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.

Tabla de contenidos

[1 Introducción 7](#_Toc43889831)

[2 Métodos de Feature Selection basados en Mutual Information 7](#_Toc43889832)

[2.1 Feature Selection 7](#_Toc43889833)

[2.2 Algoritmos Feature Selection Propuestos 7](#_Toc43889834)

[2.2.1 Mutual Information 7](#_Toc43889835)

[2.2.2 Mínima Redundancia Máxima Relevancia 8](#_Toc43889836)

[3 Metodología 9](#_Toc43889837)

[3.1 Pre procesado de datos 9](#_Toc43889838)

[3.1.1 ISBSG 9](#_Toc43889839)

[3.1.2 Filtrado 9](#_Toc43889840)

[3.1.3 Set Inicial de Features 9](#_Toc43889841)

[3.1.4 categorización 11](#_Toc43889842)

[3.2 Estimación de esfuerzo de desarrollo software basado en casos 11](#_Toc43889843)

[3.3 Comparativa de algoritmos de Feature Selection 11](#_Toc43889844)

[3.4 Multiple 3-Fold Cross Validation 11](#_Toc43889845)

[4 Resultados Experimentales 12](#_Toc43889846)

[4.1 Precisión de los algoritmos de Feature Selection 12](#_Toc43889847)

[4.1.1 Convergencia de los algoritmos 12](#_Toc43889848)

[4.2 Análisis de las features seleccionadas 12](#_Toc43889849)

[4.2.1 Información Mutua y Redundancia de las variables independientes 12](#_Toc43889850)

[4.2.2 Numero de Variables seleccionadas dependiendo de los algoritmos 14](#_Toc43889851)

[4.2.3 Preferencia de uso de las variables dependiendo del algoritmo 14](#_Toc43889852)

[5 Tecnología utilizada 14](#_Toc43889853)

[5.1 Python 14](#_Toc43889854)

[5.2 Pandas 15](#_Toc43889855)

[5.3 Scikit-learn 15](#_Toc43889856)

[5.4 Matplotlib 15](#_Toc43889857)

[6 Proceso de desarrollo 15](#_Toc43889858)

[7 Anexo 16](#_Toc43889859)

[7.1 Anexo A – Gráficas de ejecución de MI 16](#_Toc43889860)

[7.2 Anexo B – Integración de Python con R 18](#_Toc43889861)

# 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 mas 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.

## 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

## 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 las mismas 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.

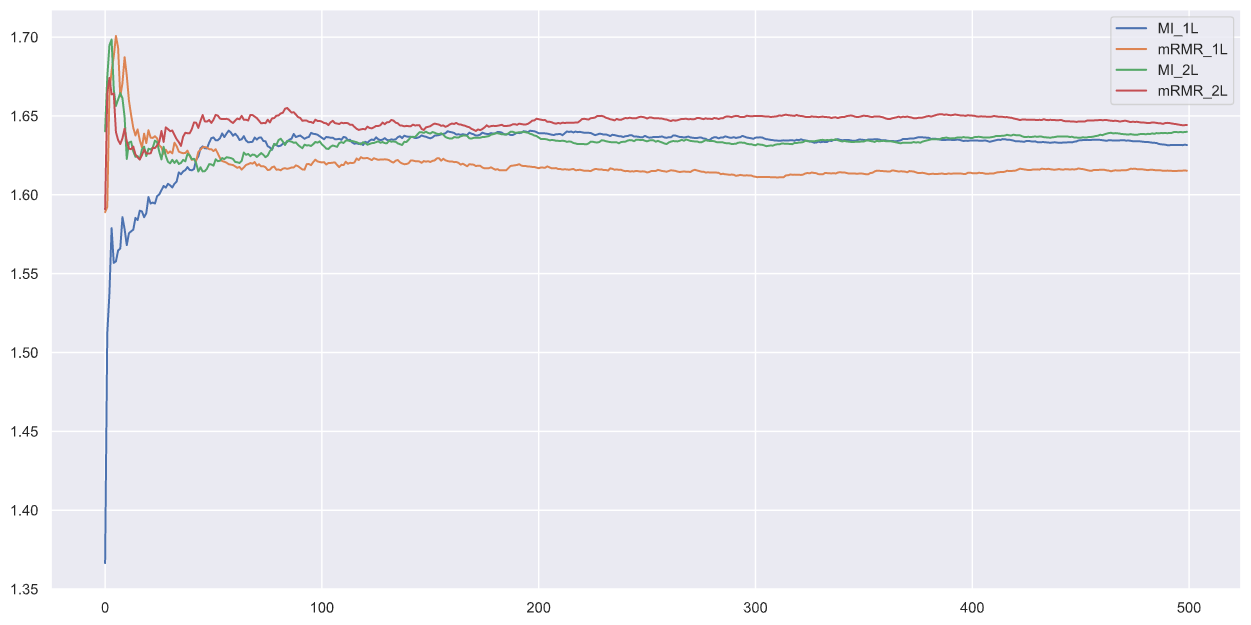


Figura 1 Evolución de las medias acumuladas de MMRE para k=1

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

### Precisión de los algoritmos de Feature Selection

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.

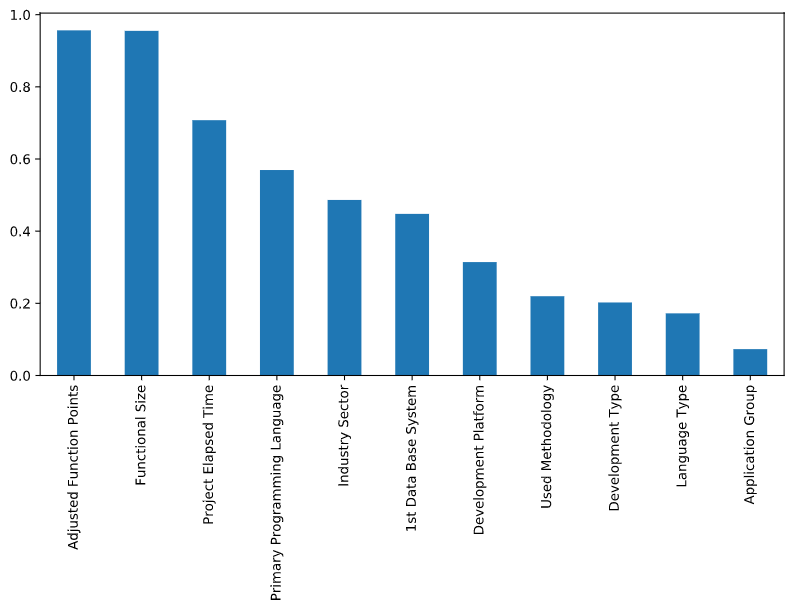
El resultado se ha obtenido con la función normalized\_mutual\_info\_score aplicada a todas las columnas del dataset. Pero se han hecho pruebas con distintas librerías tanto en Python como integrando R.

Figura 2 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 3 MRMR de las variables independientes

Figura 4 mRMR de las variables seleccionadas

### 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.

## Pandas

En computación y ciencia de datos Pandas[[2]](#footnote-2) 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[[3]](#footnote-3) 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[[4]](#footnote-4) 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

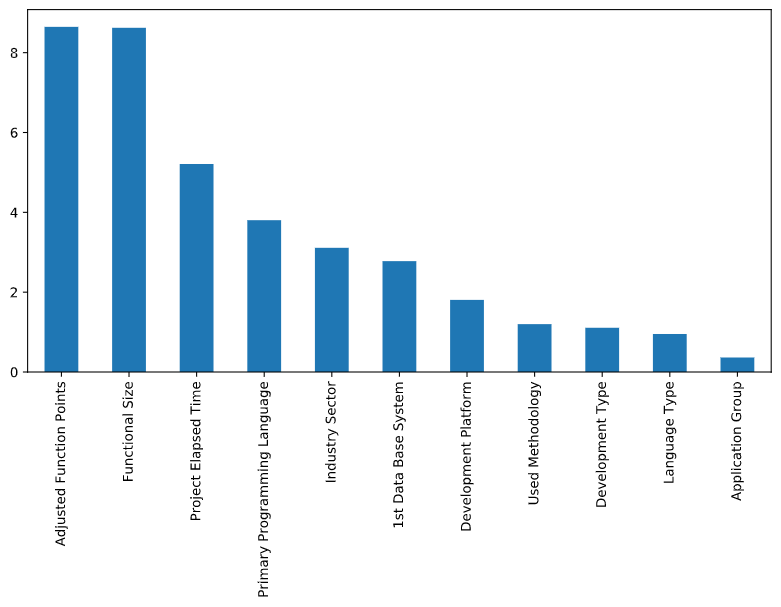
Seaborn[[5]](#footnote-5) es una librería para hacer gráficos estadísticos en Python. Está construida sobre matplot y tiene una integración muy desarrollada con las estructuras de datos de Pandas. Será la herramienta utilizada para generar gran parte de los gráficos de resultados.

# Proceso de desarrollo

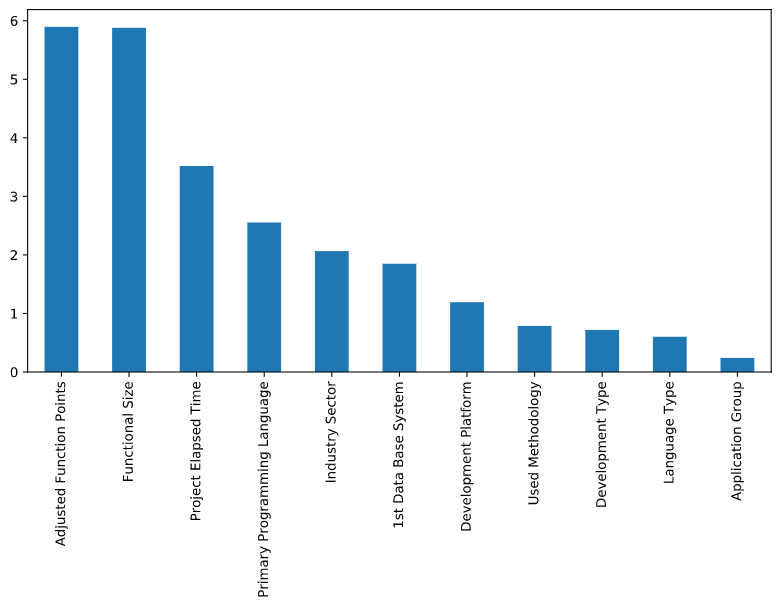
# Anexo

## Anexo A – Gráficas de ejecución de MI

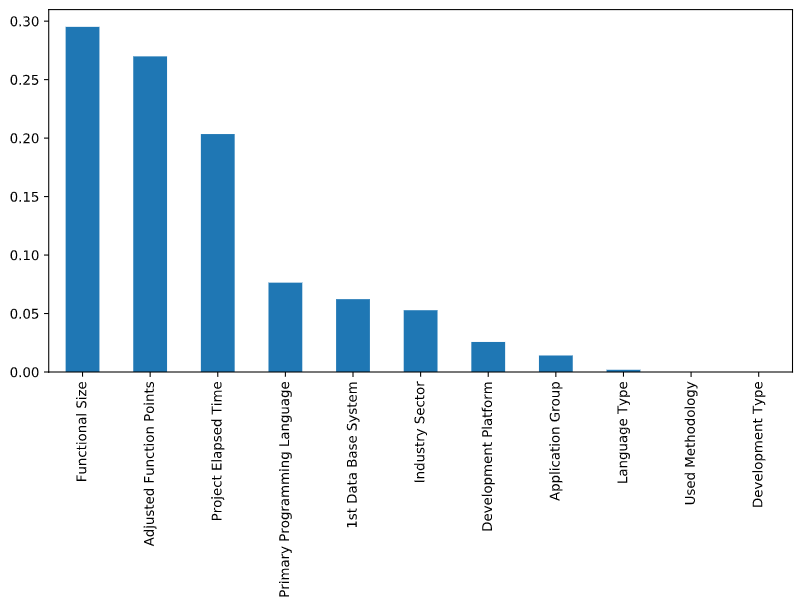
Esta primera gráfica se corresponde a la librería info\_gain.

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

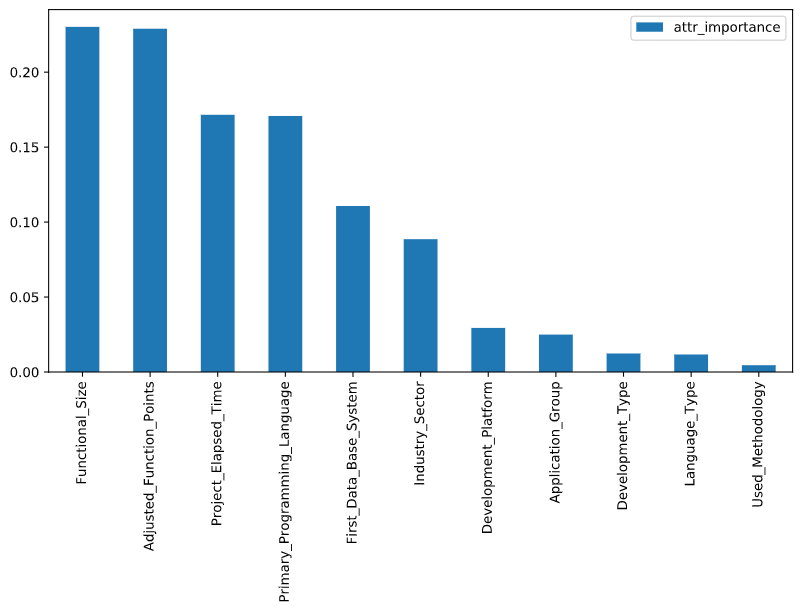
Esta grafica se hace aplicando manualmente al dataframe mutual\_info\_score de scikit

<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.

Para poder utilizar RPY2 correctamente es necesaria la iniciación de el 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.

1. <https://www.python.org/> [↑](#footnote-ref-1)
2. <https://pandas.pydata.org/> [↑](#footnote-ref-2)
3. <https://scikit-learn.org/> [↑](#footnote-ref-3)
4. <https://matplotlib.org/> [↑](#footnote-ref-4)
5. <https://seaborn.pydata.org/> [↑](#footnote-ref-5)