Universidad ORT Uruguay

Facultad de Ingeniería

Document Assistant:

Sistema para automatizar la digitalización de facturas en papel

Entregado como requisito para la obtención del título de Máster en Inteligencia Artificial y Big Data

Nicolás Eiris – 182713

Sebastián Otte - 172873

Ignacio Fiori - 192829

Tutor: Franz Mayr

2022

Declaración de autoría

Nosotros, Nicolás Eiris, Sebastián Otte e Ignacio Fiori, declaramos que el trabajo que se presenta en esta obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizamos el Trabajo final del Master en Big Data;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Nicolás Eiris	Sebastián Otte	Ignacio Fiori
27/03/2022	27/03/2022	27/03/2022

Agradecimientos

A nuestras familias y amigos, quienes nos motivaron y apoyaron incondicionalmente a lo largo de estos dos años de postgrado.

A Franz Mayr, quien con su profesionalismo y carisma buscó sacar lo mejor de cada uno de nosotros, compartiendo su conocimiento y experiencia para guiarnos permanentemente por el camino correcto.

Por último a Sergio Yovine, quien en su rol de líder de cátedra se encontró siempre disponible siendo un gran nexo entre el equipo y el proyecto en cuestión.

A todos ustedes, muchas gracias.

Abstract

Este proyecto se centra en la optimización del proceso contable asociado a la digitalización de facturas en papel. Para lograrlo, el acto de digitalizar y extraer información de un documento debe minimizar el uso de recursos humanos y de tiempo requerido.

Se detectó una necesidad insatisfecha en empresas de distintas industrias en lo que refiere a la digitalización y posterior extracción de información relevante de documentos físicos, esto nos incentiva para la realización de este proyecto aplicando los conocimientos adquiridos durante el máster cursado, permitiendo brindar una solución que resuelva dicha necesidad. En Uruguay muchas empresas están migrando sus procesos a facturación electrónica, este proceso no es obligatorio para todas las empresas y tampoco brinda una solución en lo que respecta a la problemática descripta.

En cuanto al negocio por detrás de la solución, a pesar de tratarse de un proyecto académico, el equipo toma un enfoque donde los interesados involucrados incluyen tanto a la universidad (tutor y coordinador de la carrera), así como también las empresas interesadas en adoptar una solución de este calibre. El alcance del proyecto a nivel de negocio busca obtener una primera versión de la solución con funcionalidades básicas, para luego poder continuar iterando con el fin de experimentar la reacción del mercado ante la aparición de este producto.

En lo que refiere a la solución, se integran tecnologías de inteligencia artificial junto a herramientas de software clásicas para la implementación, logrando un producto capaz de ser utilizado en producción. Las funcionalidades claves de esta primera versión tratan sobre agrupar facturas por proveedor así como también la extracción de información relevante para su posterior proceso contable.

Se construyó un producto *end-to-end* capaz de aportar valor a los procesos contables asociados a la digitalización de facturas en papel. Los próximos pasos refieren a refinar

la arquitectura para poder instalar el sistema en producción y comenzar a procesar documentos en la operativa de empresas reales.

Palabras claves:
OCR (*Optical Character Recognition*); IA (Inteligencia Artificial); NLP (*Natural Language Processing*); Procesamiento Automático; Documentos Escaneados; Extracción de información relevante; Clasificación de documentos; MVP (*Minimum Viable Product*); PoC (Proof of Concept); *Open-Source*; *Machine learning*; *Features*, *Pipeline end-to-end*; *Google cloud storage*.

Índice

1. Introducción

En los últimos años, debido al desarrollo de hardware cada vez más potente y el avance en nuevas tecnologías, áreas como Inteligencia Artificial sufrieron una explosión como nunca antes se ha visto[1]. La motivación e interés por parte de este grupo en desarrollar y aplicar estas tecnologías es lo que nos llevó al desarrollo de este proyecto, basándonos en una problemática real como lo es la extracción de información y clasificación de documentos.

A pesar de haber encontrado herramientas disponibles que intentan solucionar dicha realidad, estas se encuentran en desarrollo y no se ajustan a la problemática que tienen las distintas empresas del mercado local, ya sea por no estar adecuados a los documentos utilizados localmente, alto precio, alta complejidad, o falta de integración con sistemas de contabilidad disponibles en el mercado.

Este proyecto busca brindar una herramienta que aporte valor permitiendo procesar grandes volúmenes de documentos que se encuentran en formato papel, de los cuales se necesita realizar una digitalización estructurada de los mismos.

1.1. Selección del proyecto

Uno de los aspectos más importantes a la hora de seleccionar el proyecto fue la motivación de generar una solución ideada por el equipo con el objetivo de iterativamente potenciar una solución a un determinado segmento del mercado a partir de la tesis del máster.

Luego de varias instancias en las que el equipo se reunió para proponer distintas ideas y opciones de proyecto, surge la necesidad por parte de la empresa F. S. Otte Ltda., perteneciente a la familia de uno de los integrantes del grupo, en automatizar la digitalización de documentos en papel. Específicamente tratándose de facturas de proveedores donde se debía llevar una correcta trazabilidad contable de las mismas y se tornaba complejo debido al gran número de documentos y la carencia de recursos humanos suficientes para efectuar la tarea. El equipo se interesó mucho en esta

propuesta, llevando a cabo un análisis exhaustivo de herramientas existentes en el mercado, sumado a una recopilación de información sobre las necesidades en las que aplica la utilización de este tipo de herramientas para organizaciones del Uruguay.

Se encontraron en el mercado herramientas que intentan resolver dicha problemática, como ser *IBM Automation Document Processing* [2] o *Nanonets* [3], pero éstas presentan un alto costo de implementación, carecen de flexibilidad y/o usabilidad, o directamente no brindan la integración necesaria requerida para un sistema de este tipo. Ninguna de ellas presenta un buen soporte para el idioma español además que tienen costos muy elevados en torno de los mil dólares mensuales para el volúmen de datos a procesar (3500 facturas mensuales). En la industria uruguaya no existe producto que cumpla con las funcionalidades requeridas.

En conclusión, el equipo se nota sumamente motivado con esta iniciativa dadas las necesidades de negocio insatisfechas. Luego de algunas ideas iniciales y analizando el objetivo del producto se decide nombrarlo como *Document Assistant*. A su vez, es de gran interés afrontar el desafío técnico asociado donde es posible aplicar todos los conocimientos adquiridos durante los dos años de postgrado en esta instancia formal de evaluación.

1.2. Objetivos

A continuación se detallan los distintos objetivos que el equipo se planteó para el proyecto, el producto y lo relacionado a lo académico.

1.2.1. Objetivos del proyecto

Uno de los objetivos principales relacionado al proyecto es cumplir y superar las expectativas de los distintos interesados involucrados. Estos son, el tutor, coordinador de cátedra, empresa F. S. Otte Ltda., y el equipo de trabajo.

Otro de los objetivos a cumplir trata sobre la construcción de un mínimo producto viable (MVP), mediante el cuál se puede realizar una validación a nivel de mercado con potenciales clientes reales.

Se busca que la actividad esté alineada a una metodología de trabajo idéntica a la aplicada en la industria, con el fin de generar un producto de calidad que le aporte valor real a los potenciales clientes en el futuro. Aplicando no solamente conocimientos sobre inteligencia artificial, sino también procesos de ingeniería de software estandarizados.

1.2.2. Objetivos académicos

Dado que el proyecto final es la última instancia del máster, se aplican los conocimientos adquiridos a lo largo de la carrera en un único proyecto. En comparación al proyecto de grado, el tiempo de trabajo del máster es menor, donde se debe lograr un producto razonable en un tiempo de trabajo reducido. Por esto mismo se define un MVP que el equipo se compromete a alcanzar.

Este proyecto surge como instancia de evaluación para la obtención del título Master en Big Data. De aquí, el equipo se percibe motivado por aprobar esta instancia con una calificación de excelencia.

1.3. Descripción del equipo de trabajo

El equipo está integrado por dos ingenieros en sistemas y un ingeniero en electrónica. Fueron compañeros reiteradas veces durante la carrera y están acostumbrados a trabajar juntos, aspectos que permitieron una comunicación fluida y un entorno de confianza.

El hecho de contar con una mezcla de técnicos informáticos y electrónicos permite contar con distintos puntos de vista a la hora de abordar distintos problemas, siendo esto muy positivo en la dinámica de discusiones dentro del equipo.

Los integrantes son:

- Sebastian Otte
- Nicolás Eiris
- Ignacio Fiori
- Franz Mayr (Tutor)

1.4. Estructura del documento

Capítulo 1: Introducción

Se expone introducir a los lectores acerca de aspectos generales del proyecto, como ser la selección del mismo, composición del equipo y objetivos de distinta índole.

Capítulo 2: El problema y la solución

Pretende poner en contexto sobre el problema detectado (necesidad insatisfecha) en el mercado local del Uruguay y la solución propuesta acompañada de sus principales características generales.

Capítulo 3: Ingeniería de requerimientos

En esta sección se describen las actividades de ingeniería de requerimientos ejecutadas con el fin de relevar tanto requerimientos funcionales como no funcionales. Luego se listan ambos tipos de requerimientos detallando actores involucrados en cada uno de ellos.

Capítulo 4: Ciclo de vida del proyecto

Se presenta un diagrama explicando cada fase del ciclo de vida del proyecto

Capítulo 5: Descripción de la solución de software

Aquí se expone el detalle técnico de la solución. Mediante el análisis de la solución desde distintos puntos de vista el lector será capaz de comprender las decisiones de diseño y el funcionamiento interno del sistema.

También se presentarán todos los experimentos y actividades asociadas a técnicas de Machine Learning aplicadas a lo largo del proyecto.

Capítulo 6: Conclusiones

Se exponen las conclusiones finales del proyecto y lecciones aprendidas en el proceso, dando una reflexión por parte del equipo respecto a esta experiencia académica.

Capítulo 7: Próximos pasos

En esta última sección se exponen distintos pasos a seguir luego del trabajo realizado durante este proyecto.

2. El problema y la solución

Este capítulo describe el contexto del problema a resolver, para luego exponer cuál es la solución de alto nivel propuesta desde el punto de vista del negocio, explicando sus características fundamentales.

2.1. Contexto del problema

El procesamiento de facturas y documentos para la obtención de la información relevante, implica una elevada cantidad de horas hombre revisando uno a uno los documentos con el fin de poder extraer la información deseada. Ésta situación ocurre en la empresa con la que trabajamos para este proyecto (F.S. Otte Ltda.) donde procesar manualmente aproximadamente 3500 facturas mensuales toma una semana y media de un recurso humano a jornada completa (entre 5 y 8 minutos por factura).

Esta es una actividad generalmente tediosa, generando desmotivación por parte de la persona encargada de realizar la tarea, que a pesar de no ser compleja es repetitiva y que en muchas ocasiones requiere numerosos re-trabajos para disminuir las probabilidades de fallos al momento de transcribir los datos. Dicha tarea es la que se pretende automatizar, con el fin de evitar desgastes innecesarios por parte de las personas que transcriben/utilizan los datos, logrando así disminuir el tiempo necesario para realizar el procesamiento y por consiguiente lograr una disminución en los costos operativos.

Para contextualizar al lector, se presenta un ejemplo de flujo básico del sistema:

- Finaliza el mes y se necesita generar el análisis contable de costos y ganancias de la empresa. Por tanto, se requiere procesar todas las facturas de los distintos proveedores para luego realizar el balance mensual. Supongamos que la empresa tiene seis proveedores, de los cuales se tienen aproximadamente quince facturas de cada uno (noventa en total). El usuario encargado de la gestión de esas facturas coloca los documentos en un escáner automático el cual genera los archivos digitales de los documentos a procesar.

- Mediante un protocolo establecido, el software procede a enviar al servidor todas las facturas a procesar. Vale destacar que no tiene que subir factura por factura, sino que se publican todas juntas (mecanismo *batch*).
- El sistema implementado toma los archivos alojados en el servidor, y genera el correspondiente procesamiento con los mismos, clasificando según el proveedor y extrayendo los campos requeridos.
- Por último, se alojan los datos asociados a la extracción en un formato estándar para ser fácilmente consumido por el usuario.

2.2. Objetivos generales del producto

El objetivo primordial del producto es brindar una solución al problema identificado, permitiendo invertir esas horas en tareas que aporten valor real para el negocio. Como se mencionó anteriormente, el equipo define un MVP acorde al plazo de trabajo, pero la utilización del producto le brinda a la empresa la disponibilidad de los datos para su posterior explotación.

2.3. Descripción general del sistema

El encare propuesto de un sistema que brinde una solución a la problemática descrita comienza por una digitalización por parte del usuario de los documentos a procesar. Los documentos esperados son del tipo imagen, en formatos clásicos tipo .jpg, .png o pdf. Los archivos serán enviados para su procesamiento, como resultado el sistema devolverá al usuario una copia del documento correctamente orientada y ajustada así como también un archivo que contiene toda la información extraída para ser fácilmente integrada con sistemas externos.

En la siguiente imagen se muestra un diagrama de alto nivel del sistema, que permite visualizar fácilmente el funcionamiento esperado.

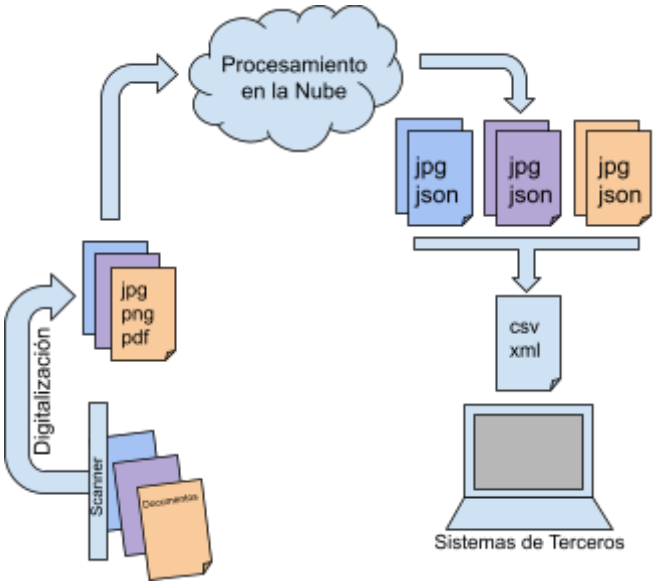


Ilustración 2.3 - Diagrama flujo de sistema alto nivel

El proceso comienza por la digitalización del documento por parte del usuario, donde se genera una representación digital del archivo original, y sobre esta representación se obtiene la información relevante. Usualmente puede ser un documento escaneado pero también puede ser un documento digital nativo como lo es un PDF, nuestro caso de estudio está basado en facturas particularmente, pero más adelante podrían ser agregados otro tipo de documentos donde se requiere sistemáticamente extraer información para su uso.

El sistema procesa el conjunto de datos digitalizados, obteniendo como resultado información valiosa como lo es por ejemplo, en caso de comprobantes fiscales, el emisor y número del comprobante, valor total, etc. También se podrían obtener atributos importantes como la presencia de sellos, firmas o referencias a otros documentos en caso de que corresponda.

Una vez obtenida la metadata asociada a los archivos digitalizados, el sistema presenta los datos en formatos capaces de ser procesados por sistemas de terceros, logrando así

cerrar el ciclo de transformación desde el documento original hasta la digitalización de los datos contenidos en el mismo.

2.4. Principales aspectos de la solución

Se espera obtener un producto que sea capaz de recibir y procesar distintos tipos de documentos en los cuales el tamaño, la rotación, formato o distribución de los mismos pueda ser muy variable. El producto aplica tecnologías de *computer vision* [69] e inteligencia artificial para lograr el procesamiento integral del documento, obteniendo como salida una copia del documento correctamente orientado junto con su metadata la cual contiene los datos más relevantes del archivo.

Al iniciar el proyecto no se contaba con un *dataset* de facturas uruguayas con el cual entrenar el/los modelo/s de inteligencia artificial, por lo tanto, como primer paso se requiere la recopilación de facturas para la creación del *dataset* a utilizar en el proyecto. El equipo cuenta con la ventaja de tener disponibles una gran cantidad y variedad de facturas gracias a la empresa F. S. Otte Ltda (*stakeholder* del proyecto). Esta tarea implica el procesamiento manual de las facturas para digitalizarlas así como también la generación del conjunto de datos que se quieren obtener de las mismas, para su posterior utilización en las distintas etapas del proyecto.

El *dataset* asocia copias digitalizadas de múltiples documentos, donde se etiquetan las características principales del mismo como ser emisor, totales, y demas datos que resulta interesante poder obtener. Para lograrlo se desarrolló una solución basada en métodos de *computer vision* clásicos apoyados por etiquetados y correcciones manuales para lograr extraer los datos relevantes. Además se realizaron ajustes estéticos sobre el documento original (especificados en la sección 5. Descripción de la solución de software), permitiendo de esta forma una extracción precisa del contenido presente en el mismo.

El proyecto se focaliza en la implementación de dos grandes bloques, un clasificador capaz de evaluar cada factura y clasificarla de acuerdo al emisor, y un modelo capaz de extraer datos relevantes para el usuario.

3. Ingeniería de requerimientos

En el presente capítulo se describen las actividades que fueron llevadas a cabo para realizar el relevamiento y posterior validación de los requerimientos del sistema. Finalmente, se detallan los distintos requerimientos funcionales, no funcionales y restricciones identificados durante las etapas de relevamiento.

3.1. Relevamiento de requerimientos

El proceso de relevamiento tal como se detalla en el artículo de ReQtest [4] refiere a la exploración de los requerimientos del proyecto, siendo sumamente importante ya que permite detectar riesgos de distinto tipo en etapas tempranas del ciclo de vida del proyecto. Dicho relevamiento conlleva comprender el problema al cual nos debemos enfrentar y lograr obtener una noción sobre los recursos que van a ser necesarios, alcance e impacto de negocio. Para esto se realizaron aproximaciones con distintos agentes del mercado local de donde se pudo llegar a comprender las necesidades existentes, complementado con investigación sobre el mercado y lluvia de ideas entre el equipo para descubrir requerimientos con alto impacto de negocio.

3.2. Listado de requerimientos

A continuación se presentan los requerimientos funcionales y no funcionales relevados.

3.2.1. Requerimientos funcionales

Id	Requerimiento	Descripción
RF01	Carga de múltiples documentos a procesar	El sistema debe poder cargar los archivos subidos por el usuario a un servidor provisto.
RF02	Preparación de documentos para ser procesados	El sistema debe preparar los documentos bajo ciertas reglas preestablecidas. Esto refiere básicamente a generar: - Recorte - Alineación - Normalización

RF03	Clasificación según proveedores	El sistema debe clasificar los documentos en grupos luego de preparados según el proveedor al que pertenecen.
RF04	Entrenamiento para grupo de proveedores	Dado un número adecuado de facturas (al menos 20 facturas) asociadas a un proveedor, el sistema debe ser entrenado para predecir la procedencia (emisor) en el proceso de inferencia.
RF05	Integración con sistemas externos	El sistema debe permitir exportar los resultados mediante API, Google Cloud, CSV o Excel.
RF06	Extracción de datos	El sistema debe permitir extraer la información relevante para persistir como salida del procesamiento.
RF07	Detección de *features* complejos	El sistema debe poder detectar firmas, sellos, logos, etc.
RF08	Restringir procesamiento	El sistema debe poder restringir procesamiento a documentos que cumplan ciertas condiciones: - Firma - Entre fechas configurables - Campos requeridos
RF08	Interfaz de usuario	El sistema debe proveer una interfaz de visualización para usuarios con las siguientes características: - *Dashboard* con métricas sobre procesamiento - Configuración de entrenamiento (grupos de documentos, mapeo de campos) - Ingesta de documentos - Inicialización de entrenamiento - Manejo de errores (e-mail) - Revisión manual sobre predicciones - Corrección de predicción y re-entrenamiento.

Tabla 3.2.1 - Requerimientos funcionales

3.2.2. Requerimientos no funcionales

Id	Descripción	Atributo de calidad
RNF01	El sistema debe restringir el acceso a personas no autenticadas	Seguridad
RNF02	El sistema debe proveer suficiente información, de alguna forma, que permita conocer el detalle de las tareas que realiza. En particular, en el caso de ocurrir una falla o cualquier tipo de error, es imprescindible que el sistema provea toda la información necesaria que permita a los administradores hacer un diagnóstico rápido y preciso sobre las causas.	Disponibilidad
RNF03	El sistema debe registrar información que permita realizar auditorías de acceso de forma de identificar los accesos autorizados y no autorizados al sistema.	Seguridad
RNF04	El sistema debe soportar un volumen elevado de documentos a ser procesados (~3500 mensuales)	Escalabilidad
RNF05	El sistema debe estar disponible y funcionando con normalidad la mayor parte del tiempo que sea posible	Disponibilidad
RNF06	El sistema debe permitir aumentar la cantidad de usuarios activos procesando documentos en paralelo.	Escalabilidad

Tabla 3.2.2 - Requerimientos no funcionales

3.2.3. Requerimientos seleccionados prueba de concepto

Como ha sido mencionado previamente en el presente documento, el equipo decide abordar el proyecto de tesis como una instancia para poder desarrollar un mínimo producto viable para una oportunidad de negocio a futuro, mediante una solución escalable y genérica.

Para la prueba de concepto se atacan los siguientes requerimientos funcionales listados previamente:

- RF01 - Carga de múltiples documentos a procesar
- RF02 - Preparación de documentos para ser procesados
- RF03 - Clasificación según proveedores
- RF04 - Entrenamiento para grupo de proveedores
- RF05 - Integración con sistemas externos
- RF06 - Extracción de datos

De los requerimientos no funcionales relevados se desprenden los siguientes atributos de calidad a favorecer con el diseño de la arquitectura:

- Escalabilidad
- Disponibilidad
- Seguridad

4. Ciclo de vida del proyecto

A continuación se presenta un esquema con las diferentes instancias del proceso a seguir en el proyecto.

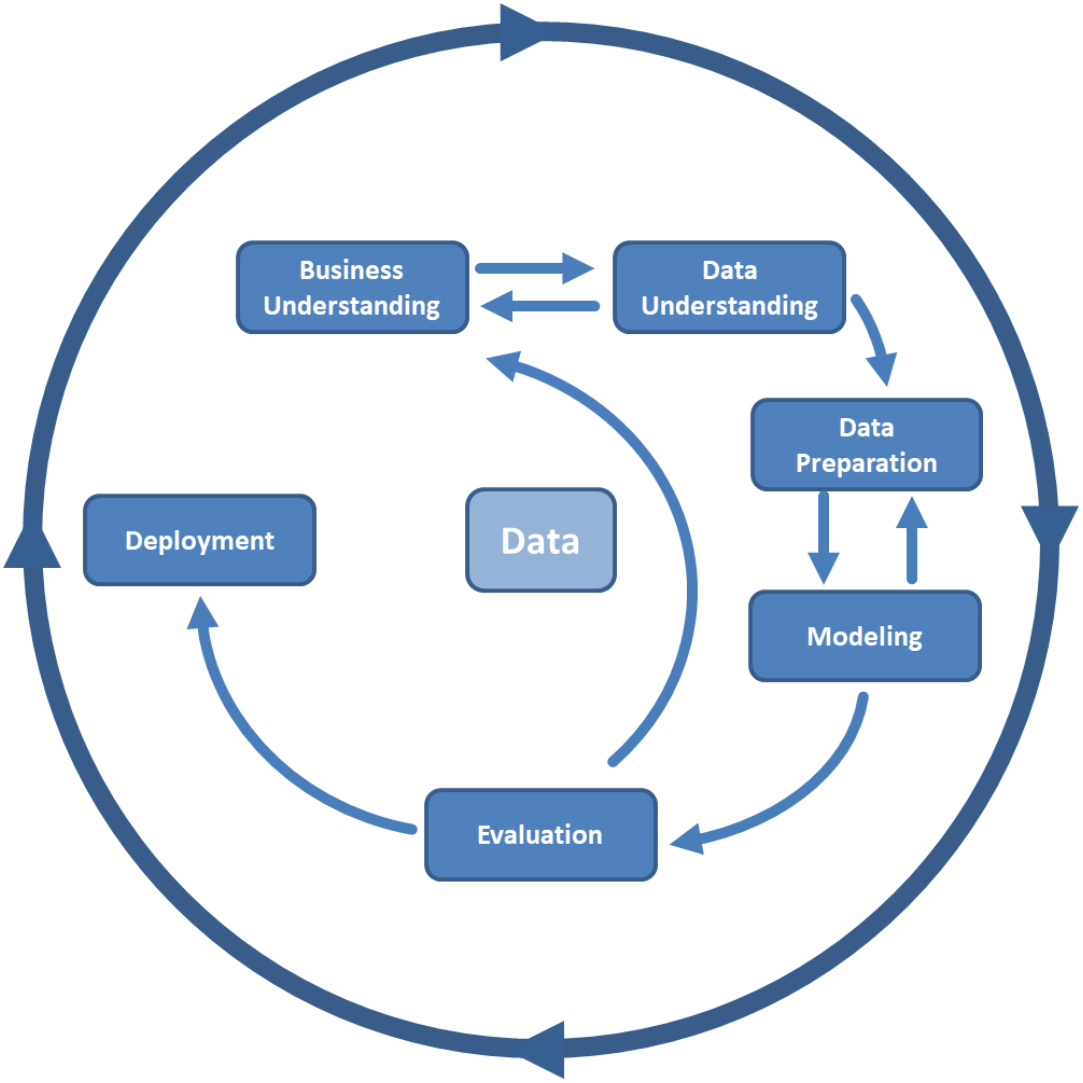


Ilustración 4.1 - Ciclo de vida del proyecto [5]

En esta primera instancia se llevaron a cabo las siguientes etapas:

Entendimiento del negocio:

Como primer paso el equipo necesita entender la problemática del negocio y su funcionamiento. Uno de los integrantes cuenta con la ventaja de que conoce el negocio de primera mano ya que su familia enfrenta los problemas mencionados anteriormente a diario. Esto le permite al equipo involucrarse en el negocio rápidamente. En base a consultas realizadas a la empresa se sabe que el procesamiento de las facturas para conseguir los datos relevantes es una tarea que a casi ninguna persona le es de interés realizar.

Entendimiento de los datos:

En esta fase el equipo se involucra con las facturas. En la fase anterior se van conociendo los principales problemas a los que se enfrentan las empresas pero hasta no ver ejemplos de facturas no se toma dimensión de la problemática. Las facturas contienen muchos datos disponibles, desde el logo, número de RUT, fecha, totales y más. Pero no siempre estos datos están bien estructurados y definidos, muchas veces las facturas están hechas a mano lo que hace difícil identificar los valores de interés. El equipo fue identificando qué datos de las facturas eran los más relevantes para las empresas.

Preparación de los datos:

Al comienzo del proyecto no se contaba con datos disponibles, por lo que se designaron recursos asociados a la digitalización de miles de documentos, para luego proceder a la creación de un *dataset* donde se tuviera identificado por cada documento digitalizado un ángulo de corrección y una categoría correspondiente al emisor de la factura.

Modelado:

Esta etapa propone la implementación de los modelos luego de tener los datos disponibles y procesados. Se busca en una primera instancia definir modelos un tanto simples para poder tener los primeros resultados y en base a estos ir definiendo modelos

más complejos. En esta etapa se pretende comparar el uso de todos los modelos para poder optimizar aquellos que obtengan los mejores resultados.

Evaluación:

En esta etapa se pretende evaluar los resultados de los modelos mediante la comparación de los datos reales contra los datos predichos por el sistema de aprendizaje automático. La idea es tener una noción del poder de generalización del mismo para de cierta forma asegurarse que tendrá una tasa de error baja a la hora de predecir nuevos datos en producción.

Puesta en producción:

Como última instancia luego de tener el prototipo estable y con resultados satisfactorios el mismo debe ser puesto en producción para la disponibilización a los potenciales clientes.

Vale aclarar que lo ejecutado en éstas etapas pertenece a la primera iteración de trabajo. La idea del equipo, tal como se realiza en la industria del software, y más específicamente en el desarrollo de sistemas de inteligencia artificial [6], inicia con iteraciones cortas sobre todas las etapas del ciclo de vida, con el objetivo de validar la viabilidad de la solución y rápidamente obtener una primera versión del producto, y luego continuar iterando sobre los mismos pasos para mejorar y agregar funcionalidades.

5. Descripción de la solución de software

5.1. Introducción

En este capítulo se hace foco en la presentación de la solución tecnológica implementada. La idea es exponer las distintas herramientas, tecnologías, tácticas y conocimientos aplicados que se obtuvieron en este máster tanto en el área de arquitectura de *software*, como de ciencia de datos y aprendizaje automático. convergiendo el contenido teórico y práctico de las distintas materias en un proyecto real de *machine learning end-to-end*.

5.2. Tecnologías y herramientas

Para la fase de implementación se trabajó sobre el lenguaje Python [7] en su versión 3.8. Este lenguaje es uno de los más utilizados en proyectos de *machine learning* y *data science*. Además de que la gran mayoría de herramientas asociadas a *big data* utilizadas en este proyecto brinda soporte para Python, todos los integrantes del equipo están familiarizados con este lenguaje, por ello no se dudó en incorporarlo para la implementación.

Se utilizó la herramienta Poetry [8] para la gestión de dependencias dentro del proyecto. Éstas son algunas de las ventajas que provee:

- Mantiene bloqueo en las versiones de las dependencias (evitando actualizaciones de las mismas por error).
- Continua compatibilidad entre librerías.
- Permite actualizaciones frecuentes con el beneficio de detectar problemas entre dependencias en etapas de desarrollo.
- Facilidad de gestionar ambientes virtuales y librerías de desarrollo.

Para el control de versiones del código fuente desarrollado se utilizó un repositorio Git [9] alojado en un servidor de GitHub [10].

Por otro lado, se tomó la decisión de utilizar una herramienta para versionado de datos, lo cual favorece la reproducibilidad del pipeline y experimentos. Se incorpora DVC [11] *(Data Version Control)*, herramienta *open-source*, que permite alojar los datos procesados por el pipeline en un repositorio remoto asociado a una rama de Git. Una de las buenas prácticas en proyectos de *machine learning* es nunca alojar *datasets* junto al código fuente, lo que genera la necesidad de gestionar manualmente el compartir datos entre desarrolladores e involucrados en el desarrollo. DVC soluciona este proceso, donde se lo puede ver como un "Git de los datos". Algunas de las funcionalidades más importantes de esta herramienta son:

- Funciona sobre cualquier repositorio de Git (GitHub, GitLab [12], etc.)
- Agnóstico al almacenamiento: se puede elegir entre una variedad de tecnologías para alojar los datos remotamente, por ejemplo: Google Cloud Storage [13], Google Drive [14], Amazon S3 [15], on-premise FTP [16], servidor SFTP [17], etc.
- Reproducible: con tan solo descargar el código fuente de una rama Git en específico y configurar el repositorio de DVC remoto, ya estamos en condiciones de ejecutar el pipeline (sin ningún otro trabajo manual) con el comando *"dvc repro"*.
- Baja fricción entre ramas: como esta herramienta está basada en Git, junto al código fuente se almacenan archivos con referencias (*hash* del archivo de datos) al repositorio remoto, por tanto cambiar de la rama "rama_A" a la "rama_B" genera un cambio de referencia lo que provoca que DVC baje los archivos asociados a la "rama_B".
- *Framework* de *machine learning*: DVC provee código fuente base necesario para rápidamente construir un pipeline *end-to-end* dando soporte de parametrización favoreciendo altamente la reproducibilidad y mantenibilidad a lo largo de todo el flujo de datos.
- Agnóstico al lenguaje y *framework*: los pipelines de DVC están basados en archivos de entrada y salida (o directorios), por esto no importa el lenguaje utilizado. Python, R [18], Scala Spark [19], Jupyer Notebooks [20], TensorFlow [21], PyTorch [22], etc son todos soportados por DVC.

- Trazabilidad de fallas: mantener conocimiento de fallas en el pipeline permite ahorrar mucho tiempo en el futuro (*debugging*, etc). DVC está diseñado para llevar trazabilidad de todo lo que sucede en una forma reproducible y fácil de acceder.
- Trazabilidad de métricas y experimentos: provee un comando para listar todas las ramas de Git con sus asociadas métricas de performance de modelo. Además permite ejecutar múltiples experimentos en paralelo (por ejemplo entrenar un modelo donde variamos un hiper parámetro en cinco experimentos distintos).

Para el procesamiento de los datos en Python se utilizaron librerías de conocidas como ser:

- Pandas [23]: Es una biblioteca de código abierto de Python, que proporciona herramientas de análisis y manipulación de datos de alto rendimiento. Entre sus características principales están: proveer herramientas para cargar datos en memoria desde diferentes formatos de archivo. Brindar estructuras de datos manipulables que permitan eliminar o insertar columnas eficientemente. Ofrecer un alto rendimiento en intersección de datos.
- Scikit-learn [24]: Es una biblioteca de código abierto para aprendizaje automático desarrollada para el lenguaje Python. Scikit-learn posee una amplia variedad de algoritmos de aprendizaje, tanto supervisados como no supervisados.

Para el procesamiento de las imágenes a lo largo del desarrollo del proyecto se utilizó la librería OpenCV [25], una librería *open source* para la aplicación de herramientas de *computer vision*. OpenCV es ampliamente utilizada a nivel global, cuenta con un amplio abanico de algoritmos diseñados para el procesamiento de imágenes, además de una enorme comunidad lo que la hace una herramienta muy accesible y práctica al momento de desarrollar soluciones basadas en *computer vision*.

Esta librería incorpora algoritmos que utilizaremos más adelante como lo son *HughLines* [26], algoritmo utilizado para la detección de líneas rectas en imágenes. Además algoritmos más complejos como SIFT [27] *(Scale Invariant Feature Transform)* con el cual la imagen procesada se descompone en puntos de interés y descriptores, para dar lugar a una comparación entre imágenes que difieren en perspectiva, angulo y tamaño, obteniendo una transformación homográfica la cual corrige los aspectos antes mencionados. La librería permite además manipular la imagen realizando transformaciones, recortes y modificaciones que permitan tener el resultado final esperado.

Para la obtención de textos desde una imagen se utilizan herramientas basadas en la tecnología Optical Character Recognition (OCR). Estas herramientas están basadas en modelos de inteligencia artificial, generalmente modelos de *deep learning*, que se encargan de localizar y reconocer los caracteres presentes en una imágen, devolviendo la transcripción de las palabras detectadas junto con un cuadro delimitador para cada una de ellas, acompañado usualmente de un valor de confianza asociado a la palabra detectada.

Existen varias librerías de OCR, algunas son descargables para ser ejecutadas de manera local y otras requieren de conexión a internet donde las imágenes se envían a un servidor para su procesamiento. Estas implementaciones pueden requerir un pago por imagen procesada, dependiendo de la implementación devuelven más atributos sobre las palabras detectadas como ser idioma, tipo de letra, tipo de escritura, entre otros, o tener mejor performance para cierto tipo de imágenes. La universidad cuenta con un convenio con Microsoft Azure [28] en el cual otorgan créditos para la utilización de Microsoft Azure OCR [29]. Google [30] posee librerías licenciadas como ser Google OCR Vision Api [31], así como también una librería de uso libre como ser Tesseract OCR [32].

Por último, vale detallar los conceptos por detrás de los distintos algoritmos experimentados en la fase de modelado y brindar una breve descripción de los mismos:

BERT

Previo a presentar el marco teórico detrás de este algoritmo de aprendizaje automático, vale introducir Natural Language Processing [33] (NLP) como concepto en cuanto a tecnología ya que BERT [34] pertenece a la familia de algoritmos dentro de NLP.

NLP es un campo de estudio dentro de la lingüística, computación e inteligencia artificial donde se estudia la interacción entre computadores y el lenguaje humano. Particularmente, cómo programar computadoras capaces de entender y analizar un gran volúmen de datos asociados al lenguaje natural.

Uno de los mayores desafíos del NLP es que no hay suficientes datos de entrenamiento. Se cuenta con una enorme cantidad de datos de texto disponibles pero si se desea crear conjuntos de datos específicos de tareas, se deben dividir. Para contrarrestar este problema, se han desarrollado varias técnicas para entrenar modelos de representación de lenguaje de propósito general. Estos modelos pre entrenados de propósito general se pueden ajustar en conjuntos de datos específicos de tareas más pequeñas, por ejemplo, cuando se trabaja con problemas como la respuesta a preguntas y el análisis de sentimientos. Este enfoque da como resultado grandes mejoras en la precisión en comparación con el entrenamiento en los conjuntos de datos específicos de tareas más pequeñas desde cero. BERT es una adición reciente a estas técnicas para el pre-entrenamiento de NLP generando buenos resultados en una amplia variedad de tareas de NLP, como la respuesta a preguntas.

Dentro de NLP, el concepto de modelo de lenguaje [35] refiere a un modelo estadístico de probabilidades, donde se busca determinar la secuencia de palabras dentro de una oración basado en palabras anteriores. Esto ayuda a predecir las próximas palabras dado un texto. Un ejemplo de uso de estas técnicas es Gmail a la hora de redactar mails donde se sugieren posibles continuaciones del texto redactado basado en el contexto del correo a enviar.

BERT es un modelo de lenguaje que se entrena bidireccionalmente lo que lo hace más poderoso. Se puede tener un sentido más profundo del contexto y el flujo del lenguaje en comparación con los modelos de lenguaje de una sola dirección. En lugar de predecir la siguiente palabra en una secuencia, BERT utiliza una técnica novedosa llamada *Masked LM* (MLM) [36]: enmascara aleatoriamente palabras en la oración y luego intenta predecirlas.

Las representaciones de lenguaje pre entrenadas pueden ser independientes del contexto o basadas en el contexto. Las representaciones basadas en el contexto pueden ser unidireccionales o bidireccionales.

BERT se basa en la arquitectura del modelo *transformer* [38]. Un transformer funciona realizando un número pequeño y constante de pasos. En cada paso, aplica un mecanismo de atención para comprender las relaciones entre todas las palabras de una oración, independientemente de su posición respectiva.

Naive Bayes

Naive Bayes [39] es uno de los algoritmos de clasificación más sencillo y rápido. Para la predicción de una clase desconocida utiliza el teorema de probabilidad de Bayes. La biblioteca Scikit-Learn implementa distintas variantes de Naive Bayes, entre ellas están:

- Bernoulli: en esta alternativa las características son variables booleanas independientes. El modelo Bernoulli es comúnmente usado para las tareas de clasificación de documentos, donde interesa la ocurrencia de términos binarios en lugar de la frecuencia de los términos.

- Multinomial: es usado cuando se tiene un número finito de clases donde no importa el orden. Con un modelo de evento multinomial, las muestras (vectores de características) representan las frecuencias con las que ciertos eventos han sido generados por una distribución multinomial (p_1 ,..., p_n) donde p_i es la probabilidad de que ocurra el evento i.

- Gaussiano: el algoritmo gaussiano se utiliza en los casos que se debe trabajar con características continuas.

En la etapa de implementación se utilizó Naive Bayes Multinomial.

Random Forest

Random forest [40] es un algoritmo de aprendizaje automático supervisado que se usa ampliamente en problemas de clasificación y regresión. Este algoritmo construye árboles de decisión en diferentes muestras y toma su voto mayoritario para la clasificación y el promedio en caso de regresión. Una de las características más importantes de *Random forest* es que puede manejar el conjunto de datos que contiene variables continuas como en el caso de la regresión y variables categóricas como en el caso de la clasificación.

Logistic Regression

La regresión logística [41] es un algoritmo de clasificación. Se utiliza para predecir un resultado binario basado en un conjunto de variables independientes. Además también es utilizado para describir datos y explicar la relación entre una variable binaria dependiente y una o más variables independientes nominales, ordinales, de intervalo o de relación.

Linear Support Vector Classifier

El modelo *Linear* SVC [42] aplica una función de kernel lineal para realizar la clasificación y funciona bien con una gran cantidad de muestras. Si se compara con el modelo SVC, el Linear SVC tiene parámetros adicionales como la normalización de penalización que aplica 'L1' o 'L2' y la función de pérdida.

Resnet-18

Este modelo se presenta en el artículo *Deep Residual Learning for Image Recognition* [43] publicado en 2015 por un equipo de investigación de Microsoft. La idea detrás de esta arquitectura de aprendizaje profundo es solucionar el problema de *vanishing gradients* [44] (en problemas de *computer vision*) el cuál complejiza el entrenamiento

de redes neuronales con alto número de capas. Durante la fase de entrenamiento, en cada iteración se actualizan los pesos de la red proporcionalmente a la derivada parcial de la función de error. En el caso de redes con múltiples capas, el gradiente se irá desvaneciendo a valores muy pequeños para las primeras capas de la red, impidiendo eficazmente el cambio de pesos. En el peor caso, esto puede impedir que la red neuronal continúe el proceso de aprendizaje.

Los autores presentan la arquitectura en bloques denominados bloques residuales, la idea es generar conexiones no solo entre convoluciones [45] contiguas sino con las siguientes. El motivo detrás de ésta estrategia es evitar pequeños cambios de pesos en las capas iniciales de la red cuando el paso de *backpropagation* [46] entra en juego.

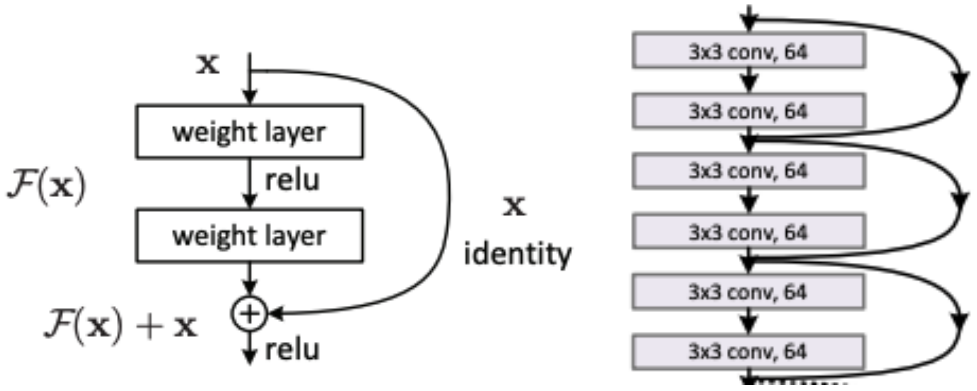


Ilustración 5.2 - Resnet-18 Bloque residual y Conexiones [43]

5.3. Arquitectura del sistema

En lo que refiere a la arquitectura del sistema, el equipo tomó varias decisiones de diseño que se pueden ver reflejadas en la ilustración 5.3.1. Dado que este proyecto tiene como objetivo la construcción de un *minimum viable product* (MVP) *end-to-end* de la solución, sumado a que académicamente se busca foco en lo que refiere a conceptos de *big data* y aplicación de conocimientos sobre aprendizaje automático, se simplificó el diseño del sistema en cuanto a la arquitectura generando un monolito con todas las funcionalidades previamente detalladas en la sección de ingeniería de requerimientos. A continuación se presenta un diagrama de la arquitectura de alto nivel comunicando la interacción entre los distintos componentes del sistema.

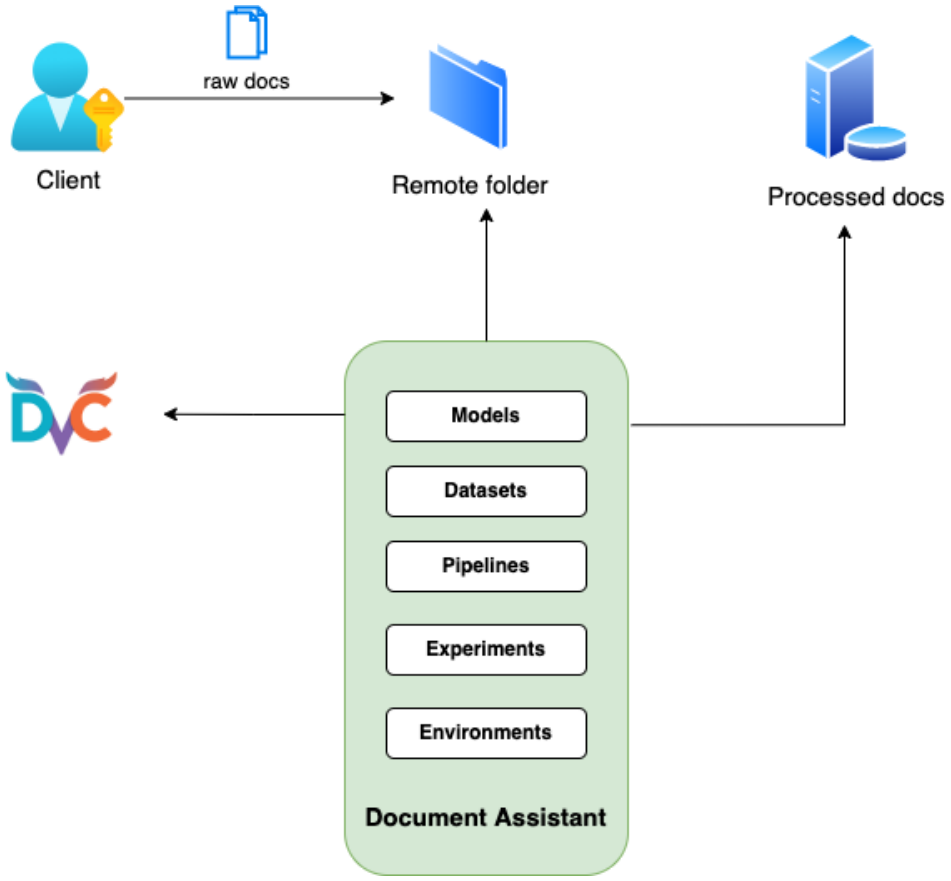


Ilustración 5.3.1 Arquitectura Document Assistant

El punto de partida del flujo normal del sistema implica la subida de documentos por parte de un cliente (cantidad indeterminada de documentos de distintos proveedores). Luego el sistema toma estos documentos a procesar, y aquí ejecuta el pipeline de producción (proceso de inferencia) utilizando los pasos de pre-procesamiento, limpieza, clasificación y posterior extracción de campos haciendo uso de DVC para cargar todo lo que refiere a datos y modelos necesarios en inferencia. Luego de procesados los documentos se alojan los resultados en un formato estándar a lo largo de todos los clientes para que éstos puedan recopilar rápidamente lo generado por el pipeline de inferencia.

En siguientes iteraciones de diseño y arquitectura del sistema, cuando el número de clientes aumente y se comience a comercializar el servicio de procesamiento de documentos, se planea escalar horizontalmente generando una ejecución distribuida y en paralelo del pipeline (por ejemplo como estrategia un servidor por cliente).

5.4. Fuente de datos

El equipo no contaba con un *dataset* al inicio del proyecto, por tanto se generó un *dataset* manualmente a partir del escaneo de más de 5.000 facturas (múltiples emisores). Los documentos fueron escaneados con un escáner automático de la marca HP modelo Smart Tank 530, el cual logra reducir sustancialmente el tiempo requerido para la digitalización y facilita el procesamiento de los documentos. El equipo se encarga de alimentar el scanner con los múltiples documentos que se coloquen en la bandeja de entrada y genera automáticamente las digitalizaciones. La digitalización de los documentos para la generación del *dataset* fue realizado en formato PNG con el fin de evitar pérdidas de calidad debido a la compresión como sí ocurre en los formatos orientados a fotografías como ser JPG o JPEG.

Una vez recopilados los datos, se obtiene un conjunto de archivos donde las digitalizaciones de los documentos pueden estar volteadas, rotadas y no siempre ubicarse en la misma posición. Para generar el *dataset* se colocaron manualmente los

documentos digitalizados en carpetas según nombre del emisor, obteniendo más de 30 emisores distintos, con varios ejemplares de documentos para cada uno.

La forma en la que se generó el *dataset* permitió etiquetar grandes volúmenes de documentos de manera ágil, dicho *dataset* está orientado al entrenamiento de un modelo de clasificación, siendo el nombre de la carpeta contenedora la categoría correspondiente al documento. Como se mencionó previamente el objetivo de este proyecto es ser capaz de clasificar las facturas según su proveedor y extraer información relevante.

5.5. Fases del pipeline

5.5.1. Preprocesamiento

Para poder obtener una mejor extracción de texto a través del OCR es necesario que el documento se encuentre correctamente orientado. Se debe corregir el ángulo de giro ajustando la imágen original para que el texto quede correctamente orientado y alineado.

En cuanto a la orientación y alineación se experimentaron varias técnicas, pero la que resultó más efectiva y logró excelentes resultados fue aprovecharse de que los documentos de texto suelen tener "líneas" tanto verticales como horizontales respecto al documento, ya sea por el texto en sí o por patrones tabulares dentro del documento. Al detectar las líneas mediante métodos clásicos y filtrado oportuno se puede predecir el ángulo de rotación de la imagen, para de esta forma alinearlas correctamente a su cuadrante más cercano.

A continuación en la Ilustración 5.5.1.1 se muestran ejemplos de imágenes originales de dos documentos que conforman el *dataset*:

Ilustración 5.5.1.1 Imágenes de documentos Originales

La Ilustración 5.5.1.2 muestra cómo aplicando métodos clásicos de *computer vision* como el algoritmo *hough lines* [47], el cuál permite detectar líneas existentes en en el documento para luego poder predecir su ángulo de rotación:

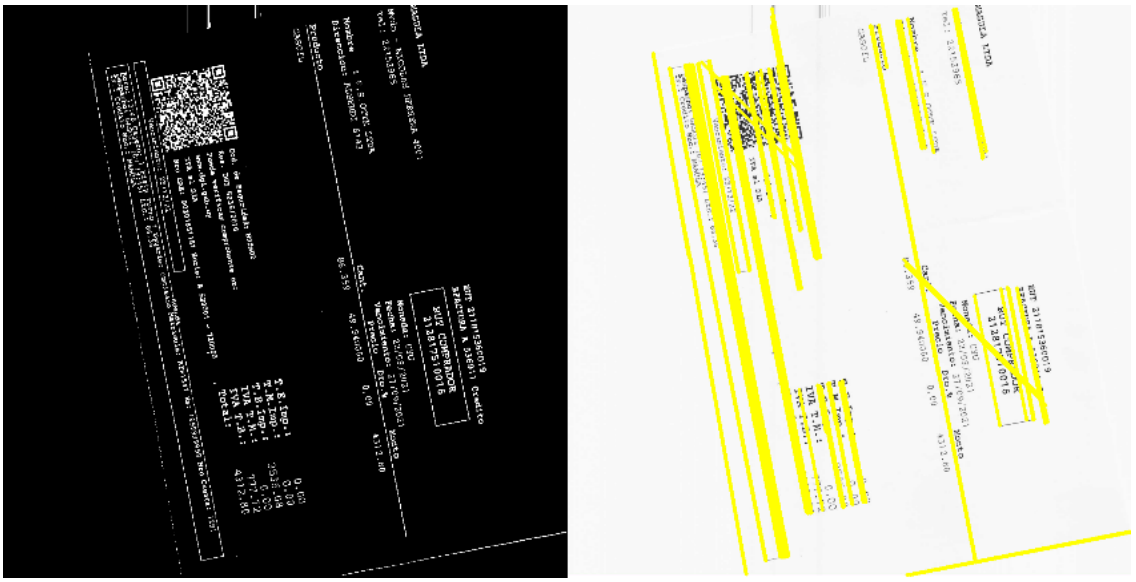


Ilustración 5.5.1.2 Detección de líneas en Imágenes de documentos Originales

Como se puede observar en la Ilustración 5.5.1.2 no todas las líneas detectadas brindan información respecto al ángulo de rotación del documento. Para poder filtrar estos datos

y poder obtener un ángulo de corrección primero se debe hacer una transformación que permita comparar los ángulos entre sí para luego proceder a realizar un filtrado de los mismos.

Se aplica una transformación sobre las líneas detectadas como se puede ver ejemplificada en la Ilustración 5.5.1.3, donde se calcula el ángulo de cada línea respecto a su cuadrante más cercano. El ángulo obtenido “A” se espera sea el mismo tanto para las líneas verticales como horizontales del documento original, logrando que los ángulos detectados sean comparables entre sí, y de esta forma realizar un filtrado acorde que permite eliminar aquellas líneas que no aportan información relevante acerca de la rotación del documento.

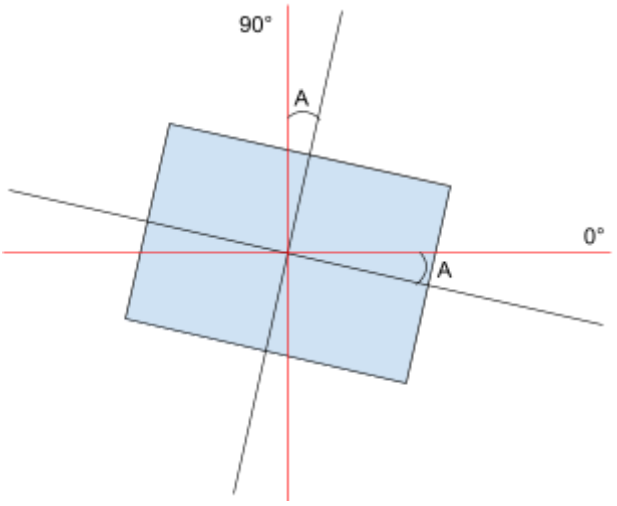


Ilustración 5.5.1.3 Ángulo de giro del documento

Al realizar un histograma de los ángulos ’A’ obtenidos como se muestra en la Ilustración 5.5.1.4, se obtiene un pico máximo indicando que sucesivas líneas detectadas en el documento coinciden en el ángulo, así como también la distribución de los demás ángulos detectados. Se tomó el pico máximo del histograma como eje y se eliminaron todas las líneas cuyo ángulo no estaba dentro de una desviación estándar para cada lado. El ángulo para la corrección de giro es el promedio resultante de los ángulos filtrados.

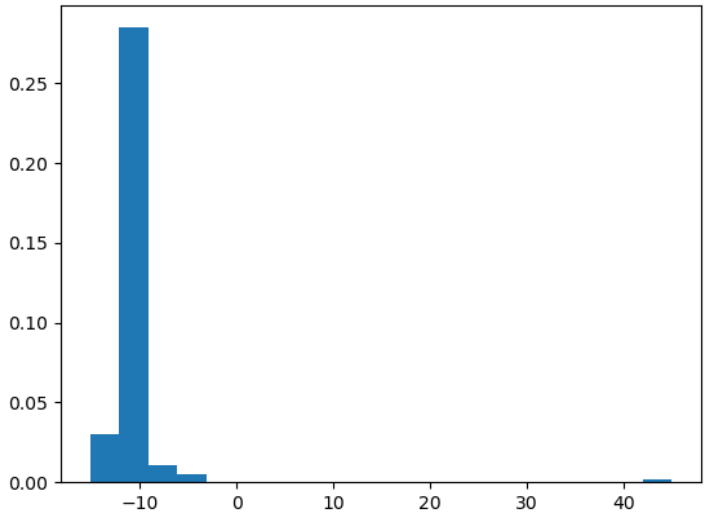


Ilustración 5.5.1.4 Histograma de ángulos en líneas detectadas

A partir de promediar los ángulos de las líneas filtradas, se obtiene una excelente aproximación para el ángulo de giro del documento como se puede observar en la Ilustración 5.5.1.5.

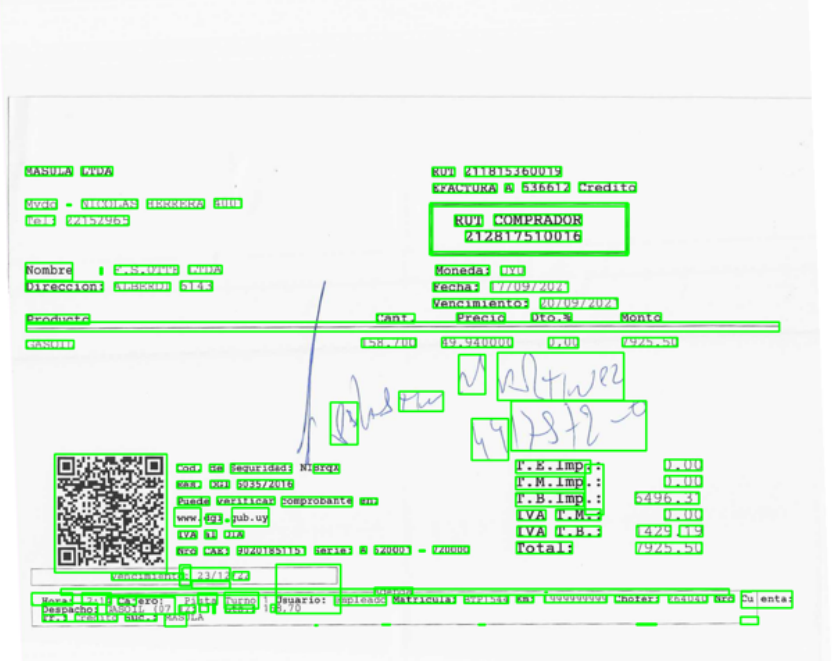


Ilustración 5.5.1.5 Corrección de ángulo sobre imagen original

La corrección obtenida alinea el documento pero no necesariamente orienta correctamente el texto.

Para orientar correctamente el documento se utilizó la herramienta de OCR llamada Google Tesseract OCR. Esta herramienta permite obtener una estimación de la orientación del texto, permitiendo corregir textos completamente volteados o girados 90°.

Basándose en la orientación de Tesseract, se realiza la rotación de la imagen obteniendo una copia del documento correctamente alineado y orientado.

5.5.2. Extracción de texto

En esta fase del procesamiento de la imagen se pretende obtener una transcripción del texto que aparece en el documento con la finalidad de procesarlo posteriormente.

Para realizar la detección del texto en la imagen a procesar se utilizara una herramienta de OCR, se evaluaron distintas alternativas disponibles en el mercado como Google API OCR Assistant, PyTesseract y Microsoft Azure OCR. Todas las herramientas procesan la imagen devolviendo los cuadros delimitadores de cada una de las palabras detectadas así como la transcripción para cada una de ellas.

En esta etapa se decidió utilizar la herramienta Tesseract ya que es gratuita y como veremos más adelante se lograron obtener muy buenos resultados al momento de entrenar los modelos. La evaluación se realizó empíricamente mirando manualmente el resultado de extracción para varias facturas distintas usando cada una de las herramientas evaluadas.

Utilizando la herramienta antes mencionada para la extracción de OCR, se extrae el texto del documento y se genera un archivo .json [48] con toda la información asociada al mismo.

Llegado a este punto se decide no continuar trabajando con las imágenes y se pasa a trabajar con los archivos .json generados, estos archivos contienen toda la información otorgada por el OCR además de toda la metadata relacionada al ajuste de la imágen.

El archivo json y la imágen tienen una relación de 1 a 1, para cada imágen existe un solo archivo json y para cada archivo json se asocia solamente una imágen, debido a esto es que podemos continuar solamente con los archivos json, y volver a la imágen solamente cuando se necesite realizar un cambio estético sobre la misma.

A partir de este punto se genera un nuevo *dataset* que replica el *dataset* original pero utilizando archivos .json en vez de imágenes, los cuales contienen la *metadata* obtenida de la imágen y los datos obtenidos del OCR, logrando de esta forma un *dataset* más liviano y práctico para entrenar nuestros modelos.

5.5.3. Limpieza de datos

La herramienta Tesseract nos devuelve el texto detectado, la posición y el recuadro que lo contiene, así como también la confianza que tiene la herramienta sobre la predicción realizada para cada una de las palabras detectadas.

En esta fase se procura filtrar y procesar los resultados obtenidos por el OCR, se realiza una limpieza básica donde se convierten todos los caracteres a minúsculas, se eliminan espacios y caracteres extraños, además se desestiman las palabras detectadas cuya confianza arrojada por Tesseract no cumple con un mínimo estipulado del 50% (seguridad que tiene la herramienta en cuanto al texto detectado).

El valor mínimo estipulado en la confianza al momento de filtrar los resultados se obtuvo empíricamente ya que las palabras detectadas con una confianza menor al 50% solían ser palabras inexistentes, con símbolos y letras que no se correspondían a una palabra válida.

Para contextualizar, lo que se busca eliminar son palabras detectadas que no aportan al problema como ser “n$4ssd” - Confianza 3% y quedarnos solamente con palabras válidas com “Subtotal” - Confianza 79%.

El valor de confianza refleja cuán seguro se encuentra Tesseract de que la palabra detectada corresponda a la transcripción generada, y es por eso que decidimos eliminar a las detecciones que no cuenten con la confianza suficiente.

Una vez filtrados y se procesados los resultados, se concatenan las palabras con espacios obteniendo un párrafo con todo el texto detectado en el documento el cual será utilizado para el entrenamiento de los modelos desarrollados, en resumen todo el texto obtenido de la imágen se transforma en un párrafo que es una simple concatenación de palabras pero carece de un sentido narrativo.

Se decide no generar procesamiento típico en tareas de NLP como stopwords, lematización, etc. ya que contamos con palabras y números independientes (el texto no contiene una narrativa). Se experimenta con estas técnicas únicamente con el algoritmo BERT.

Como resultado de esta etapa obtenemos un párrafo conteniendo todas las palabras detectadas como el que se muestra a continuación:

"CRISTAL e-Factura SERIE A 470251) CONTADO SILICONA RUC …"

5.5.4. Split del *dataset*

Luego de la fase de procesamiento, se separa el *dataset* en uno de entrenamiento y otro de *test*. Se decide dejar un 80% de los datos como entrenamiento y el 20% restante para *test*, una técnica bastante común. El *dataset* de *test* es utilizado como validación de los resultados de los modelos obtenidos (poder medir la capacidad de generalización en datos desconocidos).

En la Ilustración 5.5.4.1 se puede observar la distribución de los datos para los distintos proveedores existentes. Se identifica un desbalance entre la cantidad de documentos para cada emisor, por lo que el equipo generó el conjunto de entrenamiento y evaluación siguiendo una estrategia donde el muestreo sea del 80% y 20% respectivamente relativo a cada proveedor. Definiendo el parámetro "*stratify*" [49] en la función de *split* del *dataset* de Sklearn se logra separar los datos manteniendo 80% y 20% no solo en la

cantidad de datos sino también en la relación de elementos de cada clase, es decir, si la clase A tiene 100 facturas y la clase B tiene 200 facturas en total, 80 del grupo A y 160 del grupo B irán a *train*, mientras que el restante 20 del grupo A y 40 del grupo B irán a *test*.

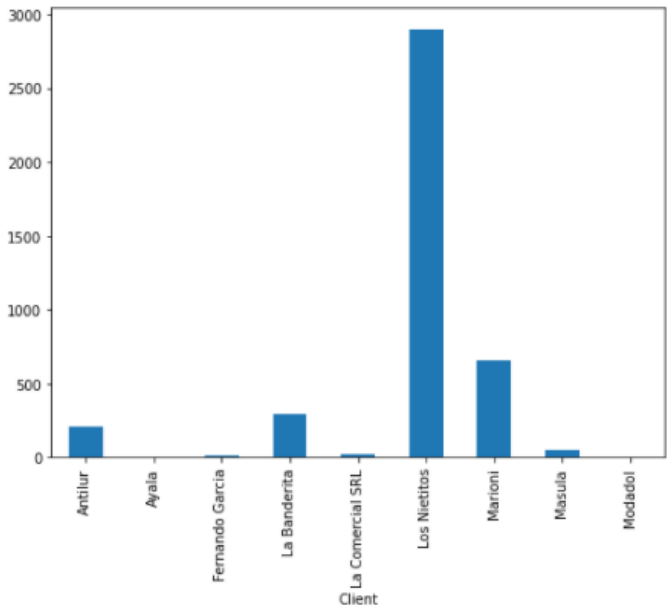


Ilustración 5.5.4.1 Distribución del *dataset*

5.5.5. Entrenamiento del modelo clasificador según proveedor

Se implementaron distintos modelos aplicando soluciones de *machine learning*, hasta modelos más avanzados de *deep learning* donde se aplican redes neuronales. El objetivo es comparar la performance de cada uno de ellos y decidir el modelo a utilizar en el prototipo. De los modelos de *machine learning* utilizados se implementó un clasificador Bayesiano, Random Forest, Linear SVC y Regresión Logística, en lo que respecta a los modelos de *deep learning* se implementó un modelo basado en imágenes como lo es Resnet-18 y un modelo de NLP perteneciente al estado del arte como lo es BERT.

Para utilizar modelos de *machine learning* es necesario procesar los datos y transformarlos a un lenguaje que los mismos puedan interpretar, por eso antes del entrenamiento se tuvo una fase de *feature engineering* [50] donde se convierte el texto

extraído a un formato de vectores numéricos. Esta transformación aplicada se denomina vectorización o tokenización y existen múltiples mecanismos para cumplir con este cometido.

En la fase de *feature engineering* se probaron dos técnicas de vectorización para el *dataset*:

- **CountVectorizer** [51]: Proporciona una manera simple de codificar una colección de palabras, construyendo un vocabulario de palabras basado en la cantidad de ocurrencias de las mismas. Devuelve un vector codificado con una longitud de todo el vocabulario (palabras únicas en el texto) y un número entero para el número de veces que cada palabra apareció en el documento. Para ser utilizado en la etapa de modelado comúnmente se realiza un corte del vector de las X palabras más frecuentes. Este método es bueno para utilizar como baseline, pero posee varios inconvenientes. Un problema y más cuando se considera un *dataset* con textos muy dispares, es que el vocabulario resultante se compone de palabras con un alto número de apariciones que quizás no sean muy significativas y que se descarten palabras muy relevantes pero que aparecen muy pocas veces.

- **TfidfVectorizer** [52]: Similar al método anterior pero en este caso se utiliza la frecuencia con la que las palabras aparecen dentro de un documento tomando en cuenta también la frecuencia relativa de las palabras en el *corpus* (conjunto de documentos). Trata de darle más importancia a las palabras más interesantes, generando un vocabulario rico y eficiente.

TfidfVectorizer tiene mayor inteligencia en la ponderación de las palabras que componen el vocabulario. Como era de esperarse, los resultados obtenidos más adelante en fase de evaluación decantaron a la elección de este algoritmo como estrategia de transformación del texto a vectores numéricos.

Luego de la vectorización se aplica PCA [53] (*Principal Component Analysis*) que es un procedimiento estadístico que convierte un conjunto de variables correlacionadas en un

conjunto de variables no correlacionadas. PCA es la herramienta más utilizada en el análisis exploratorio de datos y en el aprendizaje automático para modelos predictivos. Como resultado además se reduce la cantidad de columnas del *dataset* generado.

Teniendo el *dataset* vectorizado, se pasa a la definición de los modelos para poder entrenarlos. Como primera instancia se utilizaron varios modelos conocidos de *machine learning* 'clásico' como son *Random Forest*, *Linear* SVC , *Multinomial* NB y *Logistic Regression*.

Existen varias métricas o indicadores que se utilizan para evaluar el rendimiento de un clasificador. Las métricas más conocidas son *accuracy* [54], *precision* [55], *recall* [56] y *f1-score* [57]. Para entender cómo funcionan cada una de las métricas hace falta explicar previamente los conceptos que están presentes en sus definiciones.

La métrica *accuracy* es una medida que indica el porcentaje de predicciones que fueron acertadas. Se calcula como el total de predicciones correctas sobre el total de las predicciones. Por otro lado, las otras tres medidas son particulares para cada una de las clases que existen.

- La métrica *precision* es la cantidad de elementos de una clase clasificados correctamente, sobre la cantidad de elementos clasificados de esa clase en total.
- La métrica *recall* consiste en la cantidad de elementos clasificados correctamente de una clase, sobre la cantidad de elementos de esa clase.
- Por último, *f1-score* es una medida armónica que considera tanto la precisión (PR) como el *recall* (RC) y se calcula como:

$$F - SCORE = 2 * PR * RC/(PR + RC)$$

Debido al desbalance existente en el *dataset* se decide utilizar la métrica *f1_macro* en vez de *accuracy* ya que logra puntuar mejor los resultados de los modelos en este tipo de *datasets*. La métrica *accuracy* no es eficaz para evaluar en *datasets* desbalanceados ya que no pondera el error relativo por clases, por tanto, si el modelo aprende a clasificar bien únicamente la clase con gran volúmen de datos la métrica dará buenos resultados no siendo una realidad en el poder de generalización para todos los grupos.

En cambio la métrica *f1_macro* toma en cuenta el desbalance de clases y la puntuación no será positiva si el modelo no tiene buen poder de generalización para todas las clases.

Para realizar una comparación entre los modelos detallados se utilizó la técnica de *K-Fold Crossval* [58] con 5 *folds.* Esta técnica consiste en replicar 5 veces el *dataset* de entrenamiento para cada modelo, luego se separa aleatoriamente cada uno en un nuevo *dataset* de entrenamiento y de test. Los resultados obtenidos para cada uno de los cinco entrenamientos de cada modelo son representados mediante un diagrama de cajas como se puede ver en la Ilustración 5.5.5.1.

En esta primera instancia *Linear SVC (Support Vector Classifier)* fue el algoritmo que obtuvo mejores resultados, logrando una puntuación prácticamente de 100% *f1_score*, seguido por *Logistic Regression* y *Multinomial NB*.

Linear SVC además de obtener una puntuación muy elevada también demostró tener muy poca variabilidad al realizar la validación cruzada, ya que los cinco resultados de las evaluaciones fueron prácticamente iguales entre sí, indicando que este modelo es una excelente opción para resolver el problema que se enfrenta.

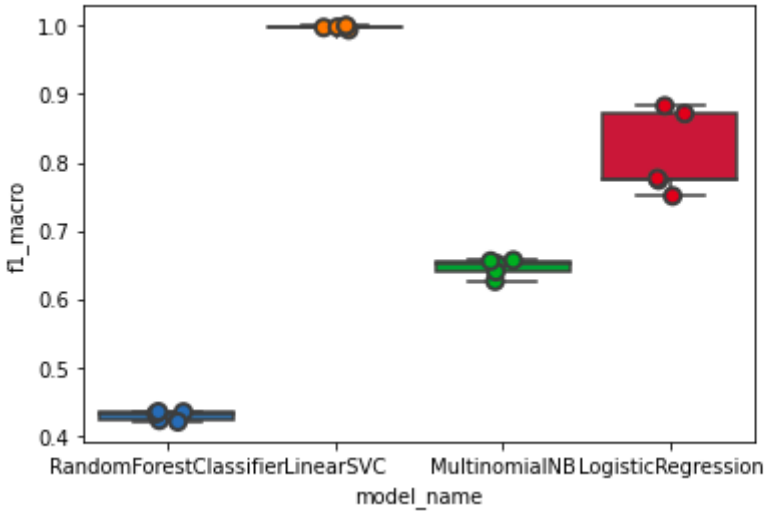


Ilustración 5.5.5.1 Comparación de modelos

Se evaluaron modelos basados en aprendizaje no supervisado como K-Means [59],donde se intenta categorizar el *dataset* a partir de la clusterización del mismo. El objetivo era experimentar si la clusterización lograba buenos resultados para categorizar los documentos. El algoritmo K-Means conoce de antemano el número de categorías en la cual debe separar los documentos, en nuestro caso además se agregan semillas iniciales con un documento por cada categoría ya que ésto aporta valor para la convergencia y una mejor categorización. Dada las posiciones iniciales y la cantidad de categorías, el algoritmo separa el *dataset* de manera no supervisada.

En la Ilustración 5.5.5.2 se puede visualizar a la izquierda una representación gráfica de los vectores correspondientes a los documentos y su categoría representada con los distintos colores. A la derecha se puede ver el resultado de la clusterización obtenida a través del algoritmo de K-Means, podemos observar que la categorización obtenida parece ser muy semejante a las categorías originales.

Los resultados obtenidos para este modelo fue de un *f1_macro* de ~88%, ubicándolo en el *top-3* de algoritmos obtenidos pero con un puntaje por debajo de *Linear SVC* y comparable con *Logistic Regression.*

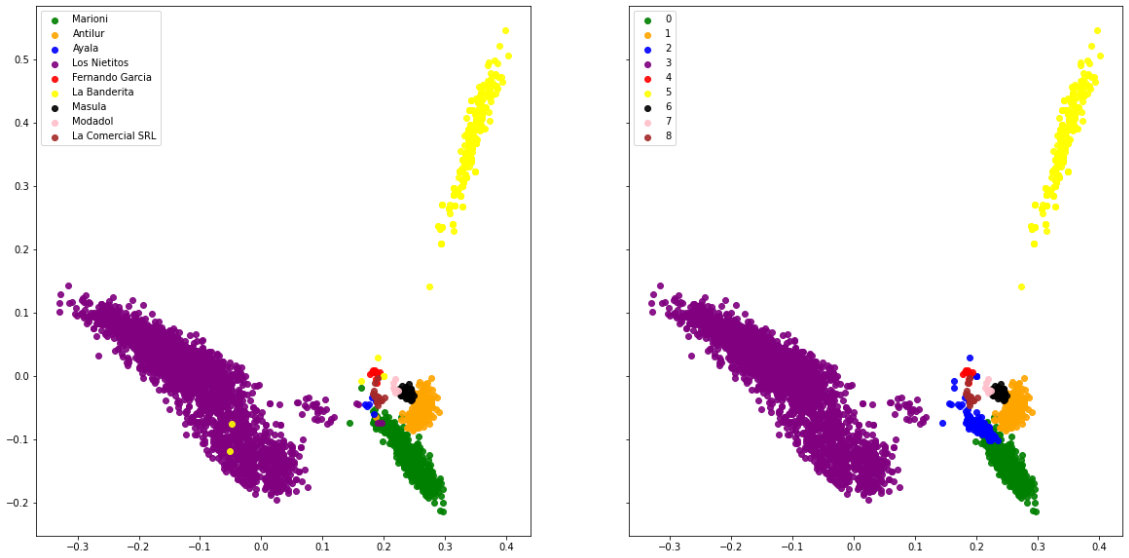
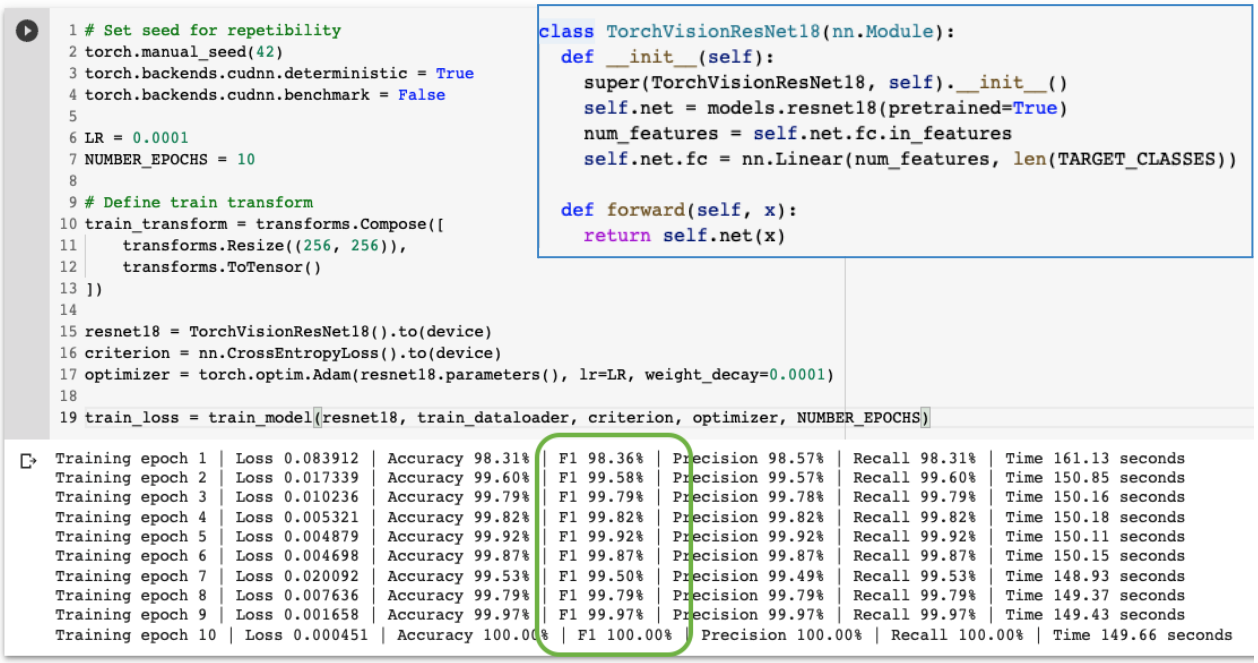


Ilustración 5.5.5.2 Modelado K-Means

Otra manera de abordar la clasificación de los documentos puede ser utilizando modelos basados en procesamiento de imágen. En particular, se experimentó con el modelo de aprendizaje profundo *Resnet-18* pre-entrenado de Pytorch, modificando la capa final de acuerdo a la cantidad de clases existentes como se puede apreciar en la Ilustración 5.5.5.3. Se alimenta el modelo directamente con las imágenes de los documentos, y a la salida el modelo devuelve las probabilidades para cada clase. Los resultados obtenidos fueron excelentes obteniendo un *f1_score* de 100% para entrenamiento y *f1_score* de 99.85% para el conjunto de test. Esto nos dice que una estrategia basada en diferenciar facturas desde su visual y no desde su contenido de texto es sumamente eficaz para este problema en particular, siempre y cuando los formatos visuales de las facturas sean diversos.

En la Ilustración 5.5.5.3. se aprecia el proceso de entrenamiento del modelo junto con la asignación de sus hiper-parámetros. Se realizaron modificaciones sobre la arquitectura del modelo *Resnet-18* pre-entrenado de Pytorch para adaptarla al problema atacado en particular (9 posibles emisores a predecir).



```
# Set seed for repetibility
torch.manual_seed(42)
torch.backends.cudnn.deterministic = True
torch.backends.cudnn.benchmark = False

LR = 0.0001
NUMBER_EPOCHS = 10

# Define train transform
train_transform = transforms.Compose([
    transforms.Resize((256, 256)),
    transforms.ToTensor()
])

resnet18 = TorchVisionResNet18().to(device)
criterion = nn.CrossEntropyLoss().to(device)
optimizer = torch.optim.Adam(resnet18.parameters(), lr=LR, weight_decay=0.0001)

train_loss = train_model(resnet18, train_dataloader, criterion, optimizer, NUMBER_EPOCHS)
```

```
class TorchVisionResNet18(nn.Module):
  def __init__(self):
    super(TorchVisionResNet18, self).__init__()
    self.net = models.resnet18(pretrained=True)
    num_features = self.net.fc.in_features
    self.net.fc = nn.Linear(num_features, len(TARGET_CLASSES))

  def forward(self, x):
    return self.net(x)
```

```
Training epoch 1 | Loss 0.083912 | Accuracy 98.31% | F1 98.36% | Precision 98.57% | Recall 98.31% | Time 161.13 seconds
Training epoch 2 | Loss 0.017339 | Accuracy 99.60% | F1 99.58% | Precision 99.57% | Recall 99.60% | Time 150.85 seconds
Training epoch 3 | Loss 0.010236 | Accuracy 99.79% | F1 99.79% | Precision 99.78% | Recall 99.79% | Time 150.16 seconds
Training epoch 4 | Loss 0.005321 | Accuracy 99.82% | F1 99.82% | Precision 99.82% | Recall 99.82% | Time 150.18 seconds
Training epoch 5 | Loss 0.004879 | Accuracy 99.92% | F1 99.92% | Precision 99.92% | Recall 99.92% | Time 150.11 seconds
Training epoch 6 | Loss 0.004698 | Accuracy 99.87% | F1 99.87% | Precision 99.87% | Recall 99.87% | Time 150.15 seconds
Training epoch 7 | Loss 0.020092 | Accuracy 99.53% | F1 99.50% | Precision 99.49% | Recall 99.53% | Time 148.93 seconds
Training epoch 8 | Loss 0.007636 | Accuracy 99.79% | F1 99.79% | Precision 99.79% | Recall 99.79% | Time 149.37 seconds
Training epoch 9 | Loss 0.001658 | Accuracy 99.97% | F1 99.97% | Precision 99.97% | Recall 99.97% | Time 149.43 seconds
Training epoch 10 | Loss 0.000451 | Accuracy 100.00% | F1 100.00% | Precision 100.00% | Recall 100.00% | Time 149.66 seconds
```

Ilustración 5.5.5.3 Arquitectura y Entrenamiento Resnet-18

Por último se experimentó con un modelo perteneciente al estado del arte *BERT*, modelo de NLP basado en *transformers*. Para esto se debió realizar un proceso adicional de lematización [60] previo a la vectorización (por ejemplo se transforman los verbos a su forma normal para no tomar como distintas palabras un verbo en distintas conjugaciones).

Se utilizó una implementación de *BERT* pre-entrando y se modificaron las capas finales para clasificar las distintas categorías existentes. Sorprendentemente los resultados fueron muy malos, obteniendo un *f1_score* apenas de 21% para el conjunto de *train*, lo cual nos sorprendió ya que no era esperado dado el gran poder de aprendizaje de este tipo de modelo. El pobre resultado obtenido queda evidenciado en la matriz de confusión que se detalla en la Ilustración 5.5.5.3. Dada la limitación de tiempo y la necesidad de experimentar con varios algoritmos distintos, se utilizó una herramienta de AutoML (AutoGluon) la cuál provee automatización de BERT. Somos conscientes que los resultados serían mejores si hubiéramos entrenado un modelo BETO el cuál se basa sobre idioma español.

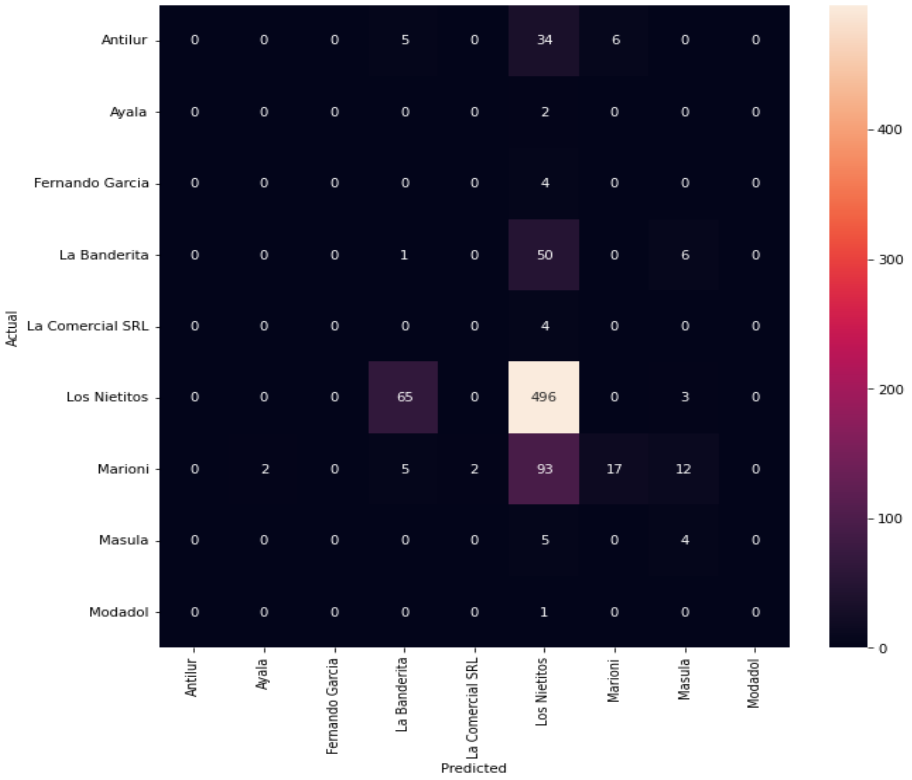


Ilustración 5.5.5.4 Matriz de confusión [62] de BERT

Luego de analizar el conjunto de modelos implementados se obtuvieron los siguientes resultados en *test*:

- *Linear SVC* con una metrica *f1_score* ~ 99.8%
- *Resnet-18* con una metrica *f1_score* ~ 99.8%
- *K-Means* con una metrica *f1_score* ~ 88%
- *Logistic Regression* con una metrica *f1_score* ~ 85%
- *MultinomialNB* con una metrica *f1_score* ~ 66%
- *Random Forest Classifier* con una metrica *f1_score* ~ 43%
- *BERT* con una metrica *f1_score* ~ 21%

Tanto *Linear SVC* como *Resnet-18* destacan por sobre los demás en la categorización del *dataset*, ambos obtuvieron la mejor métrica *f1_score* siendo de 99.8% en promedio.

Se toma la decisión de utilizar *Linear SVC* por sobre *Resnet-18*, ya que el segundo posee una elevada complejidad además de necesitar muchos más recursos como memoria y disponibilidad de GPU. *Linear SVC* demostró tener excelentes resultados, con muy poca variabilidad al realizar la validación cruzada de los datos, y es un modelo considerablemente menos complejo.

A priori no nos explicamos por qué modelos clásicos de *machine learning* lograban obtener una *performance* muy superior a modelos más complejos y que actualmente corresponden al estado del arte. Analizando los datos vimos que a diferencia del procesamiento NLP, el texto de nuestro *dataset* no tiene una semántica compleja ni sigue una estructura comparable a un texto escrito. En nuestro caso quizás tan solo una palabra sea suficiente para poder inferir correctamente la categoría. Analizar frecuencias o ocurrencias de determinadas palabras es suficiente y muy efectivo para detectar la categoría correspondiente y es por tanto que creemos que esa es la razón por la cual los algoritmos de predicción clásicos tuvieron un mejor desempeño.

5.5.6. Evaluación del mejor modelo en datos de test

Una vez entrenado el modelo de *Linear SVC*, se ejecuta el mismo sobre el conjunto de *test* para obtener resultados ante datos no vistos durante el entrenamiento.

En la Ilustración 5.5.6.1 se presenta la matriz de confusión obtenida de los datos de test para el modelo *Linear SVC*.

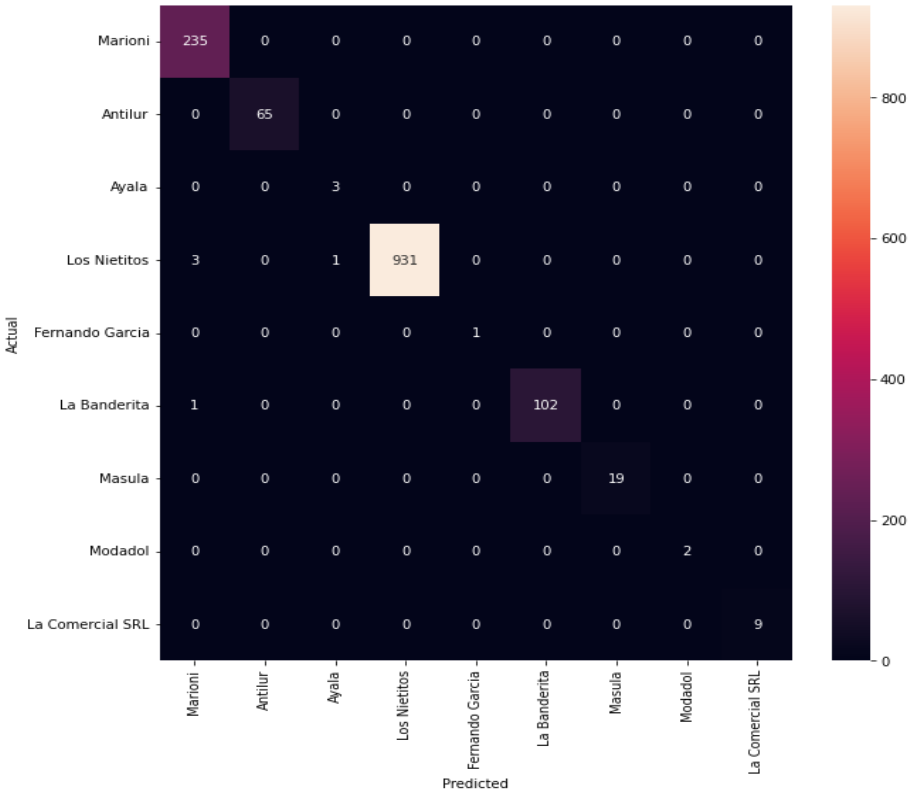


Ilustración 5.5.6.1 Matriz de confusión Linear SVC

De los 1372 documentos pertenecientes al *dataset* de test se categorizaron correctamente 1367, de los documentos mal categorizados cuatro pertenecen a la categoría "Los Nietitos" y uno corresponde a la categoría "La Banderita" arrojando como resultado un *f1-macro* de ~99.6%.

El resultado obtenido por el modelo en el conjunto de datos de *test* supera ampliamente las expectativas, categorizando consistentemente los documentos procesados y demostrando que es un modelo sólido para categorizar correctamente los documentos.

5.5.7. Extracción de campos relevantes

Como siguiente paso en el procesamiento de los documentos se deberán obtener campos relevantes para cada proveedor, como ser número de documento, fecha de emisión, receptor, totales, etc. Vale aclarar que este paso es dependiente de la clasificación del proveedor realizada en el paso anterior, ya que el template utilizado será el correspondiente al proveedor clasificado.

Se empleó la técnica de *template matching*, la cual nos permite a partir de un *template* de ejemplo, transformar la imágen procesada para que coincida con la imágen de referencia, para luego obtener los campos de interés.

Se genera un *template* por cada emisor indicando los recuadros de donde se encuentran los datos relevantes, como se puede ver en la ilustración 5.5.7.1 se seleccionan los campos “Total”, “Cliente”, “Fecha” y “Número de documento”. Los campos a extraer pueden variar dependiendo del proveedor.

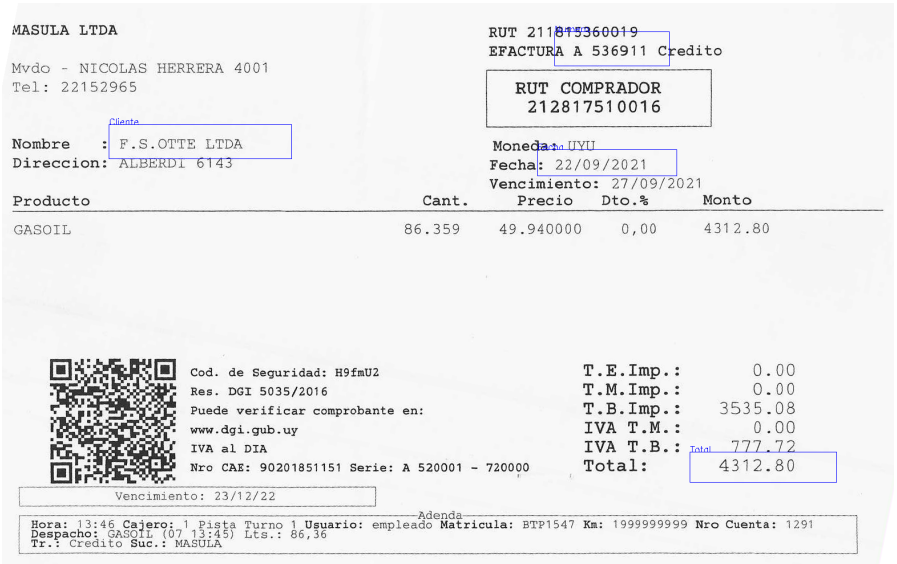


Ilustración 5.5.7.1 Template para extracción de datos

Una vez categorizado el documento se realiza un ajuste con el *template* correspondiente al emisor. El ajuste se realiza mediante la aplicación del algoritmo *Scale-Invariant Feature Transform* (SIFT), el cual nos permite detectar el template en la imágen procesada.

El algoritmo SIFT basa su funcionamiento en detectar puntos característicos entre dos imágenes y así poder determinar la transformación homográfica que existe entre ambas. La homografía nos establece que dos imágenes cualesquiera de una misma superficie plana se relacionan mediante una transformación homográfica, esta transformación nos permite rotar, recortar y corregir perspectivas en la imágen procesada.

Cómo se había detallado anteriormente el *template* utilizado en este paso del procesamiento corresponde al emisor al cual fue clasificado el documento. El algoritmo utilizado cuenta con un umbral mínimo de puntos necesarios para considerar un correcto ajuste del *template* con la imágen procesada. En caso de que la cantidad de puntos de ajuste no sea superior al umbral, la imágen es catalogada como errónea y se separa para su revisión manual.

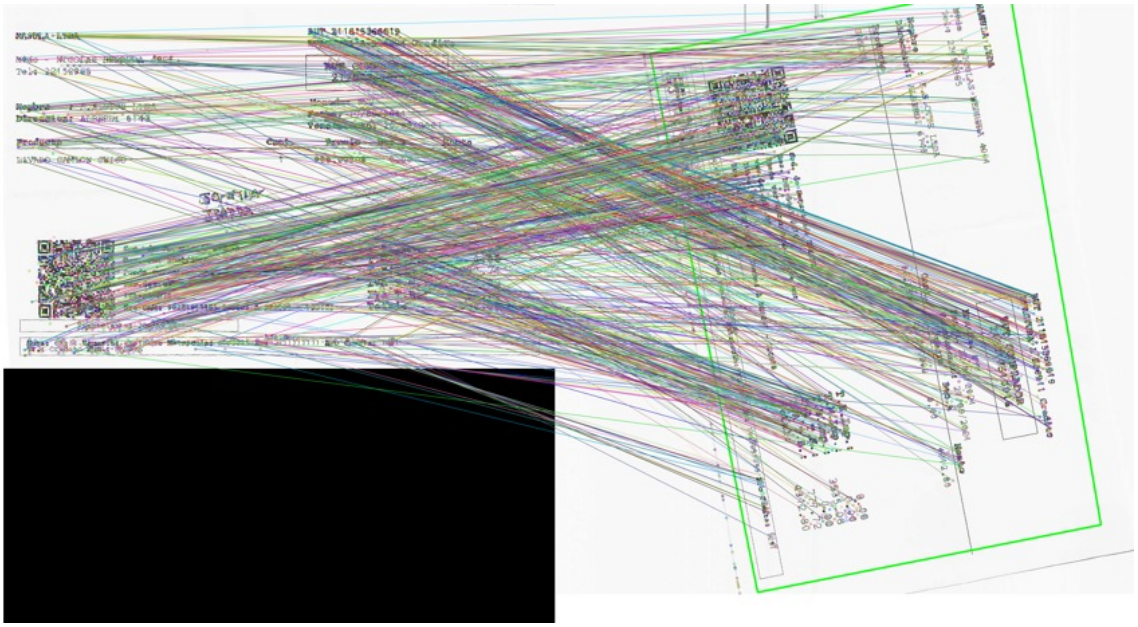


Ilustración 5.5.7.2 Algoritmo SIFT para ajuste de template en imágen

Como podemos observar en la Ilustración 5.5.7.2 el algoritmo SIFT detecta y relaciona puntos característicos entre el *template* y la imágen a procesar, obteniendo como resultado en verde la figura de recorte de la imágen. La imágen utilizada como *template* es similar pero es diferente a la imágen procesada, el algoritmo solamente relaciona los puntos que si son ajustables entre las dos imágenes, obteniendo como resultado la imágen que se muestra en la Ilustración 5.5.7.3.

Ilustración 5.5.7.3 Resultado recorte del ajuste de template

Una vez realizada la transformación y recorte se procede a extraer los campos de texto mediante OCR, en esta segunda instancia se decidió utilizar la API de OCR de Azure ya que esta mostraba mejores resultados al procesar las imágenes.

A partir de los *bounding boxes* de las regiones de interés en el template y los resultados de las palabras obtenidas por el OCR, se buscan los textos que están contenidos en cada una de las regiones de interés del template en la imágen procesada. La Ilustración 5.5.7.4 detalla gráficamente este procedimiento.

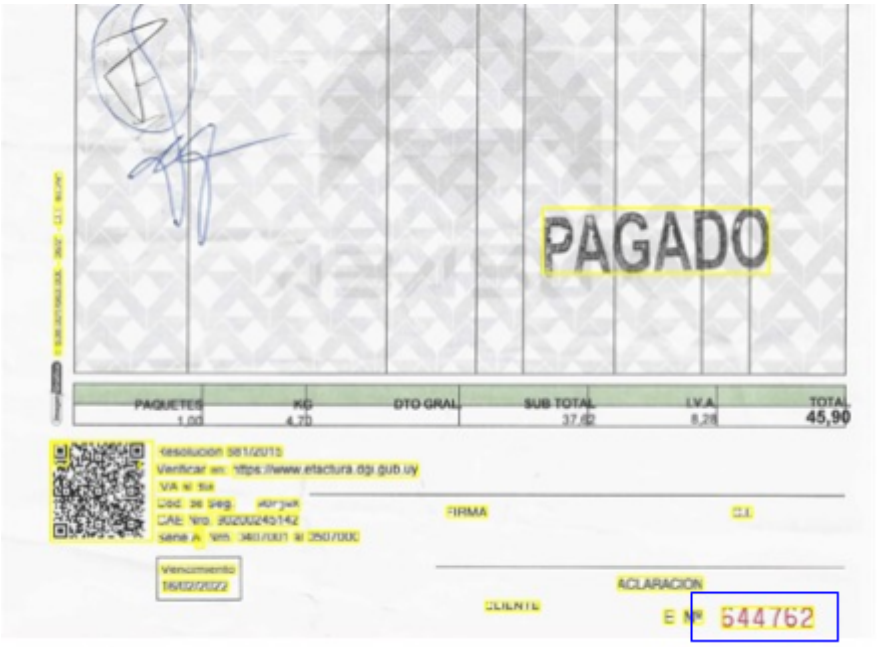


Ilustración 5.5.7.4 Ajuste OCR (Amarillo) vs Template (Azul)

Como podemos ver en la Ilustración 5.5.7.4 los recuadros amarillos se corresponden a las palabras detectadas por el OCR, y en azul se indica el recuadro del *template* correspondiente en este caso al número de documento. El programa filtra solamente las palabras cuyo centro esté incluido en el área interior del rectángulo de *template*, en nuestro caso la palabra del número de documento sería “544762”, quedando excluida la palabra “B-N°” ya que el centro de la misma se encuentra por fuera del rectángulo de *template*.

Los campos detectados son guardados en el archivo de *metadata* asociado a la imágen, indicando el campo correspondiente y el valor obtenido.

Previo a seleccionar esta estrategia para el cometido de extraer campos relevantes, se evaluaron otras alternativas para éste requerimiento como ser:

- Google Document AI [63]: *software-as-a-service* de Google donde dado un documento PDF, el mismo retorna un mapeo de campos especificando un valor asociado al nombre del campo (ej: Fecha: “01/01/2022”).
- Invoice2Data [64]: librería *open-source* para la detección de campos basada en expresiones regulares.
- InvoiceNet [65]: librería *open-source* basado en redes neuronales profundas para la extracción inteligente de campos en documentos.

Dado que éstas herramientas o bien tienen costos para su uso como ser el servicio de Google o una curva de aprendizaje bastante pronunciada para el caso de las dos librerías, se decidió continuar la solución utilizando el algoritmo SIFT ya que se obtuvieron muy buenos resultados.

5.5.8. Persistencia de información extraída

El último paso del pipeline corresponde a la persistencia de los datos relevantes extraídos de la factura. Como se mencionó anteriormente, los mismos pueden variar para cada cliente pero es de interés dejarlos disponibles de manera accesible. Hasta este punto los mismos se conservan en un archivo .json. En pasos futuros podría ofrecerse una visualización más amigable mediante una interfaz que permite al usuario interactuar con los datos.

5.6. Pipeline de entrenamiento y producción

Se generó un *pipeline* de datos apoyado sobre la tecnología DVC. En una primera instancia se puso foco en lograr tener un sistema *end-to-end*, esto permite evaluar el sistema en su totalidad y a su vez ir iterando en cada paso de manera independiente. El *pipeline* cuenta con dos grandes fases:

Fase de entrenamiento: Abarca todos los pasos correspondientes al entrenamiento del modelo de clasificación, desde el pre-procesamiento de la factura y extracción de los datos, hasta la generación del modelo dejándolo disponible para su uso en producción. Es decir, se comienza con un conjunto de imágenes etiquetadas manualmente, luego se van procesando y cada factura se pasa por el pre-ajuste para aplicación del OCR, encargado de enderezar la factura de manera correcta y alinear los datos presentes en la misma. Se extrae el texto que contiene a través de la herramienta OCR elegida. Esta información se guarda obteniendo un *dataset* de archivos .json con toda la información necesaria para entrenar el modelo deseado. Una vez generado el *dataset* se pasa a la fase de limpieza del mismo para luego proceder con la separación del mismo en un conjunto de entrenamiento y otro de *test*. El siguiente paso consta del entrenamiento donde se genera un modelo capaz de clasificar a las facturas con grandes resultados. Una vez evaluado el modelo con el *dataset* de *test* y habiendo sido aprobado, se persiste el modelo obtenido para ser utilizado en producción.

Fase de producción: En esta fase se reciben las facturas nuevas a inferir, las mismas se pre-procesan y se aplica limpieza al igual que en la fase de entrenamiento. Luego se clasifica la misma con el modelo que fue determinado por el *pipeline* de entrenamiento. Se ajusta el *template* correspondiente al emisor detectado y se extraen los datos objetivos. Estos datos son almacenados en un archivo .json los cuales se persisten para luego ser presentados al cliente de la manera que se desee.

A continuación se muestran las distintas fases para cada paso (entrenamiento y producción):

Ilustración 5.6.1: Pipeline de entrenamiento

Ilustración 5.6.1: Pipeline de producción

5.7. Preparación de un nuevo proveedor en el sistema

Ante la llegada de un nuevo proveedor se debe contar con al menos 20 ejemplares disponibles para poder agregarlos al conjunto de datos de entrenamiento y a su vez generar el *template* correspondiente a utilizar en la fase de producción.

El *template* es generado una única vez por proveedor, este *template* se utiliza en la fase de producción para la extracción de datos relevantes y se pueden elegir todos los campos que se consideren importantes.

Empíricamente se prueba que entrenar y procesar el flujo de documentos con menos de 20 ejemplares (de un mismo proveedor) el modelo no es capaz de generar predicciones precisas.

5.8. Atributos de calidad

En lo que refiere a atributos de calidad, tomando en cuenta los distintos requerimientos no funcionales relevados, el equipo decide favorecer:

- Escalabilidad
- Disponibilidad
- Seguridad

A continuación se detallan las decisiones de diseño tomadas por el equipo para favorecer cada uno de los atributos de calidad presentados.

5.8.1. Escalabilidad

Tomando en cuenta que el equipo desea continuar iterando sobre este proyecto más allá de la entrega de proyecto final, con el objetivo de generar un nicho de negocio a largo plazo, desde el inicio del diseño se considera el posible crecimiento en cantidad de clientes. Esto implica soportar un aumento tanto en el volúmen de documentos a procesar, así como también en la cantidad de proveedores a clasificar.

Para atacar este desafío y aprovechando las funcionalidades de la herramienta DVC, el equipo planea generar una instancia del sistema (escalando horizontalmente) por cada cliente. Teniendo un modelo predictivo, reglas de extracción de campos, etc por cliente.

Una de las tácticas utilizadas implica el procesamiento por lotes ya que la ejecución del pipeline de producción se realiza periódicamente bajo un rango de tiempo configurable por el administrador del sistema. Esto permite que los nodos sean autosuficientes e independientes entre sí, logrando una distribución del procesamiento de manera eficiente.

5.8.2. Disponibilidad

En lo que refiere a la disponibilidad, se aplican tácticas tanto asociadas al manejo de excepciones como al *rollback* para reintento de procesamiento. Cuando el sistema experimenta cualquier tipo de falla, se identifican y manejan las excepciones asociadas a la misma. Mediante este mecanismo no se ignora la falla, sino que se delega el error hasta el nivel de interacción con el usuario, notificando del acontecimiento y posteriormente reintentando el procesamiento.

Omitir o procesar incorrectamente una o múltiples facturas puede ser crítico para la salud financiera de la empresa. La idea es contar con un mecanismo *human-in-the-loop* [70] para revisar de manera proactiva el procesamiento generado por el sistema. Ésto permite detectar fallas así como también cuando es necesario re-entrenar el/los modelos de *machine learning* cuando ocurre el fenómeno de *data-drift* [71]. Por éste motivo se mantienen versionados los distintos modelos entrenados, resultados intermedios del *pipeline* y sus métricas asociadas. En lugar de tener un recurso humano procesando facturas a mano, se lo utiliza para revisar y re-entrenar el modelo cuando se detecta deterioro en la performance del procesamiento.

5.8.3. Seguridad

Uno de los requerimientos no funcionales más importantes es la privacidad de los datos, punto sumamente importante en la mayoría de los proyectos de *big data*. Para esto, el equipo implementa un mecanismo de autenticación con el objetivo de procesar

documentos subidos únicamente por usuarios con accesos garantizados para interactuar con el sistema. Básicamente, cada cliente cuenta con un *token* único asociado al almacenamiento de sus datos en la nube, con la opción de ser modificado cuando sea necesario. Otra de las ventajas de escalar horizontalmente con un nodo por cliente, es que fácilmente se desacopla el procesamiento de documentos asociados a distintos clientes en distintos nodos independientes. Más allá que no se considera a una factura como información crítica de la empresa, el equipo decide hacer foco en la privacidad de los datos.

6. Conclusiones

En lo que refiere a objetivos, se logra implementar una solución *end-to-end* (MVP) utilizando lineamientos y estándares de la industria, obteniendo un resultado que supera las expectativas de todos los *stakeholders*. Por un lado, mediante múltiples instancias de intercambio con el tutor, se confirma el cumplimiento de objetivos académicos en lo que respecta a la universidad. Se logra un alto nivel de autosuficiencia por parte del equipo en lo asociado a resolución de problemas y planificación del trabajo. Se balanceó correctamente la dedicación en cuanto a investigación y desarrollo, sacando máximo provecho de lo estudiado durante los distintos cursos del máster. Por otro lado, la empresa F. S. Otte Ltda. se muestra muy interesada en incorporar el software a su operativa contable luego de que el equipo presentó una *demo* del producto funcionando. Ésto permite asegurar un alto nivel de satisfacción de las partes interesadas.

Los resultados obtenidos en la clasificación por parte del modelo implementado logró cumplir con los objetivos que se habían planteado al comienzo del documento. El modelo seleccionado *Linear SVC* siendo una solución basada en *machine learning* clásico obtuvo una puntuación *f1_macro* de 99.6%, asi como también otros modelos más complejos obtuvieron resultados similares. Se aplicó del principio de Navaja de Ockham [66] (también conocido como ley de parsimonia), donde básicamente siempre

se recomienda elegir el modelo más simple y eficiente ante resultados similares de generalización en inferencia.

También se quiere destacar el aprendizaje sobre extracción de campos relevantes en documentos con formatos definidos. Se experimentaron variadas alternativas y mediante su debida justificación (detallada previamente en la presente sección) se optó por usar estrategia de *templating* para extraer los datos obteniendo excelentes resultados.

El equipo decide disponibilizar todo el código fuente [67] en su debido repositorio para su posible revisión y/o sugerencia de mejoras (en la referencia correspondiente se encuentra el *link* al mismo). En la carpeta */notebooks* se pueden encontrar todas las *Jupyter notebooks* con los experimentos realizados en la fase de modelado (algoritmos de *machine learning* y *deep learning*), la estructura del proyecto queda disponible en "Anexo 1 - Estructura del proyecto".

Se desea expresar el alto nivel de satisfacción del equipo para con la resolución de los problemas y desafíos técnicos presentados. Se ha logrado trabajar en complemento mediante los integrantes aprovechando las distintas habilidades de cada uno y lograr todos los objetivos planteados al comienzo del documento.

7. Próximos pasos

Como se menciona previamente en el documento, el objetivo del equipo para este proyecto final de máster, es contar con un MVP *end-to-end* funcionando con un conjunto de clientes. Igualmente, las aspiraciones a futuro implican continuar iterando en la solución con un gran foco en el posible negocio para salir al mercado uruguayo (y potencialmente globalizar la solución).

Para cumplir con estos objetivos a futuro, el equipo pretende continuar trabajando sobre los siguientes puntos:

1. Migrar la solución de extracción de campos relevantes utilizando un modelo de *deep learning* que logre detectar campos relevantes sobre facturas nunca antes vistas en entrenamiento.
2. Luego de fases de *testing*, el equipo percibe la existencia de un caso borde a corregir en cuanto a la extracción de campos relevantes. El mismo consiste en la extracción de campos relevantes para facturas del proveedor Fernando García S.A., donde algunos campos varían en posición dentro del documento dependiendo de la cantidad de líneas de factura.
3. Para escalar horizontalmente (un nodo en producción por cliente), el equipo considera necesario integrar la tecnología Docker [68] para facilitar el proceso de despliegue.
4. Desde el punto de vista de negocio, se pretende entrevistar empresas interesadas y comenzar a planificar el modelo de negocio detrás de esta solución tecnológica.

Referencias

[1] Urban, Tim. "*The Artificial Intelligence Revolution: Part 1.*" en Wait But Why, 2015. [En línea]. Disponible en: https://waitbutwhy.com/2015/01/artificial-intelligence-revolution-1.html.

[2] "Automation Document Processing - España." IBM. [En línea]. Disponible en: https://www.ibm.com/es-es/cloud/document-processing.

[3] "Nanonets: Intelligent document processing with AI". [En línea]. Disponible en: https://nanonets.com/.

[4] "What is Requirements Gathering Process and its Tools?" ReQtest. [En línea]. Disponible en: https://reqtest.com/requirements-blog/what-is-requirements-gathering/.

[5] Heinrichs, Florian. "*Using CRISP-DM to Grow as Data Scientist.*" Towards Data Science. [En línea]. Disponible en: https://towardsdatascience.com/using-crisp-dm-to-grow-as-data-scientist-a07ce3fd9d56.

[6] Andrew Ng. (2018). "Build your first system quickly, then iterate". En "*Machine Learning Yearning. Technical strategy for AI engineers, in the era of deep learning*" (pp. 29). Stanford, CA

[7] "Welcome to Python.org". [En línea]. Disponible en: https://www.python.org/

[8] "Poetry - Python dependency management and packaging made easy". [En línea]. Disponible en: https://python-poetry.org/

[9] "Git SCM". [En línea]. Disponible en: https://git-scm.com/

[10] "GitHub: Where the world builds software" GitHub. [En línea]. Disponible en: https://github.com/

[11] "Data Version Control" DVC. [En línea]. Disponible en https://dvc.org/

[12] “Website title:Iterate faster, innovate together” GitLab. [En línea]. Disponible en: https://gitlab.com/

[13] “Cloud Storage” Google Cloud. [En línea]. Disponible en: https://cloud.google.com/storage

[14] “Cloud Storage for Work and Home” Google. [En línea]. Disponible en: https://www.google.com/drive/

[15] “Cloud Object Storage – Amazon S3” Amazon Web Services. [En línea]. Disponible en: https://aws.amazon.com/s3/

[16] “File Transfer Protocol.” Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/File_Transfer_Protocol.

[17] “SFTP protocol, clients, servers etc. Page by the original author of SFTP.” SSH Communications Security. [En línea]. Disponible en: https://www.ssh.com/academy/ssh/sftp

[18] Dalgaard, Peter. “The R Project for Statistical Computing: R”. [En línea]. Disponible en: https://www.r-project.org/

[19] “Overview - Spark 3.2.1 Documentation.” Apache Spark. [En línea]. Disponible en: https://spark.apache.org/docs/latest/index.html

[20] “Project Jupyter | Home”. [En línea]. Disponible en: https://jupyter.org/

[21] “TensorFlow”. [En línea]. Disponible en: https://www.tensorflow.org/

[22] “PyTorch.org”. [En línea]. Disponible en: https://www.pytorch.org/

[23] “pandas - Python Data Analysis Library”. [En línea]. Disponible en: https://pandas.pydata.org/

[24] “scikit-learn: machine learning in Python — scikit-learn 1.0.2 documentation”. [En línea]. Disponible en: https://scikit-learn.org/stable/

[25] “OpenCV: Home”. [En línea]. Disponible en: https://opencv.org/

[26] “OpenCV: Hough Line Transform.” OpenCV documentation. [En línea]. Disponible en: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

[27] “OpenCV: Introduction to SIFT (Scale-Invariant Feature Transform).” OpenCV documentation. [En línea]. Disponible en: https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

[28] “Microsoft Azure: Cloud Computing Services”. [En línea]. Disponible en: https://azure.microsoft.com/en-us/

[29] “Computer Vision.” Microsoft Azure. [En línea]. Disponible en: https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/

[30] “Google”. [En línea]. Disponible en: https://google.com

[31] “Vision AI | Derive Image Insights via ML | Cloud Vision API.” Google Cloud. [En línea]. Disponible en: https://cloud.google.com/vision

[32] “Tesseract documentation | Tesseract OCR”. [En línea]. Disponible en: https://tesseract-ocr.github.io/

[33] “Searle, John. “Natural language processing.” Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Natural_language_processing

[34] Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova. (2019). “*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”*. pp. 1-16

[35] Chaudhuri, Koushiki Dasgupta. “Building Language Models in NLP.” Analytics Vidhya. [En línea]. Disponible en: https://www.analyticsvidhya.com/blog/2022/01/building-language-models-in-nlp/

[36] “Masked-Language Modeling With BERT | by James Briggs.” Towards Data Science. [En línea]. Disponible en: https://towardsdatascience.com/masked-language-modelling-with-bert-7d49793e5d2c

[37] “Word2vec.” Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Word2vec

[38] “Transformer (machine learning model).” Wikipedia [En línea]. Disponible en: https://en.wikipedia.org/wiki/Transformer_(machine_learning_model)

[39] “1.9. Naive Bayes — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/naive_bayes

[40] “sklearn.ensemble.RandomForestClassifier — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[41] “sklearn.linear_model.LogisticRegression — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[42] “sklearn.svm.LinearSVC — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

[43] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. (2015). *Deep Residual Learning for Image Recognition*. pp. 1-12

[44] “Vanishing gradient problem.” Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Vanishing_gradient_problem

[45] Krizhevsky, Alex. “Convolutional neural network.” Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Convolutional_neural_network.

[46] “Backpropagation.” Brilliant. [En línea]. Disponible en: https://brilliant.org/wiki/backpropagation/

[47] “OpenCV: Hough Line Transform.” OpenCV documentation. [En línea]. Disponible en: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html

[48] “JSON File Extension - What is a .json file and how do I open it?” FileInfo.com. [En línea]. Disponible en: https://fileinfo.com/extension/json

[49] “split - Parameter "stratify" from method "train_test_split" (scikit Learn).” Stack Overflow. [En línea]. Disponible en: https://stackoverflow.com/questions/34842405/parameter-stratify-from-method-train-test-split-scikit-learn

[50] “Feature engineering.” Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Feature_engineering

[51] “sklearn.feature_extraction.text.CountVectorizer — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

[52] “sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

[53] “sklearn.decomposition.PCA — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

[54] “sklearn.metrics.accuracy_score — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

[55] “sklearn.metrics.precision_score — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html#sklearn.metrics.precision_score

[56] “sklearn.metrics.recall_score — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.recall_score.html#sklearn.metrics.recall_score

[57] “sklearn.metrics.f1_score — scikit-learn 1.0.2 documentation.” Scikit-learn. [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html

[58] “3.1. Cross-validation: evaluating estimator performance — scikit-learn 1.0.2 documentation.” [En línea]. Disponible en: https://scikit-learn.org/stable/modules/cross_validation.html

[59] “k-means clustering.” Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/K-means_clustering

[60] “Stemming and lemmatization.” Stanford NLP Group. [En línea]. Disponible en: https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html

[61] “Cross Validation in Machine Learning.” GeeksforGeeks. [En línea]. Disponible en: https://www.geeksforgeeks.org/cross-validation-machine-learning/

[62] “Confusion matrix.” Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Confusion_matrix

[63] “Document AI.” Google Cloud. [En línea]. Disponible en: https://cloud.google.com/document-ai

[64] Brunn, Holger. “*invoice-x/invoice2data: Extract structured data from PDF invoices.*” GitHub. [En línea]. Disponible en: https://github.com/invoice-x/invoice2data

[65] "*naiveHobo/InvoiceNet: Deep neural network to extract intelligent information from invoice documents.*" GitHub. [En línea]. Disponible en: https://github.com/naiveHobo/InvoiceNet

[66] Brownlee, Jason. "*Ensemble Learning Algorithm Complexity and Occam's Razor.*" Machine Learning Mastery. [En línea]. Disponible en: https://machinelearningmastery.com/ensemble-learning-and-occams-razor/

[67] "*tesis-big-data/document-assistant*" GitHub. [En línea]. Disponible en: https://github.com/tesis-big-data/document-assistant

[68] "Docker: Home". [En línea]. Disponible en: https://docker.com/

[69] "Computer vision". Wikipedia. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Computer_vision

[70] "Human-in-the-loop". Wikipedia [En línea]. Disponibe en: https://en.wikipedia.org/wiki/Human-in-the-loop

[71] "Detect data drift on datasets". Microsoft [En línea]. Disponible en: https://docs.microsoft.com/en-us/azure/machine-learning/how-to-monitor-datasets?tabs=python

Anexo 1 - Estructura del proyecto

A continuación se muestra evidencia de la implementación mediante la presentación de la estructura del proyecto incluyendo directorios y archivos más relevantes.

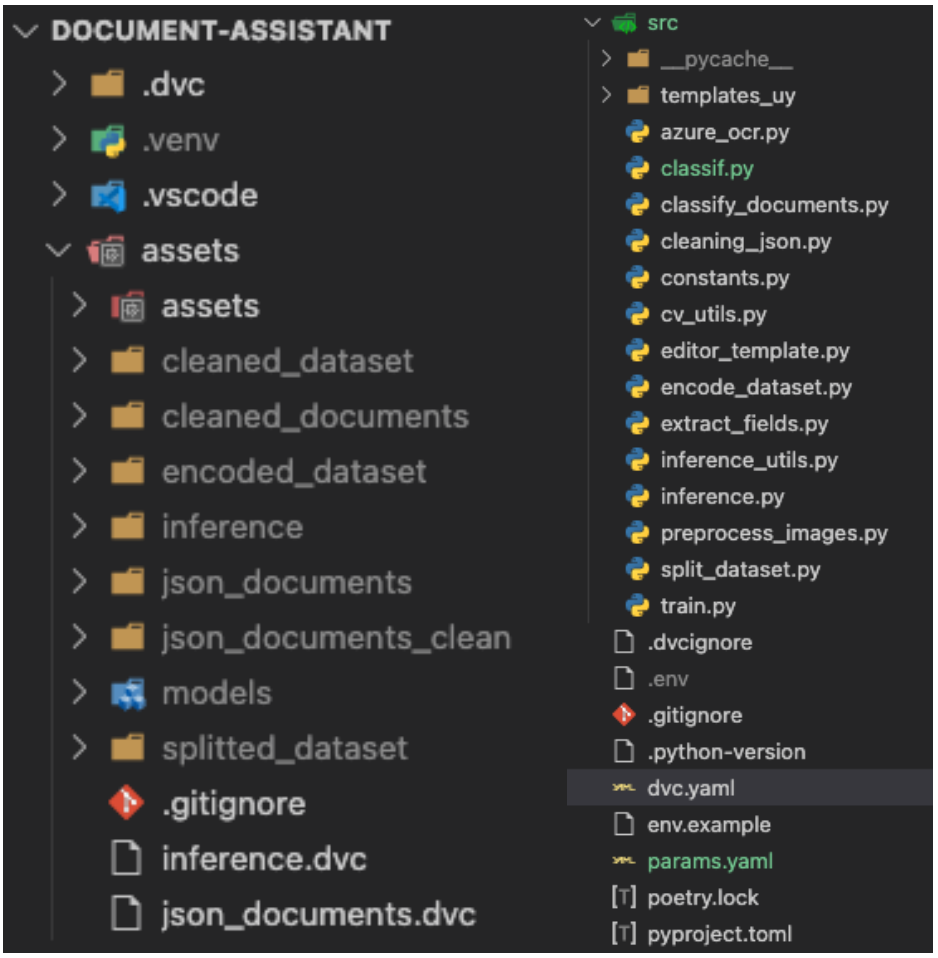


Ilustración Anexo 1.1: Estructura del proyecto

En la carpeta */assets* se encuentran todos los datos del *pipeline* (incluyendo pasos intermedios del procesamiento) los cuales son gestionados por DVC. Dentro de */src* se alojan todos los archivos Python implementados por el equipo para cumplir con todos los requerimientos funcionales.

Vale la pena presentar el archivo *dvc.yaml* el cuál contiene la definición del *pipeline* de entrenamiento así como también el *script* de inferencia (o producción):

inference.py

src > inference.py

```python
from preprocess_images import preprocess_images
from cleaning_json import clean_documents_inference
from encode_dataset import encode_inference
from classify_documents import classify_documents
from extract_fields import extract_fields
from inference_utils import save_processing, dvc_push_data

if __name__ == "__main__":
    preprocess_images()
    clean_documents_inference()
    encode_inference()
    classify_documents()
    extract_fields()
    save_processing()
    dvc_push_data()

```

Ilustración Anexo 1.2: Definición pipeline de entrenamiento en formato yaml

dvc.yaml

dvc.yaml

```yaml
stages:
  clean_documents:
    cmd: .venv/bin/python src/cleaning_json.py
    deps:
      - assets/json_documents
      - src/cleaning_json.py
    outs:
      - assets/cleaned_dataset
  split_dataset:
    cmd: .venv/bin/python src/split_dataset.py
    deps:
      - assets/cleaned_dataset
      - src/split_dataset.py
    outs:
      - assets/splitted_dataset
  encode_dataset:
    cmd: .venv/bin/python src/encode_dataset.py
    deps:
      - assets/splitted_dataset
      - src/encode_dataset.py
    outs:
      - assets/encoded_dataset
  train:
    cmd: .venv/bin/python src/train.py
    deps:
      - assets/encoded_dataset
      - src/train.py
    outs:
      - assets/models

```

Ilustración Anexo 1.3: Script de inferencia