UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Ensembles de clasificadores Multi-Label en Scikit-Learn

Presentado por Eduardo Tubilleja Calvo
en Universidad de Burgos — 12 de febrero de 2018
Tutor: Dr. Álvar Arnaiz González
y Dr. Juan José Rodríguez Díez

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. Álvar Arnaiz González y D. Juan José Rodríguez Díez, profesores del departamento de nombre departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos.

Exponen:

Que el alumno D. Eduardo Tubilleja Calvo, con DNI 71298897R, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado «Ensembles de clasificadores multi-label en Scikit-Learn».

Y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual se autoriza su presentación y defensa.

En Burgos, 12 de febrero de 2018

V^{o}. B^{o}. del Tutor:

D. Álvar Arnaiz González

V^{o}. B^{o}. del Tutor:

D. Juan José Rodríguez Díez

Resumen

La minería de datos es un campo de las ciencias de la computación, que consisten en el análisis de grandes cantidades de datos para descubrir patrones. Para ello se puede utilizar el aprendizaje automático, éste pertenece a un subcampo de las ciencias de computación y de la rama de inteligencia artificial, el objetivo del aprendizaje automático es desarrollar unas técnicas que permitan que las máquinas aprendan [32].

En la actualidad, lo más habitual es usar Single-Label, es decir, los conjuntos de datos solo pueden pertenecer a una clase. Nosotros nos enfrentaremos a Multi-Label, en el que el conjunto de datos pueden pertenecer a más de una clase, como por ejemplo en el etiquetado de imágenes en el que una imagen puede tener a la vez las etiquetas «árbol», «montaña» y «mar».

En este proyecto vamos a tratar de implementar diversos algoritmos de multi-clasificadores (*ensembles*), para Multi-Label sobre la biblioteca de Python [31] Scikit-Learn [18]. Se ha seguido la guía de estilo de Python (PeP [10]) y las convecciones que se han observado en Sklearn (Scikit-Learn). Para que se entienda mejor y sea más gráfico, se han dibujado árboles y gráficas, mostrando los resultados al ejecutar dichos algoritmos sobre unos conjuntos de datos. Los algoritmos en los que nos vamos a centrar son Disturbing Neighbors [16], Random Oracles [24] y Rotation Forest [25].

Descriptores

Python, Scikit-Learn, Ensembles, Multi-Label, Minería de datos, Aprendizaje automático.

Abstract

Data mining is an area of computer science, which involves the analysis of large data sets to discover patterns. To do this you can use machine learning, it belongs to a subarea of computer science and the artificial intelligence, the objective of machine learning is to develop techniques that allow machines to learn [32].

Currently, the most common is to use Single-Label, that is, the data sets can only belong to one class. We will challenge Multi-Label, in which the data set can belong to more than one class, for example in the labeling of images in which An image can have both tags «tree», «mountain» and «sea».

In this project we will implement some multi-classifier algorithms (*ensembles*), for Multi-Label on the Python library [31] Scikit-Learn [18]. It has followed the Python style guide (PeP [10]) and the conventions that have been observed in Sklearn (Scikit-Learn). To make it better understand and more graphic, trees and graphs have been drawn, showing the results when executing those algorithms on some data sets. The algorithms we are going to focus on are Disturbing Neighbors [16], Random Oracles [24] and Rotation Forest [25].

Keywords

Python, Scikit-Learn, Ensembles, Multi-Label, Datamining, Machine learning.

Índice general

Índice de figuras

Índice de tablas

Introducción

La minería de datos es un campo de las ciencias de la computación, que consisten en el análisis de grandes cantidades de datos para descubrir patrones. Para ello se puede utilizar el aprendizaje automático, éste pertenece a un subcampo de las ciencias de computación y de la rama de inteligencia artificial, el objetivo del aprendizaje automático es desarrollar unas técnicas que permitan que las máquinas aprendan [32]. Dentro de este aprendizaje se encuentra el aprendizaje supervisado, en él normalmente los conjuntos de datos suelen tener solo una variable a predecir, conocido como Single-Label, pero apareció el concepto de Multi-Label [29], este hace referencia a los conjuntos de datos en lo que cada elemento de la base de datos puede pertenecer a más de una clase, como por ejemplo en el etiquetado de imágenes: en el que una imagen puede tener a la vez las etiquetas «árbol», «montaña» y «mar».

En este proyecto vamos a tratar de implementar diversos algoritmos de multi-clasificadores (*ensembles*), para Multi-Label sobre la biblioteca de Python [31] Scikit-Learn [18]. Se ha seguido la guía de estilo de Python (PeP [10]) y las convecciones que se han observado en Sklearn (Scikit-Learn). Para que se entienda mejor y sea más gráfico, se han dibujado árboles y gráficas, mostrando los resultados al ejecutar dichos algoritmos sobre unos conjuntos de datos. Los algoritmos en los que nos vamos a centrar son Disturbing Neighbors [16], Random Oracles [24] y Rotation Forest [25].

A lo largo del proyecto también veremos como funcionan los *ensembles*. Los métodos de *ensembles* combinan las predicciones de unos clasificadores base, que están construidos mediante un algoritmo de aprendizaje para mejorar la solidez de si sólo tuviesemos un clasificador [26]. El éxito de un *ensemble* requiere que un clasificadores base tengan exactitud y diversidad. La ventaja de usar un *ensemble* de clasificadores base consiste en la posibilidad que algunos de ellos pueden corregir una predicción incorrecta de otros, por lo que cuantos más clasificadores tengamos mas precisas serán las predicciones. ¿Cómo puede un *ensemble* de clasificadores base que han sido generados por el mismo algo-

ritmo tener distintas salidas? Una de las estrategias que podemos usar para ello son los *ensembles* homogéneos, es decir, mismo algoritmo entrenado con distinto conjunto de datos.

Objetivos del proyecto

En este apartado, se explican los objetivos que se pretenden conseguir al final.

A continuación se muestra el esquema con todos los puntos a tratar en este proyecto.

- Implementar los algoritmos Disturbing Neighbors [16], Random Oracles [24] y Rotation Forest [25] en Scikit-Learn:
  - Que sean capaces de trabajar para datos Single-Label y Multi-Label.
  - Crear el método `fit` para entrenar un conjunto de datos.
  - Crear el método `predict` para predecir según el entrenamiento de unos datos.
  - Crear el método `predict_proba` para predecir probabilidades según el entrenamiento de unos datos.
  - Evaluar el correcto funcionamiento de los algoritmos, mediante unos test.
- Crear unos notebooks para mostrar el funcionamiento de los algoritmos:
  - Posibilidad de seleccionar el algoritmo.
  - Permitir utilizar conjuntos reales.
  - Mostrar su funcionamiento mediante el dibujado del árbol de decisión.
  - Mostrar el funcionamiento mediante gráficas en dos dimensiones con conjuntos de datos de «juguete».
  - Mostrar resultados al usar validación cruzada.

Conceptos teóricos

Este apartado va a explicar los conceptos teóricos necesarios para poder entender el proyecto.

3.1. Minería de Datos

> Es conocida la frase «los datos en bruto raramente son beneficiosos directamente». Pueden tener valor, ya que podemos extraer información útil para la toma de decisiones o exploración, y también para la compresión del fenómeno dominante en el conjunto de datos [23].

La finalidad de esto es descubrir unos patrones, una similitud o una propensión que expliquen el comportamiento de los datos. Para hacer esto utiliza los métodos de la inteligencia artificial y estadística. El objetivo del proceso de minería de datos consiste en extraer información de un conjunto de datos, luego se interpreta esta información para un uso posterior [32].

La minería de datos está basada en el aprendizaje automático, para ello se considera un conjunto con n muestras y se intenta predecir las propiedades de los datos desconocidos. Podemos separar los problemas de aprendizaje principalmente en dos [9]:

- Aprendizaje supervisado: Consiste en que a partir de un conjunto de datos, hacer predicciones basadas en el comportamiento o las características de dichos datos. Nos permite buscar patrones en datos. Dos de las tareas más comunes del aprendizaje supervisado son la clasificación y la regresión:

  - Clasificación: El programa debe aprender a predecir en que categoría o clase irán los nuevos datos, según las nuevas observaciones, por ejemplo, predecir si el precio de una acción bajará o subirá.
  - Regresión: El programa debe predecir el valor de una variable de respuesta continua, por ejemplo, predecir las ventas de un nuevo producto.

- Aprendizaje no supervisado: Usa datos que no están etiquetados. El objetivo es explorarlos para encontrar alguna forma de organizarlos.

La minería de datos está incluida en el proceso de KDD (Knowledge Discovery in Databases, Descubrimiento del conocimiento en bases de datos) Este proceso tiene una secuencia de pasos [11]:

1. Limpieza de datos. Eliminación del ruido y la inconsistencia de los datos.
2. Integración de los datos. Múltiples fuentes de datos son combinadas.
3. Selección de datos. Los datos relevantes para la tarea de análisis se recuperan de la base de datos.
4. Transformación de datos. Los datos son transformados y se consolidan como apropiados para la minería mediante la realización de operaciones simétricas o de agregación.
5. Minería de datos. Es un proceso esencial donde los métodos de inteligencia son aplicados a la extracción de patrones.
6. Evaluación de patrones. Identificar los patrones que de verdad son interesantes, que representan el conocimiento basado en medidas de interés.
7. Presentación del conocimiento. Representación visual de los resultados obtenidos.

Aunque el proceso de la minería de datos consta de más etapas nosotros nos centraremos en 4 etapas [27]:

- Determinación de los objetivos: Se tratan los objetivos que quiere conseguir el cliente bajo un asesor especialista en minería de datos.
- Preprocesamiento de los datos: Es la etapa que más tiempo se tarda en realizar el proceso. Se seleccionan, limpian, enriquecen, reducen y transforman las bases de datos.

- Determinación del modelo: Se lleva a cabo un estudio estadístico de los datos, más tarde se hace una visualización gráfica para una primera aproximación. Según los objetivos que se habían propuesto se pueden usar diferentes algoritmos de la Inteligencia Artificial.
- Análisis de los resultados: Se comprueban si los datos obtenidos tienen coherencia, después se comparan con los obtenidos en los estudios estadísticos y la visualización gráfica. El cliente es el que ve si los datos le aportan nuevo conocimiento que le permita considerar sus decisiones.

3.2. Multi-Label

Al principio el aprendizaje supervisado trataba con el análisis de datos de Single-Label donde los ejemplos de entrenamiento son asociados con un Single-Label de un conjunto que no tiene nada en común con otros conjuntos. Sin embargo, los ejemplos de entrenamiento a veces se pueden asociar a un conjunto de labels, a los que llamaremos Multi-Label [29].

La clasificación Single-Label y Multi-Label son unas tareas que se pueden resolver con distintas técnicas de minería de datos, la cual nos permite que a partir de un conjunto de instancias de entrenamiento, podamos determinar a través de unos atributos esenciales de dichas instancias, crear unas reglas que posteriormente se usarán para clasificar nuevas instancias [19]. Como por ejemplo en el etiquetado de imágenes: en el que una imagen puede tener a la vez las etiquetas «árbol», «montaña» y «mar».

3.3. *Ensemble*

Un *ensemble* es un esquema de combinación de clasificadores base con los que obtenemos unas predicciones individuales. El éxito de un *ensemble* requiere tanto exactitud como diversidad de sus clasificadores base. La diversidad representa como de diferente son las predicciones de los clasificadores base. Si los clasificadores base siempre están de acuerdo no habría diferencia entre usar sólo un clasificador base o varios combinado por un método de *ensemble*. Entonces la ventaja de usar un *ensemble* de clasificadores base consiste en la posibilidad que algunos de ellos pueden corregir una predicción incorrecta de otros, por lo que cuantos más clasificadores tengamos más precisas serán las predicciones.

Es normal obtener estos clasificadores base en un *ensemble* usando el mismo algoritmo, así que en esta situación el proceso de entrenamiento realizado por el *ensemble* es la principal fuente de diversidad. La diversidad de Bagging [3] proviene de elegir al azar diferentes instancias para entrenar a cada clasificador de base. El método Random Subspaces [12] elige diferentes subconjuntos de atributos para entrenar a cada clasificador base. Boosting [8]

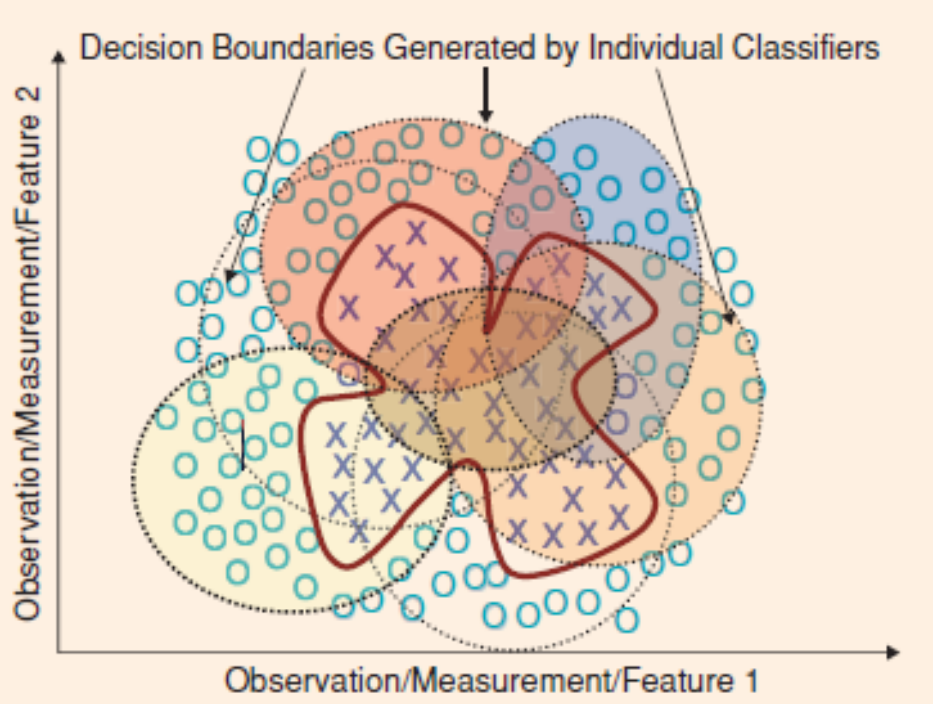

Figura 3.1: Ensemble de clasificadores que abarca un espacio de decisión

entrena de forma iterativa el conjunto de clasificadores base, modificando los pesos de las instancias para entrenar al clasificador actual. Estos nuevos pesos se calculan a partir del error de entrenamiento en el clasificador base anterior, por lo que cada nuevo clasificador de base llega más especializado en instancias que han sido mal clasificados antes. A veces los clasificadores base son muy estables y el algoritmo de entrenamiento del *ensemble* no es suficiente para proporcionar el nivel deseado de diversidad [16].

Los métodos de *ensembles* combinan las predicciones de unos estimadores base, que están construidos mediante un algoritmo de aprendizaje para mejorar la solidez de un solo estimador [26]. Se distinguen dos clases de métodos de *ensembles*:

- En los métodos de Bagging [3], se construyen varios estimadores de forma independiente y luego se calcula su promedio para las predicciones.
- En los métodos de Boosting [8], se construyen los estimadores secuencialmente y se trata de reducir el sesgo del estimador combinado.

Podemos ver un ejemplo de un ensemble de clasificadores en la siguiente figura 3.1 [20].

3.4. Disturbing Neighbors

Los Disturbing Neighbors (DN) o vecinos molestones se han utilizado con éxito para mejorar la diversidad en los bosques [16].

El método DN trabaja en cada clasificador base de la siguiente manera:

- m instancias son seleccionadas aleatoriamente de el conjunto de datos de entrenamiento para construir un clasificador 1-NN. El valor m suelen ser valores muy pequeños.
- Las dimensiones para calcular la distancia euclidea en el clasificador 1-NN, también se seleccionan. Como mínimo se seleccionan el 50 % de los atributos.
- Luego $m+1$ nuevas características son añadidas al conjunto de entrenamiento. Una de las características adicionales es la clase predicha por el clasificador 1-NN para cada instancia $\mathbf{x}$, y las otras m son características booleanas, todos los valores falsos excepto uno corresponden al vecino más cercano para esa instancia.
- El clasificador base está entrenado usando las características originales mas las nuevas $m + 1$ características.

Por lo tanto, el proceso normal de entrenamiento de los clasificadores base se altera añadiendo estas nuevas características del clasificador 1-NN. Es por eso que el método se llama vecinos molestones. La aleatoriedad aumenta la diversidad y se debe a:

- Los vecinos utilizados en cada clasificador 1-NN se seleccionan aleatoriamente. Por lo tanto, sus predicciones y las características booleanas son diferentes para cada clasificador base.
- Las dimensiones utilizadas para calcular las distancias euclidianas también se eligen de forma aleatoria, así que si dos clasificadores básicos tienen al menos los mismos m vecinos, las predicciones 1-NN y las características booleanas podrían ser diferentes.

En resumen, Disturbing Neighbors es un método para alterar el proceso de entrenamiento normal de los clasificadores base en un *ensemble*, mejorando su diversidad y mejorando la precisión general del *ensemble*.

3.5. Random Oracles

Random Oracles son mini-ensembles formados por dos o más modelos, pueden usarse como modelos base para otros métodos *ensemble*. El objetivo de usar Random Oracles es tener más diversidad entre los modelos base que forman un *ensemble* [24].

Dado un método base, el entrenamiento de un Random Oracle consiste en:

- Seleccionar aleatoriamente un Random Oracle.
- Dividir los datos de entrenamiento en dos o más subconjuntos usando el Random Oracle.
- Para cada subconjunto de datos de entrenamiento, se construye un modelo (en nuestro caso un árbol, aunque se pueden construir otros modelos). El modelo Random Oracle está formado por los modelos cada subconjunto y el propio oráculo.

La predicción del test de una instancia se realiza de la siguiente manera:

- Usa el Random Oracle para seleccionar uno de los dos o más modelos.
- Devuelve la predicción obtenida por el modelo seleccionado.

Se pueden considerar diferentes tipos de Oracles. En este trabajo, se utiliza el Linear Random Oracle. Este oráculo divide el espacio en dos o más subespacios utilizando un hiperplano. Para construir el oráculo, dos o más objetos de entrenamiento diferentes se seleccionan aleatoriamente, cada objeto de entrenamiento restante se asigna al subespacio del objeto de entrenamiento seleccionado que está más cerca.

3.6. Rotation Forest

Rotation Forest, tiene como objetivo construir clasificadores precisos y diversos. La heurística principal consiste en aplicar la extracción de características a subconjuntos de características y reconstruir un conjunto completo de características para cada clasificador en el conjunto. En el artículo original se utiliza el Análisis de Componentes Principales (PCA) [25].

Para construir el clasificador se siguen los siguientes pasos:

- Dividir los atributos del conjunto de datos en K subconjuntos. Para maximizar la diversidad se eligen subconjuntos disjuntos.

- De las clases se seleccionan las que son distintas, y de estas se selecciona una muestra, en nuestro caso la muestra tiene un tamaño del 80 %.
- Para cada subconjunto se selecciona aleatoriamente una muestra, por defecto con un tamaño del 75 %. Se ejecuta PCA sobre cada una de las muestras.
- Después de todos los pasos anteriores, para predecir se hace el `predict` del estimador base.

De esta forma, se construyen clasificadores individuales precisos.

Técnicas y herramientas

En este apartado de la memoria se presentan las técnicas metodológicas y las herramientas de desarrollo que se han utilizado para llevar a cabo el proyecto.

4.1. GitHub

Es una plataforma para alojar proyectos y utiliza git como sistema de control de versiones. Se organiza por tareas (*milestones* e *issues*). Utiliza el framework Ruby on Rails [17].

Podemos acceder a través de: https://github.com/

Ventajas:

- Es uno de los repositorios más usados, por lo que es fácil encontrar información en Internet para resolver cualquier duda.
- El código es público por lo que cualquiera puede proponer cambios en el mismo, seguirte y ver el proyecto.
- Las distintas versiones del código están alojadas en la nube por lo que si perdemos el contenido de nuestro ordenador, podremos recuperarlo.

Desventajas:

- También puedes tener proyectos privados pero para ello tienes que utilizar una cuenta de pago, aunque los estudiantes e investigadores pueden obtener esto gratuitamente.

4.2. Python

Python es un lenguaje de programación interpretado se hace hincapié en que una sintaxis que favorezca un código legible [31]. Se trata de un lenguaje de programación multiparadigma (permite crear programas utilizando mas de un estilo de programación), ya que soporta orientación a objetos, programación imperativa y programación funcional. Es un lenguaje interpretado, usa tipado dinámico (una variable puede tomar valores de distinto tipo) y es multiplataforma [34].

Python es recomendable para programadores que empiezan por primera vez, o que vienen de ostros lenguajes, ya que hay mucha documentación para dar el primer paso en este lenguaje. La comunidad organiza conferencias y también colabora con el código.

Contiene miles de módulos de terceros, a parte de la biblioteca estándar de Python, por lo que tenemos infinitas posibilidades.

Por último, Python se desarrolla bajo una licencia de código abierto aprobada por OSI, por lo que es se puede utilizar libremente y distribuir. La licencia de Python es administrada por la Python Software Foundation.

Podemos acceder a través de: https://www.python.org/

4.3. Spyder

Es un entorno de desarrollo interactivo para el lenguaje de Python, es de código abierto. Tiene funciones avanzadas de edición, pruebas interactivas, depuración e introspección. También es un entorno informático numérico [21].

Podemos acceder a través de: http://pythonhosted.org/spyder/

4.4. LaTeX

Se usa para la creación de documentos que necesiten una alta calidad tipográfica, como puede ser en artículos o libros científicos [14].

Podemos acceder a través de: https://www.latex-project.org/

Ventajas:

- Es de software libre.
- No te tienes preocupar por el diseño, ya que la herramienta se encarga de ello.

Desventajas:

- Si eres principiante necesitas un tiempo de aprendizaje para saber como funciona.

4.5. Jupyter Notebook

Es una aplicación web de código abierto, con él podemos crear y compartir documentos, que nos permiten visualizar los resultados al ejecutar nuestro código, ya sean imágenes, árboles..., que otros entornos de desarrollo (como Spyder mencionado anteriormente) no nos permiten esto. Soporta más lenguajes, pero nosotros lo usaremos para el lenguaje de Python [13].

Podemos acceder a través de: `http://jupyter.org/`

4.6. Scikit-learn

Es una librería de Python que contiene algoritmos de aprendizaje automático para problemas supervisados y no supervisados. Como esta basado en Python, puede integrarse fácilmente en aplicaciones que no suelen usarse para análisis de datos estdísticos. El trabajo futuro incluye aprendizaje en línea para escalar a grandes conjuntos de datos [18].

Podemos acceder a través de: `http://scikit-learn.org/stable/`

4.7. SonarQube

Es una plataforma que sirve para evaluar y analizar código. Es software libre, para llevar a cabo este análisis del código utiliza distintas herramientas como pueden ser Checkstyle, PMD o FindBugs, con dichas herramientas obtenemos métricas que nos ayudan a mejorar la calidad de nuestro código fuente [33].

Los aspectos que evalúa esta herramienta son:

- Technical Debt, esta parte nos indica los aspectos y métricas que no habíamos tenido en cuenta y también nos muestran la claridad del código. Una de las ventajas es que te indica donde has cometido una falta de estilo o donde tienes demasiada complejidad.
- La complejidad, los cambios de flujo que sufre el código, es decir, las condiciones if, while, for...
- Podemos ver las líneas de código que hemos escrito en cada fichero (sin contar los comentarios).
- Si queremos evaluar más aspectos, podemos instalar Plugins que nos lo permiten.

Podemos acceder a través de: https://www.sonarqube.org/

4.8. Graphviz

Es un software de visualización gráfica de código abierto. Es una forma de representar nuestra información estructural como diagramas de gráficos y redes abstractas. En nuestro caso lo utilizamos para dibujar un árbol de nuestro conjunto de datos entrenado. Su arquitectura consiste en un lenguaje de descripción de gráficos llamado DOT.

Podemos acceder a través de: https://www.graphviz.org/

4.9. Zenhub

Es un gestor de tareas es similiar a Trello https://trello.com/, tiene un modo pizarra en el que podemos ver los cambios. Una de las grandes ventajas es que podemos integrarlo desde GitHub, por lo que no es necesario el uso de una aplicación externa. Una pequeña desventaja podríamos decir que es que no se puede añadir código, pero como el repositorio de GitHub nos permite visualizar dicho código no es un gran problema.

Podemos acceder a través de: https://www.zenhub.com/

Aspectos relevantes del desarrollo del proyecto

5.1. Formación

El proyecto requería unos conocimientos técnicos de los que desconocía en un principio. Entre ellos, conocimientos sobre Minería de datos, Scikit-Learn y documentar en LaTeX. Para aprender estos conocimientos se ha necesitado leer documentos científicos, descubriendo la utilidad de ellos, ya que nunca había hecho uso de ellos.

Fue necesario instruirse en cómo implementar algoritmos, para ello tomamos de ejemplo, los algoritmos ya implementados en Scikit-Learn.

Otro de los conocimientos adquiridos en la realización de ese proyecto ha sido la librería Scikit-Learn, porque nosotros queremos implementar algoritmos en dicha librería, porque vamos a tratar la minería de datos, y como queremos clasificar un conjunto de datos para después de entrar poder predecir unos resultados con la mayor precisión posible, Sickit-Learn es una librería perfecta para esto.

5.2. Calidad del Software

Cualquier persona con pocos conocimientos en minería de datos o programación en Python, puede usarla ya que se han generado unos notebooks para ello, por lo que podremos ejecutarlos teniendo mínimos conocimientos en el tema.

Se ha conseguido una buena calidad de código, ya que se ha comprobado en SonarQube, y tenemos la mejor nota posible. Para tener código claro y fácil de entender se ha ido comentado los métodos/funciones para ver lo que hacer cada parte del código. Se ha seguido la guía de estilo de Python y Pep [10].

Se han realizado unos análisis con distintas medidas y distintos conjuntos de datos para cada uno de los algoritmos creados. Después de esto hemos podido ver que el algoritmo Rotation Forest, es mejor que los otros dos algoritmos, aunque Random Oracle le sigue muy de cerca, ya que en las distintas medidas que se han comprobado los resultados eran solo un poco peores. Respecto a Disturbing Neighbors, es el peor de los tres, ya que este si hay una diferencia notable en los resultados, respecto a los otros dos. Esto no quiere decir que siempre Distutbing Neighbors vaya ser peor, ya que se ha probado solo para algunos conjuntos de datos.

Trabajos relacionados

En este apartado vamos a analizar y comparar nuestro proyecto con otros relacionados. Lo dividiremos en tres partes, una en la que hablaremos de otras librerías ensembles, otra parte sobre librerías de aprendizaje automático y una última de servicios de computación en la nube.

6.1. Librerías con implementaciones de *ensembles*

A parte de de la librería que hemos usado de Scikit-Learn, existen librerías similares. A continuación, analizaremos algunas de ellas.

EnsembleSVM

Esta biblioteca de aprendizaje automático es de software libre. Esta librería nos ofrece la funcionalidad para llevar a cabo el aprendizaje de ensembles utilizando modelos base de la máquina de vectores de soporte (SVM). Permite entrenar eficientemente modelos para grandes conjuntos de datos [6].

H_2O *Ensemble*

Es una librería de R, que nos proporciona funcionalidad para crear conjuntos a partir de algoritmos básicos de aprendizaje que podemos acceder a través de este paquete. Este tipo de aprendizaje de ensembles se conoce como «súper aprendizaje». Este algoritmos aprende de la combinación óptima de los entrenamientos [15].

6.2. Librerías de aprendizaje automático Single-Label

Vamos a ver diferentes librerías de aprendizaje automático para Single-Label.

Weka

Es una colección de algoritmos de aprendizaje automático para tareas de minería de datos. Los algoritmos se pueden aplicar sobre un conjunto de datos o llamar desde su propio código de java. Contiene herramientas para el preprocesamiento de datos, clasificación, clustering, reglas de asociación y visualización. Es un software de código abierto con la licencia de GNU [35].

KEEL

Es una herramienta de software de código abierto de Java, que se puede utilizar para una gran cantidad de tareas de descubrimiento de datos de conocimiento diferentes. Proporciona una GUI simple basada en el flujo de datos para diseñar experimentos con diferentes conjuntos de datos y algoritmos de inteligencia computacional, con el fin de evaluar el comportamiento de los algoritmos. Contiene una amplia variedad de algoritmos clásicos de extracción de conocimiento, técnicas de preprocesamiento, algoritmos de aprendizaje basados en inteligencia computacional, modelos híbridos...

Permite realizar un análisis completo de nuevas propuestas de inteligencia computacional en comparación con las existentes [2].

TensorFlow

Es una librería de computación numérica que computa gradientes automáticamente. Es un sistema flexible y se puede utilizar para gran variedad de algoritmos, incluidos de entrenamiento e inferencia para modelos de redes neuronales. Se ha utilizado para realizar investigaciones y para implementar sistemas de aprendizaje automático en diferentes áreas [1].

Ha sido desarrollada por Google, y la utilizan empresas como Dropbox, Uber y Snapchat [4].

Pytorch

Es una biblioteca de aprendizaje de máquina de código abierto para Python que permite un crecimiento rápido de Deep Learning. Su mayor característica es que utiliza grafos computacionales dinámicos.

Ha sido desarrollado principalmente por el grupo de investigación de inteligencia artificial de Facebook, y por ejemplo el software «Pyro» de Uber para la programación probabilística se basa en él [7].

6.3. Librerías de aprendizaje automático Multi-Label

Vamos a ver diferentes librerías de aprendizaje automático para Multi-Label.

Meka

Proporciona una implementación de código abierto de métodos para aprendizaje y evaluación de Multi-Label. En la clasificación Multi-Label, queremos predecir múltiples variables de salida para cada instancia de entrada. Esto es diferente del caso estándar que involucra solo una única variable objetivo. MEKA se basa en Weka, incluye docenas de métodos Multi-Label de la literatura científica, y está relacionado con el framework de MULAN [22].

Mulan

Es una biblioteca de código abierto de Java para aprender de conjuntos de datos Multi-Label. Actualmente, esta biblioteca incluye una variedad de algoritmos para realizar unas tareas principales de aprendizaje Multi-Label [30]:

- Clasificación: Esta tarea se refiere a la salida de una bipartición de los labels relevantes e irrelevantes de una instancia de entrada.
- Ranking: Esta tarea es el orden de salida de los labels, de acuerdo con su relevancia para un elemento de datos.
- Clasificación y ranking: Mezcla de las dos tareas anteriores.

Esta biblioteca también nos ofrece dos características:

- Selección de características.
- Evaluación.

Scikit-Multilearn

Es una implementación de Python de una variedad de algoritmos de clasificación Multi-Label. Se implementa una clase de Sklearn, para dar acceso a todos los métodos disponibles de Meka, Mulan y Weka [28].

Mldr

Esta librería hace un análisis exploratorio de datos y funciones de manipulación para conjuntos de datos Multi-Label, para ello usa una aplicación interactiva llamada Shiny para facilitar si uso [5]. Pertenece al lenguaje de R.

6.4. Servicios de computación en la nube

A continuación se exponen distintos servicios de computación en la nube que nos sirven para procesar conjuntos de datos.

Google Cloud Dataproc

Es un servicio de Apache Hadoop, Spark, Pig y Hive para procesar grandes conjuntos de datos con poco esfuerzo y bajo coste. Se pueden crear clústeres y desactivarlos cuando acabemos para controlar los gastos.

Podemos cambiar el tamaño de los clústeres en cualquier momento. Cada acción de clúster tarda muy poco tiempo, así podemos dedicar más tiempo a información valiosa porque necesitamos menos tiempo revisando la infraestructura.

Más información en https://cloud.google.com/dataproc/?hl=es.

Azure

Microsoft Azure Machine Learning Studio es una herramienta que se puede usar para crear, probar y desplegar soluciones de análisis predictivo en sus datos. Machine Learning Studio publica modelos como servicios web que pueden ser ejecutados fácilmente por aplicaciones personalizadas o herramientas como Excel.

Más información en https://docs.microsoft.com/en-us/azure/machine-learning/studio/what-is-ml-studio.

AWS

En AWS muchos de sus sistemas internos están basados en algoritmos de aprendizaje automático. Nos ofrece las siguientes características:

- Servicios ML basados en API.
- Amplio soporte de framework.
- Amplitud de operaciones de computo.
- Análisis completo.
- Seguridad

Más información en https://aws.amazon.com/es/machine-learning/.

Conclusiones y Líneas de trabajo futuras

En esta sección, explicaremos las conclusiones a las que hemos llegado después de realizar el proyecto. También mencionaremos partes que se pueden mejorar en un futuro por desarrolladores.

7.1. Conclusiones

Este apartado lo vamos a dividir en distintas partes, cada una con una conclusión. Con los aspectos más relevantes que hemos realizado.

- Dinámica del proyecto: Estoy muy satisfecho con los aspectos que se han llevado a cabo. Algunas de las partes han supuesto un reto, por el desconocimiento del tema. Al conseguir completarlos han supuesto una satisfacción.
- Lenguaje de desarrollo: Se ha utilizado el lenguaje de Python, y hemos visto que para realizar cálculos, necesarios para nuestros algoritmos, dispone de una gran cantidad de librerías fáciles de usar, por ello para este proyecto Python es más eficaz que otro lenguaje.
- Aprendizaje: Este proyecto me ha servido para profundizar más en el aprendizaje de la minería de datos, y todo lo relacionado con la captura, administración y procesamiento de grandes cantidades de datos. En cuanto al lenguaje elegido, Python que mayores conocimientos he adquirido de la carrera, y por lo tanto he trabajado con mayor agilidad durante el desarrollo; me ayudado ha ampliar y afianzar los conocimientos de este lenguaje.

7.2. Lineas de trabajo futuras

Este apartado vamos a mencionar diferentes mejoras para el proyecto.

- Algunos de los métodos/funciones, se podrían simplificar su complejidad algorítmica, con el objetivo de agilizarlos.
- Se podría implementar los algoritmos en otros lenguajes, que estén preparados para realizar cálculos, ya que sino se ralentizaría mucho el análisis de los datos. Uno de los lenguajes aconsejables sería `R`.
- Ya que no se ha conseguido hacer un notebook en el que se muestren las gráficas Multi-Label de los ensembles es otro de los mejoras que se pueden llevar a cabo.

Bibliografía

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] J Alcalá-Fdez, L Sánchez, and S García. Keel: A software tool to assess evolutionary algorithms to data mining problems. *URL http://www. keel. es*, 2012.

[3] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[4] Nicole Chapaval. Cuatro librerías de machine learning: Tensorflow, scikit-learn, pytorch y keras, 2017. https://platzi.com/blog/librerias-de-machine-learning-tensorflow-scikit-learn-pythorch-y-keras/.

[5] Francisco Charte and David Charte. Working with multilabel datasets in R: The mldr package. *The R Journal*, 7(2):149–162, dec 2015. http://journal.r-project.org/archive/2015-2/charte-charte.pdf.

[6] Marc Claesen, Frank De Smet, Johan A.K. Suykens, and Bart De Moor. Ensemblesvm: A library for ensemble learning using support vector machines. *Journal of Machine Learning Research*, 15:141–145, 2014.

[7] Wikipedia contributors. Pytorch — wikipedia, the free encyclopedia, 2017. https://en.wikipedia.org/w/index.php?title=PyTorch&oldid=817120667.

[8] Yoav Freund and Robert E Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[9] Andrés González. Conceptos básicos de machine learning, 2015. http://cleverdata.io/conceptos-basicos-machine-learning/.

[10] Nick Coghlan Guido van Rossum, Barry Warsaw. Style guide for python code, 2013. https://www.python.org/dev/peps/pep-0008/.

[11] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques*. Elsevier, 2011.

[12] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, 20(8):832–844, 1998.

[13] jupyter. Jupyter-notebook, 2017. http://jupyter.org/.

[14] latex project. Latex, 2017. https://www.latex-project.org/about/.

[15] Erin LeDell. H2o ensemble, 2017. https://github.com/h2oai/h2o-3/tree/master/h2o-r/ensemble.

[16] Jesús Maudes, Juan José Rodríguez, and César Ignacio García-Osorio. Disturbing neighbors ensembles for linear svm. In *MCS*, pages 191–200. Springer, 2009.

[17] Carlos Paramio. Github, 2011. https://www.genbetadev.com/sistemas-de-control-de-versiones/conociendo-github-el-servicio-donde-alojar-tus-repositorios-git-como-el-nuestro.

[18] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[19] Julio Antonio Hernández Pérez, Raudel Hernández León, and Autor. Clasificación multi-etiquetas basada en reglas de asociación de clases. *Multi-Label*, 2017.

[20] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[21] pythonhosted. Spyder, 2017. https://pythonhosted.org/spyder/.

[22] Jesse Read, Peter Reutemann, Bernhard Pfahringer, and Geoff Holmes. Meka: A multi-label/multi-target extension to weka. *Journal of Machine Learning Research*, 17(21):1–5, 2016.

[23] José Cristóbal Riquelme Santos, Roberto Ruiz, and Karina Gilbert. Mineria de datos: Conceptos y tendencias. *Inteligencia artificial: Revista Iberoamericana de Inteligencia Artificial*, 10(29):11–18, 2006.

[24] Juan Rodríguez and Ludmila Kuncheva. Naïve bayes ensembles with a random oracle. *Multiple Classifier Systems*, pages 450–458, 2007.

[25] Juan José Rodriguez, Ludmila I Kuncheva, and Carlos J Alonso. Rotation forest: A new classifier ensemble method. *IEEE transactions on pattern analysis and machine intelligence*, 28(10):1619–1630, 2006.

[26] scikit learn. Ensemble, 2017. http://scikit-learn.org/stable/modules/ensemble.html.

[27] sinnexus. Minería de datos, 2016. http://www.sinnexus.com/business_intelligence/datamining.aspx.

[28] Piotr Szymanski. A scikit-based python environment for performing multi-label classification. *arXiv preprint arXiv:1702.01460*, 2017.

[29] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Mining multi-label data. In *Data mining and knowledge discovery handbook*, pages 667–685. Springer, 2009.

[30] Grigorios Tsoumakas, Eleftherios Spyromitros-Xioufis, Jozef Vilcek, and Ioannis Vlahavas. Mulan: A java library for multi-label learning. *Journal of Machine Learning Research*, 12:2411–2414, 2011.

[31] Guido Van Rossum and Fred L Drake. *Python language reference manual*. Network Theory, 2003.

[32] Wikipedia. Minería de datos — wikipedia, la enciclopedia libre, 2017. https://es.wikipedia.org/w/index.php?title=Miner%C3%ADa_de_datos&oldid=102214612.

[33] Wikipedia. Sonarqube — wikipedia, la enciclopedia libre, 2017. https://es.wikipedia.org/w/index.php?title=SonarQube&oldid=98980020.

[34] Wikipedia. Python — wikipedia, la enciclopedia libre, 2018. https://es.wikipedia.org/w/index.php?title=Python&oldid=105012532.

[35] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.