PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

FACULTAD DE INGENIERÍA

ESCUELA DE INGENIERÍA INFORMÁTICA

Detección y corrección de errores basados en reglas gramaticales del inglés en conjunto con el mejoramiento de estilos de escritura, aplicado en artículos científicos mediante Algoritmos de Procesamiento de Lenguaje Natural (NLP)

Jorge Patricio Montiel Montiel

Profesor Guía: **Héctor Allende Cid.**

Profesor Co-referente: **Wenceslao Palma Muñoz.**

INFORME DE PROYECTO PARA OPTAR A TITULO PROFESIONAL DE INGENIERO CIVIL EN INFORMÁTICA

DICIEMBRE, 2018

Índice

Resumen

Bajo el contexto de que existen diversos problemas para detectar errores gramaticales relacionados con diversos aspectos del lenguaje inglés y a la vez asumiendo la existencia de herramientas para realizar procesamiento del lenguaje natural, podemos suponer que la utilización de dichas herramientas nos podría ayudar a resolver muchos de esos problemas. Es por esto que una de las principales temáticas de este trabajo es desarrollar ideas tanto innovadoras como ya existentes, en las cuales serán utilizadas dichas herramientas con el fin de solventar problemáticas de carácter complejo o simple. En cuanto al desarrollo del contenido, en una primera instancia se busca aclarar el contexto tanto del problema principal como sus posibles soluciones, abordando diversos conceptos claves como Machine Learning, Deep learning, procesamiento del lenguaje natural (NLP), errores semánticos, etc. Para luego relacionar las diversas herramientas disponibles en el mercado, en donde algunas de estas herramientas son las bibliotecas de Python como "Part of Speech" (POS)[1] o simplemente plataformas web, cabe mencionar que la mayoría de ellas se pueden encontrar con libre acceso para la comunidad. Por otro lado, también se da un espacio para proponer ideas poco convencionales para el procesamiento de texto, tales como conversión de texto a ondas o series de datos, pero siempre enmarcado en un contexto de carácter explicativo-teórico. En último lugar se presenta un modelo que intenta abordar la predicción de un punto utilizando tecnología referente a modelos de clasificación, tales como SGDClassifier, SVM, GaussianNB y logistic Regression [9]. Adicionalmente, para buscar una mayor exactitud en dichos modelos y obtener una variabilidad más amplia de resultados se utiliza el método de descarte de características RFE con diversos estimadores. Finalmente, para concluir el trabajo se dan a conocer algunos aspectos importantes a considerar para la utilización de las herramientas antes descritas, también se dan a conocer los resultados de los experimentos realizados en base al modelo propuesto, en donde dichos resultados son comparados bajo 4 métricas acordes a problemas de clasificación.

Palabras-claves: inglés, errores gramaticales, errores semánticos, NLP, POS, series de datos, SGDClassifier, SVM, GaussianNB, logistic regression, RFE, estimadores.

[1] Página: https://spacy.io/usage/linguistic-features#section-pos-tagging, último acceso 3 de Abril del 2018.

Abstract

Under the context that there are various problems to detect grammatical errors related to various aspects of the English language and at the same time assuming the existence of tools to perform natural language processing, we can assume that the use of these tools could help us solve many of those problems. That is why one of the main themes of this work is to develop both innovative and existing ideas, in which these tools will be used in order to solve complex or simple problems. Regarding the development of the content, in the first instance it seeks to clarify the context of both the main problem and its possible solutions, addressing various key concepts such as Machine Learning, Deep Learning, natural language processing (NLP), semantic errors, etc. To then relate the various tools available in the market, where some of these tools are Python libraries such as "Part of Speech" (POS) or simply web platforms, it should be mentioned that most of them can be found with free access to community. On the other hand, there is also a space to propose unconventional ideas for the text processing, such as the text conversion to waves or series of data, but always framed in a context of theoretical explanatory nature. Finally, we present a model that tries to approach the prediction of a point using technology related to classification models, such as SGDClassifier, SVM, GaussianNB and logistic Regression [9]. Additionally, to seek greater accuracy in these models and obtain a wider variability of results, the discard method of RFE characteristics with various estimators is used. Finally, to conclude the work, some important aspects to be considered for the use of the tools described above are disclosed, the results of the experiments carried out based on the proposed model are also disclosed, where said results are compared under 4 metrics according to classification problems.

Keywords: English, grammatical errors, semantic errors, NLP, POS, data series, SGDClassifier, SVM, GaussianNB, logistic regression, RFE, estimators.

Siglas

AI: Artificial Intelligence.
API: Aplication Programming Interface.
BSD: Berkeley Software Distribution.
DL: Deep Learning.
JSON: JavaScript Object Notation.
ML: Machine Learning.
GNB: Gaussian Naïve Bayes.
NLP: Natural Language Processing.
NLTK: Natural Language Toolkit.
POS: Part of Spreech.
RFE: Recursive Features Elimination.
SGD: Stochastic Gradient Descent.
SVC: Support Vector Classification.
SVM: Support Vector Machine.

Lista de Figuras

Lista de Tablas

1. Introducción

1.1. Motivación del Proyecto

Hoy en día la generación de innovación es fundamental para incrementar la productividad de los países ayudando en su crecimiento y desarrollo, una de las principales fuentes es la investigación, la cual se da a conocer a través de documentos científico-técnicos que generan un canal de comunicación escrito para los científicos, dicho canal les permite difundir y compartir descubrimientos con sus pares, en el caso de considerar la producción de dichos artículos permitiría medir y acreditar a un país o entidad si está realizando y produciendo investigación de forma objetiva.

Según el contexto antes mencionado, uno de los grandes problemas que está afectando a los países latinoamericanos es la baja productividad científica en comparación con otros países más desarrollados. Bajo rigurosos estudios se destaca que uno de los factores clave en dicha problemática es la falta de entrenamiento en distintos aspectos de la gramática inglesa que poseen los autores de artículos científicos latinoamericanos, idioma el cual es uno de los más frecuentes en este tipo de documentos, pese a la existencia de alrededor de 7000 idiomas en el mundo[2], imposibilitando que sus publicaciones sean aceptadas debido a la falta de coherencia o simples errores basados en reglas de dicha gramática.

Es por esto que se decide ahondar en los aspectos de la corrección gramatical en el lenguaje Ingles, abordando específicamente el tema de la reglamentación en el aspecto de cuándo debe ir un punto en un párrafo, esto haciendo una analogía con el tratamiento de las series de datos y la clasificación de características antes de una etiqueta en particular.

1.2. Definición de objetivos

Los objetivos a abordar en este trabajo se basan principalmente en tareas de investigación sobre herramientas y tecnologías para el procesamiento del lenguaje natural, sin embargo, también se da el espacio para proponer ideas innovadoras con el objetivo de realizar análisis de textos de forma no convencional.

Objetivo general:

- Desarrollar un sistema para la detección y/o clasificación de patrones lingüísticos de alto nivel en artículos científicos escritos en inglés.

Objetivos específicos:

- Investigar sobre herramientas o plataformas web que ayuden a solucionar problemas con respecto a errores gramaticales.
- Implementación y pruebas con bibliotecas relacionadas a la corrección de errores gramaticales.
- Implementación de funciones que intenten resolver errores gramaticales, específicamente errores de puntuación.

[2] Página: http://www.ethnologue.com/, último acceso 3 de Abril del 2018.

2. Estado del Arte

2.1. Herramientas Relacionadas

Pese a la existencia de diversas herramientas disponibles en el mercado, según las cuales pueden cubrir con éxito algunos de los problemas más sofisticados en el ámbito de los errores basados en reglas, no se ha podido encontrar alguna herramienta que refleje de forma precisa y con detalles los errores de carácter gramatical (basados en reglas) de los cuales pudiesen entregar un "feedback" más preciso o con un nivel de detalle que permita hacer correcciones constructivas en algún texto. Esto ya sea de carácter científico como cualquier otro que necesite algún formato en específico, sin embargo, algunas de ellas permiten acotar el esfuerzo en crear funciones desde cero entregando funcionalidades clave que nos ayudan a no reinventar la rueda.

Para obtener una idea y un primer acercamiento con dichas herramientas se realizó un seguimiento de diversas plataformas web y bibliotecas que permiten la autocorrección de palabras mal escritas o con errores ortográficos ("Spell Checking"). Cabe mencionar que la priorización en la selección de estas bibliotecas se basa en el lenguaje de programación Python en su versión 3.6 debido a su gran comunidad de desarrolladores que otorgan una gran cantidad de bibliotecas complementarias para el manejo de grandes volúmenes de datos, también posee una sintaxis más simple y abstracta que permite hacer un trabajo más impecable en cuanto a la lectura de código, ya que se reduce considerablemente la cantidad de líneas de código al momento de desarrollar funciones complejas.

Finalmente otra de las razones en la utilización de dicha herramienta se debe a la "Data" disponible para la realización de pruebas en un entorno controlado, haciendo referencia a dicho entorno de desarrollo se utiliza "Jupyter" en su versión web, esto debido principalmente a dos razones, en primer lugar permite un acceso controlado por un sistema de cuentas de usuarios en las cuales los recursos del servidor son administrados de forma óptima para la ejecución de pruebas, en segundo lugar permite el acceso a los datos de prueba[3] de forma remota con solo ingresar a la IP de una VPN creada por el laboratorio de "BigData" de la escuela de ingeniería en informática de la PUCV.

Manteniendo en cuenta los componentes anteriormente descritos podemos ahondar en algunas de las categorías propuestas en el análisis de herramientas disponibles hasta la fecha, dichas categorías corresponden a herramientas de corrección ortográfica y gramatical, herramientas de NLP y herramientas de manejo del corpus.

2.1.1. Herramientas de corrección ortográfica y gramatical

La corrección automatizada de la ortografía y la gramática es una realidad que se encuentra presente en diversas herramientas de acceso libre en la web. Uno de los casos más atendidos es la biblioteca para Python en su versión 3.3 o 2.7 llamada "Language-check 1.1"[4]. el cual si bien presenta una serie de funcionalidades que permiten corregir entradas de palabras, no logra cumplir con la entrega de un "feedback" más refinado en cuanto a las reglas de puntuación en la gramática del inglés que se quiere abordar.

[3] Para más detalles sobre los datos utilizados referirse a la sección 3.1 Datos de Prueba.

[4] Página: https://pypi.python.org/pypi/language-check, último acceso 3 de Abril del 2018.

Bajo el mismo lenguaje antes mencionado pero solo en su versión 3, se tiene el caso particular de la biblioteca llamada "Autocorrect - Spelling Corrector"[5], el cual permite auto-corregir palabras mal escritas en el idioma inglés o incluso es capaz de llegar a una suposición de corrección, por ejemplo al introducir la palabra "inteligen" el autocorrector es capaz de redefinir la cadena de caracteres y formar la palabra a la cual se quiere llegar ("intelligent"), sin embargo su uso carece de una retroalimentación que permita discernir los errores ortográficos y gramaticales al igual que "Python-Ginger"[6].

Por otro lado se tiene la opción que ofrece Microsoft Azure, el cual entrega servicios de "Spell Check API"[7] a través de una API compatible con 6 lenguajes de programación hasta la fecha, este servicio requiere de la creación de una cuenta de usuario para poder recibir una "key" que permitirá el acceso a sus servicios complementarios, el cual se basa en un JSON con las características "token", "type" y "suggestion" indexadas por una característica llamada "offset", este output permite corregir la palabra ingresada mostrando un listado de sugerencias ordenadas por un puntaje que determina su grado de relevancia.

De igual forma como Microsoft se impone en esta área, existen otros como "Virtual Writing Tutor"[8] el cual ofrece una API que proporciona un servicio web para verificar la ortografía, la gramática y la puntuación de cualquier texto en inglés con la facilidad de integrar dichos servicios de revisión en cualquier sitio web o aplicación. Luego para finalizar el análisis de bibliotecas de apoyo tenemos "After the Deadline"[9], el cual posee una amplia cantidad de bibliotecas de código abierto, además dentro de sus dominios se encuentran varios proyectos disponibles en "Github", entregando la facilidad de utilizarlos para crear nuevas versiones.

Como caso anexo, tenemos bibliotecas que nos permiten hacer "Deep Learning" en Python, tales como "Keras"[10] y "Scikit-learn"[11]. Bajo el primer caso la creación de prototipos es fácil y rápida gracias a su facilidad de uso y modularidad, además es estable con los recursos de GPU y CPU teniendo compatibilidad con Python 2.7-3.6. Para el segundo caso otorga herramientas simples y eficientes para la minería de datos o el análisis de estos, sin embargo, una de las mejoras características es que es accesible para todos y reutilizable en varios contextos.

2.1.2. Herramientas NLP

Para el tratamiento de problemáticas más complejas en el área de la investigación informática, en donde dichos problemas relacionados con el procesamiento del lenguaje natural se hacen presente bajo una serie de inconsistencias gramaticales o incluso semánticas, es que se busca utilizar herramientas que permiten simplificar dichos problemas, debido a lo

[5] Página: https://github.com/phatpiglet/autocorrect, último acceso 3 de Abril del 2018.
[6] Página: https://github.com/zoncoen/python-ginger, último acceso 3 de Abril del 2018.
[7] Página: https://docs.microsoft.com/en-us/azure/cognitive-services/bing-spell-check/proof-text, último acceso 4 de Abril del 2018.
[8] Página: https://virtualwritingtutor.com/api.php, último acceso 4 de Abril del 2018.
[9] Página: http://www.afterthedeadline.com/development.slp, último acceso 4 de Abril del 2018.
[10] Página: https://keras.io/, último acceso 5 de Abril del 2018.
[11] Página: http://scikit-learn.org, último acceso 5 de Abril del 2018.

antes descrito es que nacen propuestas tales como "Natural Language Toolkit"[12] (NLTK), el cual consiste en una plataforma líder para construir programas de Python, que permiten trabajar con datos de lenguaje humano proporcionando interfaces fáciles de usar a más de 50 recursos corporales y léxicos como WordNet, junto con un conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización, derivación, etiquetado, análisis y razonamiento semántico como se propone según el libro "Deep-Learning with Python" [6].

Por otro lado, también tenemos "Gensim"[13] cuyo comienzo se basó en una colección de varios scripts de Python para la Biblioteca Checa de Matemática Digital dml.cz en 2008, sin embargo, dicho trabajo sirvió para generar una breve lista de los artículos más similares a un artículo dado, de allí su composición del nombre como gensim ("generar similar").

Por último, tenemos "SpaCy"[14] que ofrece como estructuras de datos centrales 2 objetos llamados "Doc" y "Vocab", el primer objeto posee una secuencia de tokens y todas sus anotaciones. Mientras que el objeto "Vocab" posee un conjunto de tablas de búsqueda que hacen que la información común esté disponible en todos los documentos, por ende al centralizar cadenas, vectores de palabras y atributos léxicos, evitamos almacenar copias múltiples de datos con la finalidad de ahorrar memoria y asegura que hay una única fuente, la arquitectura de esta solución propuesta se basa en las referencias de los trabajos realizados por Eliyahu Kiperwasser y Yoav Goldberg [5] junto con una serie de otras investigaciones referentes al tema [2, 11, 12, 15, 16].

2.1.3. Herramientas para el manejo del corpus

Para manejar datos de texto existen diversas herramientas web y bibliotecas que permiten realizar esta acción de forma eficiente y cómoda, como ejemplos claros tenemos "Pandas"[15] y "Numpy"[16], los cuales son reconocidas distribuciones dentro del área científica de la informática.

Para el caso de "Numpy" posee funciones sofisticadas relacionadas con el álgebra lineal, transformada de Fourier y muchas otras que permiten una mayor abstracción de desarrollo por parte de los usuarios.

Mientras que en el caso de "Pandas" nace para complementar el entorno de desarrollo en "Python", con el objetivo de mejorar aspectos del análisis y el modelado de datos, permitiendo llevar a cabo un flujo de trabajo de análisis de datos más estable sin necesidad de cambiar de lenguaje o plataforma.

[12] Página: http://www.nltk.org/, último acceso 5 de Abril del 2018.
[13] Página: https://radimrehurek.com/gensim/, último acceso 5 de Abril del 2018.
[14] Página: https://spacy.io/, último acceso 5 de Abril del 2018.
[15] Página: https://pandas.pydata.org/, último acceso 5 de Abril del 2018.
[16] Página: http://www.numpy.org/, último acceso 5 de Abril del 2018.

3. Marco Teórico

3.1. Datos de Prueba

A lo largo de este trabajo se pueden apreciar una gran cantidad de herramientas capaces de procesar datos de texto, en donde dichas herramientas requieren de una fuente confiable que proporcione una cantidad considerable de estos datos y en lo posible los suficientes como para generar información relevante para un posterior análisis.

Debido a las condiciones antes descritas también se debe considerar que dicha fuente reúna las características necesarias para tratar el tema de interés a estudiar, esto conlleva a la creación de la base de datos que reúne una cantidad considerable de papers, los cuales permitirán realizar diversos experimentos en base al procesamiento del lenguaje natural.

Por otra parte, dichos datos de texto deben ser almacenados de tal forma que puedan ser tratados con diversas herramientas externas, las cuales permitan un manejo fácil y eficiente de todos los datos de texto.

3.1.1. El trabajo con datos de texto

Dentro de los distintos tipos de datos que se pueden utilizar para realizar aprendizajes tenemos los datos de texto, los cuales son una de las formas más extendidas de datos de secuencia, esto se entiende como una secuencia de caracteres o una secuencia de palabras, cabe mencionar que es más común el trabajo a nivel de palabras sobre las demás categorías.

Existen modelos "Deep-Learning sequence-processing"[17] que pueden utilizar texto para producir una forma básica de comprensión del lenguaje natural, la cual pueden llegar a ser lo suficientemente aptas para aplicaciones que incluyen clasificación de documentos, análisis de sentimientos, identificación del autor e incluso preguntas y respuestas, los cuales deben estar en un contexto restringido. Cabe destacar que estos modelos solo pueden mapear la estructura estadística del lenguaje escrito, el cual si bien es algo acotado puede ser suficiente para resolver muchas tareas textuales simples.

3.1.2. El origen de los datos de texto utilizados y su estructuración

Bajo el contexto del estudio de los corpus sobre papers enfocados en el área de la biología molecular escritos en inglés, nace la necesidad de poseer datos de prueba que ayuden a discernir entre un texto de buena calidad contra uno de menor calidad especialmente enfocados en esta área.

Por ello se plantea la utilización de herramientas informáticas para realizar "Scraping" de las páginas de editoriales con la finalidad de obtener corpus completos divididos en secciones. En cuanto a los criterios de selección se basaron en 3 categorías principales desde la más relevante a la menos relevante: Área del paper (Biología Molecular), Calidad del paper (Q1-Q4) y editorial (Elsevier, Grupo Omics, Nature, etc.).

Finalmente, la modularización del corpus se reparte en 9 categorías relevantes: Journal, Title, Doi, Abstract, Keywords, Introduction, Materials and Methods, Results and Discussions

[17] Para más detalles sobre el modelo referirse a la sección 3.5 Deep Learning for Natural Lenguage Processing.

y Conclusion, para mayor información sobre el contenido de las 9 categorías relevantes ver anexo A, tabla A3.1.

3.2. Tipos de Errores

Al momento de realizar un análisis exhaustivo de un texto escrito por una persona que podría dominar el lenguaje utilizado ya sea de forma nativa o no, podemos encontrar errores lingüísticos que probablemente ocurren en diversos idiomas y géneros, tal como si se hablase de un patrón de errores basados en una estructura genérica, a modo de ejemplo, en textos científicos dichos errores son intrínsecos en la escritura de papers, en donde las reglas del idioma son más rígidas y poseen menos flexibilidad, ya sea con un conjunto de reglas como por ejemplo la utilización de la coma o el punto como la conjugación de los verbos y el orden en la posición de los artículos, sin embargo, la corrección de problemas relacionados con la dicción y el estilo de escritura puede ser una cuestión de opinión que no solo está arraigado a propiedades lingüísticas, como por ejemplo las abreviaciones.

3.2.1. Errores simples

Los errores de ortografía son de los casos más frecuentes en esta categoría [9], como un ejemplo común tenemos el caso de la utilización errónea de los artículos definidos en donde suelen ser colocados innecesariamente antes de los sustantivos genéricos en el idioma inglés, bajo el mismo idioma tenemos otros casos referentes a la numeración, en donde se deletrea cuando no se debería, también tenemos los casos de las referencias, contracciones, pluralización, verbos irregulares, preposiciones o hasta incluso la repetición de palabras.

Por lo tanto, estos casos representan errores que pueden ser acotados por reglas y estándares definidos según el idioma y contexto en que se escriban, es por ello que también se les define como errores basados en reglas y pertenecen a la problemática de menor complejidad.

3.2.2. Errores complejos

Los errores complejos son más difíciles de encontrar e interpretar, ya que no poseen patrones simples como los descritos en la categoría anterior, esto se ve reflejado por ejemplo en la falta de palabras o el simple hecho de no haber un orden consistente en la redacción de un tramo de palabras, por ejemplo en la gramática del inglés los errores más comunes para esta categoría son los errores de puntuación en una oración [9], ya sea desde la utilización innecesaria de caracteres como los dos puntos hasta la mala ubicación de la misma coma.

También existen otros errores frecuentes que se dan en el desacuerdo numérico, el cual puede tener ocurrencias en cláusulas pasivas o activas, por otro lado, también tenemos errores relacionados con las abreviaturas específicas de trabajo, pero uno de los problemas más complejos de esta categoría es la dicción y el estilo.

3.3. Machine Learning (ML)

El aprendizaje automático o mejor conocido como “machine learning” presenta como objetivo el desarrollo de métodos computacionales que sean capaces de "aprender" con experiencias acumuladas [3, 4, 7, 10, 13, 14]. Usualmente se puede encontrar dos tipos de aprendizaje bajo el contexto de “machine learning”.

El primer caso se conoce como aprendizaje no supervisado, cuya tarea principal consiste en revelar las estructuras intrínsecas que están integradas en las relaciones de datos, por lo cual en este caso el proceso de aprendizaje solo se guía por los datos proporcionados, ya que no se tiene ningún conocimiento previo sobre los datos [8]. Para este tipo de aprendizaje se pueden destacar algunas de las principales tareas tales como: agrupación, detección de valores anómalos, la reducción de dimensionalidad y la asociación.

Para el segundo caso de aprendizaje se tiene el supervisado, cuyo objetivo es deducir conceptos sobre los datos, en donde esta inferencia se realiza utilizando las instancias etiquetadas que se han presentado, comúnmente se les conoce como el conjunto de entrenamiento, bajo este sentido, el proceso de aprendizaje trata de construir una función de mapeo condicionada al conjunto de entrenamiento proporcionado [1].

Por lo tanto, la principal diferencia entre paradigmas de aprendizaje supervisados y no supervisados es la siguiente. En el primero, el alumno explora la información externa del conjunto de entrenamiento, que está disponible en la etapa de entrenamiento, para inducir la hipótesis del clasificador. En una tarea de clasificación, por ejemplo, esta información externa se lleva al proceso de aprendizaje en forma de clases o etiquetas.

3.4. Natural Language Processing (NLP)

Se puede definir como el campo que combina las tecnologías de la ciencia computacional, tales como la inteligencia artificial (“Artificial Intelligence” o AI), el aprendizaje automático (“Machine Learning”), la inferencia estadística y la lingüística aplicada, todo esto con el objetivo de hacer posible la comprensión y el procesamiento asistido por un computador que contiene la información expresada en lenguaje humano para determinadas tareas, como la traducción automática, sistemas de diálogo interactivos, el análisis de opiniones, etc.

Por ende, cuando se habla del NLP en forma global se hace referencia a que se ocupa de la formulación e investigación de mecanismos eficaces computacionalmente para la comunicación entre personas y máquinas por medio de lenguajes naturales, los cuales se ven soportados a través de programas que puedan ser ejecutados o que realicen simulaciones de la comunicación. Dichos modelos aplicados se enfocan no solo a la comprensión del lenguaje, sino que también en aspectos generales cognitivos humanos y a la organización de la memoria[18].

3.5. Deep Learning for Natural Language Processing

El concepto de aprendizaje profundo (“Deep Learning” o DL) para el procesamiento del lenguaje natural (NLP) se puede entender como el reconocimiento de patrones aplicado a palabras, oraciones y párrafos. A modo de ejemplo se puede interpretar como la forma en que trabaja una computadora reconociendo a través de la visión patrones aplicados en cada uno de los pixeles [6].

Luego, para el caso de la etapa de los datos de entrada, como es de suponer en las redes neuronales, los modelos de aprendizaje profundo no suelen procesar texto en bruto, es por esta situación que nacen los tensores numéricos, estos dan origen a la vectorización del texto

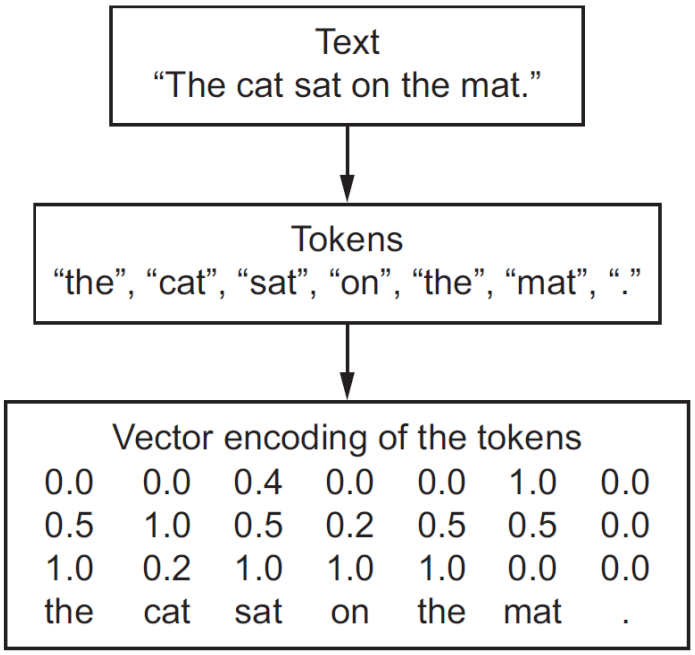
[18] Página: es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales, último acceso 3 de Abril del 2018.

(proceso de transformar texto en tensores numéricos). Existen varias formas de realizar este proceso, entre las cuales tenemos:

- Segmentación del texto en palabras para luego transformar cada palabra en un vector.
- Segmentación del texto en caracteres para luego convertir cada carácter en un vector.
- Extraer n-grams[19] de palabras o caracteres, y transformar cada n-gram en un vector.

Dichos vectores, empaquetados en tensores de secuencia, ahora alimentan a las redes neuronales profundas ("Deep neural networks"), para una mayor claridad ver figura 3.1. Ahora bien, existen múltiples formas de asociar un vector con un token, entre las cuales las más destacables son: "one-hot encoding of tokens" y "token embedding" en donde esta última normalmente se usa exclusivamente para palabras.

Figura 3. 1: "De texto a tokens y luego a vectores"



[Fuente por Deep Learning with Python, Francois Chollet, 2018]

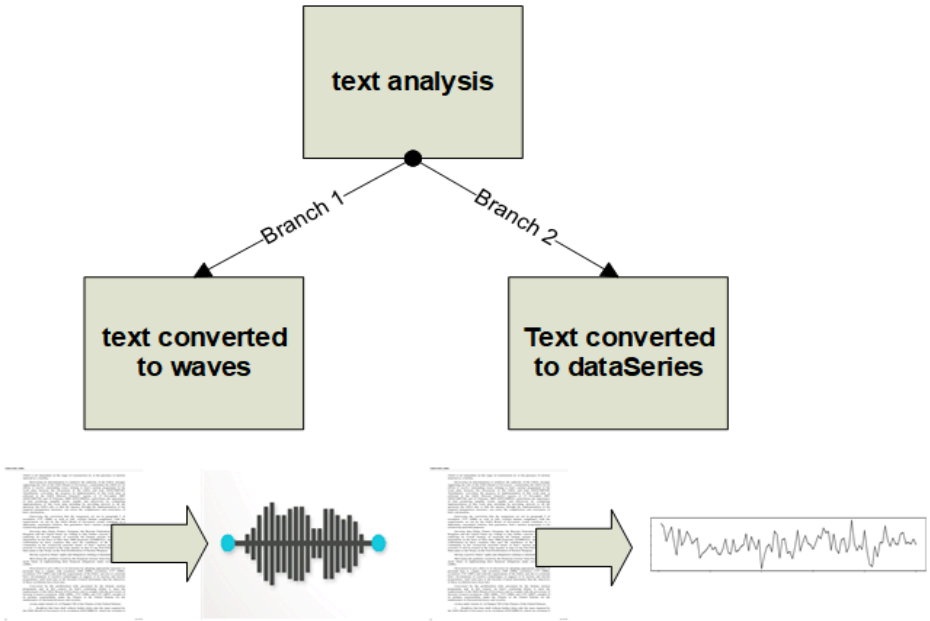
[19] Los N-grams son grupos superpuestos de múltiples palabras o caracteres consecutivos.

4. Propuestas para el análisis de Texto

Gracias a la existencia de diversas metodologías para el análisis de textos cabe mencionar la posible extrapolación de métodos utilizados en otras áreas de la inteligencia artificial tales como métodos de predicción o pronósticos en base a series de tiempo, métodos con regresión lineal, etc.

Sin embargo, también ciertos datos podrían simular fenómenos físicos los cuales podrían ampliar el espectro de análisis de dichos datos, un ejemplo seria la simulación de ondas a través de textos, bajo este planteamiento se desprende el siguiente esquema (ver figura 4.1) con dos métodos propuestos.

Figura 4. 1: "Propuestas para el análisis de texto"



En el esquema anterior (figura 4.1) se presentó dos de las principales ramas propuestas en el presente trabajo, en donde la primera rama (Branch 1) se propone la simulación de ondas en base a datos de texto, los cuales vienen formateados y pre-procesados[20] para reducir el ruido.

Para el caso de la segunda propuesta (Branch 2) en la figura 4.1 se contempla la posibilidad de convertir texto en Series de tiempo, con el objetivo de utilizar mecanismos de predicción o procesamiento en base a valores discretos, de igual forma que en la propuesta anterior los datos son pre-procesados para luego pasar a la fase de "tokenization"[21] y posteriormente a la fase de "Part of Speech Tagging"[22], esto finalizando con el procesos

[20] Para más detalles sobre el modelo referirse a la sección 5.1 Data Pre-procesing.
[21] Para más detalles sobre la fase referirse a la sección 5.1.4 Tokenization by sentence, word or character.
[22] Para más detalles sobre la fase referirse a la sección 5.1.5 Part of Speech Tagging and create Dictionary.

"Numeric Tagging"[23] el cual requiere un vector de datos más su diccionario (Dictionary) para codificar y crear la serie de tiempo.

Por lo tanto, los dos métodos propuestos utilizan la misma fase de pre-procesamiento, el punto de diferenciación comienza al momento de transformar los datos en el etiquetado. Cabe mencionar que la primera propuesta (Branch 1) será especificada como un modelo y solo pretende dar nociones de cómo podría conformarse una onda en base a texto utilizando las características comunes de una onda y como dichas características podrían ser obtenidas a partir de herramientas para el procesamiento del lenguaje natural y minería de datos.

En el caso de la segunda propuesta (Branch 2) presenta mayor detalle en cada una de sus fases, ya que pretende ser un modelo para el descarte de características funcional, enfocado en solucionar la problemática de encontrar patrones de configuración antes de un punto (carácter punto) y lograr discernir cuáles de estas características presentan una mayor relevancia antes de llegar a dicho carácter.

4.1. Branch 1: Text converted to waves

El concepto de convertir texto en ondas radica en utilizar herramientas para el procesamiento del lenguaje natural a favor de la extracción de ciertos componentes de un texto para simular el comportamiento oscilatorio de una onda, esto permitirá generar una secuencia de oscilaciones por cada palabra involucrada, aun así, existen algunos problemas con este modelo ya que la relación entre palabras no se considera.

4.1.1. Características de una onda

Desde un punto de vista matemático la onda más sencilla o fundamental es la onda sinusoidal descrita por la función en la ecuación 4.1.1.1, donde A es la amplitud de una onda (elongación máxima o altura máxima de la cresta de la onda), las unidades de amplitud dependerán del tipo de onda y puede ser constante o variar con el tiempo y/o posición. Por otro lado, tenemos lambda, la cual es la longitud de onda, o mejor dicho es la distancia entre dos crestas o valles seguidos, dicha distancia se mide en unidades de longitud, tales como metro o sus múltiplos o submúltiplos según convenga. Una vez obteniendo la longitud de onda se puede asociar al número de onda angular K (ecuación 4.1.1.2). Sin embargo, también necesitamos el periodo T el cual es el tiempo requerido para que el movimiento de oscilación de la onda describa un ciclo completo, mientras que la frecuencia f es el número de ciclos completos transcurridos en la unidad de tiempo (ecuación 4.1.1.3). Finalmente, la frecuencia angular w representa la frecuencia de radianes por segundo (ecuación 4.1.1.4).

$$f(x,t) = A\sin(\omega t - kx) \qquad (4.1.1.\ 1)$$

$$k = \frac{2\pi}{\lambda} \qquad (4.1.1.\ 2)$$

$$f = \frac{1}{T} \qquad (4.1.1.\ 3)$$

$$\omega = 2\pi f = \frac{2\pi}{T} \qquad (4.1.1.\ 4)$$

[23] Para más detalles sobre la fase referirse a la sección 5.1.6 Numeric Tagging.

4.1.2. Tipos de ondas

Básicamente las ondas se pueden clasificar abarcando diferentes aspectos, ya sea en función del medio en el que se propagan; Ondas mecánicas, Ondas electromagnéticas, Ondas gravitacionales. En función de su dirección; Ondas unidimensionales, Ondas bidimensionales o superficiales, Ondas tridimensionales o esféricas. En función del movimiento de sus partículas; Ondas Longitudinales, Ondas transversales. En función de su periodicidad; Ondas periódicas, Ondas no periódicas.

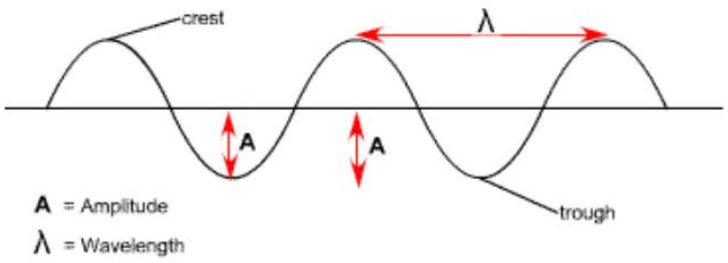
4.1.3. Simulación de ondas en base a texto

Existen variadas tecnologías en el mercado que permiten calcular ciertas propiedades de un set de datos, propiedades tales como la frecuencia de aparición de una misma palabra en un texto determinado, o calcular las probabilidades de asociación con respecto a palabras similares, etc.

Una de las bibliotecas más utilizadas es “pymining” para Python, la cual permite calcular la frecuencia de aparición de un token determinado, para este caso es factible utilizarla para suplir el factor lambda (longitud de onda), mientras que para el caso de la amplitud puede ser cubierta por un factor de peso en base a la rareza de la etiqueta, por dar un ejemplo, se pueden extraer de la gramática general de un texto las etiquetas determinante y sustantivo, en cuyo caso las determinantes tienden a aparecer más veces y tienen menos variabilidad, por ende tienen un factor de peso más acotado.

Por otro lado las palabras etiquetadas como sustantivos son menos frecuentes y suelen ser únicas, para este caso el factor peso tendería a ser más alto, si se combinan estas dos representaciones se podrá observar que al aplicarlos en una secuencia de palabras se podrá simular la oscilación de una onda como se muestra en la figura 4.2, donde la frecuencia de aparición de la palabra es lambda y el factor peso es la amplitud, para el caso del horizonte de tiempo puede denotarse por una frecuencia constante.

Figura 4. 2: "Elementos principales de una Onda"



4.2. Branch 2: Text converted to dataSeries

Para este caso el concepto de convertir texto en dataSeries se basa en la utilización de herramientas para el procesamiento del lenguaje natural enfocados en el reconocimiento de palabras en su forma gramatical general (“Part of Speech”), con el fin de realizar una fase de “Tagging” para codificarlos en valores numéricos discretos, los cuales simularan una

secuencia de valores bajo un horizonte de tiempo definido, en donde dicho horizonte puede estar definido por el cierre de una oración o la aparición de un carácter de puntuación. Para que esta transformación sea de utilidad se propone utilizar dichos datos codificados para el descarte de características con el modelo "Recursive Feautures Elimination" (RFE)[24], el cual permite hacer un ranking con las características recibidas como input.

4.2.1. ¿Qué es una Serie de Tiempo?

Básicamente una serie de tiempo es un registro de una secuencia de valores muestreados generalmente a una frecuencia regular, dichas secuencias podemos separarlas en 2 tipos; Secuencias Estocásticas y Secuencias Determinísticas, en donde el primer tipo denota una secuencia de valores aleatorios que representan la evolución de un sistema en el tiempo, dicha evolución puede ser según una ley de probabilidad.

Por otro lado, tenemos las secuencias determinísticas, que son aquellas donde el azar no está involucrado en la evolución temporal de los estados pasado y futuros, también como sub-categoría tenemos las secuencias determinísticas estacionarias y no estacionarias, en donde las estacionarias se materializan cuando la serie es estable a lo largo del tiempo, cuando la media y varianza son constantes en el tiempo. Esto se refleja gráficamente en que los valores de la serie tienden a oscilar alrededor de una media constante y la variabilidad con respecto a esa media también permanece constante en el tiempo.

Mientras que las no estacionarias son series en las cuales la tendencia y/o variabilidad cambian en el tiempo. Los cambios en la media determinan una tendencia a crecer o decrecer a largo plazo, por lo que la serie no oscila alrededor de un valor constante.

4.2.2. Componentes de una serie de tiempo

Según el análisis clásico de las series temporales, estas se basan en la suposición de que los valores que toma la variable de observación es la consecuencia de tres componentes, cuya actuación conjunta da como resultado los valores medidos, estos componentes son:

- Componente tendencia: El cual se puede definir como un cambio a largo plazo que se produce en la relación al nivel medio, o el cambio a largo plazo de la media, la tendencia se identifica con un movimiento suave de la serie a largo plazo.
- Componente estacional: Muchas series temporales presentan cierta periodicidad o variación de cierto período (semestral, mensual, etc.). Por dar un ejemplo clásico tenemos las ventas al detalle en una localidad X aumentan por los meses de noviembre y diciembre por las festividades navideñas. Estos efectos son fáciles de entender y se pueden medir explícitamente o incluso se pueden eliminar de la serie de datos, a este proceso se le llama desestacionalización de la serie.
- Componente aleatoria: Esta componente no responde a ningún patrón de comportamiento, sino que es el resultado de factores fortuitos o aleatorios que inciden de forma aislada en una serie de tiempo.

[24] Para más detalles sobre la función referirse a la sección 5.2.3 Recursive Feature Elimination (RFE).

4.2.3. Simulación de una serie de tiempo en base a texto

Según el comportamiento y los componentes de una serie de tiempo, esta puede ser representada bajo una secuencia de valores que oscilan dentro de un rango determinado, para generar dicho estado es posible utilizar la extracción de la gramática general de cada palabra, ya que cada forma general de la palabra es una clase la cual puede ser categorizada o codificada en un valor numérico discreto, bajo esta premisa, una vez pre-procesado el texto y disminuido el ruido que pudiese generar palabras que no tienen relación con el texto, se puede pasar a la fase de "Part of Speech Tagging", la cual genera la gramática general que resume la palabra o el carácter involucrado.

Luego de obtener la gramática general de todo el texto se procede a generar un diccionario el cual le entrega un ID a cada clase única encontrada a lo largo del texto. En la siguiente fase "Numeric Tagging" se procede a entregar el valor de la ID del token en el diccionario a cada una de las clases del texto, obteniendo como resultado una secuencia de números discretos que oscilan entre los valores de 1 y la cantidad máxima de tokens únicos identificados en el texto, para una mayor idea sobre lo descrito (ver figura 4.3).

Figura 4. 3: "Proceso General de Etiquetado"

5. Modelo para ranking de características en base a función RFE

Bajo el interés de encontrar una secuencia de patrones antes de un carácter determinado, como por ejemplo un punto, se establece a continuación un modelo que propone la utilización de los datos en la forma simulada “Data Series” (Branch 2). Dicho modelo se compone de 2 fases principales.

La primera de ellas es el Pre-procesamiento de los datos ver figura 5.1 en donde los datos de texto son convertidos a una secuencia numérica discreta para luego generar muestras con sus respectivas etiquetas Verdadero (Punto) o Falso (Otro signo de puntuación).

Mientras que en la segunda fase ver figura 5.2 se plantea el descarte de características mediante la utilización de la función RFE (Recursive Feature Elimination) recibiendo como input la matriz de muestras generadas X (n_samples_data_matrix) tanto para entrenamiento como para pruebas con su correspondiente vector de etiquetado Y (n_samples_target_array).

Cabe mencionar que a modo experimental se han configurado 3 tipos diferentes de estimadores para hacer el descarte de las características, mientras que por otro lado también se realizó el experimento predictivo sin el uso del método RFE con fines comparativos.

Sin embargo, también se varia la cantidad muestras y la distribución con que se presentan las muestras, ya sea de modo intercalado o random, todo esto con el fin de lograr una variación en el ranking y así poder comparar resultados buscando una mayor precisión en cuanto al descarte. Para una vista más detallada de cada proceso ver anexo B, figura B5.1 y figura B5.2 respectivamente.

Figura 5. 1: "Data Pre-processing"

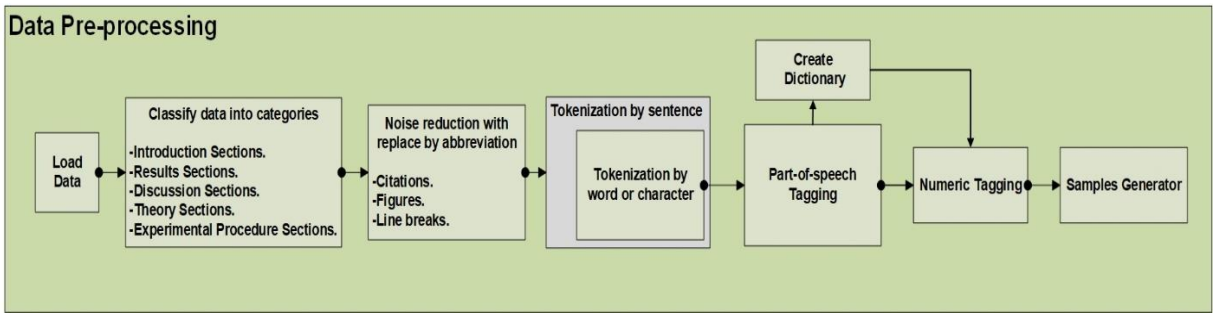
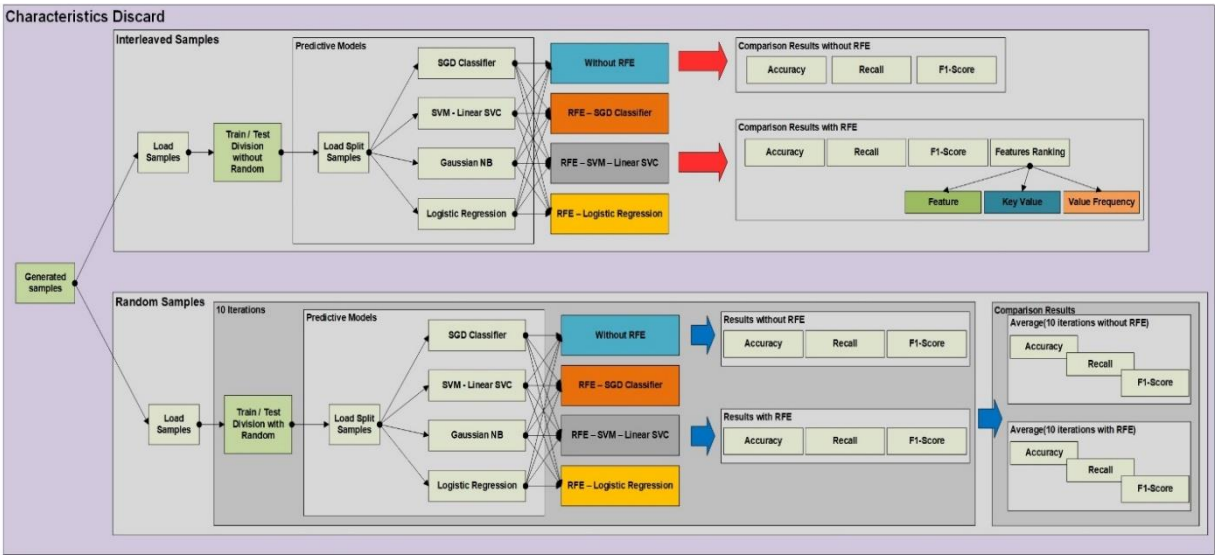


Figura 5. 2: "Characteristics Discard"



5.1. Data Pre-processing

Tomando en cuenta las subdivisiones del diagrama en la figura 5.1, la cual hace referencia al pre-procesado de los datos, se pueden identificar fases clave para una correcta generación de la data a utilizar, dichas fases serán tratadas en esta sección por separado en sub-secciones para poder detallar el proceso que acontece en cada una de ellas y poder definir ciertos supuestos y conocimientos previos de materias involucradas.

5.1.1. Load Data

La carga de los datos de texto se basa en un archivo JSON que contiene más de 2000 papers Q1 anexados uno tras otro[25], en donde dicho contenido puede ser cargado con una biblioteca directa desde Python llamada “pandas”, ya que esta posee la función “read_json()” y facilita la lectura inicial, dicha función es implementada dentro de la función “data_reader(inputText)” contenida en el archivo “give_data.py”, esta función lee un archivo con la dirección guardada en la variable “inputText” y genera como output un dataFrame en una nueva variable.

5.1.2. Classify data into categories

Debido a la temática de la fuente de datos (papers) es conveniente realizar una previa clasificación de los datos centrada en las partes genéricas que posee un paper y con ellas generar una categorización, dichas categorías principalmente son: secciones de introducción, secciones de resultados, secciones de discusión, secciones de teoría y secciones de procedimientos experimentales.

Una vez fragmentadas es más fácil procesar los datos debido al contexto en común que poseen las secciones del mismo tipo, para llevar a cabo este proceso se ha implementado una

[25] Para más detalles sobre el origen de los datos de texto utilizados ir a la sección 3.1 Datos de Prueba.

función llamada “get_from_dataframe(dataText, title)” contenida en el archivo “give_section.py”, esta función recibe como parámetro el dataFrame o conjunto de datos leído como “dataText” y el nombre de la sección que se quiere adquirir como el parámetro “title”, luego retorna un arreglo con todas las secciones de ese tipo.

5.1.3. Noise reduction with replace by abbreviation

Otro de los problemas a considerar es el ruido que puede haber en la carga de los datos, este ruido puede presentarse como restos de palabras sin sentido con el texto, citaciones con formatos en mayúsculas, figuras o incluso los saltos de línea.

Para solventar este problema se propone estandarizar dichos errores en base al reemplazo por abreviaciones que resuman la incoherencia, dicho de otro modo, es tokenizar estos elementos bajo palabras claves tales como -CITE- para las citas, -FIGURE- para las figuras, etc.

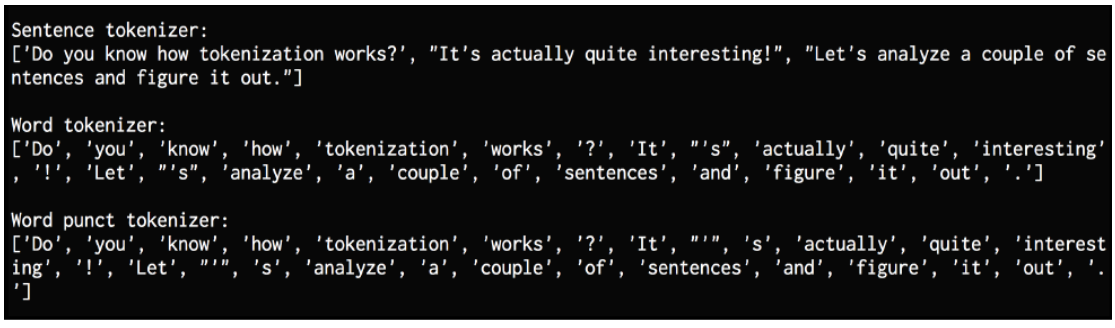
5.1.4. Tokenization by sentence, word or character

En general cuando tratamos el texto, tenemos que dividirlo en partes más pequeñas para su análisis es así como aparece la tokenización de un texto, ya que es el proceso de dividir el texto de entrada en un conjunto de piezas como palabras u oraciones en donde estas piezas se llaman tokens.

Según lo que queramos hacer, podemos definir nuestros propios métodos para dividir el texto en muchos tokens ya sea de la forma por oraciones (sentence tokens), por palabras (word tokens) o incluso por puntuación y palabras (word punct tokens), todo esto utilizando la biblioteca NLTK de Python (para tener una mejor idea del funcioinamiento ver figura 5.3).

Sin embargo, para poder realizar la siguiente fase de Tagging se ha optado por descomponer la data primero por oraciones, y luego por cada oración realizar una tokenización por cada punto, palabra o abreviación[26] que la componga, todo esto con el fin de realizar un proceso más controlable.

Figura 5. 3: "Formas de Tokenizar por NLTK"



[26] Para más detalles sobre tanto referirse a la sección 5.1.3 Noise reduction with replace by abbreviation.

5.1.5. Part of Speech Tagging and create Dictionary

En esta fase se plantea la utilización de la biblioteca “Spacy” de Python para extraer la forma gramatical genérica de la palabra o carácter (en el caso de pertenecer a la categoría puntuación), dicha extracción puede ser realizada a partir de dos tipos de propiedades.

En primera instancia tenemos la propiedad “.tag_” en donde nuestra palabra tokenizada puede ser etiquetada bajo una clasificación que oscila en alrededor de 30 clases distintas, para el caso de la segunda instancia tenemos la propiedad “.pos_” en donde nuestra palabra tokenizada puede ser etiquetada bajo una clasificación que oscila en alrededor de 20 clases distintas, esto quiere decir que posee un etiquetado más abstracto. Para aprovechar estos dos tipos de clasificación se propone crear 2 tipos de taggeo, uno extendido (usando la clasificación “.tag_”) y otro delimitado (utilizando la clasificación “.pos_”).

En conjunto con esta fase se crea un diccionario (Dictionary), la idea consiste en crear una variable de tipo diccionario la cual reúne todos los tipos de clases utilizados dentro del tramo de datos procesados, luego se les asigna una ID que comienza desde el 1 hasta la cantidad máxima de tipos de etiquetas utilizadas, esta herramienta será de utilidad al momento de codificar la secuencia de datos a un valor numérico discreto.

5.1.6. Numeric Tagging

Esta fase propone realizar una codificación numérica de valores discretos en base a un diccionario que posee las ID’s de todas las clases presentes en el texto (generado anteriormente). Para ello se ha implementado una función parser llamada “numeric_parser()” contenida en el archivo “give_numeric_data.py”, esta función recibe como parámetros una variable con los datos etiquetados llamada “data_tagged” y una variable de tipo diccionario con las keys de cada tipo de etiqueta llamada “translation_dict”, luego retorna una nueva variable con los datos codificados según sus clases.

En los experimentos llevados a cabo solo se utilizó el “delimited_classifier”, con el objetivo de simplificar el etiquetado de las muestras, con esto se obtendrán dos tipos de etiquetas para los signos de puntuación, en el caso de las muestras consideradas verdaderas su etiqueta correspondiente será la key del tag que represente explícitamente el punto (ver tabla 5.1, el tag marcado con azul), mientras que para las muestras consideradas falsas su etiqueta será la key del tag que represente todo el resto de los signos de puntuación que no sean puntos (ver tabla 5.1, el tag marcado con rojo), por ende, el diccionario generado sólo se compone de 18 keys, las cuales en este caso poseen la siguiente distribución:

Tabla 5. 1: "Delimited Classifier Dictionary"

Key	Tag
1	PRON
2	VERB
3	DET
4	ADJ
5	NOUN
6	ADP
7	CITE
8	PUNCAT2
9	NUM
10	CCONJ
11	PUNCAT1
12	PROPN
13	ADV
14	PART
15	X
16	SYM
17	INTJ
18	SPACE

5.1.7. Samples Generator

Por último, en la fase de generación de muestras se propone el almacenamiento de muestras para la matriz X (o explícitamente en el código “n_samples_data_matrix”) y con sus respectivas etiquetas en el arreglo Y (o explícitamente en el código “n_samples_target_array”), proceso el cual se desarrolla de forma intercalada, esto quiere decir que se recolecta una muestra con etiqueta verdadero (Punto) y luego una con etiqueta falso (Otro signo de puntuación) sucesivamente.

La idea de realizar una recolección de esta forma se fundamenta bajo el interés de equilibrar la cantidad de muestras para las dos etiquetas, sin embargo, esta forma de recolectar las muestras de la data puede provocar perdida de información, ya que las variables pueden estar relacionadas entre ellas bajo un contexto sucesivo.

Por otro lado, la cantidad de muestras generadas dependerá de la cantidad de secciones a considerar según la base de datos con los papers almacenados, dichas secciones hacen referencia a las partes antes mencionadas de un paper, en donde una sección a modo de ejemplo es la introducción, por ende, se tomarán todas las secciones introducción necesarias para generar la cantidad de muestras pertinentes.

En los experimentos que se dan a conocer en este trabajo se utilizaron 5 variaciones de la cantidad de secciones, estas fueron específicamente de la sección introducción y la sección resultados, las cuales generaron 5 variaciones de cantidad de muestras, estas variantes fueron: para el test 1 se consideraron 10 secciones introducción las cuales generaron una cantidad de 217 muestras, para el test 2 se consideraron 500 secciones introducción la cuales generaron 20559 muestras, para el test 3 se consideraron 1000 secciones introducción en donde se generaron 41511 muestras, mientras que para el test 4 se consideraron todas las secciones introducción de la base de datos generando 88337 muestras, finalmente para el test 5 se consideraron 5000 secciones las cuales se componían de secciones introducción junto con secciones resultados, todo esto genero una cantidad de 157213 muestras.

5.2. Characteristics Discard

En este caso, tomando en cuenta el diagrama de la figura 5.2 “Characteristics Discard” presentado anteriormente, en el cual se hace referencia al descarte de características mediante la utilización del método RFE, se pueden observar 4 secciones importantes en el proceso.

Dicho proceso comienza con la división de las muestras en los conjuntos de entrenamiento y pruebas cuya distribución permitirá generar resultados variados según el porcentaje de asignación a cada uno.

Luego de la separación de las muestras estas son utilizadas por los modelos para clasificación de características, en donde son entrenados para luego generar los resultados predichos, sin embargo, en esta parte y en este caso se plantean 4 caminos distintos para generar los resultados, en donde dichos resultados pueden ser generados sin haber utilizado el descarte de características o bien utilizar el RFE con una variedad de estimadores (en este caso 3 estimadores distintos).

Finalmente se obtienen los resultados basándose en las siguientes métricas: precisión (Accuracy), recuperación (Recall) y Puntaje F1 (F1-Score), en donde estas métricas utilizadas

permitirán saber el nivel de precisión de cada modelo ya sea con RFE o sin él, adicionalmente se podrán observar las características mejor correlacionadas entre ellas para el caso de utilizar RFE, ya que según la cantidad de características a considerar en el ranking permitirá realizar un análisis a la secuencia que presentan estas características antes de llegar a una etiqueta en específico.

5.2.1. División de muestras en conjuntos de entrenamiento y pruebas

En la primera sección importante del modelo propuesto anteriormente se debe crear una subdivisión de los datos, con el objetivo de crear un set de entrenamiento (data training) y otro set para pruebas (data testing). En donde la proporción por default en la función está configurada para una relación de 70–30 respectivamente, dicha función se llama "train_test_split()" y es propia de la biblioteca de Python "sklearn.model_selection".

Esta función recibe como parámetros una matriz X con las muestras generadas y pre-procesadas a valores enteros discretos llamada "n_samples_data_matrix" y su respectivo arreglo Y con las etiquetas correspondientes a cada muestra llamada "n_samples_target_array", adicionalmente son requeridas las variables "test_size" la cual entrega el porcentaje dedicado a las muestras para pruebas y la variable "random_state" la cual si es configurada igual a 1 ordenara las muestras de forma aleatoria (configuración por defecto), como retorno de esta función se obtienen 4 nuevos objetos, los cuales son 2 matrices de muestras (n_samples_train_data_matrix, n_samples_test_data_matrix) y sus arreglos de etiquetado correspondientes (n_samples_train_target_array, n_samples_test_target_array), todo esto en las proporciones definidas anteriormente para entrenamiento y pruebas con un orden definido como random.

Para el caso de los experimentos que se presentan en este trabajo la configuración de esta función tuvo variaciones en cuanto a la utilización del random, ya que por cada prueba se hicieron 2 configuraciones distintas, en donde la primera utilizaba las muestras de forma intercalada (en cuanto a sus etiquetas Verdadera y Falsa) y por ende el random fue desactivado usando solamente la distribución que trae por defecto 70-30 (a continuación ver tabla 5.2), mientras que para la segunda configuración se necesitaba de 10 iteraciones con distinto orden para las muestras por cada iteración, por ende se utilizó el random en cada iteración con el objetivo de obtener el promedio de cada métrica utilizada en el conjunto de esas 10 iteraciones.

Tabla 5. 2: "Distribution of tests with interleaved samples"

Nº Test	Train True Samples	Train False Samples	Test True Samples	Test False Samples	Samples distribution	Nº Features per Sample	Train/Test distribution
Test 1	76	76	33	32	interleaved	10	70 / 30
	152 Train Samples		65 Test Samples				
	217 Total Samples						
Test 2	7.196	7.195	3.084	3.084	interleaved	10	70 / 30
	14.391 Train Samples		6.168 Test Samples				
	20.559 Total Samples						
Test 3	14.529	14.529	6.227	6.226	interleaved	10	70 / 30
	29.058 Train Samples		12.453 Test Samples				
	41.511 Total Samples						
Test 4	30.918	30.918	13.251	13.250	interleaved	10	70 / 30
	61.836 Train Samples		26.501 Test Samples				
	88.337 Total Samples						
Test 5	55.025	55.024	23.582	23.582	interleaved	10	70 / 30
	110.049 Train Samples		47.164 Test Samples				
	157.213 Total Samples						

5.2.2. Modelos para la clasificación y predicción de características

Bajo el margen de la segunda sección importante del modelo propuesto se debe considerar la utilización de algoritmos de clasificación, los cuales puedan ser utilizados para realizar la tarea de predicción de características.

Con la posibilidad de elegir una gran variedad de algoritmos que pudiesen cumplir con esta tarea se eligieron 4 diferentes algoritmos para realizar los experimentos, en donde la elección de estos fue respaldada por la guía que otorga la comunidad de desarrolladores de Python, más específicamente en "scikit-learn" (a modo de referencia ver figura 5.4 "Algorithm cheat-sheet").

Esta guía propone una división de los diferentes modelos separándolos según su área de desempeño y las características de los datos a utilizar, por ende, según las características que se buscan en el modelo propuesto anteriormente (ver figura 5.2 "Characteristics Discard") se debe tratar el área de la clasificación, adicionalmente según la cantidad de muestras propuestas anteriormente (ver tabla 5.2 "Distribution of tests with interleaved samples") se debe deducir el camino adecuado para la selección de los modelos.

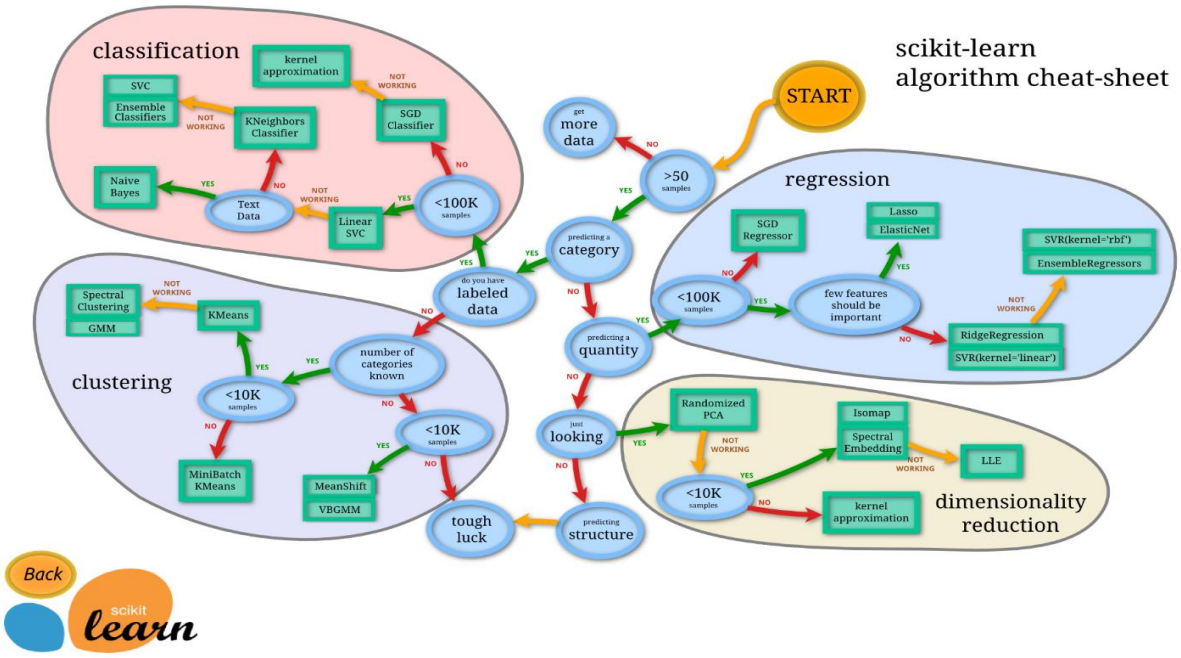
A consecuencia de lo anterior y según la guía se consensa que los modelos funcionales para dicho proceso serian SGD Classifier para el caso en que la cantidad de muestras fuese superior a los 100 k, por otro lado, si la cantidad es menor se propone la utilización de SVM – Linear SVC, en el caso de que este último no funcionase se propone la utilización de Gaussian Naive Bayes.

Finalmente, para los experimentos tratados en este trabajo se utilizan los 3 modelos antes mencionados más el modelo Logistic Regression, este último es un algoritmo de clasificación de aprendizaje automático que se utiliza para predecir la probabilidad de una variable dependiente categórica.

Estos modelos fueron sometidos a entrenamientos con distintas cantidades de muestras, específicamente con una variación de 5 pruebas distintas, y también se hicieron unas variaciones en el orden que recibían las muestras, específicamente en 2 tipos de orden, todo esto con la finalidad de probar sus desempeños bajo distintas condiciones.

Por otro lado, para aumentar la diversidad de resultados fueron sometidos a una reducción de características, debido a que en la fase de clasificación recibían normalmente 10 características por muestra, para los experimentos con RFE la cantidad de características por muestra es reducida a 5 características, las cuales son las mejor correlacionadas entre ellas.

Figura 5. 4: "Algorithm cheat-sheet"



[Fuente por Scikit-learn, www.scikit-learn.org, última visita 25 de noviembre del 2018]

5.2.2.1. Stochastic Gradient Descent (SGD Classifier)

El Stochastic Gradient Descent (SGD) es un enfoque simple pero muy eficaz para el aprendizaje discriminativo para los clasificadores lineales bajo funciones de pérdida convexas como las máquinas de vectores de soporte (específicamente lineales) y la regresión logística como tal.

Sin embargo, su principal característica es su especial posición en el contexto del aprendizaje a gran escala, en donde se ha aplicado con éxito a problemas de aprendizaje de máquina dispersos que a menudo se encuentran en la clasificación de textos y el procesamiento de lenguaje natural, los clasificadores en este módulo se escalan fácilmente a problemas con más de 100 k muestras de entrenamiento.

Por otro lado, hay que asegurarse de mezclar las muestras de entrenamiento antes de ajustar el modelo después de cada iteración, en donde por cada iteración se cargan dos objetos, la matriz X de tamaño (n_samples, n_features) la cual contiene las muestras de entrenamiento, y un arreglo Y de tamaño [n_samples] el cual contiene los valores objetivo (etiquetas True o False correspondientes a cada muestra) en el caso de las muestras de entrenamiento.

Para los experimentos desarrollados en este trabajo, los parámetros básicos que se utilizaron para esta función fueron el parámetro “loss” configurado como “hinge” que entrega un SVM lineal, el parámetro “penalty” o también conocido como termino de regularización configurado como “l2” ya que es el regularizador estándar para modelos SVM lineales y una cantidad máxima de iteraciones “max_iter” configurada como 100.

5.2.2.2. Support Vector Machine (SVM – LinearSVC)

Support Vector Machines son un conjunto de métodos de aprendizaje supervisado, utilizados con la finalidad de realizar clasificación, regresión y detección de valores anormales. Algunas de las ventajas más destacables son la eficacia en espacios de muchas dimensiones, también en casos donde el número de dimensiones es mayor que el número de muestras, adicionalmente utiliza un subconjunto de puntos de entrenamiento en la función de decisión (vectores de soporte) y finalmente tiene una versatilidad para especificar diferentes funciones del Kernel para la función de decisión.

Por otro lado, LinearSVC es otra implementación de la Clasificación de vectores de soporte para el caso de un kernel lineal, es similar a SVC pero con el parámetro “kernel” igualado a “Linear”, por lo que tiene más flexibilidad en la elección de penalizaciones y funciones de pérdida y debe escalar mejor a un gran número de muestras, también admite entradas tanto densas como dispersas.

5.2.2.3. Gaussian Naive Bayes (GNB)

Los métodos Naive Bayes son un conjunto de algoritmos de aprendizaje supervisado basados en la aplicación del teorema de Bayes con el supuesto "naive" de independencia condicional entre cada par de características dado el valor de la variable de clase, sin embargo, a pesar de sus supuestos aparentemente simplificados, los clasificadores Naive Bayes suelen funcionar muy bien en muchas situaciones reales, a modo de ejemplo está la situación de la reconocida clasificación de documentos y el filtrado de correo no deseado.

Por otra parte, requiere una pequeña cantidad de muestras de entrenamiento para estimar los parámetros necesarios, otras de las ventajas para los Naive Bayes learners y clasificadores es que pueden llegar a ser extremadamente rápidos en comparación con métodos más sofisticados. El desacoplamiento de las distribuciones de características condicionales de clase significa que cada distribución puede estimarse independientemente como una distribución unidimensional, esto significa que ayuda a aliviar los problemas derivados de la dimensionalidad.

5.2.2.4. Logistic Regression

La regresión logística es un tipo de análisis de regresión utilizado para predecir el resultado de una variable categórica, la cual puede adoptar un número limitado de categorías, esta variable a la vez se encuentra en función de otras variables independientes o predictorias. Por ende, es el modelo perfecto para el formato de las muestras que se utilizan en el experimento, además la regresión logística se utiliza para describir datos y para explicar la relación entre una variable binaria dependiente y una o más variables independientes, siendo estas de forma nominal, ordinal, de intervalo o de proporción.

Finalmente, mezclando la problemática respecto a la predicción de un punto y este modelo, se llega al consenso de que es útil para modelar la probabilidad de un evento ocurrido como función de otros factores, en donde dicho evento puede ser clasificado de forma binaria, sin embargo, se debe destaca que lo más importante no es predecir el evento como tal, sino la probabilidad de que este evento resulte como un punto (éxito) u otro signo de puntuación (Fracaso), y la medida en que esa probabilidad depende de las variables predictoras.

5.2.3. Recursive Feature Elimination (RFE)

Para la tercera sección importante se procede a utilizar el método RFE, cuyo objetivo es realizar un descarte de características (de las menos relacionadas) entre aquellas que conforman una muestra, todo esto dado un estimador externo que asigna ponderaciones a las características, por lo tanto el fuerte es la eliminación recursiva de características o en otras palabras su finalidad es seleccionar las características considerando recursivamente conjuntos de características cada vez más pequeños.

Para lograr esto, primero el estimador se entrena en el conjunto inicial de características y la importancia de cada característica se obtiene a través de un atributo "coef" o mediante un atributo "feature_importances_". Por consiguiente, las características menos importantes se eliminan del conjunto actual de características. Ese procedimiento se repite recursivamente en el conjunto podado hasta que finalmente se alcanza el número deseado de características para seleccionar, a modo de ejemplo ver figura 5.5 "Diagrama descarte de características".

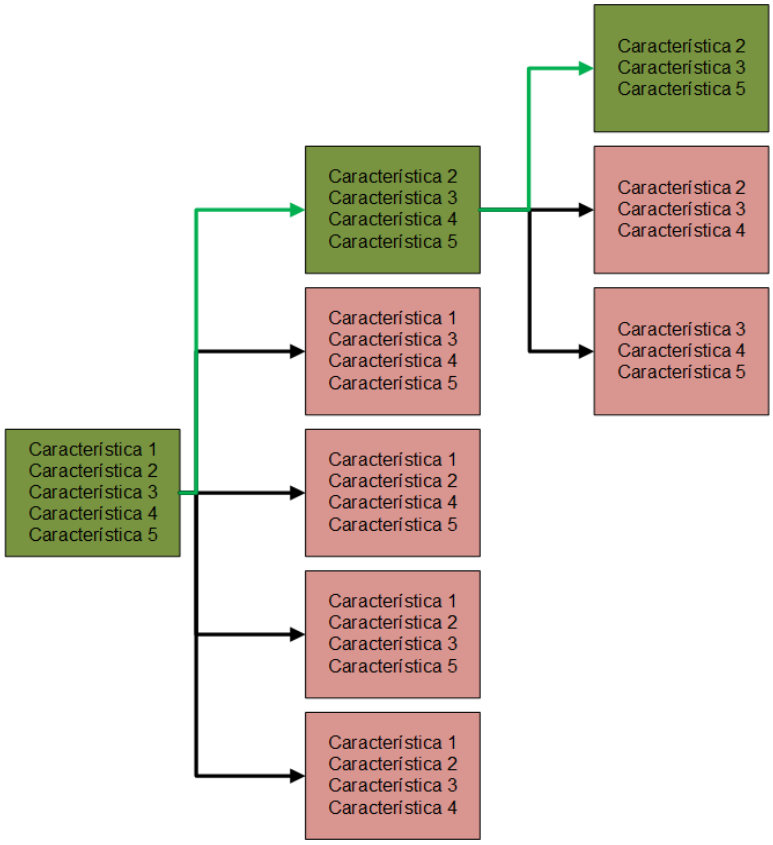
En los experimentos realizados la participación del RFE es fundamental para la variación de resultados, ya que nos permite hacer la reducción de características en base a diferentes tipos de estimadores, estos estimadores al igual que los clasificadores antes mencionados son modelos matemáticos que se ajustan a las muestras de entrenamiento para poder presentar una estimación sobre futuras características. Los estimadores utilizados en este caso varían en 3 modelos: SGD Classifier, SVM-Linear SVC y Logistic Regression.

Al utilizar estos estimadores se puede realizar una comparación de su desempeño ya sea en cuanto a su precisión, recuperación o el F1-Score (promedio entre las dos primeras), el objetivo principal es ver qué modelo es el mejor estimador para utilizar el RFE frente a los 4 modelos de clasificación antes mencionados (vistos en la sección 5.2.2), cabe mencionar que también se realizaron pruebas sin la utilización del método RFE sólo para aumentar la cantidad de variaciones posibles.

En cuanto a la configuración por defecto que se estableció para el RFE se propone la generación de un ranking para las 5 mejores características de un total de 10 características por cada muestra, por ende, el resultado de este no sólo genera el ranking, sino que se aprovecha de recaudar la información con respecto al índice de las 5 mejores columnas, los valores que más se repitieron por cada columna seleccionada (como mejor correlacionada) y adicionalmente la cantidad de veces que apareció el valor dentro del margen de la cantidad de muestras totales, todo esto para poder establecer un análisis con respecto a los valores entregados en conjunto con las métricas antes descritas.

Finalmente, para cerrar el proceso de la tercera sección importante con respecto al RFE las muestras son readaptadas, esto quiere decir que tanto el conjunto de muestras de entrenamiento como de pruebas X son reducidas a la cantidad de columnas configuradas en el ranking RFE, conservando sólo las columnas de los índices rescatados anteriormente, dichos índices son respectivos a las características mejor correlacionadas, a modo de ejemplo si se contaba con una matriz de 2400 muestras (filas) y cada muestra con 10 características (columnas) se vería como X[2400][10], una vez realizado el ranking y pre-procesada la matriz X con las respectivas columnas quedaría de la siguiente forma X[2400][5], en donde esas 5 columnas serían las mejores características.

Figura 5. 5: "Diagrama descarte de características"



5.2.3.1. Detalles para la configuración del RFE

Debido a la variación de pruebas con distintos estimadores de la función RFE hay que tener en cuenta los aspectos claves al momento de configurar la función, para ello se establecen a continuación una vista rápida del código que engloba toda la función descrita anteriormente, código el cual se encuentra en su versión original disponible y documentado a la fecha en la comunidad de Python www.scikit-learn.org, más específicamente en la biblioteca "sklearn" bajo la subsección "feature_selection".

En la función recolectada tenemos 2 parámetros importantes al momento de configurarla, en donde el parámetro "estimator" es el modelo a utilizar en dicha función y por ende debe ser instanciado en un objeto para que esta pueda recibirlo como input, a modo de ejemplo en el código que se muestra a continuación se carga el modelo SVR desde la misma biblioteca "sklearn.svm" el cual es configurado con un parámetro "kernel" igual a "linear".

Para el caso del segundo parámetro de la función RFE tenemos el parámetro "ranking_features" el cual nos indica la cantidad de características a considerar como las mejores siendo en este caso las 5 características mejor correlacionadas en un universo de características mayor o igual a este número.

Finalmente, se obtiene un objeto "selector" el cual contiene la instancia del RFE configurada bajo los parámetros descritos anteriormente, en donde debe ser entrenado con el

conjunto de muestras de entrenamiento de la matriz con dimensiones X[cantidad_muestras][cantidad_caracteristícas] y su respectivo arreglo de etiquetados de dimensión Y[cantidad_muestras] para poder realizar el descarte y obtener el ranking generado mediante el método "selector.ranking_", esta forma modular de tratar el método RFE permite realizar fácilmente una variación en los estimadores que se pueden instanciar y así obtener diferentes resultados para su comparación, a continuación se muestra un extracto del código ejemplo:

```
from sklearn.feature_selection import RFE

from sklearn.svm import SVR

estimator = SVR(kernel="linear")

ranking_features = 5

selector = RFE(estimator, ranking_features)

selector.fit(X,Y)

selector.ranking_
```

5.2.3.2. Estimadores utilizados

Los modelos que se utilizan para las distintas instancias del estimador en los experimentos tratados en este trabajo fueron elegidos en base a los ya vistos anteriormente en función para la predicción, para más detalles ver la guía que ofrece la comunidad de Python (ver figura 5.4 "Algorithm cheat-sheet"), esta elección fue debido a su área de desempeño, la cual ayuda al descarte de características con respecto al tipo de muestras que se están tratando, muestras las cuales se basan en texto tokenizado según su gramática genérica.

Por ende, los modelos que se rescatan y son de utilidad para ser usados como estimador son el Stochastic Gradient Descent (SGD) y el Support Vector Machine (SVM-LinearSVC), cabe mencionar que el modelo Gaussian Naive Bayes no es utilizado debido a que los objetos retornados por este modelo, ya que no son compatibles para realizar un descarte de características.

Por otra parte, también se utiliza el modelo Logistic Regression, ya que es el apropiado para cuando la variable dependiente es binaria, esto debido a que se da el caso de que las muestras dependen de 2 tipos de etiquetado (Punto u otro signo de puntuación), además entrega un análisis predictivo y generalmente ayuda a describir datos para explicar la relación entre una variable binaria dependiente y una o más variables independientes nominales, ordinales, de intervalo o de proporción.

5.2.4. Métricas utilizadas

Para poder medir el desempeño de las distintas configuraciones en cada modelo clasificador se establecieron 4 métricas predominantes, dichas métricas fueron elegidas debido a la información que proporcionan para poder realizar comparaciones entre los diversos resultados, estas métricas son: Exactitud, Precisión, Recuperación y el Puntuación F1.

La exactitud es una métrica para evaluar modelos de clasificación, ya que es la fracción de predicciones que el modelo realizó correctamente, cabe mencionar que no suele ser un buen indicador cuando la cantidad de muestras están desequilibradas, por ende, bajo su definición será igual al total de predicciones correctas dividido por el total de predicciones, ver ecuación 5.2.4.1, en donde VP es Verdaderos Positivos, VN es Verdaderos Negativos, FP es Falsos Positivos y FN es Falsos Negativos.

Por otro lado, la precisión nos da a conocer la proporción de identificaciones positivas que fueron correctas, esto quiere decir que es igual a la cantidad de verdaderos positivos dividido por la suma de los verdaderos positivos más los falsos positivos, ver ecuación 5.2.4.2.

Mientras que la recuperación o exhaustividad nos entrega la proporción de positivos reales que se identificaron correctamente, esto se define como la cantidad de verdaderos positivos dividido por la suma de los verdaderos positivos más los falsos negativos, para este caso ver la ecuación 5.2.4.3.

Para evaluar completamente la efectividad de un modelo, se deben examinar la precisión y la recuperación por igual, pero con frecuencia hay tensión entre precisión y recuperación, esto quiere decir que, al mejorar la precisión, generalmente se reduce la recuperación y viceversa.

Para ello en el análisis estadístico de la clasificación binaria, existe la puntuación F1, la cual es una medida de la precisión de una prueba en donde se considera tanto la precisión como la recuperación para calcular la puntuación. Cabe mencionar que el puntaje de F1 es el promedio armónico de la precisión y la recuperación, donde un puntaje de F1 alcanza su mejor valor en 1 (precisión y recuperación perfectas) y el peor en 0.

$$Exactitud = \frac{VP+VN}{VP+VN+FP+FN} \qquad (5.2.4.1)$$

$$Precision = \frac{VP}{VP+FP} \qquad (5.2.4.2)$$

$$Exhaustividad = \frac{VP}{VP+FN} \qquad (5.2.4.3)$$

5.2.5. Resultados

En esta sección se tratarán los resultados obtenidos de las distintas configuraciones para los experimentos de clasificación y utilización del método RFE, se debe considerar que esta serie de configuraciones se basan en variaciones de 4 factores clave. El primero de ellos es la forma en que se encuentran ordenadas las muestras según su etiqueta (verdadero o falso), esto quiere decir que se obtuvieron resultados tanto de intercalación de muestras positivas con negativas como también resultados con distribuciones random, cabe mencionar que los fuertes de los resultados se enfocan en las muestras intercaladas.

El segundo factor es la variación de la cantidad de muestras utilizadas, las cuales varían en 5 cantidades distintas (ver sección 5.2.1, en la tabla 5.2), mientras que el tercer factor variante consiste en la utilización de 4 modelos de clasificación distintos (ver sección 5.2.2 para más detalles). Por último, el cuarto factor hace referencia a las 4 instancias distintas del RFE (estimadores diferentes y el caso en que no se utiliza RFE), los cuales se dieron a conocer en la sección 5.2.3. Sin embargo, también se debe mencionar aquellas variables que se mantuvieron fijas en todos los experimentos, dichas variables fueron la cantidad de características por muestra (10), la división Train-Test (70-30) y la cantidad de características a considerar en el ranking (5).

A continuación, se ahondará en el rendimiento individual de cada modelo de clasificación con respecto al porcentaje de exactitud frente a la cantidad de muestras utilizadas. En el gráfico de la figura 5.6 podemos apreciar los resultados de exactitud del modelo SGDClassifier sin usar RFE (azul) frente a las variaciones de la cantidad de muestras (eje X). Para interpretar mejor los valores descritos en la gráfica puede ver la tabla 5.3, en donde podemos destacar que el mayor porcentaje de exactitud (55%) se obtuvo con 41.511 muestras y sin utilizar el RFE (marcado en rojo), mientras que el menor porcentaje (46%) fue con 217 muestras y sin utilizar el RFE (marcado en blanco).

Figura 5. 6: "Exactitud modelo SGDClassifier"

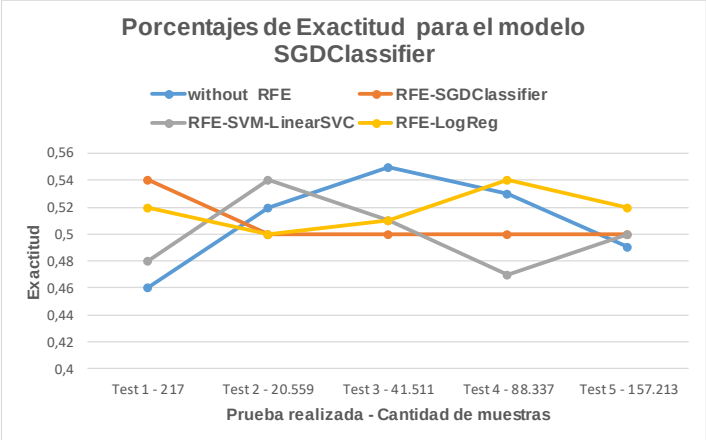


Tabla 5. 3: "Exactitud modelo SGDClassifier"

Porcentajes de Exactitud para el modelo SGDClassifier				
N° Test - Samples	without RFE	RFE-SGDClassifier	RFE-SVM-LinearSVC	RFE-LogReg
Test 1 - 217	0,46	0,54	0,48	0,52
Test 2 - 20.559	0,52	0,5	0,54	0,5
Test 3 - 41.511	0,55	0,5	0,51	0,51
Test 4 - 88.337	0,53	0,5	0,47	0,54
Test 5 - 157.213	0,49	0,5	0,5	0,52

Ahora bien, para saber sobre las métricas de precisión, recuperación y puntuación F1 asociadas a todos los modelos en cada uno de los test, puede acudir al conjunto de anexos D, en donde la figura D5.1 contiene los resultados de dichas métricas para el test 1, en la figura D5.3 encontrara los resultados para el test 2, luego en la figura D5.5 para las del test 3, en la figura D5.7 los resultados del test 4 y en la figura D5.9 para los del test 5, allí podrá analizar el comportamiento de dichas métricas en formato de gráficos de barras.

Para el caso del modelo clasificador SVM-LinearSVC tenemos el gráfico de la figura 5.7 y sus respectivos valores en la tabla 5.4, en donde podemos comparar su exactitud bajo los distintos escenarios que entregan las cantidades de muestras utilizadas. Según se puede apreciar este modelo posee su mejor porcentaje de exactitud (58%) cuándo es utilizado con RFE-SGDClassifier y con una cantidad de muestras igual a 41.511, sin embargo, su porcentaje más bajo (45%) se dio con el RFE-Logistic Regression y una cantidad de muestras igual a 157.213.

Figura 5. 7: "Exactitud modelo SVM-LinearSVC"

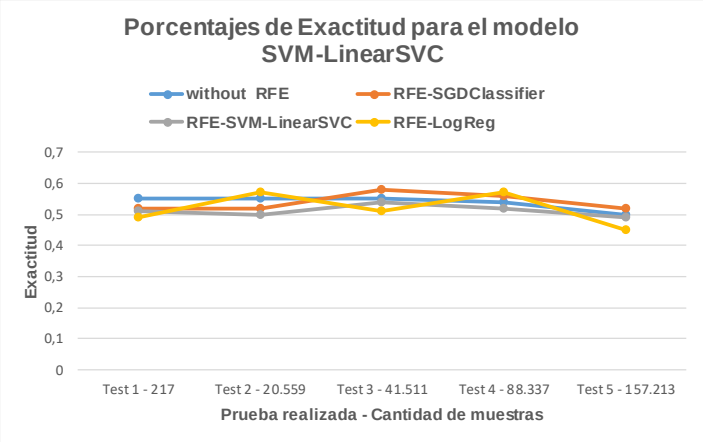


Tabla 5. 4: "Exactitud modelo SVM-LinearSVC"

Porcentajes de Exactitud para el modelo SVM-LinearSVC				
Nº Test - Samples	without RFE	RFE-SGDClassifier	RFE-SVM-LinearSVC	RFE-LogReg
Test 1 - 217	0,55	0,52	0,51	0,49
Test 2 - 20.559	0,55	0,52	0,5	0,57
Test 3 - 41.511	0,55	0,58	0,54	0,51
Test 4 - 88.337	0,54	0,56	0,52	0,57
Test 5 - 157.213	0,5	0,52	0,49	0,45

En cuanto al modelo para clasificación Gaussiana Naive Bayes podemos corroborar que posee uno de los mejores desempeños en comparación a los otros modelos, esto debido a que presenta 3 instancias de mejor desempeño en cuanto a exactitud (63%). Según el grafico de la figura 5.8 y respaldando los datos en la tabla 5.5, estos notorios resultados se dieron 2 veces sin utilizar RFE tanto con 217 como con 88.337 muestras, y 1 vez con RFE-Logistic Regression utilizando 157.213 muestras, cabe mencionar que fueron los mejores resultados para exactitud a nivel global en comparación a los otros modelos de clasificación y sus distintas configuraciones.

Por otro lado, este modelo de clasificación obtuvo 2 resultados con el peor porcentaje de exactitud (51%), siendo estos la configuración con RFE-SVM-LinearSVC y RFE-Logistic Regression dados en el mismo test 1 con 217 muestras.

Figura 5. 8: "Exactitud modelo Gaussian NB"

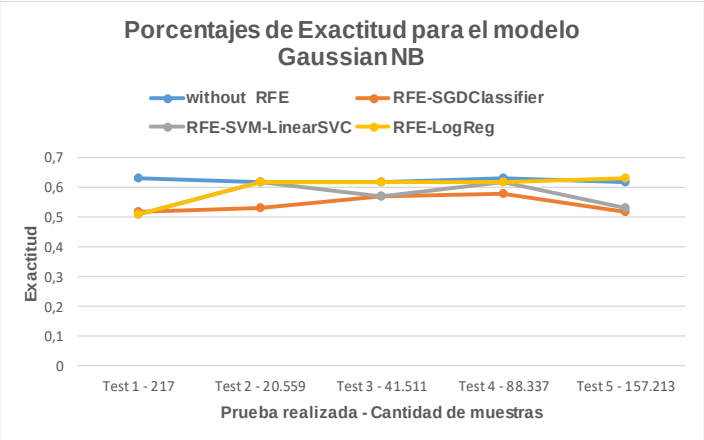


Tabla 5. 5: "Exactitud modelo Gaussian NB"

Porcentajes de Exactitud para el modelo Gaussian NB				
N° Test - Samples	without RFE	RFE-SGDClassifier	RFE-SVM-LinearSVC	RFE-LogReg
Test 1 - 217	0,63	0,52	0,51	0,51
Test 2 - 20.559	0,62	0,53	0,62	0,62
Test 3 - 41.511	0,62	0,57	0,57	0,62
Test 4 - 88.337	0,63	0,58	0,62	0,62
Test 5 - 157.213	0,62	0,52	0,53	0,63

Mientras que en el modelo para clasificación Logistic Regression, según el gráfico de la figura 5.9 y sus datos respaldados en la tabla 5.6, obtuvo como mejor porcentaje de exactitud un 60% con la configuración RFE-SGDClassifier y utilizando 88.337 muestras. Sin embargo, presento su peor desempeño (45%) utilizando la misma configuración RFE-SGDClassifier pero con menos cantidad de muestras (Test 1 - 217 muestras).

Figura 5. 9: "Exactitud modelo Logistic Regression"

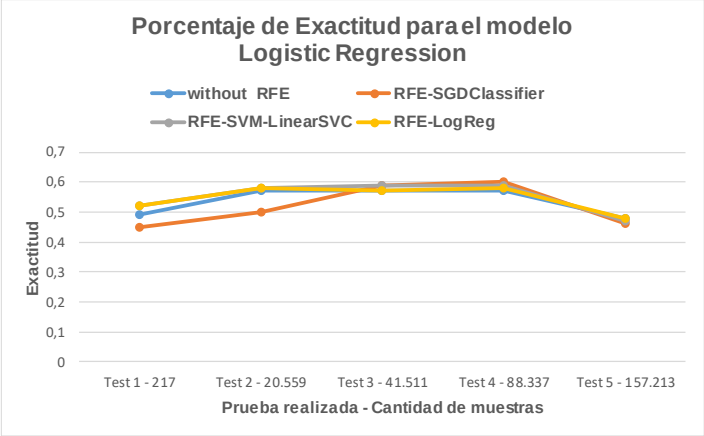


Tabla 5. 6: "Exactitud modelo Logistic Regression"

Porcentajes de Exactitud para el modelo Logistic Regression				
N° Test - Samples	without RFE	RFE-SGDClassifier	RFE-SVM-LinearSVC	RFE-LogReg
Test 1 - 217	0,49	0,45	0,52	0,52
Test 2 - 20.559	0,57	0,5	0,58	0,58
Test 3 - 41.511	0,57	0,59	0,59	0,57
Test 4 - 88.337	0,57	0,6	0,59	0,58
Test 5 - 157.213	0,48	0,46	0,47	0,48

A continuación, se darán a conocer los resultados más destacables en cuanto a la exactitud de cada modelo de clasificación frente a las pruebas realizadas, todo esto con el fin de comparar cual tuvo mejor rendimiento en función de la cantidad de muestras utilizadas. Para poder visualizar los correspondientes gráficos y sus tablas asociadas referirse al anexo C.

Para el test 1 con 217 muestras el mejor porcentaje de exactitud fue del 63%, este resultado generado por el modelo GaussianNB sin utilizar RFE, mientras que su contraparte fue del 45%, generado por el modelo Logistic Regression utilizando RFE-SGDClassifier, ver anexos C, figura C5.1 y la tabla C5.1.

En el caso del test 2 con 20.559 muestras el mejor porcentaje de exactitud fue del 62%, en donde dicho resultado se repitió 3 veces para el modelo GaussianNB sin RFE y utilizando RFE con estimadores SVMLinearSVC y Logistic Regression. Por otra parte, el peor resultado fue de un 50% el cual se repitió 4 veces, esto utilizando los modelos para clasificación SGDClassifier con RFE-SGDClassifier y RFE-LogReg, SVM-LinearSVC con RFE-SVM-LinearSVC y Logistic Regression con RFE-SGDClassifier, ver anexos C en la figura C5.2 y su correspondiente tabla C5.2.

Por otro lado, el test 3 con 41.511 muestras tuvo el mejor porcentaje de exactitud con un porcentaje del 62% en 2 ocasiones, por el modelo GaussianNB sin RFE y utilizando RFE-LogReg, mientras que el peor resultado fue de un 50% del modelo SGDClassifier con RFE-SGDClassifier, ver anexos C, en la figura C5.3 y su tabla C5.3.

Para el test 4 con 88.337 muestras el mejor resultado fue de un 63% generado por el modelo de clasificación GuassianNB sin utilizar RFE, mientras que su contraparte fue de un 47% generado por el modelo SGDClassifier utilizando RFE-SVM-LinearSVC, ver anexos C, en la figura C5.4 y la tabla C5.4.

En el último caso de los tests presentados, tenemos el test 5 con 157.213 muestras, en donde se obtuvo que el mejor resultado para el rendimiento de la exactitud de un modelo fue del 63%, esto nuevamente por parte del modelo para clasificación GaussianNB, pero esta vez utilizando RFE-LogReg, mientras que su contraparte fue del 45% generado por el modelo SVM-LinearSVC con RFE-LogReg, ver anexos C, en la figura C5.5 y su respectiva tabla C5.5.

A modo de síntesis, se puede determinar que el mejor resultado global para la exactitud de un modelo de clasificación que ha sido configurado bajo todas las combinaciones descritas anteriormente es el modelo GaussianNB, el cual presento 3 variaciones de instancias con el mismo resultado (un 63%), dichas variaciones fueron; sin RFE utilizando 217, sin RFE utilizando 88.337 muestras y con RFE-LogReg utilizando 157.213 muestras. Mientras que sus contrapartes globales fueron; el modelo de clasificación SVM-LinearSVC utilizando RFE-LogReg y una cantidad de 157.213 muestras, el cual arrojo un 45% de exactitud y el modelo de clasificación Logistic Regression utilizando RFE-SGDClassifier con 217 muestras, para más detalles ver anexos C en la tabla C5.6.

Finalmente, en la tabla 5.7 se presentan los resultados obtenidos por las diferentes configuraciones del ranking RFE, en donde dichas configuraciones varían de acuerdo a los estimadores utilizados y la cantidad de muestras en cada prueba realizada, cabe mencionar que

dicha herramienta es utilizada para la selección de las mejores características dentro de un conjunto finito de estas, debido a esto los resultados poseen 3 conjuntos de datos importantes.

El primer conjunto de datos llamado “Feature” muestra los índices de las características mejor correlacionadas según el estimador utilizado en el RFE, para el caso que se muestra en la tabla 5.7 se establecieron máximo 5 características a considerar como las mejores dentro de un universo de 10 características en total.

En el segundo conjunto de datos llamado “Key value” se da a conocer el valor codificado de la clase que más apariciones tuvo bajo el índice de la característica seleccionada, para el caso que se muestra en la tabla 5.7 el valor más frecuente fue un 5, el cual decodificado (según la tabla 5.1 vista anteriormente en la sección 5.1.6) es un sustantivo, también alcanzo a aparecer el valor 8, cuya decodificación es un signo de puntuación que no pertenece a la sub-categoría punto.

Para el tercer conjunto de datos llamado “Value frequency” se obtiene la cantidad de veces que dicho valor codificado apareció en el índice de la característica seleccionada, considerando como la cantidad máxima el número de muestras utilizadas.

Tabla 5. 7: "Resultados Ranking de Características"

		RFE estimators		
Samples	**Concepts**	**Ranking RFE-SGDClassifier**	**Ranking RFE-SVM-LinearSVC**	**Ranking RFE-LR**
217	Feature	[1, 4, 6, 7, 8]	[4, 5, 7, 8, 9]	[4, 5, 7, 8, 9]
	Key value	[5, 5, 5, 5, 8]	[5, 5, 5, 8, 5]	[5, 5, 5, 8, 5]
	Value frequency	[44, 35, 44, 51, 35]	[35, 38, 51, 35, 43]	[35, 38, 51, 35, 43]
20559	Feature	[1, 3, 4, 5, 7]	[4, 5, 8, 9, 10]	[4, 7, 8, 9, 10]
	Key value	[5, 5, 5, 5, 5]	[5, 5, 5, 5, 5]	[5, 5, 5, 5, 5]
	Value frequency	[3826, 3884, 3876, 3951, 4133]	[3876, 3951, 3410, 4578, 6818]	[3876, 4133, 3410, 4578, 6818]
41511	Feature	[2, 3, 4, 6, 9]	[2, 4, 6, 8, 9]	[4, 7, 8, 9, 10]
	Key value	[5, 5, 5, 5, 5]	[5, 5, 5, 5, 5]	[5, 5, 5, 5, 5]
	Value frequency	[7677, 7682, 7811, 8256, 9324]	[7677, 7811, 8256, 6964, 9324]	[7811, 8289, 6964, 9324, 13538]
88337	Feature	[3, 5, 6, 8, 9]	[3, 4, 8, 9, 10]	[4, 7, 8, 9, 10]
	Key value	[5, 5, 5, 5, 5]	[5, 5, 5, 5, 5]	[5, 5, 5, 5, 5]
	Value frequency	[16397, 17000, 17649, 14783, 19830]	[16397, 16480, 14783, 19830, 28734]	[16480, 17619, 14783, 19830, 28734]
157213	Feature	[1, 2, 3, 5, 9]	[3, 4, 5, 7, 9]	[3, 4, 7, 9, 10]
	Key value	[5, 5, 5, 5, 5]	[5, 5, 5, 5, 5]	[5, 5, 5, 5, 5]
	Value frequency	[28543, 28685, 28702, 29730, 33070]	[28702, 29064, 29730, 31296, 33070]	[28702, 29064, 31296, 33070, 50208]

6. Conclusiones

En síntesis, una vez comprendido el contexto de la problemática y realizado una exploración pertinente en cuanto a las tecnologías que se pueden encontrar en el mercado, se ha podido concluir que la utilización de herramientas libres como Python, ofrecen una amplia gama de bibliotecas que pueden ayudar y/o facilitar resolver muchos problemas con respecto al manejo de grandes volúmenes de datos y el procesamiento de estos mismos, esto debido a la gran comunidad que hasta la fecha se encuentra activa y en constante desarrollo.

Por otro lado, haciendo una retrospectiva de forma global en base a los resultados obtenidos para cada modelo de clasificación frente a distintas configuraciones y utilizando como métrica de comparación su exactitud (ven anexos C, tablaC5.6) se puede inferir que el modelo que destaca entre los demás con el mejor porcentaje de exactitud es el modelo GaussianNB con un 63%, en donde dicho porcentaje es alcanzado por este modelo bajo tres configuraciones distintas; por medio de RFE-LogReg en una cantidad de 157.213 muestras, sin utilizar RFE con una cantidad de 88.337 muestras y nuevamente sin utilizar RFE pero esta vez con 217 muestras. Sin embargo, para el caso contrario el porcentaje más bajo fue de 45%, obtenido por 2 modelos distintos: el modelo SVM utilizando RFE-LogReg con una cantidad de 157.213 muestras y el modelo Logistic Regression utilizando RFE-SGDClassifier con 217 muestras.

En cuanto al modelo de clasificación que mejor se ajustó a cada configuración de RFE (estimador utilizado) tenemos; para RFE-SGDClassifier el mejor modelo de clasificación fue el Logistic Regression alcanzando un porcentaje de exactitud del 60%, mientras que para el RFE-SVM-LinearSVC fue el GaussianNB con un 62%, y por ultimo para el RFE-LogReg fue el GaussianNB con un 63%.

Otro de los aspectos a concluir es la aparición reiterada de sustantivos como resultado de los valores decodificados dentro de los índices de las características mejor correlacionadas (haciendo referencia a los resultados que entrega el ranking generado por el RFE usando distintos estimadores, ver tabla 5.7), esto quiere decir que en la mayoría de las veces habrá por lo menos 5 sustantivos antes de un punto o signo de puntuación dentro de un rango de 10 palabras anteriores a dicho carácter, todo esto englobado en las características más importantes según el método RFE y sus correspondientes estimadores.

Volviendo a los modelos de clasificación, si bien los resultados en cuanto a exactitud de estos modelos fueron generalmente bajos, se estima que una de las principales causas es por la forma en que se recolectaron las muestras desde la data ya pre-procesada, en donde es posible que al tratar de intercalar la recolección para poder obtener una cantidad de muestras equilibradas (etiquetado tanto positivo como negativo) se pierda información importante en cuanto a la coherencia entre estas cadenas de valores, ya que esta secuencia de valores puede presentar una mayor cantidad de patrones o coherencia si se ampliara el horizonte de características, para tratar de solventar esta situación se recomendaría variar la cantidad de características a considerar antes de una etiqueta (o variable dependiente) a una cantidad mayor.

7. Trabajo futuro

Como trabajo futuro se desprende la posibilidad de realizar nuevamente la serie de experimentos tratados en este trabajo, ya que dichos experimentos no fueron desarrollados al cien por ciento en cuanto a la diversidad de resultados, puesto que hubo variables que permanecieron constantes y que tal vez su variación hubiese entregado información relevante, ya sea frente al desempeño de los modelos para clasificación como los distintos estimadores utilizados en el RFE.

También cabe mencionar que las variables más relevantes para una variación de resultados más contundente en futuras experimentaciones seria probar con distintas cantidades de características, las cuales se encuentran antes de la etiqueta clasificadora, esto quiere decir que se puede intentar aumentar o disminuir la cantidad de palabras, de igual forma se puede intentar variar la cantidad de características a considerar en el ranking entregado por el método RFE, ya que según el tipo de estimador puede funcionar mejor o peor si presenta más o menos características para el ranking.

Otra de las razones para continuar con las experimentaciones se debe a la falta de resultados en base a muestras con un orden random, ya que, si bien se lograron obtener algunos resultados sacando el promedio de las métricas básicas para 10 iteraciones con muestras en orden random, ver las métricas básicas en los anexos D, en la figura D5.2, figura D5.4, figura D5.6, figura D5.8 y figura D5.10. Los resultados obtenidos para la cantidad de variables consideradas hacían que fuese un trabajo monumental detallar cada una de las iteraciones desembocando en la omisión de algunos factores importantes como la matriz de confusión o la exactitud de los modelos en cada iteración.

8. Referencias

[1] Aggarwal, C.C., Reddy, C.K.: Data Clustering: Algorithms and Applications. CRC, Boca Raton (2014)

[2] Anders Søgaard, Yoav Goldberg: Deep multi-task learning with low level tasks supervised at lower layers (2016)

[3] Bishop, C.M.: Pattern Recognition and Machine Learning (Information Science and Statistics). Springer, New York (2007)

[4] Duda, R.O., Hart, P.E., Stork, D.G.: Patrón de clasificación. Wiley-Interscience, Chichester (2000)

[5] Eliyahu Kiperwasser, Yoav Goldberg: Simple and Accurate Dependency Parsing Using Bidirectional LSTM Feature Representations (2016)

[6] Francois Chollet: Deep Learning with Python. Manning (2018)

[7] Kodratoff, Y., Michalski, R.S.: Machine Learning: An Artificial Intelligence Approach, vol. 3. Morgan Kaufmann, San Mateo (2014)

[8] Kuhn, M., Johnson, K.: Applied Predictive Modeling. Springer, New York (2013)

[9] Madeline Remse, Mohsen Mesgar and Michael Strube: Feature-Rich Error Detection in Scientific Writing Using Logistic Regression (2016)

[10] Marsland, S.: Machine Learning: An Algorithmic Perspective. CRC, Boca Raton (2014)

[11] Matthew Honnibal, Mark Johnson: An Improved Non-monotonic Transition System for Dependency Parsing (2015)

[12] Matthew Honnibal: Parsing English in 500 Lines of Python (2013)

[13] Mitchell, T.M.: Machine Learning. McGraw-Hill Science/Engineering/Math, New York, NY (1997)

[14] Müller, P., Quintana, F.A., Jara, A., Hanson, T.: Bayesian Nonparametric Data Analysis. Springer, New York (2015)

[15] Yoav Goldberg, Joakim Nivre: A Dynamic Oracle for Arc-Eager Dependency Parsing (2012)

[16] Yuan Zhang, David Weiss: Stack-propagation: Improved Representation Learning for Syntax (2016)

9. Anexos

A: Tabla de categorías relevantes del corpus

Figura A3. 1: "Definición de las 9 categorías relevantes del corpus"

Columna	Descripción
Journal	Revista a la que pertenece
Title	Título del paper.
Doi	Digital Object Identifier.
Abstract	Abstract del paper.
Keywords	Palabras claves
Introduction	Lista de parrafos correspondientes a la introducción.
Materials and Methods	Lista de parrafos correspondientes a la sección Materials and Methods.
Results and Discussions	Lista de parrafos correspondientes a la sección de Results and Discussions.
Conclusion	Lista de parrafos correspondiente a la sección Conclusions.

B: Esquemas sobre el flujo de datos

Figura B5. 1: "Data Pre-processing"

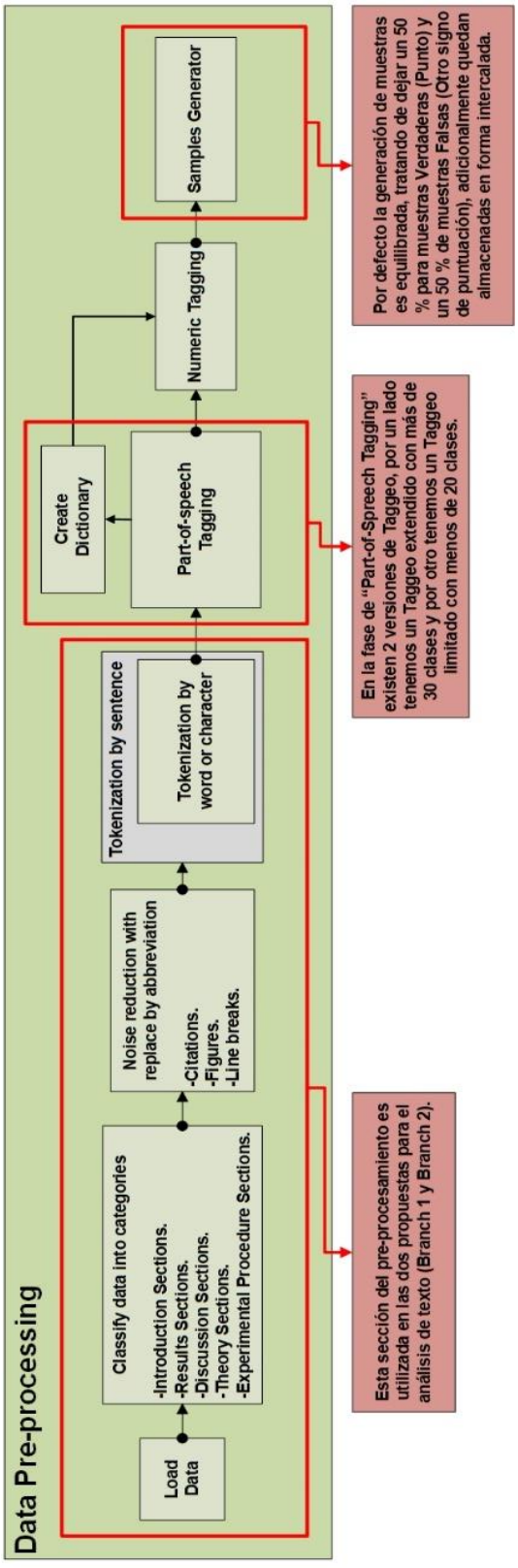
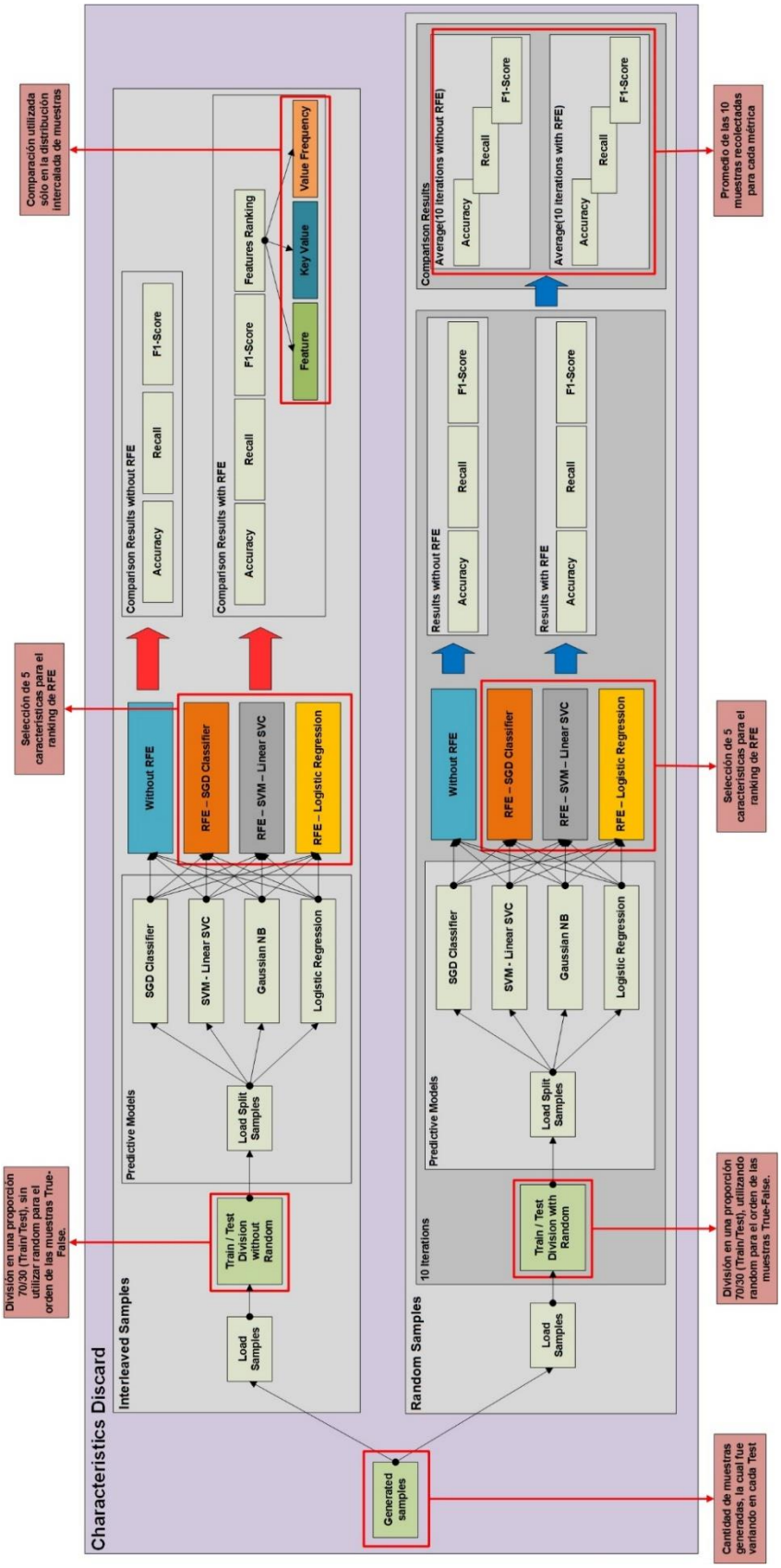


Figura B5. 2: "Characteristics Discard"



C: Gráficos y tablas de comparación entre modelos

Figura C5. 1: "Comparación Exactitud de modelos - Test 1"

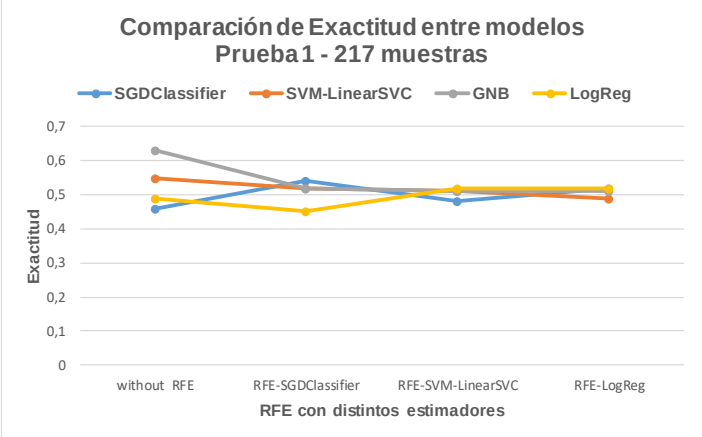


Tabla C5. 1: "Comparación Exactitud de modelos - Test 1"

Porcentajes de Exactitud de cada modelo para la Prueba 1 - 217 muestras				
Models	**without RFE**	**RFE-SGDClassifier**	**RFE-SVM-LinearSVC**	**RFE-LogReg**
SGDClassifier	0,46	0,54	0,48	0,52
SVM-LinearSVC	0,55	0,52	0,51	0,49
GNB	0,63	0,52	0,51	0,51
LogReg	0,49	0,45	0,52	0,52

Figura C5. 2: "Comparación Exactitud de modelos - Test 2"

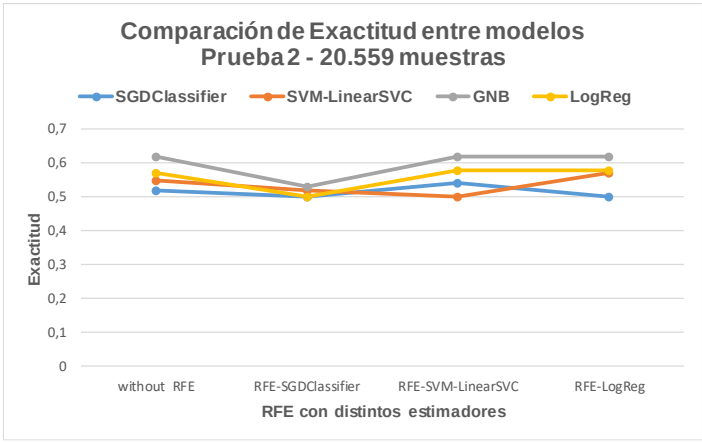


Tabla C5. 2: "Comparación Exactitud de modelos - Test 2"

Porcentajes de Exactitud de cada modelo para la Prueba 2 - 20.559 muestras				
Models	**without RFE**	**RFE-SGDClassifier**	**RFE-SVM-LinearSVC**	**RFE-LogReg**
SGDClassifier	0,52	0,5	0,54	0,5
SVM-LinearSVC	0,55	0,52	0,5	0,57
GNB	0,62	0,53	0,62	0,62
LogReg	0,57	0,5	0,58	0,58

Figura C5. 3: "Comparación Exactitud de modelos - Test 3"

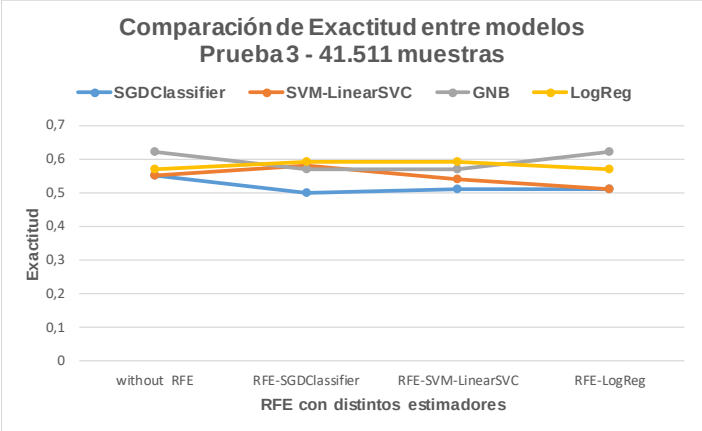


Tabla C5. 3: "Comparación Exactitud de modelos - Test 3"

Porcentajes de Exactitud de cada modelo para la Prueba 3 - 41.511 muestras				
Models	**without RFE**	**RFE-SGDClassifier**	**RFE-SVM-LinearSVC**	**RFE-LogReg**
SGDClassifier	0,55	0,5	0,51	0,51
SVM-LinearSVC	0,55	0,58	0,54	0,51
GNB	0,62	0,57	0,57	0,62
LogReg	0,57	0,59	0,59	0,57

Figura C5. 4: "Comparación Exactitud de modelos - Test 4"

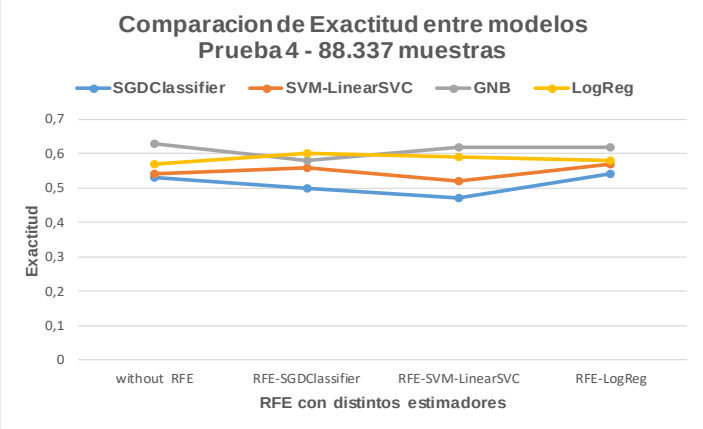


Tabla C5. 4: "Comparación Exactitud de modelos - Test 4"

Porcentajes de Exactitud de cada modelo para la Prueba 4 - 88.337 muestras				
Models	**without RFE**	**RFE-SGDClassifier**	**RFE-SVM-LinearSVC**	**RFE-LogReg**
SGDClassifier	0,53	0,5	0,47	0,54
SVM-LinearSVC	0,54	0,56	0,52	0,57
GNB	0,63	0,58	0,62	0,62
LogReg	0,57	0,6	0,59	0,58

Figura C5. 5: "Comparación Exactitud de modelos - Test 5"

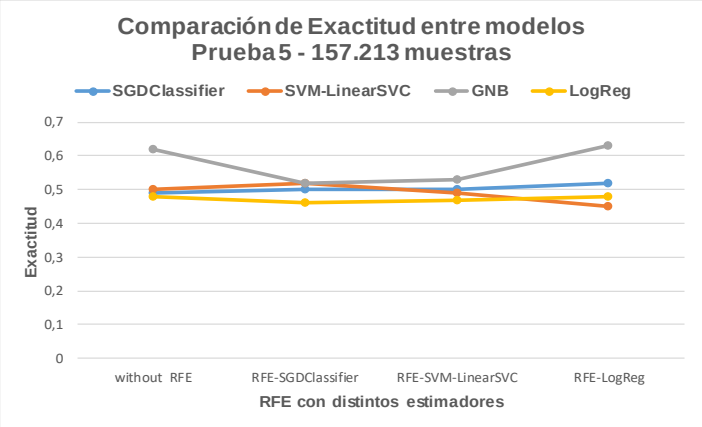


Tabla C5. 5: "Comparación Exactitud de modelos - Test 5"

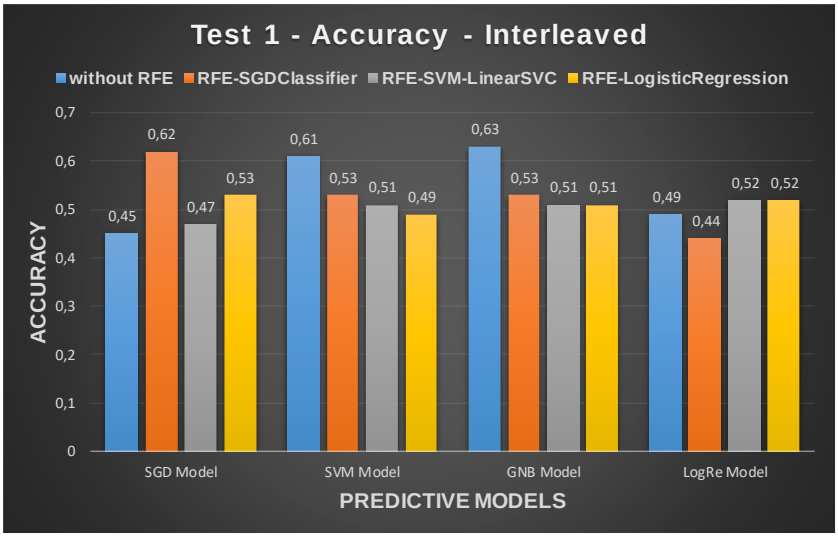
Porcentajes de Exactitud de cada modelo para la Prueba 5 - 157.213 muestras				
Models	**without RFE**	**RFE-SGDClassifier**	**RFE-SVM-LinearSVC**	**RFE-LogReg**
SGDClassifier	0,49	0,5	0,5	0,52
SVM-LinearSVC	0,5	0,52	0,49	0,45
GNB	0,62	0,52	0,53	0,63
LogReg	0,48	0,46	0,47	0,48

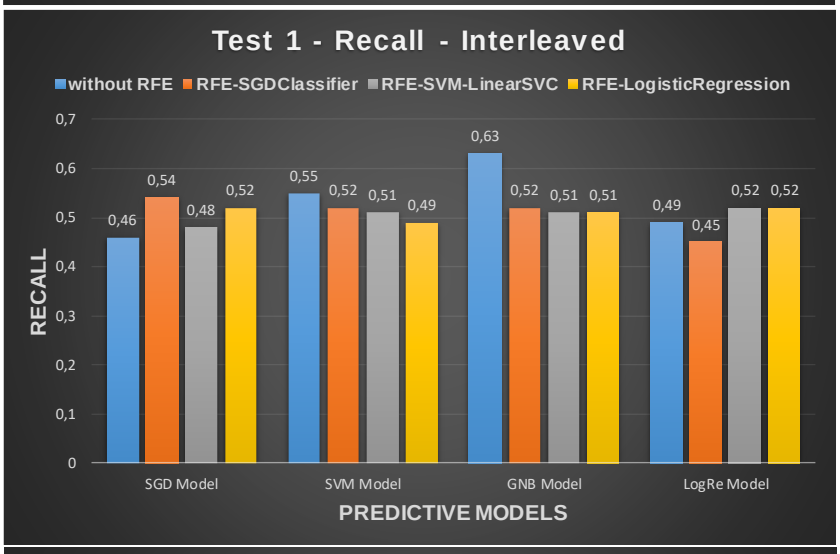
Figura C5. 6: "Comparación general de Exactitud para cada modelo"

Tabla general de porcentajes de Exactitud para cada modelo					
N° Test	**Models**	**without RFE**	**RFE-SGDClassifier**	**RFE-SVM-LinearSVC**	**RFE-LogReg**
Test 1	SGDClassifier	0,46	0,54	0,48	0,52
	SVM-LinearSVC	0,55	0,52	0,51	0,49
	GNB	0,63	0,52	0,51	0,51
	LogReg	0,49	0,45	0,52	0,52
Test 2	SGDClassifier	0,52	0,5	0,54	0,5
	SVM-LinearSVC	0,55	0,52	0,5	0,57
	GNB	0,62	0,53	0,62	0,62
	LogReg	0,57	0,5	0,58	0,58
Test 3	SGDClassifier	0,55	0,5	0,51	0,51
	SVM-LinearSVC	0,55	0,58	0,54	0,51
	GNB	0,62	0,57	0,57	0,62
	LogReg	0,57	0,59	0,59	0,57
Test 4	SGDClassifier	0,53	0,5	0,47	0,54
	SVM-LinearSVC	0,54	0,56	0,52	0,57
	GNB	0,63	0,58	0,62	0,62
	LogReg	0,57	0,6	0,59	0,58
Test 5	SGDClassifier	0,49	0,5	0,5	0,52
	SVM-LinearSVC	0,5	0,52	0,49	0,45
	GNB	0,62	0,52	0,53	0,63
	LogReg	0,48	0,46	0,47	0,48

D: Métricas para comparación entre modelos

Figura D5. 1: "Conjunto de métricas para Test 1 - Muestras Intercaladas"





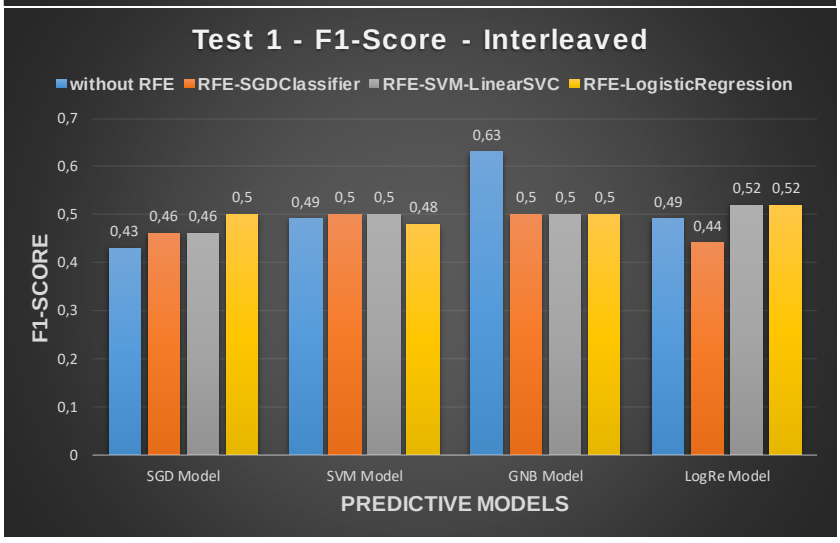
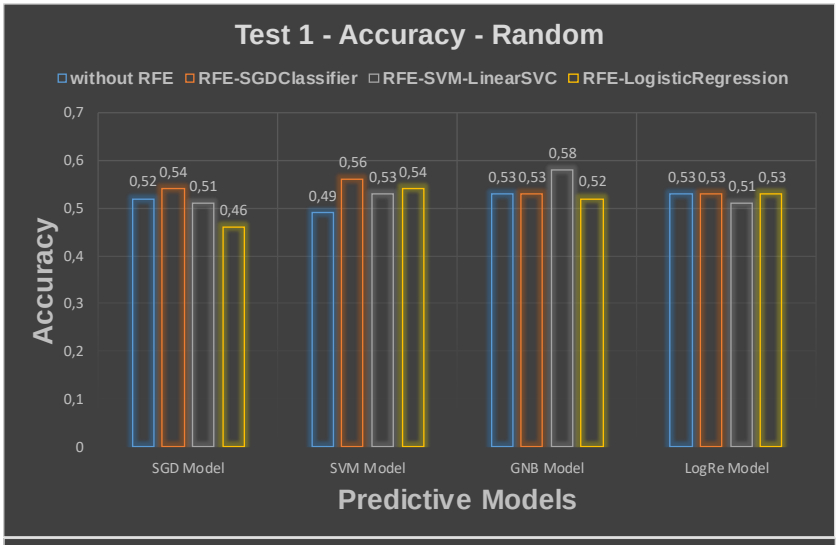
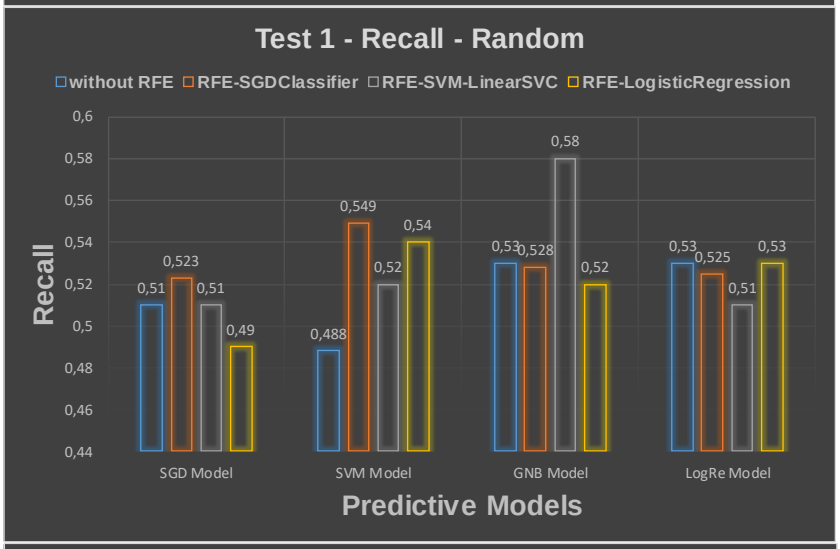


Figura D5. 2: "Conjunto de métricas para Test 1 - Muestras Random"





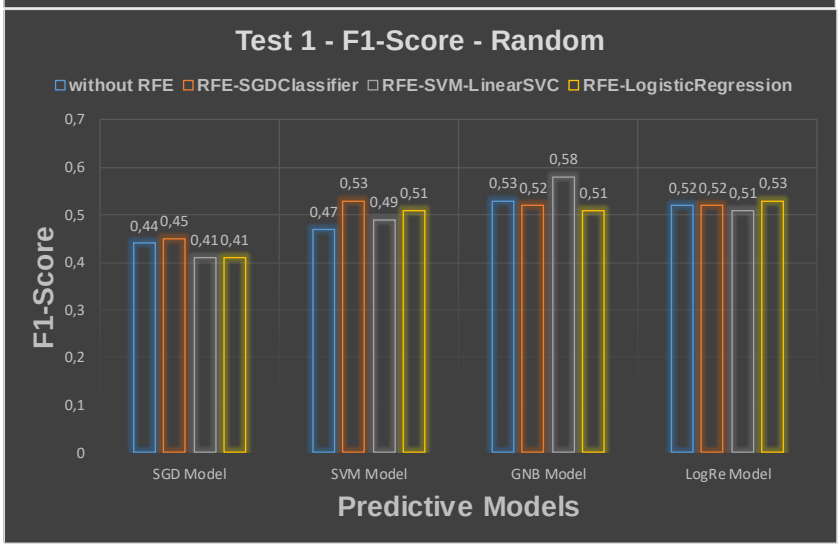


Figura D5. 3: "Conjunto de métricas para Test 2 - Muestras Intercaladas"

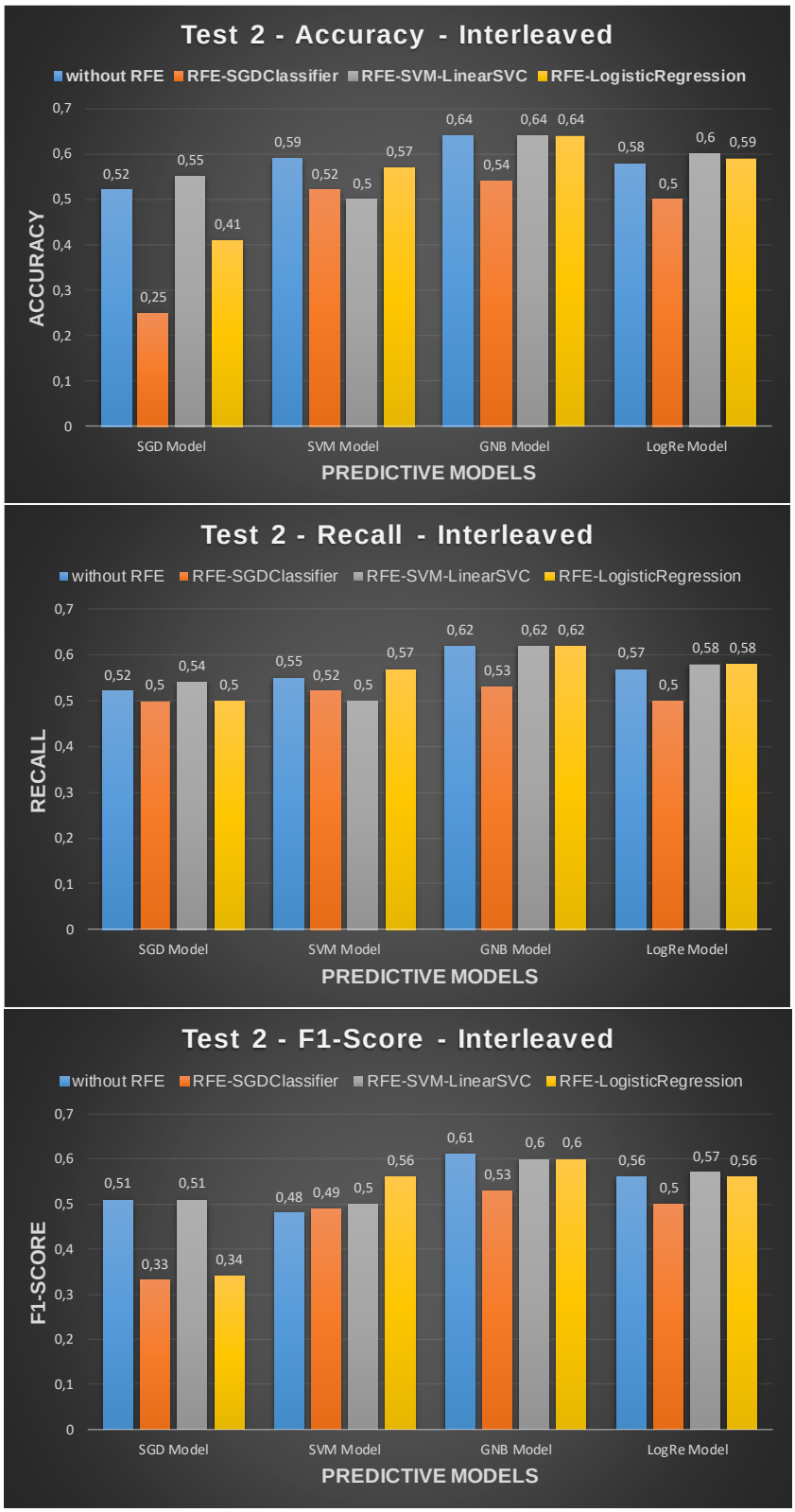


Figura D5. 4: "Conjunto de métricas para Test 2 - Muestras Random"

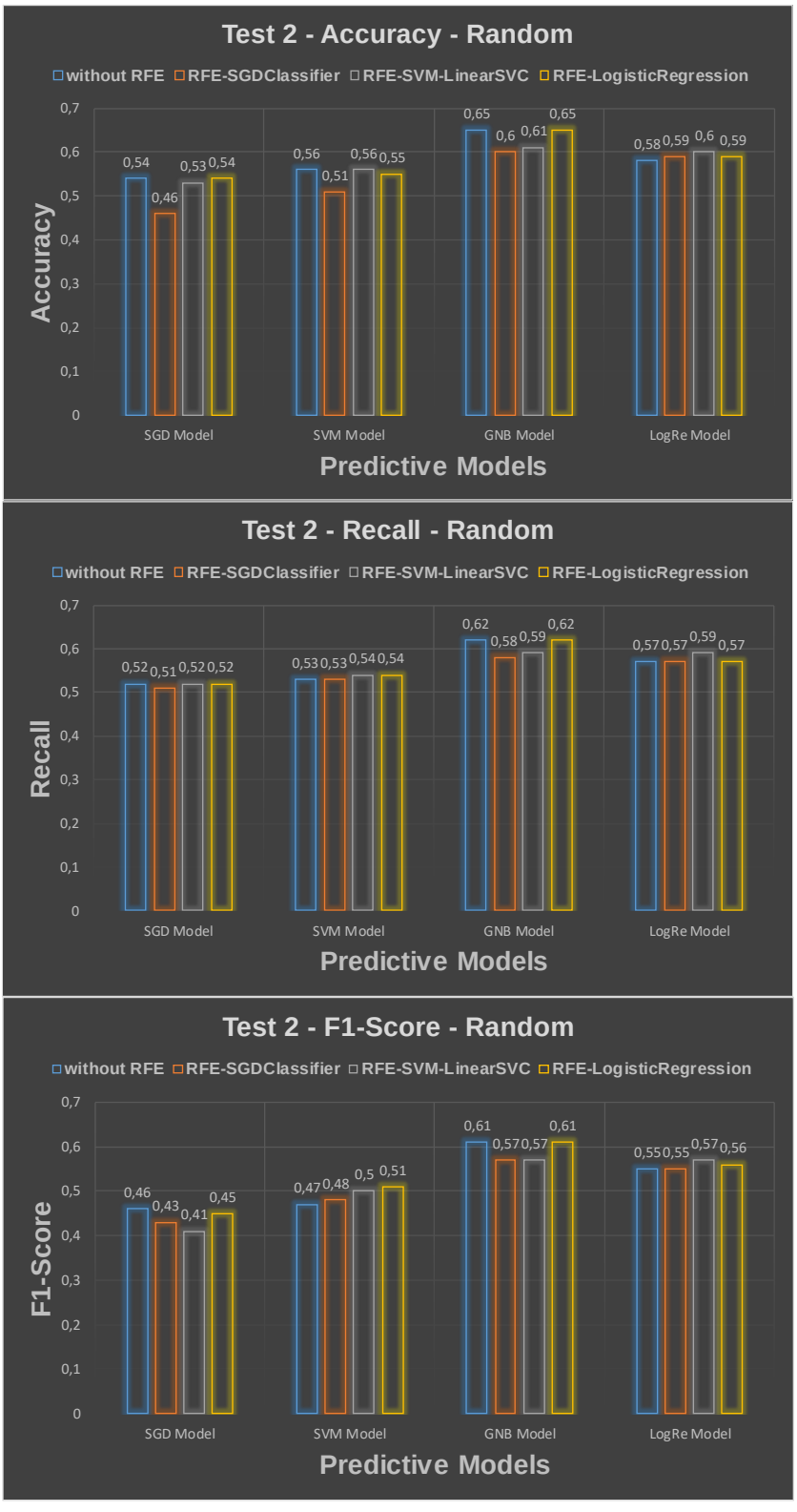


Figura D5. 5: "Conjunto de métricas para Test 3 - Muestras Intercaladas"

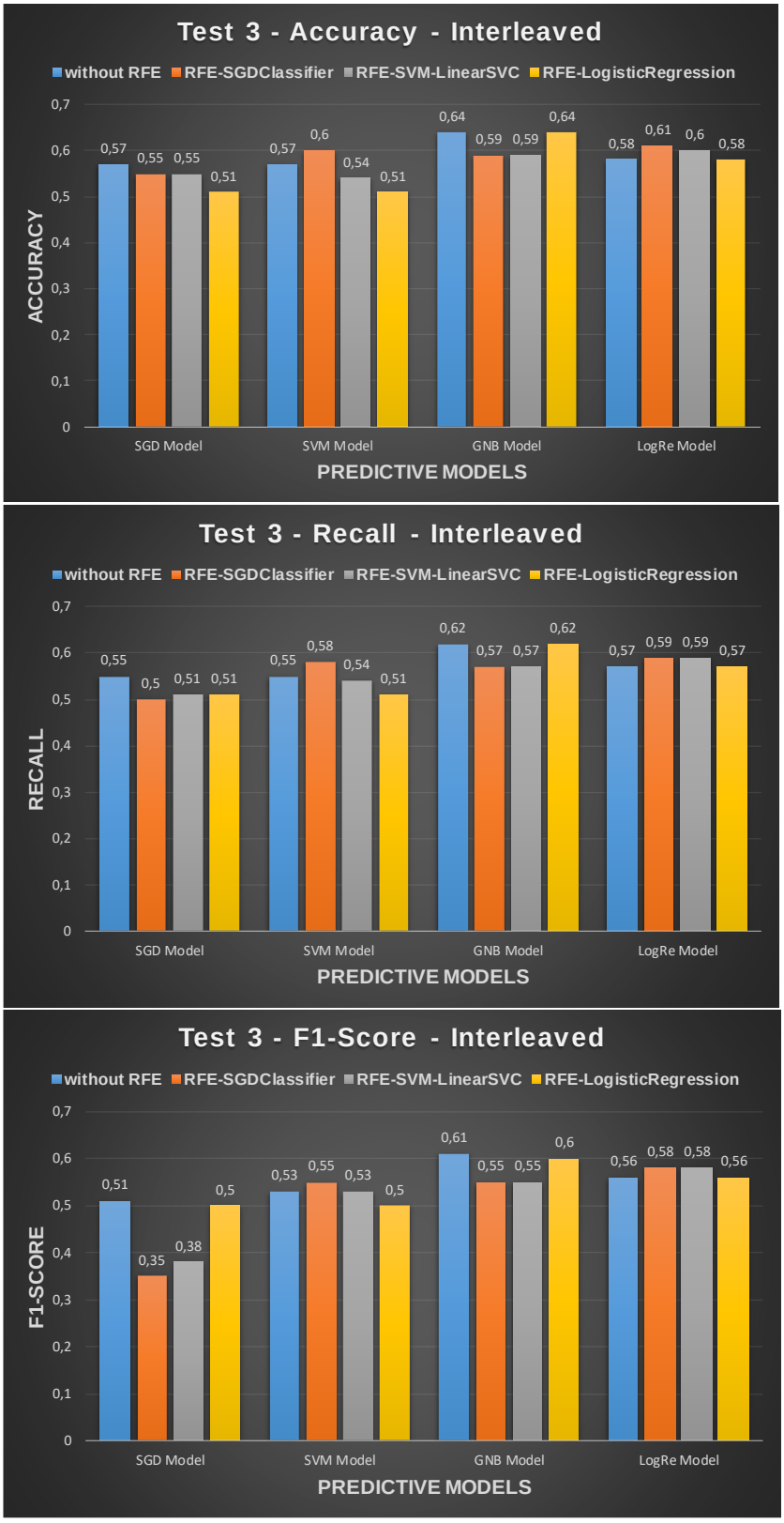


Figura D5. 6: "Conjunto de métricas para Test 3 - Muestras Random"

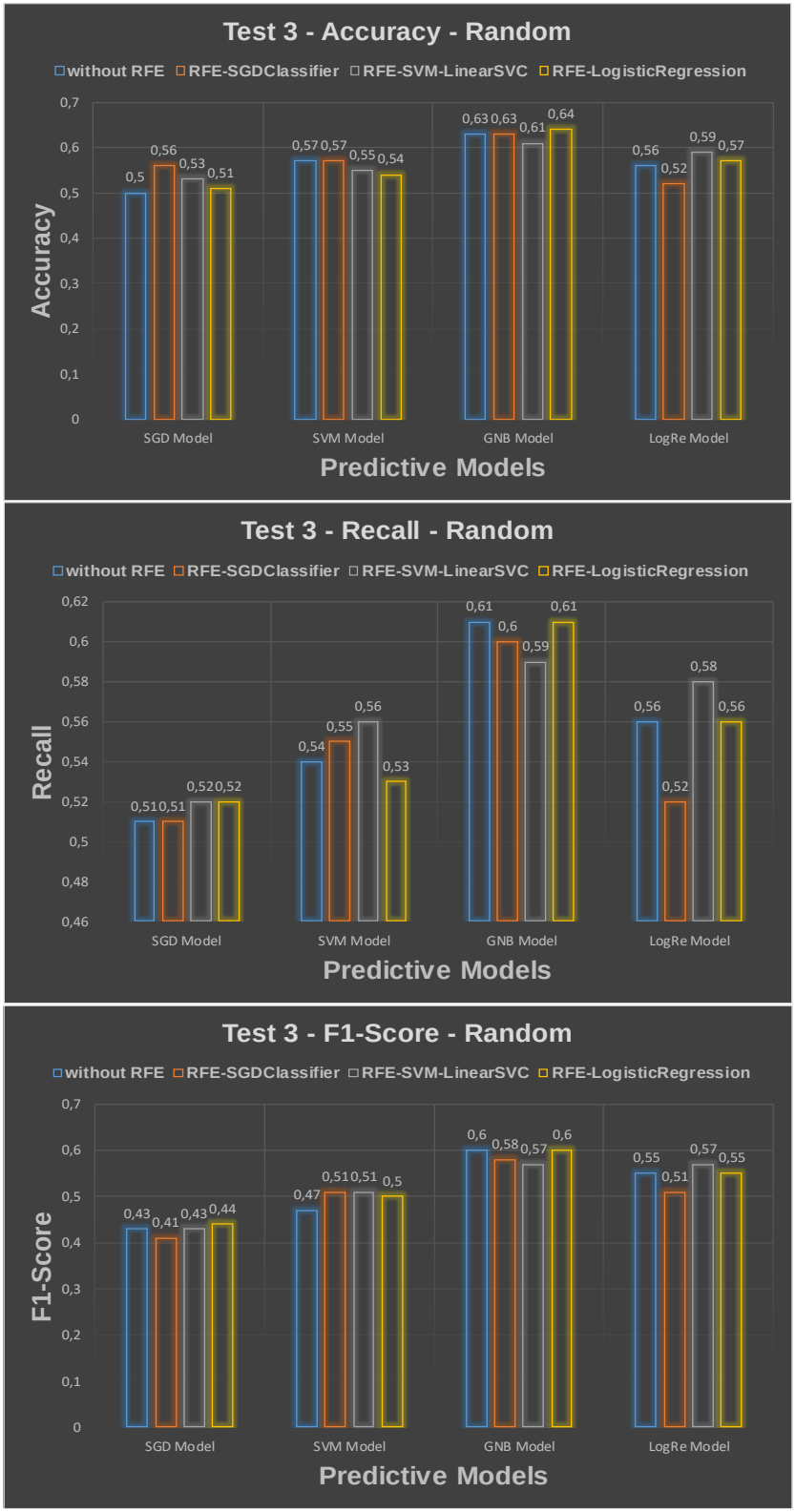
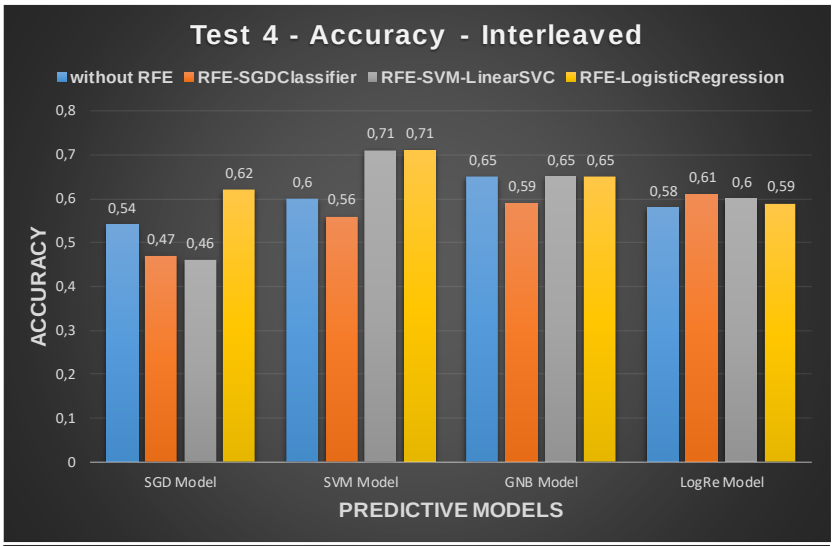
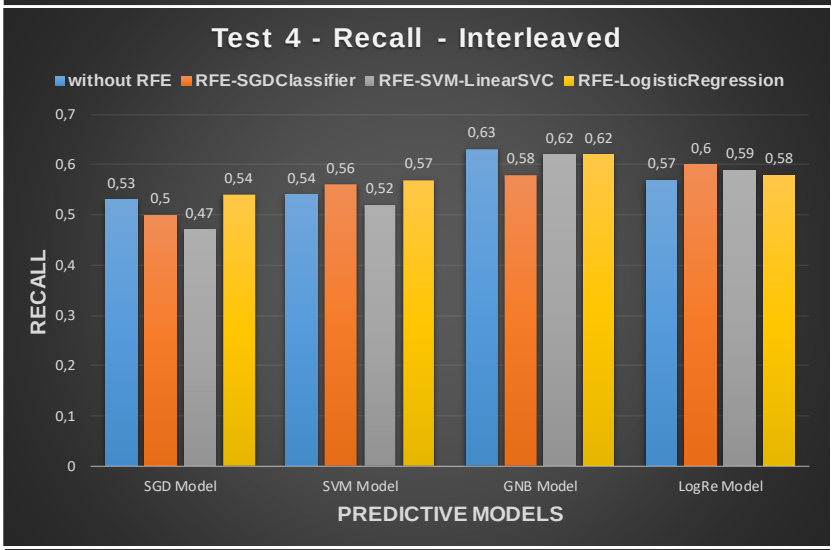


Figura D5. 7: "Conjunto de métricas para Test 4 - Muestras Intercaladas"





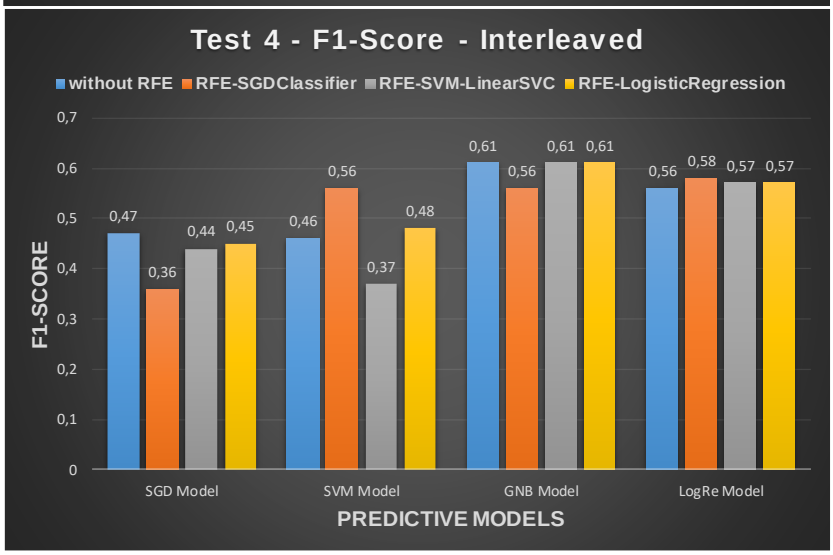


Figura D5. 8: "Conjunto de métricas para Test 4 - Muestras Random"

Figura D5. 9: "Conjunto de métricas para Test 5 - Muestras Intercaladas"

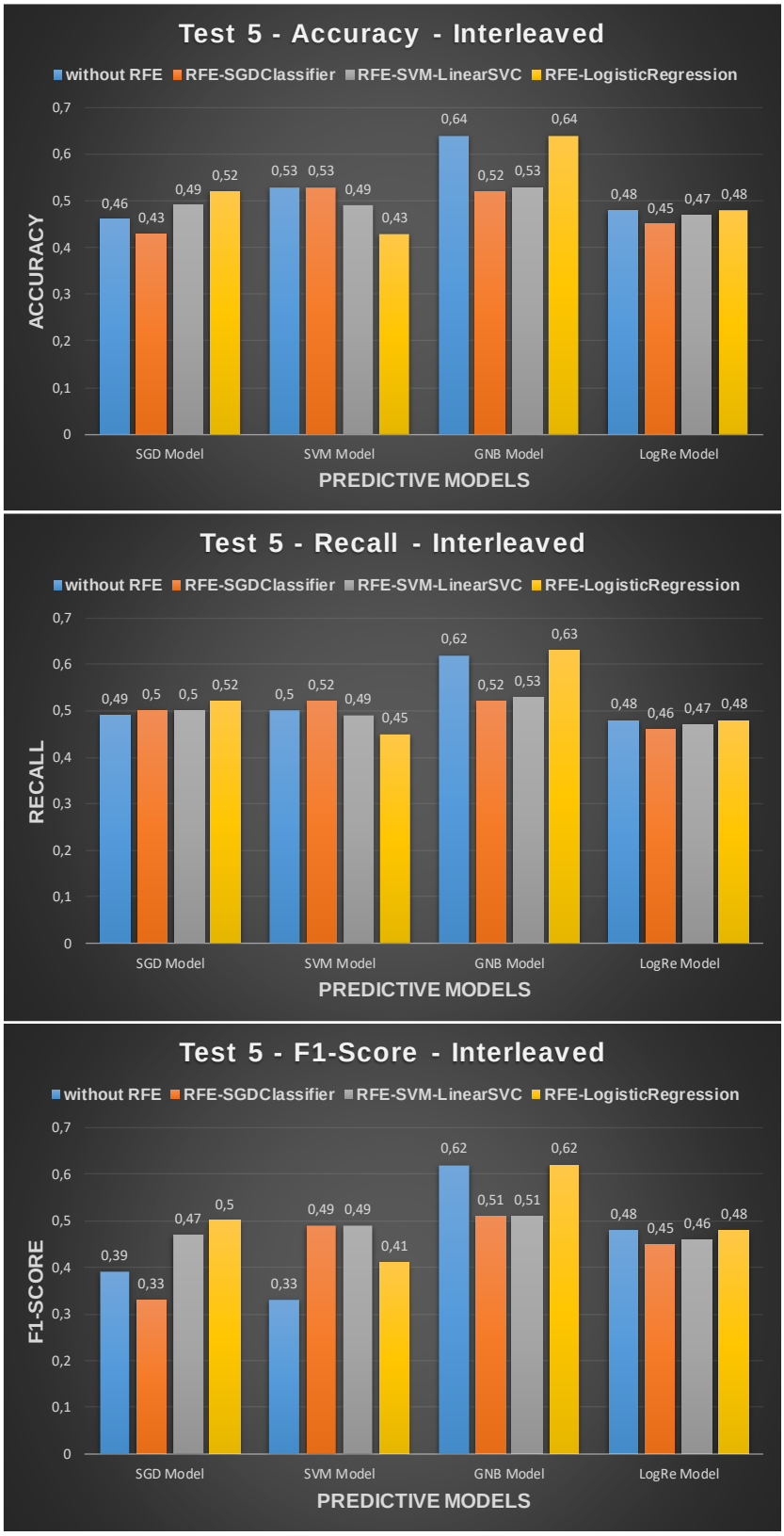


Figura D5. 10: "Conjunto de métricas para Test 5 - Muestras Random"