UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

TFG del Grado en Ingeniería Informática

Extracción y procesamiento de datos de Amazon para su utilización en un estudio de marketing.

Presentado por Daniel Arnaiz Gutierrez
en Universidad de Burgos — 18 de junio de 2019
Tutores: Dr. José Francisco Díez Pastor
Dr. César Ignacio García Osorio

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

Dr. José Francisco Díez Pastor y Dr. César Ignacio García Osorio, profesores del departamento de Ingeniería Civil, área de Lenguajes y sistemas informáticos.

Exponen:

Que el alumno D. Daniel Arnaiz Gutierrez, con DNI 71296333, ha realizado el Trabajo final de Grado en Ingeniería Informática titulado «Extracción y procesamiento de datos de Amazon para su utilización en un estudio de marketing».

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, 18 de junio de 2019

Vº. Bº. del tutor:	Vº. Bº. del tutor:
D. José Francisco Díez Pastor	D. César Ignacio García Osorio

Resumen

El principal objetivo de este proyecto es la recolección de datos de determinados productos del sitio web «Amazon.com» para después procesar y almacenar los resultados de forma relacional. Además, parte de este proceso consiste en, dada una imagen del producto, etiquetarla de forma automática según las características de esta.

Para realizar la parte de la recolección de datos se utilizará una técnica de minería de datos llamada «web scraping» y para el procesado y etiquetado de las imágenes se harán uso de redes neuronales, otra rama de la minería de datos.

Los resultados, tanto del *scraper* como del clasificador de imágenes, serán almacenados en una base de datos para su posterior uso en estudios de marketing.

Descriptores

Minería de datos, *Web scraping*, Clasificador, Base de datos

Abstract

A **brief** presentation of the topic addressed in the project.

Keywords

keywords separated by commas.

Índice general

Índice de figuras

Índice de tablas

Introducción

Uno de los principales problemas que se encuentran a la hora de hacer un estudio de mercado es el conjunto de datos sobre el que trabajar. Existen muchas maneras de recolectar esos datos como podrían ser encuestas, sondeos, entrevistas o incluso observaciones directas. Es aquí donde entra el proceso de recolección de datos usado en este proyecto: *Web scraping*. Este concepto se puede definir como una técnica de obtención de datos de sitios web de forma automatizada, forma parte de un concepto que lo engloba llamado *Data mining* o Minería de datos.

Este proceso conlleva una serie de ventajas sobre la recolección de datos de forma manual:

- Una de las principales razones por la que utilizar un *web scraper* es la automatización, dado que siempre vas a querer extraer los mismos campos o características de un sitio web, este proceso se convierte rápidamente en repetitivo y aburrido. Un *web scraper* es capaz de realizar estas acciones de forma relativamente autónoma y mucho más rápido de lo que sería hacerlo de forma manual.
- Otra ventaja importante de utilizar un *web scraper* sobre la extracción de datos de forma manual sería el almacenamiento de la información extraída. Aunque se podrían llegar a almacenar de la misma forma los datos extraídos de ambas formas, con un *web scraper* tienes la posibilidad de personalizar el modo en que se almacenen los datos, de tal forma que sea una propia extensión del proceso de extracción además de poder guardarlo en varios formatos a la vez sin apenas alargar la duración del proceso.

Una vez que se dispone de la información necesaria para iniciar el estudio, es probable que se necesite de un procesamiento de esta información para clasificar o identificar los datos deseados. Existen infinidad de procesamientos

de datos posibles, desde procesamiento de textos como *Text sentiment analysis*[1] o categorización, a procesamiento de imágenes como se ha hecho en este proyecto. En este caso, se ha hecho una clasificación automática de imágenes para más tarde extraer información de éstas dependiendo de la categoría en la que han sido clasificadas.

[1] Proceso automático de identificación y extracción de información subjetiva de un texto.

Objetivos del proyecto

Este apartado explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se puede distinguir entre los objetivos marcados por los requisitos del software a construir y los objetivos de carácter técnico que plantea a la hora de llevar a la práctica el proyecto.

2.1. Objetivos de carácter general

El principal objetivo de este proyecto es la creación de varias herramientas capaces de extraer y procesar información sobre un familia de productos concreta de Amazon, en este caso camisetas, para su posterior uso en estudio de mercado.

A su vez este proceso está divido en diferentes fases que podemos identificar como objetivos secundarios:

- Ser capaz de extraer información de los distintos productos. Esta información incluye campos como el nombre de la marca, el rango de precios, número de valoraciones o imágenes del producto. Este objetivo se realiza con técnicas de *web scraping*.
- Como complemento al anterior objetivo, otro objetivo es la implementación de un clasificador automático de imágenes. Este clasificador etiquetará cada imagen en función de si aparece un modelo o no, y si la cara del modelo es visible o no. Esto se consigue con técnicas de inteligencia artificial y redes neuronales, y más concretamente con el uso de *deep learning*.
- Otro de los principales objetivos del proyecto es el almacenamiento de la información descargada y de las imágenes clasificadas. Para cumplir con este objetivo se hará uso de un gestor de bases de datos y se almacenará la información en múltiples tablas con sus respectivos campos.

- Por último, un objetivo impuesto para la realización de este proyecto ha sido la visualización de la información descargada y procesada en un documento Excel.

2.2. Objetivos de carácter técnico

A continuación se listan los objetivos técnicos utilizados para la correcta implementación de los objetivos de carácter mas generales expuestos anteriormente.

- Utilizar Python como lenguaje de programación para la totalidad del proyecto.
- Hacer uso de la herramienta Scrapy para implementar todo lo relacionado con la extracción de datos y *Web scraping*.
- Un objetivo técnico importante y necesario para la creación del clasificador automático es la creación de un *dataset* o conjunto de datos personalizado de forma manual para posteriormente entrenar el clasificador. Para ello se ha usado la herramienta Dataturks[2].
- Usar JSON como forma de almacenamiento temporal de los productos descargados debido la simplicidad de su estructura.
- Utilizar SQLite[3] como gestor de base de datos para el almacenamiento final de los productos descargados.
- Implementar el clasificador de imágenes con la ayuda de la librería Keras[4] y de Google Colab[5].
- Utilizar el servicio de GitHub[6] para hacer uso de un control de versiones del proyecto.
- Seguir una metodología ágil para la el desarrollo y el enfoque del proyecto. Para esto se ha usado la herramienta ZenHub[7] como una extensión de GitHub.

[2] https://dataturks.com/
[3] https://www.sqlite.org
[4] https://keras.io/
[5] https://colab.research.google.com/
[6] https://github.com/
[7] https://www.zenhub.com/

Conceptos teóricos

3.1. Inteligencia artificial

Un concepto tan amplio como lo es la inteligencia artificial se puede definir de múltiples formas, a continuación se citan algunas:

1. "La habilidad, como sistema, de interpretar correctamente datos externos, aprender de estos datos, y usar lo aprendido para lograr objetivos y tareas concretas a través de una adaptación flexible" [3].

2. "... la habilidad de un sistema de actuar adecuadamente en un ecosistema incierto en el que la acción apropiada es la que aumenta la probabilidad de alcanzar el objetivo, siendo el objetivo completar una serie de logros menores que ayuden a alcanzar el objetivo final" [1].

3. "Estudio de del diseño de agentes inteligentes" [4].

Historia de la inteligencia artificial

Durante la segunda guerra mundial, el aclamado científico Alan Turing descifró el «Código Enigma» usado por las fuerzas alemanas para mandar mensajes de forma segura. Alan Turing y Enigma conforman los pilares de la inteligencia artificial y el *machine learning*.

En 1956, el científico John McCarthhy[8], usó el término inteligencia artificial por primera vez en la Conferencia Dartmouth[9], dando lugar a la creación de varios centros de investigación en Estados Unidos con el propósito del estudio del potencial de la inteligencia artificial. Los investigadores Allen Newell[10]

[8] https://es.wikipedia.org/wiki/John_McCarthy
[9] https://es.wikipedia.org/wiki/Conferencia_de_Dartmouth
[10] https://es.wikipedia.org/wiki/Allen_Newell

y Herbert Simon[11] fueron clave para impulsar inteligencia artificial como un nuevo campo en la Informática que podría cambiar el mundo.

A partir de los años 60, el principal enfoque de los investigadores se centró en desarrollar algoritmos para resolver problemas y teoremas matemáticos. Este proceso dio lugar a los primeros conceptos sobre *machine vision learning*[12] y inteligencia artificial aplicada a la robótica.

Tras un largo periodo sin avances notables debido a una escasez de financiaciones para el desarrollo de la inteligencia artificial, a finales de los años 90, el desarrollo resurge gracias a grandes corporaciones interesadas en usar este campo para lograr unas ventajas y beneficios nunca vistos, Es así como en 1997 la máquina Deep Blue[13] de IBM se convirtió en la primera computadora capaz de derrotar a un campeón mundial de ajedrez como lo fue Garry Kasparov.

En los ultimos 15 años, las principales empresas mundiales han utilizado la inteligencia artificial como una herramienta más a la hora de crecer y convertirse en el sector más importante del mundo, gracias a campos como la visión artificial, minería de datos o el procesamiento de lenguajes naturales [5].

Machine learning (Aprendizaje automático)

El *Machine learning* se considera un campo de las ciencias de computación y a su vez una rama de la inteligencia artificial. Tiene como objetivo principal el desarrollo de técnicas y la búsqueda de algoritmos y heurísticas que permitan que las computadoras aprendan por si mismas a través de la experiencia.

Las aplicaciones que conforman este campo son muy variadas, desde su uso en diagnósticos médicos, análisis de patrones, reconocimiento de lenguajes naturales hablados y escritos, a juegos o robótica [6].

A su vez los diferentes algoritmos de aprendizaje automático se pueden agrupar por tipos según la salida de los mismos. Encontramos así:

Aprendizaje supervisado Produce una función que establece una correspondencia entre las entradas y salidas deseadas. Un posible uso de estos algoritmos podría ser un problema de clasificación en el que hay que etiquetar una serie de valores usando una entre varias categorías. En este tipo de aprendizaje, la base del conocimiento estaría formada por ejemplos de etiquetados anteriores.

Aprendizaje no supervisado En este caso el proceso se lleva a cabo por un conjunto de datos únicamente formado por entradas al sistema que no

[11] https://es.wikipedia.org/wiki/Herbert_Alexander_Simon
[12] https://es.wikipedia.org/wiki/Vision_artificial
[13] https://es.wikipedia.org/wiki/Deep_Blue_(computadora)

contienen información sobre las categorías de cada ejemplo. El sistema debe reconocer los patrones de forma autónoma para encontrar grupos de elementos comunes y asignar las nuevas entradas a dichos grupos.

Aprendizaje semi-supervisado Estos algoritmos combinan los dos anteriores para clasificar teniendo en cuenta los datos clasificados y los no clasificados.

Aprendizaje por refuerzo En este tipo de algoritmos, el sistema aprende observando lo que le rodea, es decir, su información de entrada se basa en la retroalimentación que obtiene como respuesta a sus acciones. Se podría decir que aprende a base de prueba-error.

Transducción Un algoritmo similar al aprendizaje supervisado tratado anteriormente. La principal diferencia sería que no construye de forma explícita una función. Un posible ejemplo de este tipo de algoritmos sería el k de vecinos mas cercanos[14].

Aprendizaje multi-tarea Los métodos incluidos en esta categoría se basan en la utilización de el conocimiento aprendido previamente para enfrentarse a problemas similares a los ya vistos.

3.2. Redes neuronales

Una Red Neuronal Artificial (RNA) se basa en un modelo inspirado en el comportamiento de las neuronas y como se organizan para crear una estructura en el cerebro.

El cerebro se puede llegar a ver como un sistema inteligente, aunque de manera distinta a como trabajan los ordenadores hoy en día. Si bien las computadoras actuales son muy rápidas y eficientes en el procesamiento de información, existen otro tipo de tareas y procesos mas complejos que requieren más tiempo y más esfuerzo computacional que a un cerebro humano. Este tipo de tareas están relacionadas con el reconocimiento y clasificación de patrones [2].

Neurona artificial

El primer modelo de neurona artificial, con el objetivo de llevar a cabo tareas muy simples, fue creado por el neuroanatomista Warren McCulloch y el matemático Walter Pitts y consta de las siguientes partes:

- Conjunto de entradas x_1, x_2, ... x_n
- Pesos sinápticos w_1, w_2, ... w_n, correspondientes a cada entrada

[14] https://es.wikipedia.org/wiki/K_vecinos_mas_proximos

- Una función de agregación, $\sum$
- Una función de activación, f
- Una salida, Y

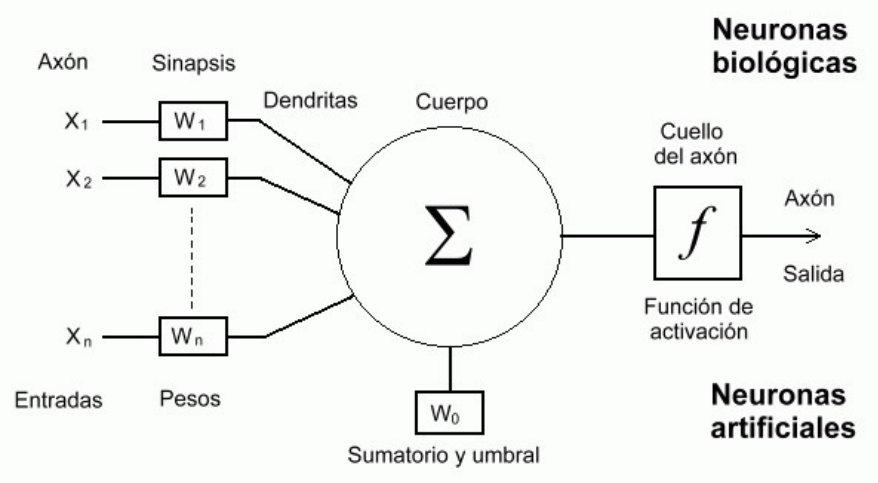

Figura 3.1: Diferenciación entre una neurona biológica y una neurona artificial

Perceptrón

Un perceptrón es el modelo más simple de una neurona artificial, está basado en una única neurona y posee una única salida. El perceptrón puede combinarse con otros tipos de perceptrones o neuronas artificiales para formar una red neuronal mas compleja.

A su vez, un perceptrón puede organizar sus neuronas en capas: capa de entrada, una capa de salida, y múltiples capas ocultas formadas por neuronas. En el caso de existir capas ocultas nos referimos a la red neuronal como perceptrón multicapa[15]

[15] https://es.wikipedia.org/wiki/Perceptron_multicapa

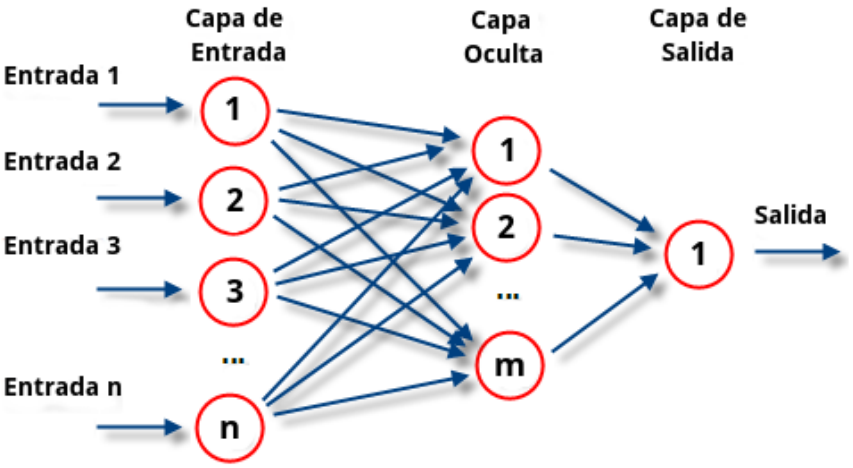

Figura 3.2: Diagrama de un perceptrón multicapa

3.3. Minería de datos

La minería de datos se considera un campo de la estadística y de las ciencias de la computación. El proceso de la minería de datos se basa en el descubrimiento de patrones en grandes conjuntos de datos utilizando métodos de inteligencia artificial, aprendizaje automático, estadística y bases de datos. El principal objetivo es el de extraer información de un conjunto de datos y procesarlo para transformarlo de forma estructurada y comprensible para su uso posterior [10].

El proceso típico de minería de datos está constituido por los siguientes pasos generales:

Selección del *dataset* o conjunto de datos Selección de los campos que se quieren predecir y de los necesarios para el proceso.

Análisis de las propiedades de los datos Comprobación de histogramas, diagramas de dispersión y búsqueda de valores atípicos o nulos.

Transformación del conjunto de datos de entrada Preparación de los datos obtenidos para aplicar la técnica de minería de datos que mejor se adapte al problema a resolver.

Selección y aplicación de la técnica de minería de datos Construcción del modelo a aplicar: predictivo, o de clasificación.

Extracción de datos Proceso por el que se recopilan los datos a ser tratados, ya sea registrando las medidas de sensores, consultando bases de datos, o como en este proyecto, realizando tareas de *web scraping*.

Interpretación y evaluación Una vez obtenido el modelo, se procede a su validación y evaluación.

Técnicas de minería de datos

Redes neuronales Tal y como se han descrito anteriormente, esta técnica de procesamiento automático y los diferentes tipos de red neuronal (perceptrón y perceptrón multicapa) son una técnica utilizada en la minería de datos.

Regresión lineal Técnica utilizada para identificar relaciones lineales entre dos variables.

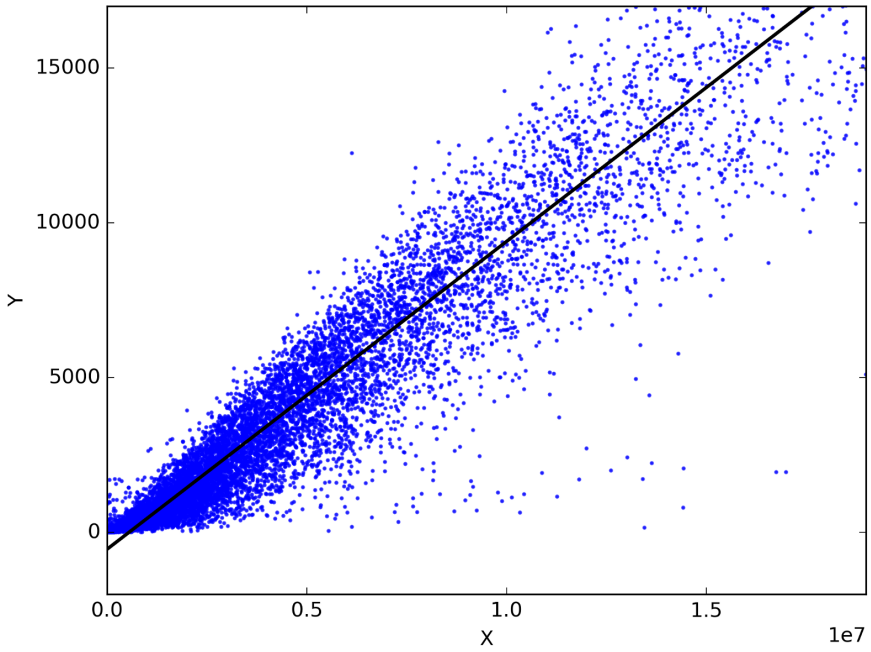

Figura 3.3: Ejemplo de diagrama de regresión lineal

Árboles de decisión Modelo de predicción utilizado en el campo de la inteligencia artificial y el análisis predictivo[16] donde a partir de un conjunto de datos se construyen diagramas para representar y categorizar condiciones de forma sucesiva hasta resolver el problema dado.

[16] https://es.wikipedia.org/wiki/Analisis_predictivo

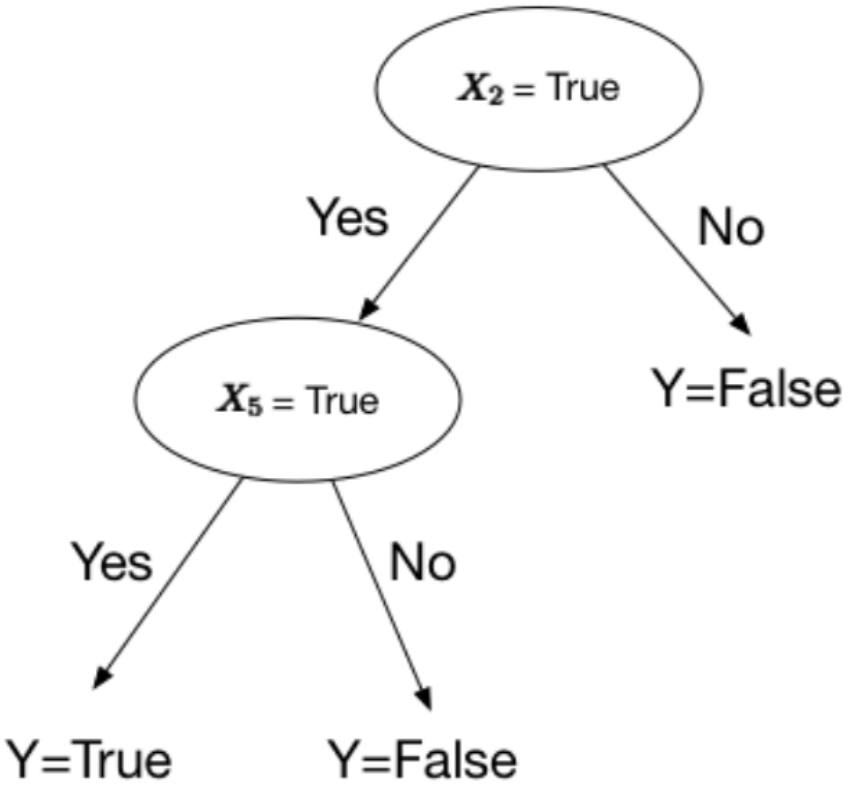

Figura 3.4: Ejemplo de árbol de decisión

Modelos estadísticos Expresión o ecuación utilizada con el fin de indicar diferentes factores que modifican la variable de una respuesta.

Clustering Proceso de agrupación de una series de valores según un criterio de distancia o similitud.

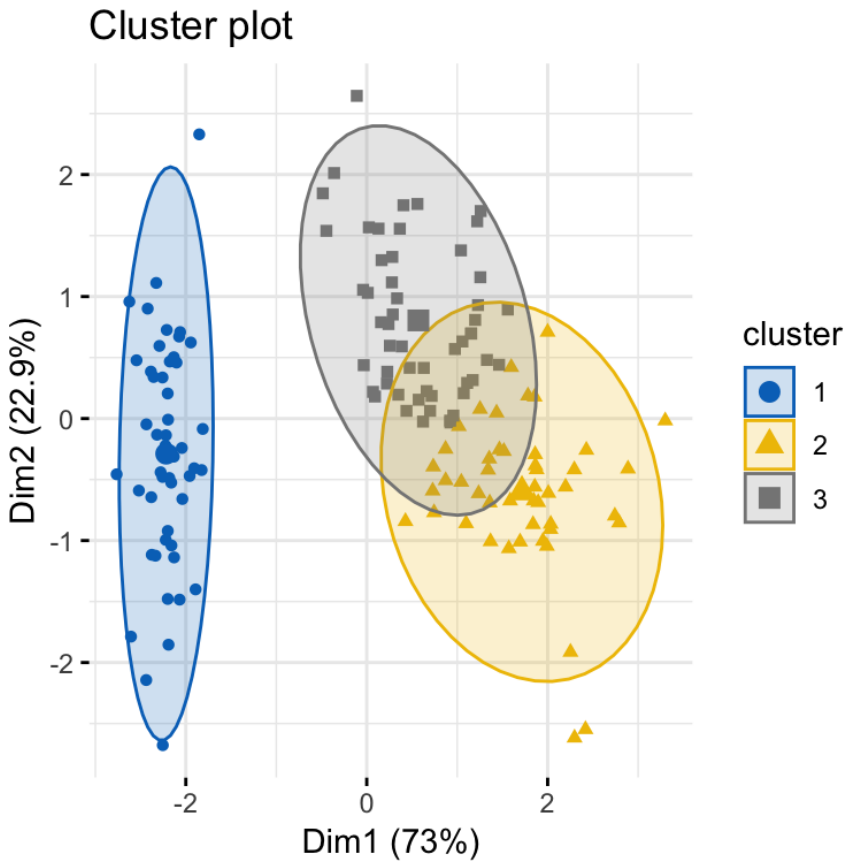

Figura 3.5: Ejemplo de diagrama mostrando *clustering*

Reglas de asociación Procedimiento utilizado para reconocer hechos en común dentro de un conjunto de datos.

3.4. Web Scraping

El web *scraping* es un a técnica utilizada para extraer información desde páginas web a través de uno o varias herramientas de software, estos programas simulan la navegación por el sitio web a la hora de extraer los datos deseados. La técnica de web *scraping* es muy utilizada a día de hoy por servicios como motores de búsqueda ya que es necesario para indexar la web, sin embargo, los principales usos de esta técnica son la transformación de datos sin estructurar que podemos encontrar en cualquier web en formato HTML[17] a estructuras de datos que pueden ser almacenados en bases de datos, hojas de cálculo, o en cualquier forma de almacenamiento estructurada [16].

[17] https://es.wikipedia.org/wiki/HTML

En cuanto a las técnicas utilizadas para realizar *web scraping* sobre un sitio web encontramos varios tipos en función del nivel de automatización del proceso:

Expresiones regulares Es posible extraer información de una página web utilizando expresiones regulares.

Peticiones HTTP Se basa en extraer información por medio de peticiones HTTP al servidor remoto utilizando sockets.

Algoritmos de minería de datos Aplicación de algunas de las técnicas de minería de datos tratadas en el apartado anterior.

Parser HTML Existen lenguajes capaces de procesar documentos HTML y extraer de forma estructurada todo el contenido de una web.

Aplicaciones de web scraping Programas completos de web scraping que combinan algunas de las técnicas anteriores para ofrecer un proceso personalizado para cada caso, facilitando y simplificando la obtención del código de la aplicación de obtención de datos a partir de páginas web.

Técnicas y herramientas

Se han utilizado diferentes herramientas para la realización de este proyecto, a continuación se listan con una pequeña descripción de cada una:

4.1. Lenguajes y bibliotecas de programación

Python

Python[18] es un lenguaje de programación nacido en los años 90 y ampliamente utilizado en el presente. Se trata de un lenguaje interpretado, dinámicamente tipado y multiplataforma. Soporta programación orientada a objetos y sigue una filosofía de legibilidad y transparencia [12].

Ha sido el lenguaje utilizado en la mayor parte de este proyecto, desde la parte de *web scraping* a la clasificación de imágenes.

Scrapy

Scrapy[19] es una herramienta gratuita y de código abierto escrita en Python por Scrapinghub Ltd. y originalmente diseñada para *web scraping* aunque también utilizado para extraer datos mediante APIs [13].

Scrapy está compuesto por una serie de características que lo hacen la herramienta más adecuada para este proyecto. Entre ellas encontramos:

- Soporte para extraer y seleccionar datos de fuentes HTML/XML usando selectores CSS[20] y expresiones XPath[21].

[18] https://www.python.org/
[19] https://scrapy.org/
[20] https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
[21] https://es.wikipedia.org/wiki/XPath

- Soporte para exportar los datos extraídos en múltiples formatos tales como JSON[22], CSV[23] y XML[24].
- Fuerte extensibilidad, ya que permite conectar funcionalidades externas tales como bases de datos.

Keras

Keras[25] se trata de una librería de redes neuronales de código abierto escrita en Python por François Chollet y capaz de ser ejecutada sobre TensorFlow. Las principales características que lo componen son su modularidad, extensibilidad y el ser amigable con el usuario. Además del soporte para las redes neuronales estándar, Keras ofrece soporte para las Redes Neuronales Convolucionales y Redes Neuronales Recurrentes [8].

OpenCV

OpenCV[26] es una biblioteca multiplataforma originalmente desarrollada por Intel y orientada a la visión artificial. Nace en 1999 y actualmente es utilizada en infinidad de aplicaciones. Contiene más de 500 funciones abarcando una gran gama de áreas en el proceso de visión, como reconocimiento de objetos (reconocimiento facial), calibración de cámaras y visión robótica [11].

SQLite

SQLite[27] es un sistema de gestión de bases de datos relacional multiplataforma que forma parte de una biblioteca escrita en C por Richard Hipp. Su principal característica es que la base de datos se almacena en un único archivo.

4.2. Herramientas de desarrollo

Visual Studio Code

Visual Studio Code es un editor de código desarrollado por Microsoft anunciado en 2015 y lanzado al público en 2016. Cuenta con soporte para multitud

[22] https://es.wikipedia.org/wiki/JSON
[23] https://es.wikipedia.org/wiki/Valores_separados_por_comas
[24] https://es.wikipedia.org/wiki/Extensible_Markup_Language
[25] https://keras.io/
[26] https://opencv.org/
[27] https://www.sqlite.org/index.html

de lenguajes de programación y entre sus características mas destacadas encontramos soporte para depuración, control integrado de Git[28], resaltado de sintaxis, refactorización de código y finalización inteligente de código [15].

Este editor ha sido utilizado para escribir la mayoría de código que encontramos en el proyecto.

Google Colab (Colaboratory)

Google Colab[29] es un entorno de Jupyter Notebook[30] ya configurado y completamente ejecutado en remoto. Permite editar y ejecutar código, guardar y compartir proyectos y tener acceso a potentes recursos desde un navegador web.

Esta herramienta ha sido usada a la hora de desarrollar y más tarde entrenar el clasificador de imágenes utilizado en el proyecto.

4.3. Herramientas de gestión de código

Git

GitHub

GitHub[31] es una plataforma de desarrollo colaborativo utilizado para alojar proyectos y utilizar el sistema de control de versiones Git [7].

ZenHub

4.4. Herramientas de documentación

LaTeX

LaTeX[32] es un sistema de composición de textos orientado a la creación de documentos con una alta calidad tipográfica como podrían ser artículos científicos u otros textos con, por ejemplo, expresiones matemáticas.

Se trata de un software libre basado en un conjunto de macros de TeX, que a su vez se trata de un lenguaje escrito por Leslie Lamport en 1984 [9].

[28] https://git-scm.com/
[29] https://colab.research.google.com
[30] https://jupyter.org/
[31] https://github.com/
[32] https://es.wikipedia.org/wiki/LaTeX

Overleaf

4.5. Otras técnicas y herramientas

JSON

JSON (JavaScript Object Notation) es un formato de texto utilizado para el intercambio de datos que desde el año 2019 es considerado como un formato independiente de lenguaje. La principal característica de este lenguaje y la razón por la que se ha utilizado en este proyecto es la facilidad de extraer los datos que lo conforman con ayuda de *parsers* o analizadores sintácticos.

Dataturks

Dataturks[33] es una herramienta web cuya finalidad es anotar imágenes de forma manual con el propósito de crear conjuntos de entrenamiento para modelos de redes neuronales como Keras o TensorFlow[34].

En este proyecto, esta herramienta ha sido indispensable para crear el modelo que clasifica automáticamente las imágenes de cada producto.

[33] https://dataturks.com/
[34] https://www.tensorflow.org/

Aspectos relevantes del desarrollo del proyecto

Este apartado se recogen los aspectos más interesantes del desarrollo del proyecto, el ciclo de vida utilizado y los detalles de mayor relevancia de las fases de análisis, diseño e implementación.

5.1. Metodologías aplicadas

Para la realización de este proyecto se ha decidido seguir una metodología ágil siempre que ha sido posible. Por ello, se han seguido ciertas líneas de trabajo. A continuación se listan:

- Se ha dividido el desarrollo del proyecto en múltiples *sprints*.
- Se han ido proponiendo nuevos requisitos y objetivos a medida que se iban cumpliendo los anteriores.
- Se han realizado reuniones entre todos los integrantes al acabar cada *sprint*.
- Al acabar cada *sprint* se ha realizado una demostración siempre que los cambios eran sustanciales.
- Se han organizado las diferentes tareas y objetivos a completar en un tablero Kanban[35].
- Se ha asociado una duración estimada a cada tarea antes de realizarla.

[35] https://kanbantool.com/es/tablero-kanban

5.2. Propuesta del proyecto

La idea de este proyecto nace con el objetivo de crear una herramienta que ayude a obtener y descargar información sobre una familia de productos concreta de la web de compras por internet *Amazon.com* con el objetivo de hacer múltiples estudios sobre los diferentes conjuntos de datos obtenidos.

Desde un primer momento se ha querido obtener la mayor cantidad de campos de información, es por ello que una vez el proceso ha sido completado, obtenemos la siguiente información de cada producto:

ASIN: Código identificativo de cada producto.

Sexo: Sexo al que va dirigido cada artículo.

Rango de precios: El precio de cada producto puede variar en función del color o la talla. Este campo recoge el precio máximo y mínimo de cada producto.

Marca: El nombre de la marca de cada artículo.

Descripción: Texto descriptivo que el vendedor ha usado para cada producto.

Puntuación: Media de las puntuaciones que los clientes han votado a cada producto. Va del 1 al 5.

Número de valoraciones: Número de clientes que han valorado el producto.

Comentarios sobre el producto: Comentarios completos sobre un producto en el que los compradores dejan su opinión. El número de comentarios a extraer debe ser definido al comenzar el proceso.

URL: Dirección web de cada producto

Imágenes: Conjunto de imágenes asociadas a cada artículo.

Contenido de la imagen principal: Este campo describe el contenido de la imagen principal de cada producto en función de si aparece un modelo o no, y si la cara del modelo es visible o no.

La mayoría de estos campos se pueden extraer directamente del código *HTML* de la web de cada producto, pero para ciertos casos es necesario procesar antes la información ya extraída y usarla para crear los campos que faltan. Como es el caso del campo que recoge el contenido de la imagen principal.

A continuación se irán presentando y detallando las diferentes fases de las que se compone el proceso de extracción y procesamiento de datos.

5.3. Extracción de los datos

La herramienta utilizada en este proyecto para realizar el *web scraping* ha sido Scrapy. Se han barajado otras opciones como BeautifulSoup[36] o Selenium[37].

Las principales diferencias y ventajas que lo separan del resto son:

- Es más rápido ya que trabaja de forma asíncrona.
- Tiene mejor soporte para parseado de html.
- Soporta caracteres unicode.
- Soporta redirecciones http.
- Exportación directa de los resultados a JSON, CSV o XML.
- Se puede usar junto a otras librerías como urllib[38] o BeautifulSoup.

Arquitectura de Scrapy

En cuanto a la arquitectura de Scrapy, hay que destacar los tres principales bloques que lo conforman:

***Items*:** Los diferentes elementos o campos a extraer.

***Spiders*:** Clases escritas por el usuario que contienen los diferentes procedimientos para parsear y extraer los *items* de una o varial urls. En este caso hemos hecho uso de 3 *spiders*: Una para obtener las url de cada artículo, otra para extraer los comentarios, y la última para el resto de campos.

***Pipelines*:** Apartado responsable de procesar los *items* una vez han sido extraidos por los *spiders*. En este proyecto se ha usado para almacenar los campos extraídos en una base de datos.

[36] https://www.crummy.com/software/BeautifulSoup/bs4/doc/
[37] https://www.seleniumhq.org/
[38] https://docs.python.org/3/library/urllib.html

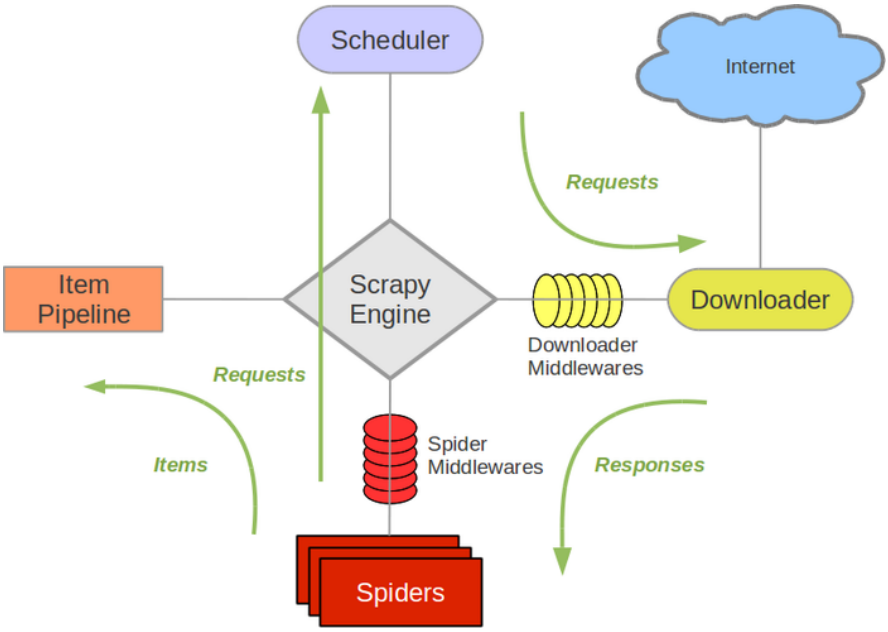

Figura 5.6: Arquitectura que sigue Scrapy y sus diferentes componentes

Spiders utilizados

El primer *spider* se encarga de extraer la url de los productos. Para ello partimos de la página principal donde se encuentran todos los artículos a extraer. Cabe destacar que no existe una url que contenga los productos orientados a ambos sexos, por lo que en caso de querer artículos para hombres y mujeres se necesita una url para cada sexo.

Este *spider* además genera un archivo JSON que contiene todas las urls que ha extraido. Este archivo será utilizado como punto de partida de los siguientes *spiders* para la extracción de los campos de cada artículo.

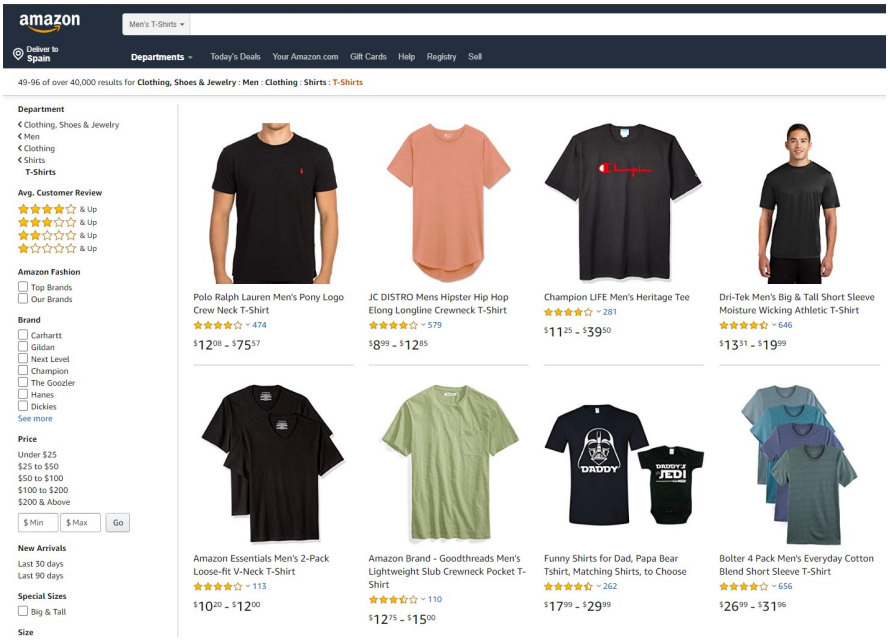

Figura 5.7: Productos a extraer

El siguiente *spider* es el encargado de extraer la mayoría de campos de un producto. Es necesario destacar los diferentes métodos utilizados para la extracción de estos campos:

- Expresiones regulares: El campo del ASIN se extrae directamente de la url con ayuda de una expresión regular.
- La mayoría de campos se han extraido con la ayuda de expresiones XPath[39], que consisten en parsear el código html de la web e identificar el campo que se desea extraer.
- Para las imágenes ha sido necesaria la utilización de una librería externa llamada js2xml[40]. Esta librería facilita la extracción de elementos embebidos en código JavaScript usando XPath.

[39] https://es.wikipedia.org/wiki/XPath
[40] https://github.com/scrapinghub/js2xml

Figura 5.8: Posición de algunos de los campos a extraer

Por último, el tercer *spider* del que hacemos uso es el encargado de extraer los comentarios de cada producto. Para ello usamos una expresión XPath sobre una página cuya url creamos a partir del ASIN de cada artículo.

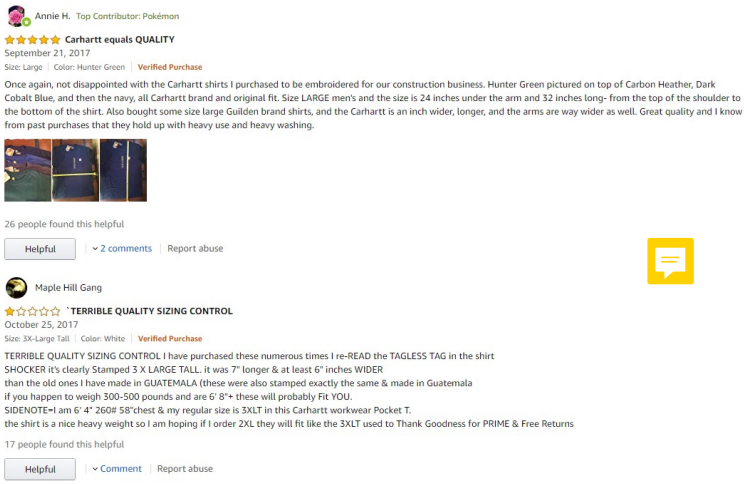

Figura 5.9: Comentarios sobre un producto

En cuanto al *pipeline* de nuestro proyecto de Scrapy, cada vez que se extrae un *item*, éste es almacenado en una base de datos. El *pipeline* se encarga de conectar con la base de datos, crear las tablas necesarias con sus respectivos campos, y cerrar la conexión una vez se han almacenado todos los productos.

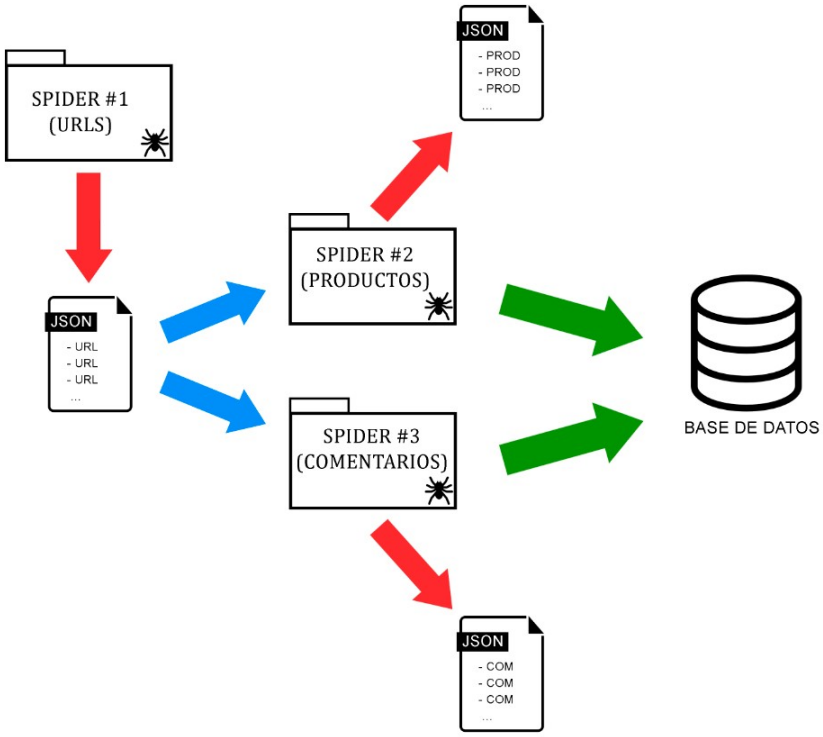

Figura 5.10: Proceso de *web scraping* del proyecto.

5.4. Etiquetado manual de las imágenes

Antes de etiquetar automáticamente las imágenes, es necesario entrenar el clasificador para que éste sea capaz de identificar cada imagen. Para ello se ha tenido que crear un conjunto de datos etiquetado manualmente que sirva de entrenamiento para el clasificador.

Ese conjunto de datos etiquetados o *dataset*, se ha generado con la ayuda de la web *Dataturks*[41].

El funcionamiento de esta herramienta es sencillo: Basta con crear una lista con las urls de las imágenes que se desean etiquetar e indicar las posibles etiquetas que una imagen puede tener. A partir de aquí, la herramienta descarga y muestra secuencialmente todas las imágenes y se ha de decidir que etiqueta corresponde a cada una.

Una vez se han etiquetado todas las imágenes, se puede exportar el conjunto de imágenes como un archivo JSON que asocia cada url con las etiquetas seleccionadas.

[41] https://dataturks.com

Para este proyecto se han creado dos *datasets*. En el primero se diferencian las imágenes en las que aparece un modelo y en las que no, el segundo *dataset* parte del anterior e identifica en cuales de las imágenes en las que si hay modelo, a éste se le ve la cara.

Problemas encontrados

El primero de los problemas sufridos relacionados con el etiquetado de imágenes fue precisamente decidir qué etiquetas seleccionar. En un principio la idea era identificar si aparecía un maniquí o no, o incluso si en la imagen no aparecía el producto anunciado. Estas ideas se desecharon ya que había demasiados casos en los que no era demasiado claro qué etiqueta correspondía a cada producto.

Otro problema encontrado ha sido a la hora de identificar si al modelo se le ve la cara o no. Hay muchos casos en los que la cara del modelo está cortada y solo es visible parcialmente. También hay casos en los que el modelo está de espaldas y mirando hacia un lado y solo se intuye la cara de éste. Lo primero que se pensó para estos casos fue un algoritmo que identificase las caras parcialmente, y aunque se han encontrado varios recursos que parecen dar respuesta a este problema, al final se optó por etiquetar las imágenes de forma que si parte de la cara es visible, se clasificaría como que si hay cara.

Algunos de los recursos que se han utilizado para una posible detección parcial de la cara han sido los siquientes:

- Partial Face Recognition: http://www.faceforensics.com/PartialFaceRecog.aspx
- Partial Face Detection in the Mobile Domain: https://arxiv.org/pdf/1704.02117.pdf
- Dynamic Feature Learning for Partial Face Recognition: http://openaccess.thecvf.com/content_cvpr_2018/papers/He_Dynamic_Feature_Learning_CVPR_2018_paper.pdf
- A robust face detector under partial occlusion - IEEE Conference Publication: https://ieeexplore.ieee.org/document/1418825
- Dealing with Inaccurate Face Detection for Automatic Gender Recognition with Partially Occluded Faces: http://repositori.uji.es/xmlui/bitstream/handle/10234/23941/34130.pdf?sequence=1&isAllowed=y

Figura 5.11: Modelo con la cara parcialmente visible

Figura 5.12: Modelo de espaldas con la cara visible

5.5. Etiquetado automático de las imágenes

Para el etiquetado automático de imágenes se ha decidido usar Keras. Existen otros *frameworks* que se han tenido en cuenta para esto proyecto como pueden ser Pytorch[42] o Caffe[43], pero se ha usado Keras por varias razones:

Experiencia con este *framework*: Ya había trabajado con anterioridad con Keras para clasificación de imágenes.

Facilidad de uso: Después de leer e informarme sobre otros *frameworks*, Keras me ha parecido el más facil de aprender y de poner en marcha.

Flexibilidad: Keras se integra con TensorFlow[44], es por esto que permite implementar cualquier cosa implemetada con TensorFlow como base.

Compatibilidad: Keras es compatible con Google Cloud[45] y Tarjetas gráficas NVIDIA[46], gracias a ello funciona de forma nativa con Google Colab.

Validación cruzada

Para el entrenamiento de estos clasificadores se ha usado la técnica de validación cruzada. Esta técnica consiste en dividir el *dataset* en dos conjuntos complementarios y realizar el análisis sobre un subconjunto para luego validarlo con los datos del otro subconjunto. La principal ventaja de este método es que es muy rápido de computar.

Existen variaciones de este método, en el que se suceden varias iteraciones, obteniendo así una mayor precisión. En este caso se ha optado por el método de validación cruzada aleatoria ya que la precisión era aceptable y Keras dispone de una función que facilita este procedimiento. [14]

[42] https://pytorch.org/
[43] https://caffe.berkeleyvision.org/
[44] https://www.tensorflow.org/guide/keras
[45] https://cloud.google.com/tpu/
[46] https://developer.nvidia.com/deep-learning

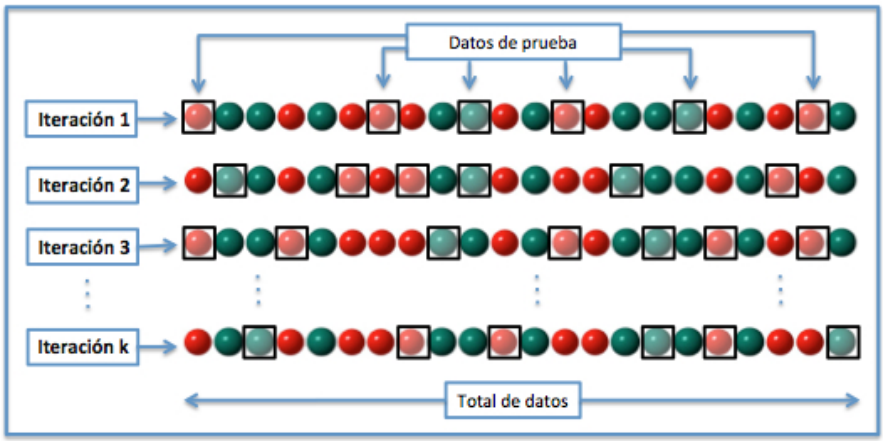

Figura 5.13: Validación cruzada aleatoria con k iteraciones

Para los clasificadores entrenados en este proyecto, se ha hecho una división del 75 % para el conjunto de entrenamiento y un 25 % para el conjunto de test. En el caso del clasificador que identifica si aparece un modelo o no, se obtiene una precisión de aproximadamente un 93 %. En cambio, para el clasificador que indica si parece la cara del modelo, la precisión es de únicamente un 80 %.

```
Train on 1410 samples, validate on 470 samples
Epoch 1/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.4792 - acc: 0.7702 - val_loss: 0.3187 - val_acc: 0.8766
Epoch 2/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.2352 - acc: 0.9128 - val_loss: 0.2231 - val_acc: 0.9128
Epoch 3/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.1854 - acc: 0.9355 - val_loss: 0.2155 - val_acc: 0.9277
Epoch 4/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.1635 - acc: 0.9390 - val_loss: 0.1878 - val_acc: 0.9383
Epoch 5/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.1342 - acc: 0.9496 - val_loss: 0.2125 - val_acc: 0.9213
Epoch 6/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.1035 - acc: 0.9674 - val_loss: 0.1979 - val_acc: 0.9383
Epoch 7/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.0941 - acc: 0.9624 - val_loss: 0.2507 - val_acc: 0.9255
Epoch 8/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.0616 - acc: 0.9816 - val_loss: 0.2044 - val_acc: 0.9489
Epoch 9/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.0408 - acc: 0.9894 - val_loss: 0.2097 - val_acc: 0.9404
Epoch 10/10
1410/1410 [==============================] - 4s 3ms/step - loss: 0.0342 - acc: 0.9879 - val_loss: 0.2415 - val_acc: 0.9362
470/470 [==============================] - 0s 533us/step
Test loss: 0.24146071662928195
Test accuracy: 0.9361702120050471
```

Figura 5.14: Precisión del clasificador Modelo/No modelo

```
Train on 594 samples, validate on 198 samples
Epoch 1/10
594/594 [==============================] - 2s 4ms/step - loss: 0.7223 - acc: 0.7508 - val_loss: 0.4951 - val_acc: 0.8081
Epoch 2/10
594/594 [==============================] - 2s 3ms/step - loss: 0.4979 - acc: 0.8131 - val_loss: 0.5158 - val_acc: 0.8081
Epoch 3/10
594/594 [==============================] - 2s 3ms/step - loss: 0.4681 - acc: 0.8131 - val_loss: 0.4917 - val_acc: 0.8081
Epoch 4/10
594/594 [==============================] - 2s 3ms/step - loss: 0.4400 - acc: 0.8131 - val_loss: 0.4773 - val_acc: 0.8081
Epoch 5/10
594/594 [==============================] - 2s 3ms/step - loss: 0.4105 - acc: 0.8232 - val_loss: 0.4787 - val_acc: 0.8131
Epoch 6/10
594/594 [==============================] - 2s 3ms/step - loss: 0.3902 - acc: 0.8266 - val_loss: 0.4578 - val_acc: 0.7980
Epoch 7/10
594/594 [==============================] - 2s 3ms/step - loss: 0.3441 - acc: 0.8434 - val_loss: 0.4638 - val_acc: 0.8030
Epoch 8/10
594/594 [==============================] - 2s 3ms/step - loss: 0.3183 - acc: 0.8569 - val_loss: 0.4557 - val_acc: 0.8182
Epoch 9/10
594/594 [==============================] - 2s 3ms/step - loss: 0.3066 - acc: 0.8771 - val_loss: 0.5042 - val_acc: 0.8081
Epoch 10/10
594/594 [==============================] - 2s 3ms/step - loss: 0.2611 - acc: 0.8956 - val_loss: 0.4915 - val_acc: 0.8030
198/198 [==============================] - 0s 561us/step
Test loss: 0.4915361502329047
Test accuracy: 0.8030303036323702
```

Figura 5.15: Precisión del clasificador Cara/No cara

El número de iteraciones para el entrenamiento ha sido de 10 para ambos clasificadores. Se ha probado con distintos números de iteraciones pero la precisión obtenida no variaba o decrecía.

Problemas encontrados

El principal inconveniente que se ha encontrado en el proceso de clasificación está directamente relacionado con el problema encontrado a la hora de etiquetar las imágenes manualmente. Para el clasificador que identifica si la cara de un modelo es visible, se pensó en utilizar una función de la librería de OpenCV que detecta caras en una imagen con la utilización de *Haar Cascades*[47]. Los resultados parecían prometedores al principio ya que actúa con una precisión muy alta. Pero el principal problema que se encontró es que no detectaba las caras cortadas. Esto hizo que se desechara la idea y se pasase a entrenar dos clasificadores.

[47] https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html

Figura 5.16: Detección errónea de caras con la librería OpenCV

5.6. Visualización y almacenamiento de todos los productos junto a sus campos

En este punto, ya tendríamos los clasificadores funcionando y los campos de los productos extraídos. Los pasos siguientes se encargan de satisfacer los objetivos de almacenamiento y visualización de los artículos extraídos:

1. Cargar los clasificadores previamente entrenados
2. Importar todos los productos que se desea clasificar. Esto se hace a partir del archivo JSON que he generado Dataturks combinado con el archivo JSON creado por Scrapy.
3. Clasificamos la imagen principal de cada producto.
4. Conectamos con la base de datos y creamos una nueva tabla que incluirá las predicciones de cada imagen. Es necesario recordar que la base

de datos hasta este momento contiene 3 tablas: Productos, Imágenes y Comentarios. Esta nueva tabla asocia cada predicción con el ASIN del artículo al que pertenece.

Con esto la base de datos quedaría completada. Cabe destacar que se ha elegido crear una nueva tabla en vez de añadir el campo *predicción* a una tabla ya existente ya que en un futuro se tendría pensado clasificar mas aspectos de cada artículo como podrían ser los comentarios u otras características de las imágenes.

Almacenamiento de los datos

La base de datos utilizada para almacenar los diferentes productos es muy simple. Una vez finalizada la extracción y el procesamiento de los datos, cuenta con 4 tablas con el ASIN de cada artículo como identificador.

- La primera tabla, la más general, tiene los diferentes campos de cada producto y el ASIN como clave primaria. Cada entrada pertenece a un artículo diferente.
- La tabla de las imágenes almacena las urls de cada imagen junto al ASIN del producto al que pertenece. Puede haber varias imágenes para un único producto.
- La tabla de los comentarios almacena comentarios asociados a cada artículo. Puede haber varios comentarios de cada producto.
- La tabla de predicciones asocia cada artículo con la predicción que el clasificador ha generado para la imagen principal de dicho producto. Cada entrada es de un producto diferente.

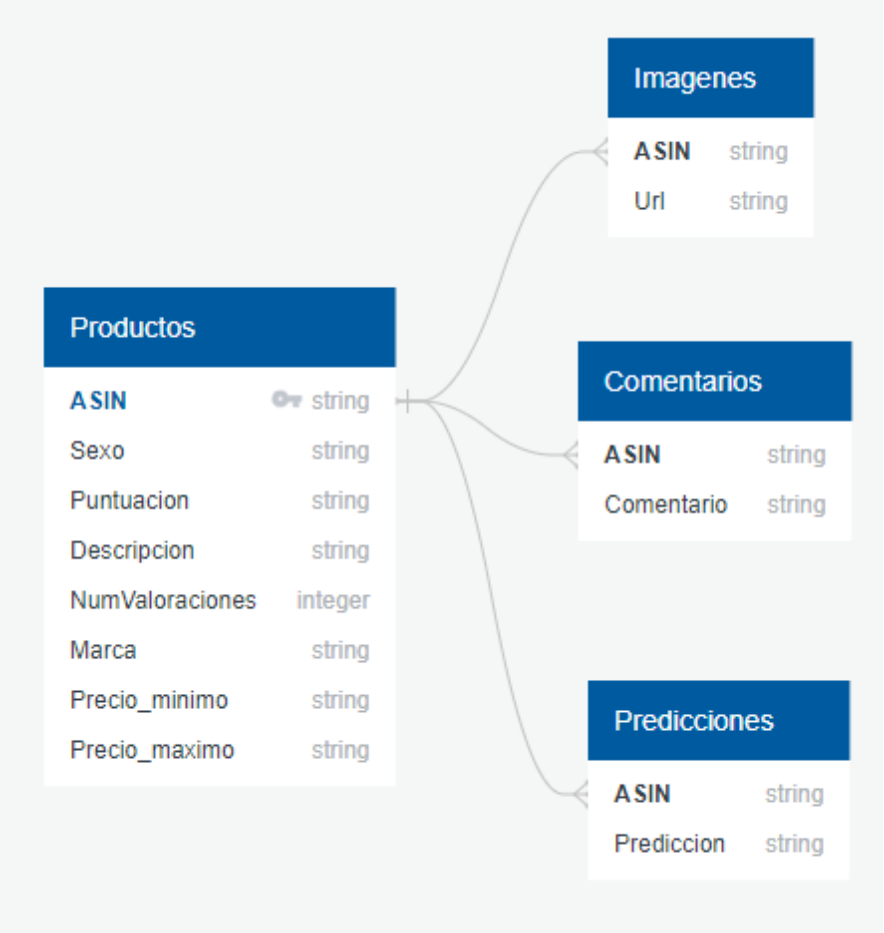

Figura 5.17: Diagrama de la base de datos utilizada.

5.7. Ciclo de vida del proceso completo

Trabajos relacionados

Este apartado sería parecido a un estado del arte de una tesis o tesina. En un trabajo final grado no parece obligada su presencia, aunque se puede dejar a juicio del tutor el incluir un pequeño resumen comentado de los trabajos y proyectos ya realizados en el campo del proyecto en curso.

Conclusiones y Líneas de trabajo futuras

Todo proyecto debe incluir las conclusiones que se derivan de su desarrollo. Éstas pueden ser de diferente índole, dependiendo de la tipología del proyecto, pero normalmente van a estar presentes un conjunto de conclusiones relacionadas con los resultados del proyecto y un conjunto de conclusiones técnicas. Además, resulta muy útil realizar un informe crítico indicando cómo se puede mejorar el proyecto, o cómo se puede continuar trabajando en la línea del proyecto realizado.

Bibliografía

[1] J.s. Albus. Outline for a theory of intelligence. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(3):473–509, 1991.

[2] Fernando Sancho Caparrini. Redes neuronales: una visión superficial, Dic 2018. http://www.cs.us.es/~fsancho/?e=72.

[3] Andreas Kaplan and Michael Haenlein. Siri, siri, in my hand: Who's the fairest in the land? on the interpretations, illustrations, and implications of artificial intelligence. *Business Horizons*, 62(1):15–25, 2019. https://www.sciencedirect.com/science/article/pii/S0007681318301393.

[4] David L. Poole, Alan Mackworth, and Randy Goebel. *Computational intelligence: a logical approach*. Oxford University Press, 1998. http://people.cs.ubc.ca/~poole/ci/ch1.pdf.

[5] Ray Shaan. History of ai, Aug 2018. https://towardsdatascience.com/history-of-ai-484a86fc16ef.

[6] Wikipedia. Aprendizaje automático — wikipedia, la enciclopedia libre, 2019.

[7] Wikipedia. Github — wikipedia, la enciclopedia libre, 2019.

[8] Wikipedia. Keras — wikipedia, la enciclopedia libre, 2019.

[9] Wikipedia. Latex — wikipedia, la enciclopedia libre, 2019.

[10] Wikipedia. Minería de datos — wikipedia, la enciclopedia libre, 2019.

[11] Wikipedia. Opencv — wikipedia, la enciclopedia libre, 2019.

[12] Wikipedia. Python — wikipedia, la enciclopedia libre, 2019.

[13] Wikipedia. Scrapy — wikipedia, la enciclopedia libre, 2019.

[14] Wikipedia. Validación cruzada — wikipedia, la enciclopedia libre, 2019.

[15] Wikipedia. Visual studio code — wikipedia, la enciclopedia libre, 2019.

[16] Wikipedia. Web scraping — wikipedia, la enciclopedia libre, 2019.