6/4/2021

Miguel Ángel Muñoz Vázquez - A01423629

Aarón Pérez Ontiveros – A01422524

ITESM campus cuernavaca

Proyecto Final: Agentes Basados en Conocimiento

Sistemas inteligentes



# Introducción

Los algoritmos para la clasificación de datos son: supervisados, que parte de un conjunto de datos ya clasificado y que sus atributos sirven para caracterizar dicha clasificación; y los no supervisados, que parte de un conjunto de datos no clasificado y que dicha clasificación depende de un procedimiento estadístico. Los algoritmos utilizados a continuación para la clasificación de datos fueron implementados con la librería scikit-learn utilizando el lenguaje de programación Python 3.9.4. Los algoritmos seleccionados son los siguientes:

MultinomialNaiveBayes: es una variante del algoritmo de Naive Bayes, es de aprendizaje supervisado, se utiliza para el análisis de textos.

DecisionTreeClassifier: es un algoritmo de aprendizaje supervisado que su complejidad depende del número de atributos de entrada (suele ser más rápido que una red neuronal).

RandomForestClassifier: es un algoritmo de aprendizaje supervisado, se compone de arboles de decisión y su predicción está dada por la creación aleatoria de los conjuntos de datos en arboles de decisión.

MLPClassifier: Multi-layer Perceptron, es un algoritmo de aprendizaje supervisado. Consta de múltiples redes neuronales que asignan un peso a los datos para entrenar el modelo.

KNeighborsClassifier: es un algoritmo de aprendizaje supervisado. Memoriza las instancias de entrenamiento como su base de conocimiento y para predecir las clases.

A continuación, se explica el procedimiento de clasificación de los datos. El conjunto de datos consta de dos archivos: el primero con el nombre ‘Gold-Ingles.csv’ contiene los identificadores de cada instancia y dos atributos más que corresponden a la clasificación, uno para el rango de edad y otro para el género; el otro archivo ‘English.txt’ contiene los identificadores de cada instancia, un atributo que especifica el lenguaje del texto y por último el texto (inglés en esta práctica)

# Descripción del conjunto de datos (género)

Un problema clásico que se puede encontrar en muchos ámbitos diferentes es la de detectar si un texto fue escrito por una mujer o por un hombre, ejemplo de caso real en donde se pudiera necesitar esta detección es la investigación antropológica de textos antiguos, donde es importante saber el género de quien lo escribió como uno de los primeros pasos, para delimitar posibilidades y posteriormente aplicar demás técnicas para obtener más información de ellos, como lo es la identidad de quien lo escribió (personajes históricos sobre todo).

Para los experimentos realizados en este proyecto se tomó en cuenta una base de datos, donde para cada instancia se tiene el identificador de esta, el idioma y el texto correspondiente. Por un lado, el identificador consiste en una cadena alfanumérica de 32 caracteres; por ejemplo: *0526eb9cfcee11c0036f3fa6d11158d5*. Por otro lado, el idioma esta codificado en la representación ISO 639-1, que está formado de 2 caracteres, para el caso de nuestros datos siempre es *en* (de inglés). Finalmente, el texto tiene un formato arbitrario que está delimitado por el carácter de nueva línea *\n*. Usando el texto, después de cierto preprocesamiento se pretende construir un modelo que tenga la capacidad de predecir el género (*male* o *female*)

# Preprocesamiento

El procesamiento de los datos comienza normalizando los textos para eliminar el conjunto de caracteres que no son letras y eliminando aquellas que no pertenecen a un lema que las represente. A partir de este resultado se obtiene la descripción de los datos que se van a procesar como el número de palabras existente por clase, cantidad de textos por clase y el vocabulario resultante del procesamiento anterior.

Para esta clasificación se procesaron los datos para filtrar los datos de la clase de ‘genero’ junto con sus características.

Las métricas, que se calcularon de la base de datos para conocer más a detalle los datos, se muestran en la siguiente tabla:

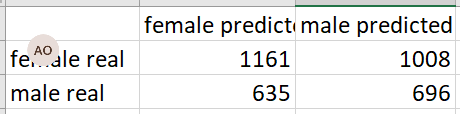
|  |  |
| --- | --- |
| **Métrica** | **Valor** |
| Número de instancias | 5000 |
| Longitud promedio de los textos por clase | male: 2216.11 | female: 2519.98 |
| Número de palabras por clase | male: 764091 | female: 905415 |
| Vocabulario | 42232 |

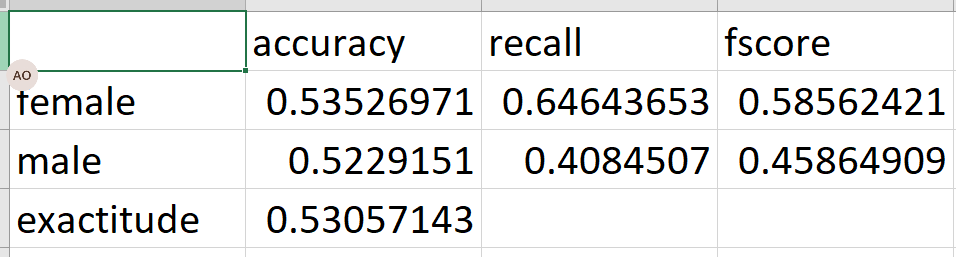
# Clasificación y Evaluación

♣ A continuación se muestra la evaluación con la clasificación de los datos con el filtro de frecuencia

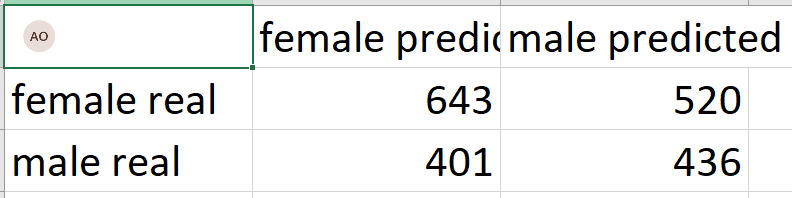
♠ Nota : Todos los nombres de los atributos de las filas y columnas están al revés. El nombre de las filas es el de las columnas y el nombre de las columnas es el de las filas.

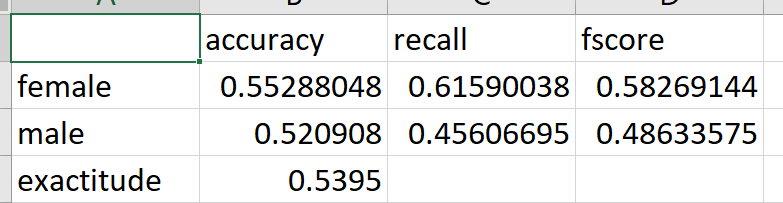
♦ DecisionTreeClassifier -> Entrenamiento: 30% de los datos



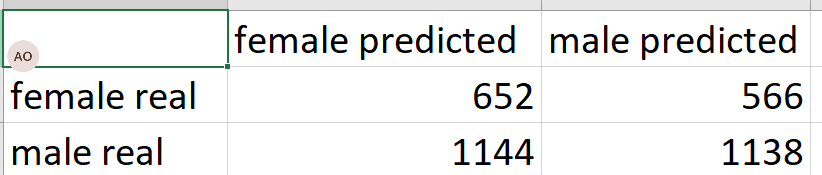


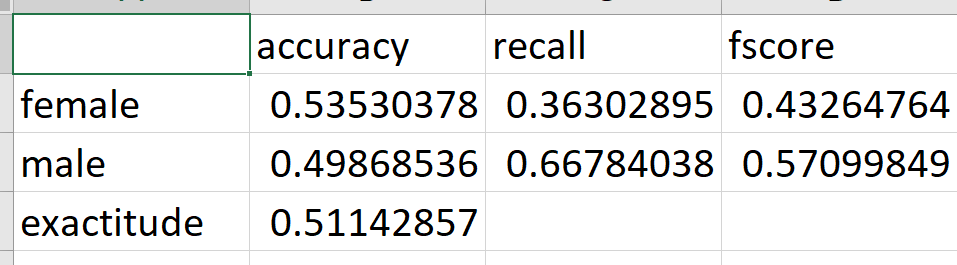
♦ DecisionTreeClassifier -> Entrenamiento: 60% de los datos



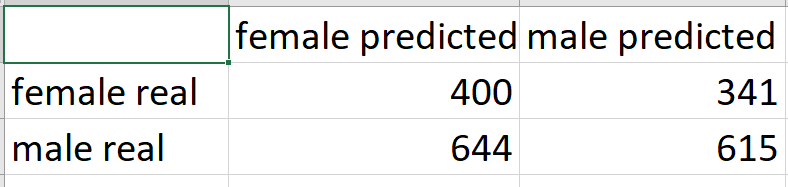


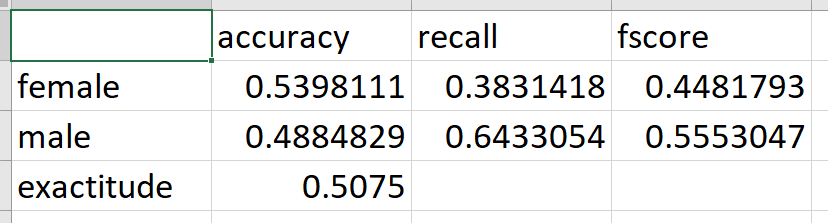
♦ KNeighborsClassifier: 30% de los datos



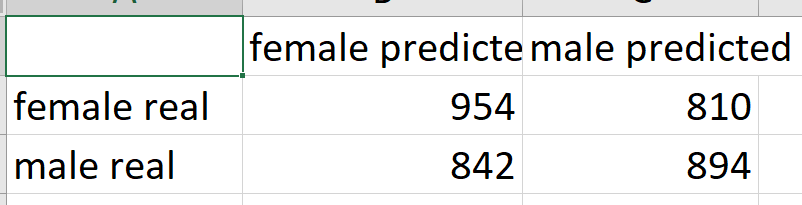


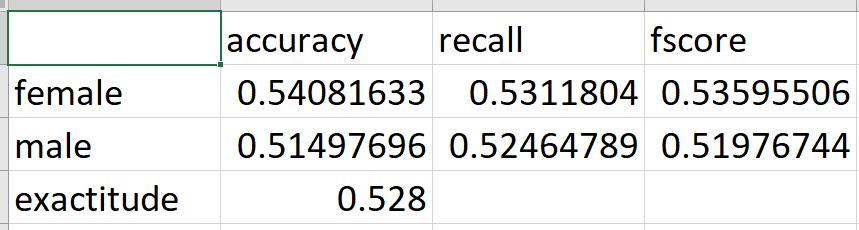
♦ KNeighborsClassifier: 60% de los datos



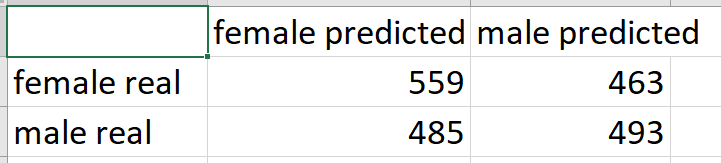


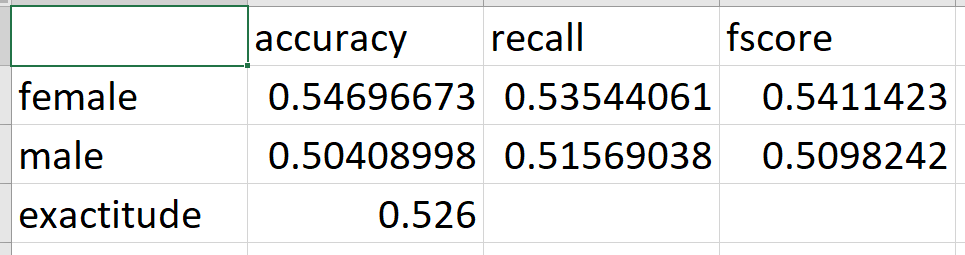
♦ MLPClassifier: 30% de los datos



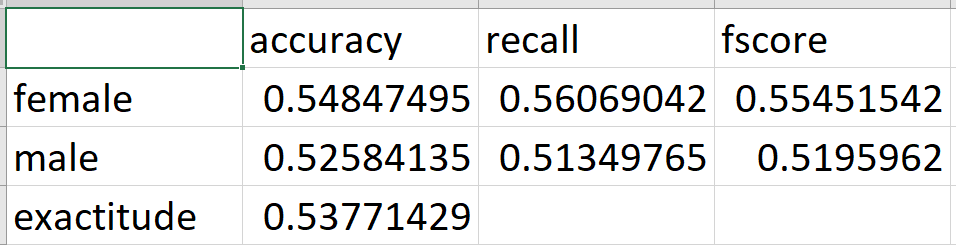
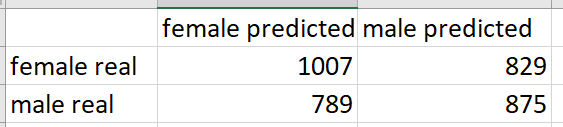


♦ MLPClassifier: 60% de los datos

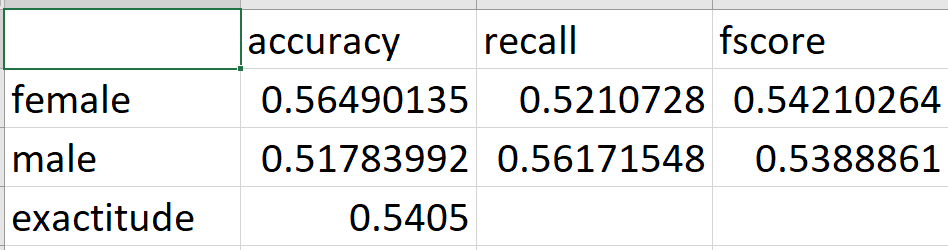
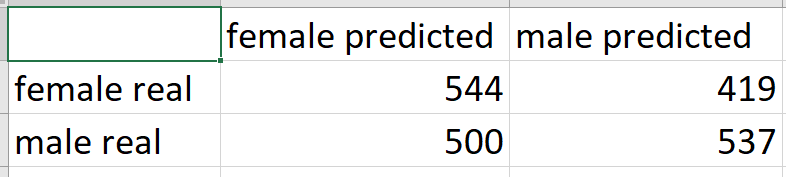




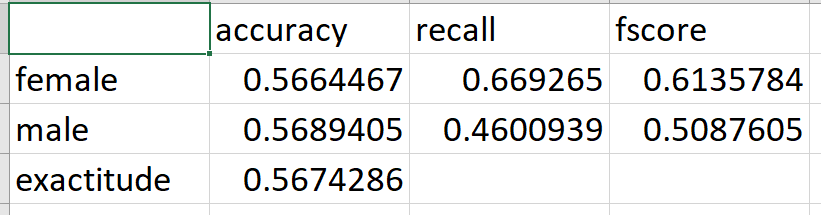
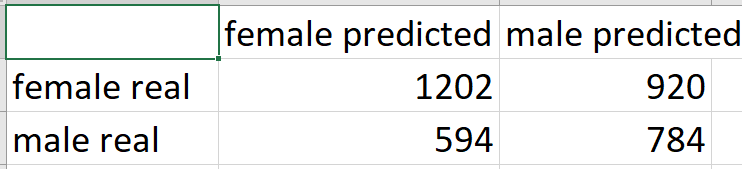
♦ MultinomialNaiveBayes: 30% de los datos



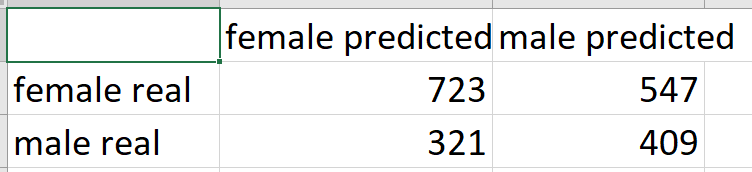
♦ MultinomialNaiveBayes: 60% de los datos

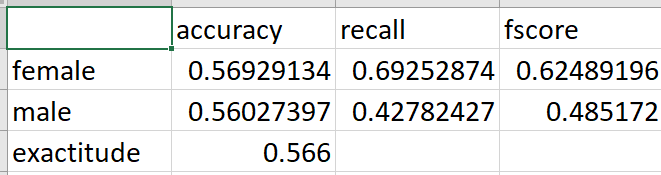


♦ RandomForestClassifier: 30% de los datos



♦ RandomForestClassifier: 60% de los datos

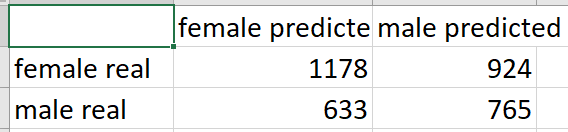


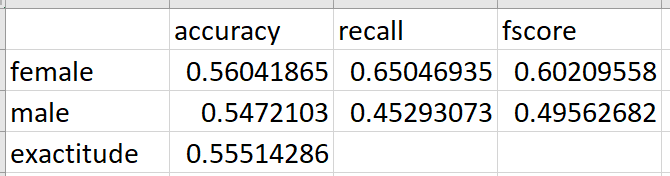


♣ A continuación se muestra la evaluación con la clasificación de los datos con el filtro de TF-IDF

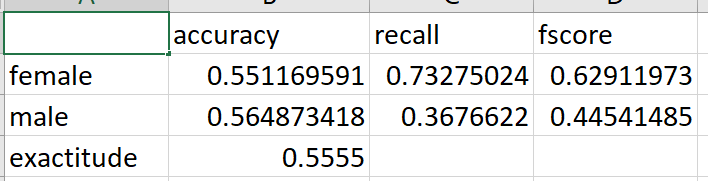
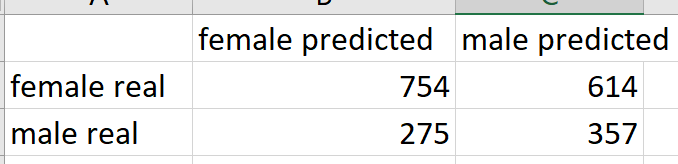
♠ Nota : Todos los nombres de los atributos de las filas y columnas están al revés. El nombre de las filas es el de las columnas y el nombre de las columnas es el de las filas.

♦ DecisionTreeClassifier -> Entrenamiento: 30% de los datos

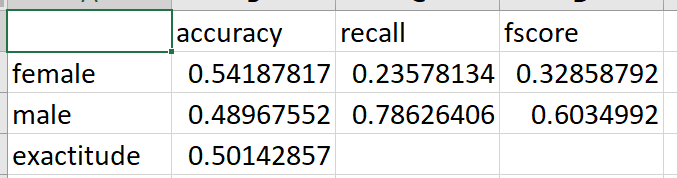
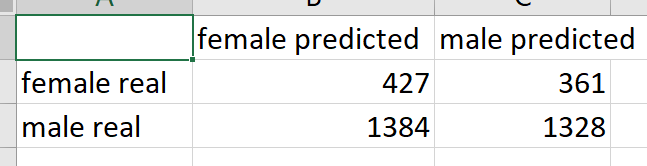




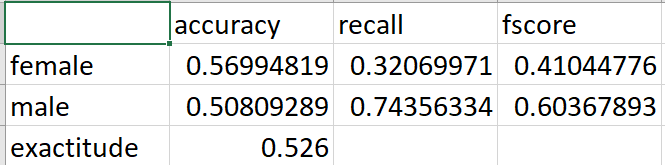
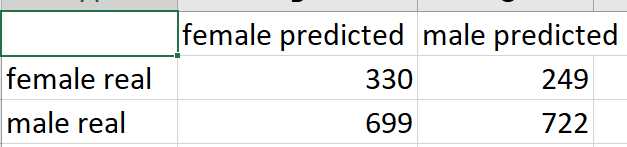
♦ DecisionTreeClassifier -> Entrenamiento: 60% de los datos



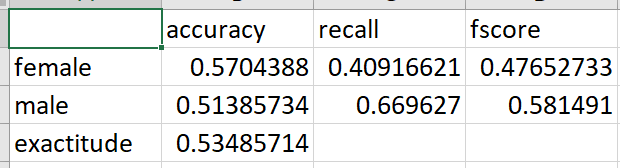
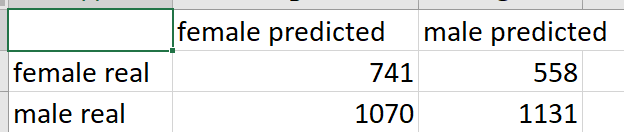
♦ KNeighborsClassifier: 30% de los datos



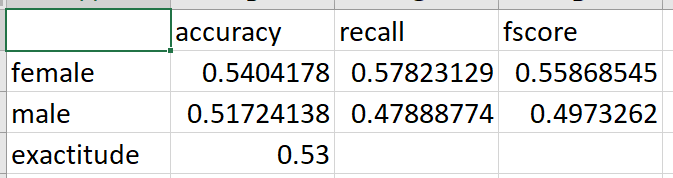
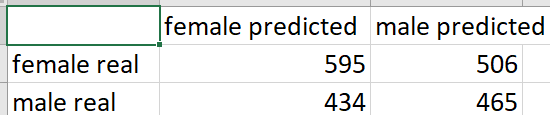
♦ KNeighborsClassifier: 60% de los datos



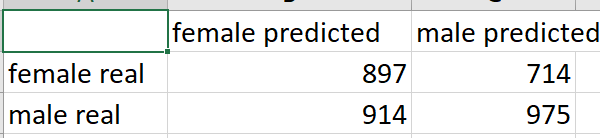
♦ MLPClassifier: 30% de los datos

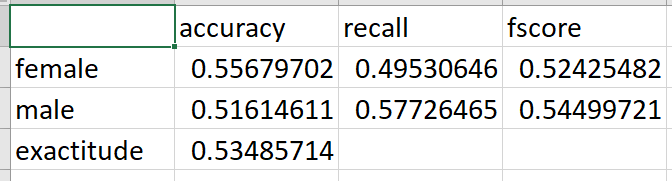


♦ MLPClassifier: 60% de los datos

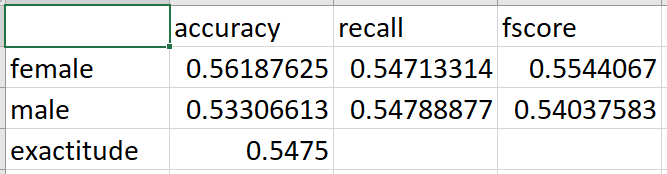
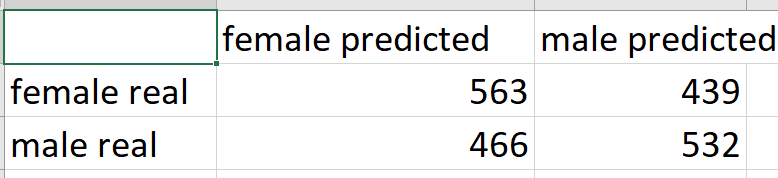


♦ MultinomialNaiveBayes: 30% de los datos

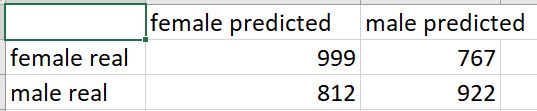


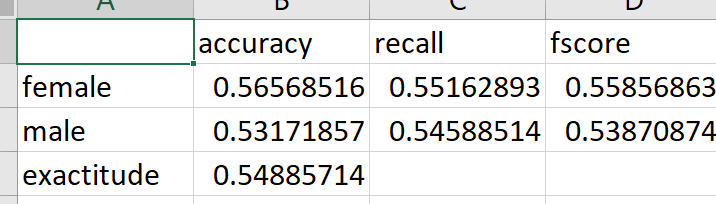


♦ MultinomialNaiveBayes: 60% de los datos

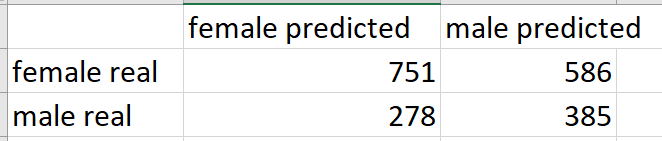


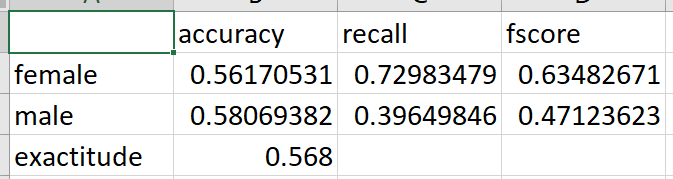
♦ RandomForestClassifier: 30% de los datos





♦ RandomForestClassifier: 60% de los datos





# Descripción del conjunto de datos (edad)

Otra forma de clasificar los datos de la base de datos a la que tenemos acceso es por medio de la edad. La clasificación se hace por rango de edades (10s, 20s y 30s). Se puede recabar mucha información en las redes sociales donde todas las personas escriben publicaciones de cierto tipo y dependiendo de la edad se expresan a su manera. En este caso se desconoce la fuente de los textos, pero leyéndolos se puede apreciar el tipo de lenguaje y por los temas se puede inferir si se trata de una puberto, un adolescente o un adulto joven.

# Preprocesamiento

El procesamiento de los datos comienza normalizando los textos para eliminar el conjunto de caracteres que no son letras y eliminando aquellas que no pertenecen a un lema que las represente. A partir de este resultado se obtiene la descripción de los datos que se van a procesar como el numero de palabras existente por clase, cantidad de textos por clase y el vocabulario resultante del procesamiento anterior.

Para esta clasificación se procesaron los datos para filtrar los datos de la clase de ‘edad’ junto con sus características.

Las métricas, que se calcularon de la base de datos para conocer más a detalle los datos, se muestran en la siguiente tabla:

|  |  |
| --- | --- |
| **Métrica** | **Valor** |
| Número de instancias | 5000 |
| Longitud promedio de los textos por clase | 10s: 2635.2 | 20s:2079.09 | 30s:2521.29 |
| Número de palabras por clase | 10s:128914 | 20s:525648 | 30s:1014944 |
| Vocabulario | 42232 |

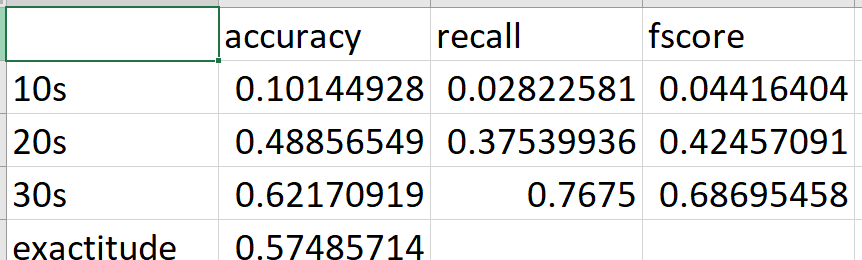
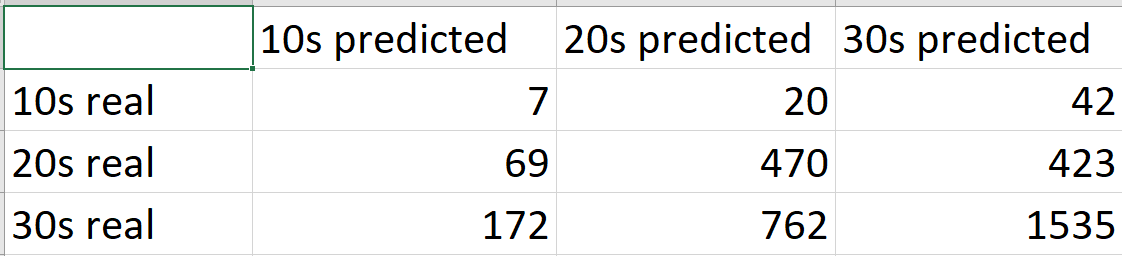
Con estas métricas conocemos la cantidad de datos existentes por cada clase que vamos a trabajar, el número de instancias y el vocabulario.

# Clasificación y Evaluación

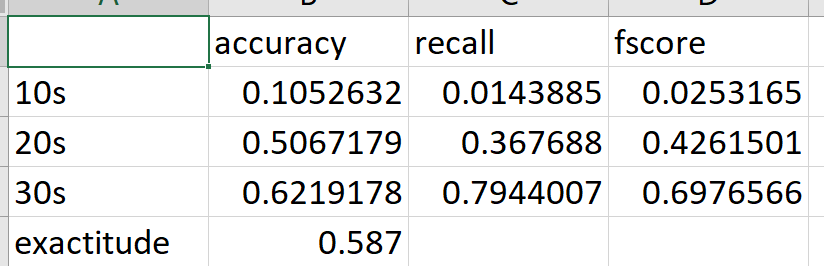
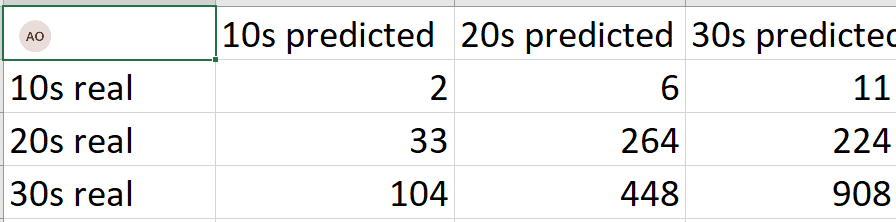
♣ A continuación se muestra la evaluación con la clasificación de los datos con el filtro de frecuencia

♠ Nota : Todos los nombres de los atributos de las filas y columnas están al revés. El nombre de las filas es el de las columnas y el nombre de las columnas es el de las filas.

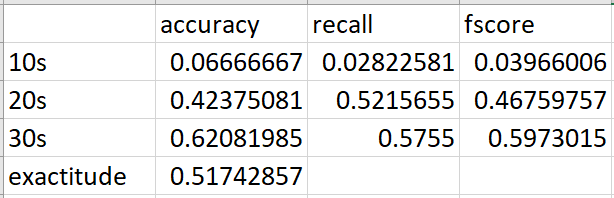
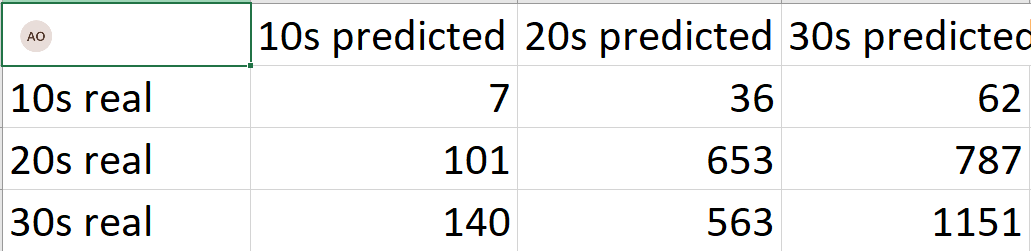
♦ DecisionTree -> Entrenamiento: 30% de los datos



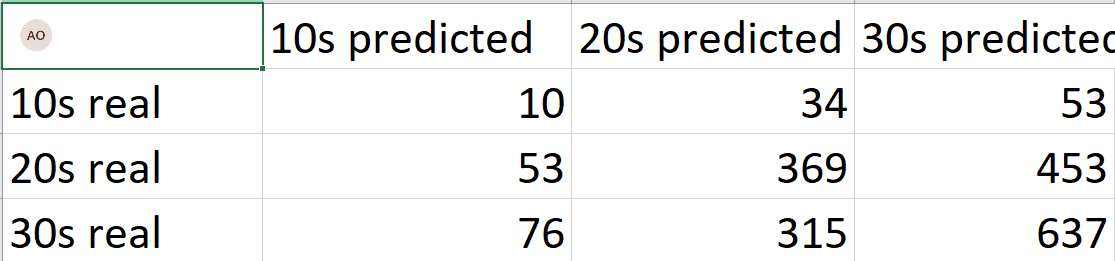
♦ DecisionTree -> Entrenamiento: 60% de los datos

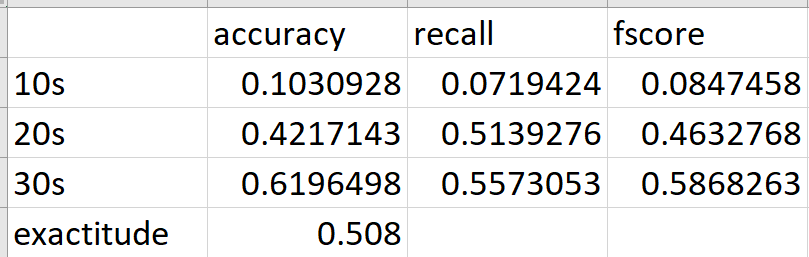


♦ KNeighborClassifier -> Entrenamiento: 30% de los datos

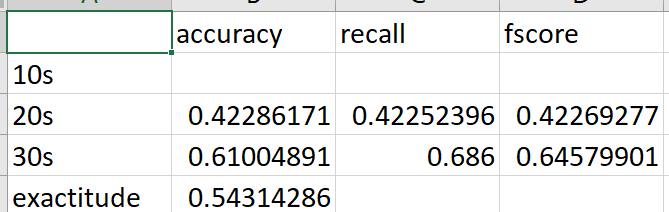
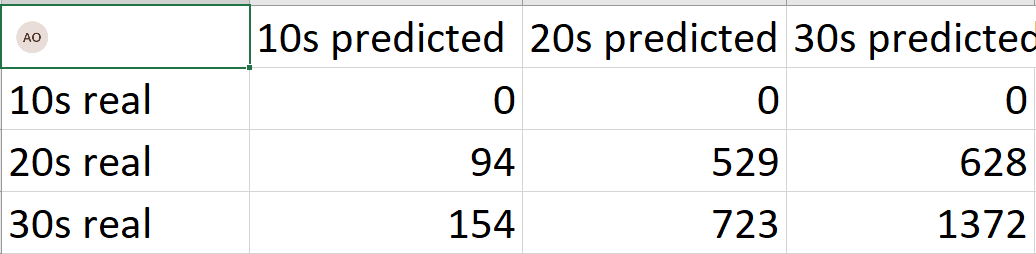


♦ DecisionTree -> Entrenamiento: 60% de los datos

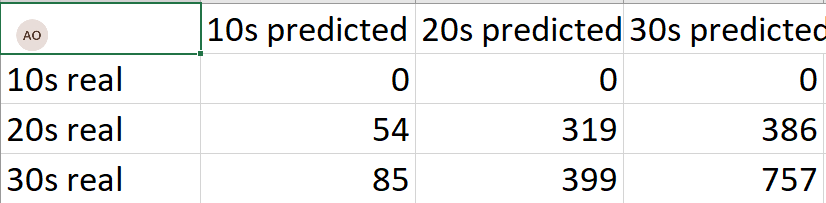


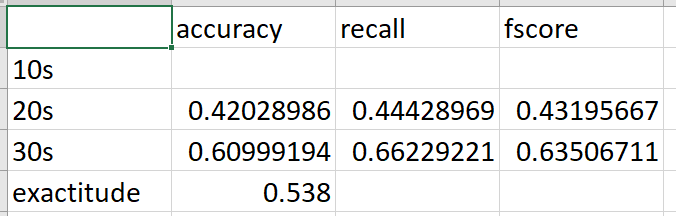


♦ MLPClassifier -> Entrenamiento: 30% de los datos

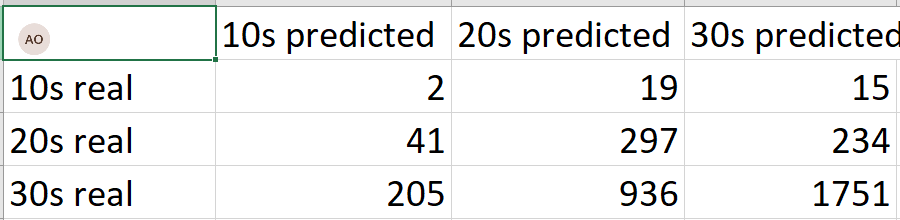


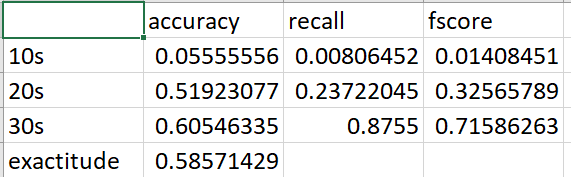
♦ MLPClassifier -> Entrenamiento: 60% de los datos



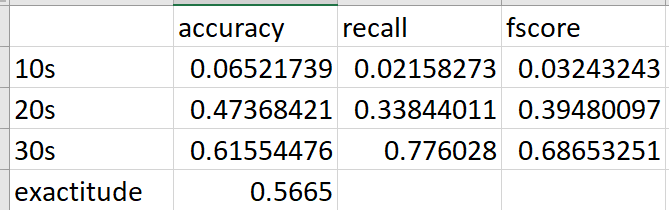
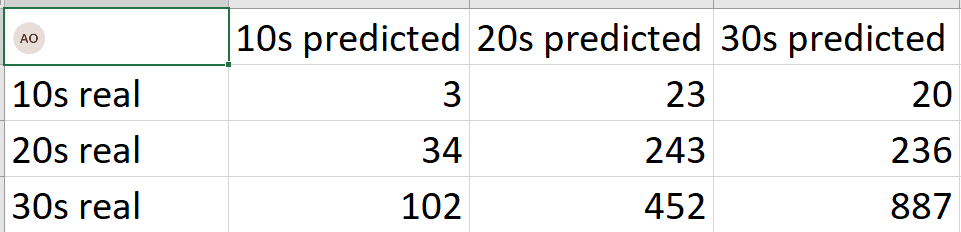


♦ MultinomialNaiveBayes -> Entrenamiento: 30% de los datos

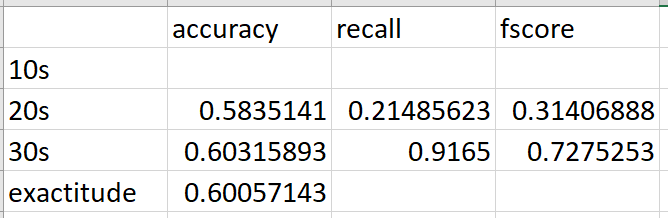
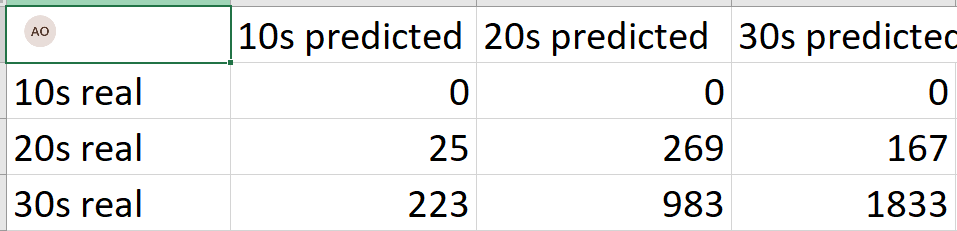




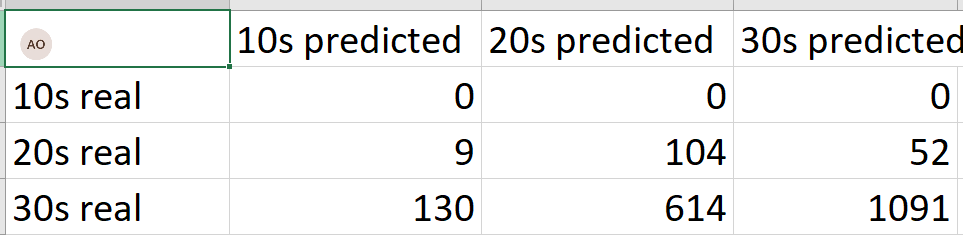
♦ MultinomialNaiveBayes -> Entrenamiento: 60% de los datos

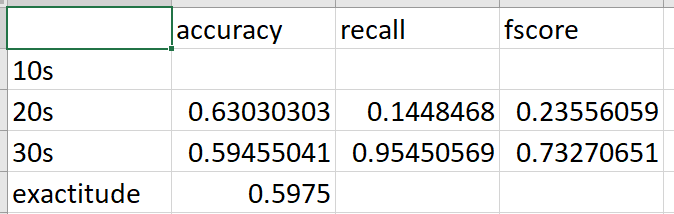


♦ RandomForest -> Entrenamiento: 30% de los datos



♦ RandomForest -> Entrenamiento: 60% de los datos

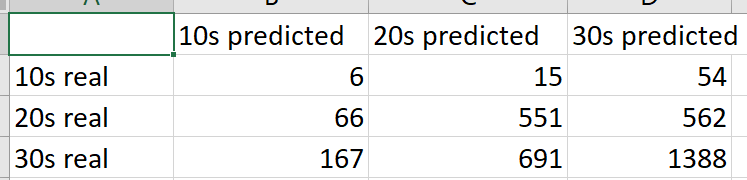
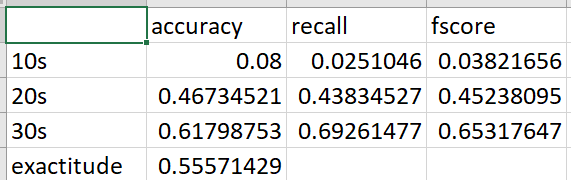




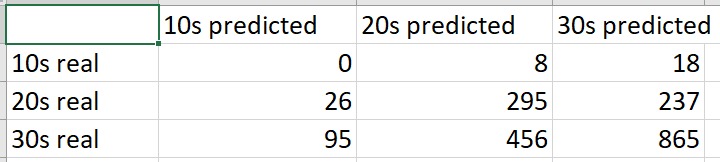
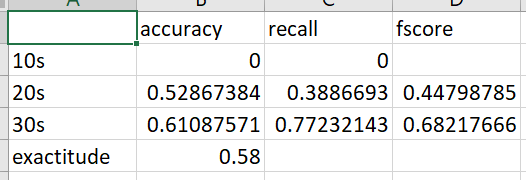
♣ A continuación se muestra la evaluación con la clasificación de los datos con el filtro de TF-IDF

♠ Nota : Todos los nombres de los atributos de las filas y columnas están al revés. El nombre de las filas es el de las columnas y el nombre de las columnas es el de las filas.

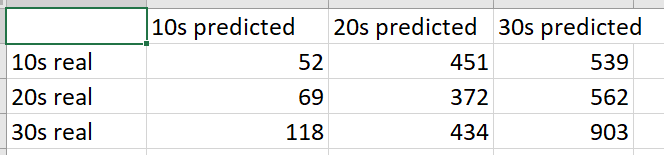
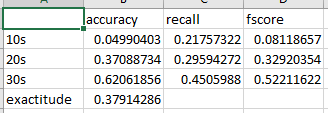
♦ DecisionTree -> Entrenamiento: 30% de los datos

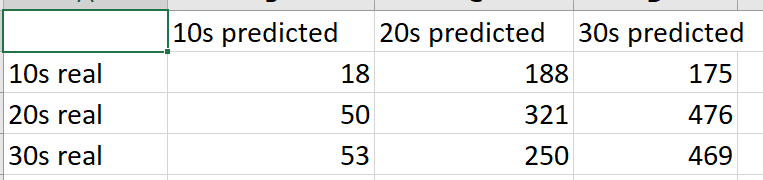
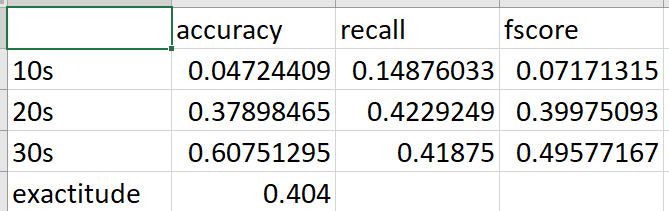
♦ DecisionTree -> Entrenamiento: 60% de los datos

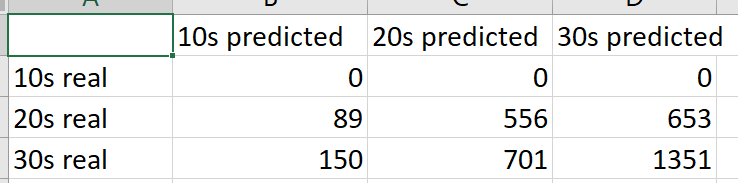
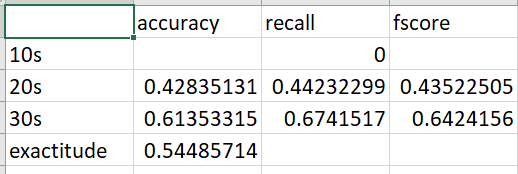
♦ KNeighborClassifier -> Entrenamiento: 30% de los datos

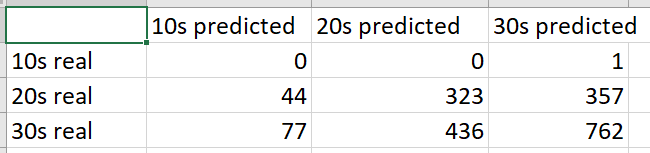
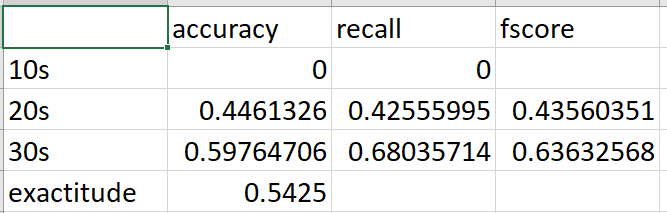
♦ KNeighborClassifier -> Entrenamiento: 60% de los datos

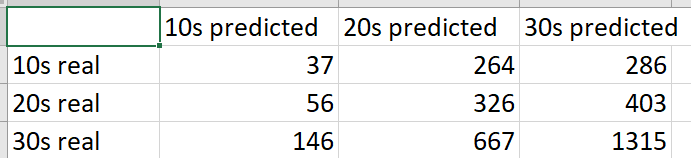
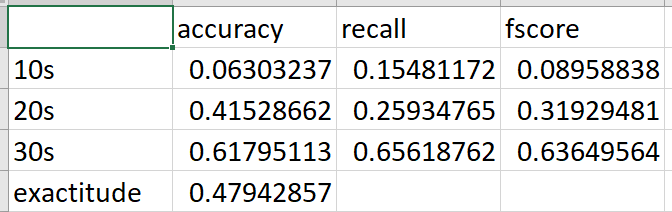
♦ MLPClassifier -> Entrenamiento: 30% de los datos

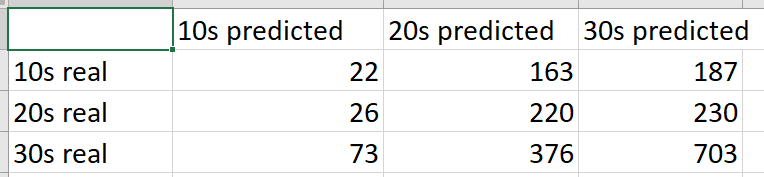
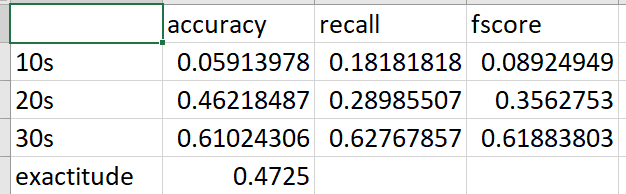
♦ MLPClassifier -> Entrenamiento: 60% de los datos

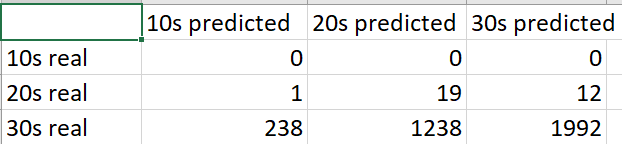
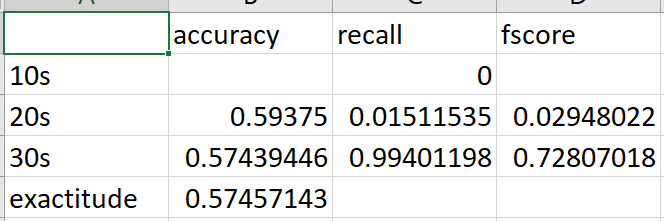
♦ MultinomialNaiveBayes -> Entrenamiento: 30% de los datos

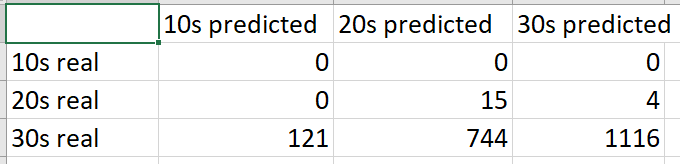
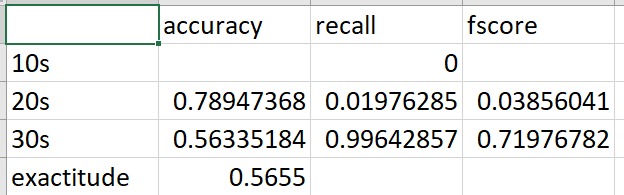
♦ MultinomialNaiveBayes -> Entrenamiento: 60% de los datos

♦ RandomForest -> Entrenamiento: 30% de los datos

♦ RandomForest -> Entrenamiento: 60% de los datos

Conclusión

Pudimos observar a primera vista que para la característica por frecuencia simple, RandomForestClassifier con 30% de los datos como entrenamiento fue el mejor algoritmo con una exactitud de 0.57 para los datos de género. De manera similar con 30% de los datos como entrenamiento, RandomForestClassifier fue el mejor algoritmo con exactitud de 0.60 para los datos de edad. Por otro lado, DecisionTree con entrenamiento de 60% fue el que peor rindió para los datos de edad con una exactitud de 0.50; mientras que KNeighborsClassifier con 60% de los datos fue el peor para los datos de género. Probablemente random forest fue el mejor con pocos datos de entrenamiento porque con ello no se presentó problema de overfitting. Decision Tree fue malo porque presentó el problema de overfitting para edades, ya que teníamos más clases. KNeighborsClassifier fue malo probablemente porque se trataban de sólo dos clases. En general se podría decir que Random Forest es la mejor opción en general.

Con la caracteristica de frecuencia inversa sucede que el algoritmo de random forest obtiene mejores predicciones.

# Bibliografía

Russell, S., Norvig P. (2004). *Inteligencia Artificial: un enfoque moderno.*

Russell, S., Norvig P. (2016). *Artificial Intelligence: a modern approach.*

Clasificación supervisada y no supervisada en ArcGIS | El blog de franz. (2021). Retrieved 9 June 2021, from <https://acolita.com/clasificacion-supervisada-no-supervisada-en-arcgis/>

1.9. Naive Bayes — scikit-learn 0.24.2 documentation. (2021). Retrieved 9 June 2021, from <https://scikit-learn.org/stable/modules/naive_bayes.html>

Navlani, A. (2021). Retrieved 9 June 2021, from <https://www.datacamp.com/community/tutorials/decision-tree-classification-python>

Navlani, A. (2021). Retrieved 9 June 2021, from <https://www.datacamp.com/community/tutorials/random-forests-classifier-python>

Robinson, S. (2021). Introduction to Neural Networks with Scikit-Learn. Retrieved 9 June 2021, from <https://stackabuse.com/introduction-to-neural-networks-with-scikit-learn>

1.6. Nearest Neighbors — scikit-learn 0.24.2 documentation. (2021). Retrieved 9 June 2021, from <https://scikit-learn.org/stable/modules/neighbors.html>