



Universidad
Internacional
de Valencia

Análisis de Créditos y Subsidios para vivienda en Colombia con IA y ML

Titulación:
Master en Inteligencia
Artificial

Curso académico:
2022 - 2023

Alumno/a: URREA
MULLER, ORLANDO

D.N.I: 80.850.095

Director/a de TFM: JESUS
CIGALES

Convocatoria:
Tercera

10 Mayo 2023

De:
 Planeta Formación y Universidades

Índice general

Índice de figuras	1
1. Introduccion	2
1.1. SincoSoft-CBR	2
1.2. Resumen	4
1.3. Hipotesis	4
1.4. Objetivos Generales	4
1.5. Objetivos Especificos	4
2. Estado del Arte Extraccion de Texto	6
2.1. Reconocimiento Óptico de Caracteres (OCR)	6
2.2. Procesamiento de Lenguaje Natural (NLP)	6
2.3. Aprendizaje Automático (Machine Learning) y Aprendizaje Profundo (Deep Learning)	6
2.4. Optimización de PDF y Preprocesamiento	6
2.5. Herramientas y Librerías Específicas	7
3. Marco Teorico Tecnologias Propuestas	8
3.1. Minería de Datos y Machine Learning	8
3.1.1. Minería de Datos	8
3.1.2. Machine Learning	8
3.2. CRISP-DM	9
3.3. TECNICAS DE PROCESAMIENTO DE IMAGEN	10
3.3.1. Filtrado de Ruido	10
3.3.2. Mejora de Contraste	12
3.3.3. Binarización	13
3.3.4. Correccion de Inclinacion	14
3.4. OCR (Optical Character Recognition)	15
3.5. Aprendizaje Supervisado	16
3.6. Clasificacion Multiclase	17
3.7. SVM	19

3.7.1. TfidfVectorizer	21
3.8. Expresiones Regulares	21
3.9. Matriz de Confusion y Metricas	22
4. Desarrollo del Proyecto	24
4.1. Recoleccion y Generacion de Cartas	24
4.2. Entendimiento y Exploracion de los Cartas	24
4.3. Transformacion y Preprocesamiento de Cartas	25
4.3.1. Conversion y Validacion de Imagenes	25
4.3.2. Preprocesamiento de Imagenes	26
4.3.3. Extraccion de Caracteristicas y Limpieza de texto	27
4.3.4. Creacion Listas con asignacion de Etiquetas	27
4.4. Modelado y Entrenamiento con SVM	28
4.5. Extraccion de Texto	31
4.5.1. valores IDF del Vectorizador usado en SVM	31
4.5.2. Librerias de NPL	32
4.5.3. Expresiones regulares	32
4.5.4. Redes Neuronales	33
4.6. Evaluacion	35
4.6.1. Matrices de confusion	35
4.6.2. Metricas y Resultados	35
5. Conclusiones y Limitaciones Futuras	36
6. Referencias	37
7. Anexos	38

Índice de figuras

1.1. CBR	3
3.1. CRISP-DM	9
3.2. Tecnicas Filtrado	11
3.3. Tecnicas Contraste	13
3.4. Tecnicas Binarización	14
3.5. Aprendizaje Supervisado	17
3.6. Clasificacion Multiclase	18
3.7. SVM	19
3.8. SVM Espacio de representación	20
3.9. Matriz Confusion	22
3.10. Medidas Matriz Confusion	23
3.11. Indice Jaccard	23
4.1. Ejecucion modelo SVM	31
4.2. Ejecucion valores IDF del Vectorizador	32
4.3. Ejecucion Expresiones regulares Cedula/Valor	33
4.4. Ejecucion Expresiones regulares Fecha	33

Introduccion

1

1.1. SincoSoft-CBR

SincoSoft es un software colombiano de gestión online que permite controlar el negocio y administrar la información de todas las áreas de las compañías del Sector Constructor, a través de un solo sistema. Está compuesto por aplicaciones modulares que aumentan la eficiencia y productividad de los equipos de trabajo, al conectar los procesos dentro de la organización de manera transversal. El Software se enfoca en tres grupos estrategicos para apoyar este tipo de compañías : Gestion del Negocio, Gestion Financiera y Apoyo Empresarial. En este caso vamos a ubicarnos en el Grupo de Gestion del Negocio principalmente en el Modulo CBR (Comercializadora de Bienes Raices), este es un modulo desarrollado hace 15 años y que ha ido creciendo de la mano de mas de 300 clientes del sector que ya hacen uso de el.

El Modulo CBR se encarga de administrar y gestionar todos los porcesos necesarios desde la captuara de Leads o Prospectos que muestren interes en la compra de vivienda nueva , hasta los procesos de Postventas o Garantias de las mismas. Pasando por los Submodulos CRM, Ventas, Tramites, Cartera, Escrituracion , Documentos, Reformas,Comisiones y Postventas.

En el Sector Constructor varios actores intervienen en el proceso de adquisicion de vivienda nueva, pero entre los mas relevantes estan las Entidades de Creditos, las cajas de compensacion y las Notarias, en este caso nos enfocaremos en Entidades de Credito y Cajas de compensacion las primeras son las encargadas de Aprobar y Entregar los dineros de los creditos y las segundas entregan los Subsidios que asigana el gobierno a los diferentes compradores, la cantidad de asignacion se evaluan y asignan bajo unas condiciones que no sean relevantes para la ejecucion de este proyecto.

Entendiendo todo esto es donde empieza a visualizarse unos problemas que son los que con este proyectos buscara minimizar, el primer inconveniente es que los datos de la Constructora no estan integrados de manera y automatizada con las entidades mencionadas, esto se da por dos cosas generalmente costos tecnologicos opara el Constructor y falta de interes por las entidades de Creditos y Subsidios para apoyar estas tareasse mejor manera; al tener este vacio se generan riesgos administrativos y financieros que en la mayoria de las veces no se

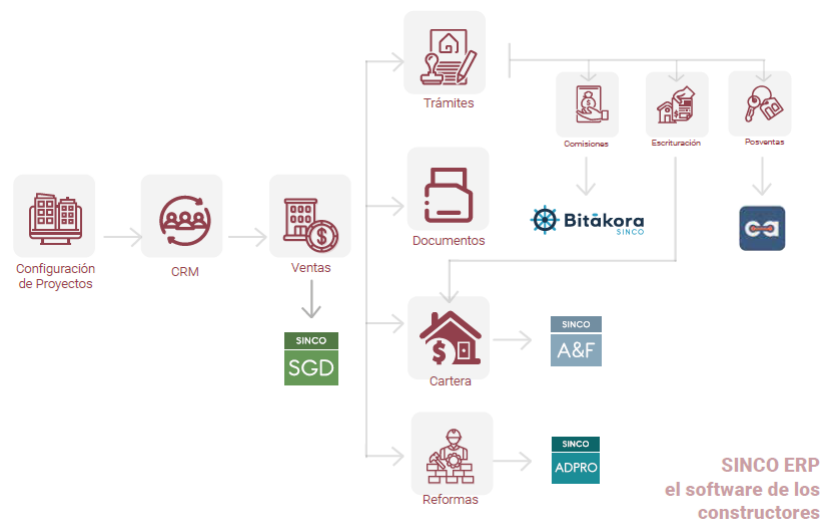


Figura 1.1: Procesos Modulo CBR

logran evidenciar a tiempo los cuales tienen que ver con temas como endeuamiento del comprador y por ende no desembolsan dineros o asignan menores montos, lo mismo vencimientos de las aprobaciones de desembolsos o cédulas y nombres errados en la asignación generando desistimientos del negocio ya proyectados por el Constructor. También al no estar integrados de forma directa esta información llega vía correo electrónico para que sea diligenciado por los usuarios de la constructora en el sistema SincoSoft-CBR dejando abierta la posibilidad de error en la digitación y cargas operativas muy altas.

1.2. Resumen

En este proyecto buscamos por medio de la IA apoyar las tareas de análisis de riesgo, verificación de información y automatización de registro de datos en las aprobaciones de créditos y subsidios que se asignan por el sector bancario y las cajas en Colombia a los compradores de vivienda nueva, información que se entrega actualmente a las constructoras vía correo electrónico en formatos de carta para que los usuarios realicen de forma manual lo descrito al comienzo de este párrafo.

Vale la pena resaltar que no existen trabajos previos ni avances en esta tarea puntual en nuestro país, ni se tiene conocimiento en el sector de herramientas que lo hagan en la actualidad.

1.3. Hipotesis

La Extraccion y Analisis de texto en imagenes servira para obtener la informacion relevante con el que el sistema Sinco CBR y sus usuarios podran tomar desiciones tempranas y reverificar Datos del Comprador, Valores, y Fechas de vigencia en la asignacion de Creditos y Subsidios en Colombia.

1.4. Objetivos Generales

1. Investigar y desarrollar un sistema de machine learning basado en algoritmos de clasificación y/o redes Neuronales que hagan un aporte al Desarrollo del sector de la Construcción en Colombia por medio del Software SINCO ERP - CBR.
2. Evaluar la eficacia de este proyecto en la captura de informacion relevante que aporte en la mitigacion de riesgos y la disminucion de error humano al analizar y centralizar la informacion de Creditos y Subsidios Asignados en Colombia para la compra de vivienda Nueva.

1.5. Objetivos Especificos

1. Revisar y analizar la teoria existente sobre el uso de algoritmos de clasificación y/o redes neuronales para el reconocimiento de texto en imagenes o documentos PDF con el fin de extraer informacion especifica.
2. Recopilar algunas cartas reales de Asignacion o Aprobacion de Creditos y Subsidios teniendo en cuenta tambien que sean de diferentes entidades bancarias y cajas de compensacion.
3. Generar conjuntos de datos relevantes de forma manual y/o mediante tecnicas de ML para entrenar los Algoritmos a utilizar.

4. Preprocesar y limpiar los conjuntos de datos para garantizar que sean datos consistentes y equilibrados con el fin de buscar un buen rendimiento y calidad del entrenamiento.
5. Identificar diferentes tecnicas que apliquen para lo que se busca en el proyecto planteado.
6. Evaluar el rendimiento de los algoritmos seleccionados con base a las metricas mas comunes e identificar que resultados generan.
7. Optimizar los parámetros de los algoritmos ya sea de forma Manual o mediante tecnicas de ML con el fin de buscar mejores resultados.
8. Identificar y Evaluar cuales fueron las variables y/o parametros mas influyentes en los cambios o mejoras de resultados.
9. Identificar posibles limitaciones de alcance que no se abarquen en este proyecto.
10. Contribuir al crecimiento del producto SincoERP-CBR y el uso de ML dentro del mismo.
11. Promover la adopción de soluciones desarrolladas con machine learning en el sector constructor, destacando sus beneficios y ventajas.
12. Establecer una base para motivar investigaciones y desarrollos de Machine Learning en SINCO ERP - CBR.

Estado del Arte Extraccion de Texto

2

En la actualidad ya existen varias tecnologías, métodos y enfoques para extraer y procesar texto de documentos PDF o imagenes con texto digital sin importar su contenido. Entre ellos tenemos :

2.1. Reconocimiento Óptico de Caracteres (OCR)

Permite convertir imágenes de texto en texto digital editable, aunque esta muy avanzado a hoy se siguen realizando investigaciones para mejorar su capacidad y efectividad enfocandose mas que todo en su precision y adaptabilidad cuando estos textos manejan diferentes formatos, fuentes o tamaños.

2.2. Procesamiento de Lenguaje Natural (NLP)

Las técnicas de NLP pueden emplearse para mejorar la calidad de la extracción de texto y comprender mejor el contenido capturado. un ejemplo de ello son BERT o GPT-3 modelos muy potentes ya que no solo capturan la informacion sino que contextualizan y comprenden la idea general del texto.

2.3. Aprendizaje Automático (Machine Learning) y Aprendizaje Profundo (Deep Learning)

Estas técnicas se aplican para mejorar la precisión de la extracción de texto y estructuras puntuales. Pueden ser entrenados para identificar patrones específicos en el formato de las cartas y asi lograr mas precision en la busqueda, y a su vez involucrar tecnicas de segmentacion para aislar las partes no relevantes de la carta.

2.4. Optimización de PDF y Preprocesamiento

Antes de realizar la extracción de texto, es posible aplicar técnicas de optimización a los documentos PDF para mejorar la legibilidad y la estructura. Esto podría incluir la eliminación de ruido, la mejora de la calidad de la imagen y la normalización del formato.

2.5. Herramientas y Librerías Específicas

Herramientas y Librerías Específicas: Existen varias herramientas y librerías de software que se especializan en la extracción de texto de documentos PDF, como PyMuPDF, pdf2text, Textract, etc. Estas herramientas a menudo incorporan varias de las técnicas mencionadas anteriormente. También en sitios como <https://huggingface.co/> pueden encontrarse soluciones open source para solucionar problemas de este tipo.

Marco Teorico Tecnologias

Propuestas

3

3.1. Minería de Datos y Machine Learning

3.1.1. Minería de Datos

La Minería de datos es un proceso para encontrar o descubrir patrones en tamaños relevantes de información con el fin de extraer conocimiento y ubicar información que no es visible o entendible a simple vista de manera que se puedan tomar decisiones más acertadas y en algunos casos anteponiéndose a los posibles resultados. Este proceso tiene varias etapas como son por ejemplo recolectar los datos, preprocesar estos datos, identificar que método o técnica aplica más para ese conjunto de datos y da mejores resultados, aplicar la técnica y evaluar e interpretar lo encontrado para al final poder tomar decisiones sobre esa información.

Aunque la información que se adquiere es en su mayoría valiosa y se toman decisiones más informadas todavía existen limitantes o impedimentos por superar entre los que encontramos la privacidad de información, seguridad de información e interpretaciones correctas de los resultados.

Vale la pena resaltar que siempre se podrán encontrar mejores resultados pero para ellos es necesario hacer tareas de refinamiento iterativas y así buscar ser más precisos en los resultados.

3.1.2. Machine Learning

El aprendizaje automático es una rama de la IA que se enfoca en la creación de modelos basados en algoritmos que son capaces de aprender y tomar decisiones basándose en datos, ellos pueden en vez de recibir instrucciones ir mejorando con base a la experiencia que se genera con más información suministrada.

Estos modelos buscan patrones, tendencias, correlaciones entre otras cosas para utilizarlos posteriormente en la realización de predicciones o toma de decisiones.

Esto se puede aplicar en muchas áreas y para diversas tareas como por ejemplo reconocimiento de voz y de imagen, recomendar productos, medicina con el fin de identificar posibles

enfermedades, entre otras; este enfoque apoya e impulsa investigaciones de manera significativa en esas areas generando valor de manera significativa y precisa.

3.2. CRISP-DM

CRISP-DM (Cross Industry Standard Process for Data Mining) es un modelo estándar ampliamente utilizado en la minería de datos y el análisis de datos. Fue desarrollado en 1996 por un grupo de expertos de diferentes organizaciones en el campo de la minería de datos, con el objetivo de proporcionar una metodología estructurada y sistemática para guiar proyectos de minería de datos desde el inicio hasta la implementación.

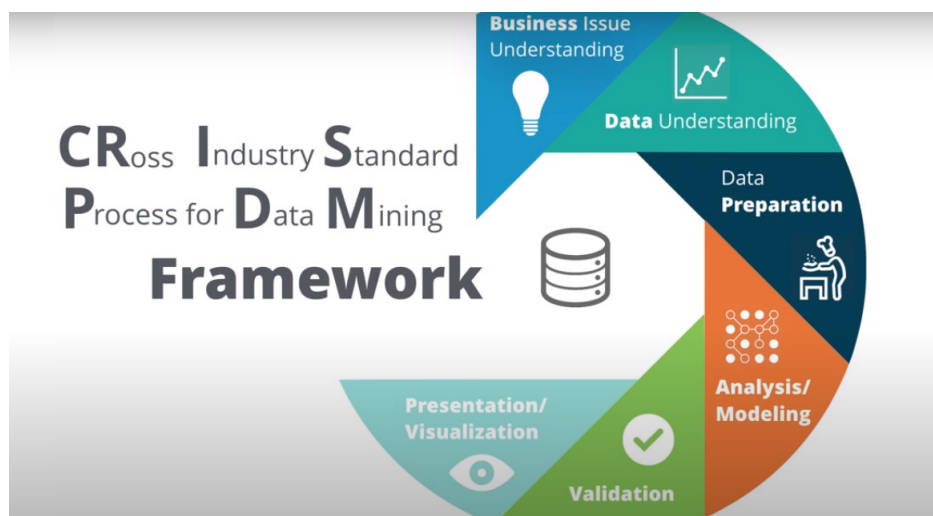


Figura 3.1: Grafico de Metodologia CRISP-DM

El modelo CRISP-DM está compuesto por seis fases principales en las que indicare su concepto teorico y al mismo tiempo como podria aplicarlo en el proyecto:

- **Comprensión del negocio (Business Understanding):** En esta fase, se establece una comprensión clara de los objetivos finales y los requisitos del proyecto. Se definen las metas que se desean alcanzar y se identifica como saber si estamos en los rangos de exito. Qué tipo de información textual se quiere extraer de las imágenes, Cuál es el propósito y la utilidad de esa extracción o para que voy a usarla, Qué calidad deben tener las imágenes, cuál es la cantidad de imágenes a procesar y de que tipo, son varios de los aspectos a tener presentes en esta fase del proyecto.
- **Comprensión de los datos (Data Understanding):** En esta fase se recopila y explora el conjunto de datos, se realiza un análisis para familiarizarse con los datos e identificar problemas de calidad con el fin de entender la estructura y detectar patrones. Revisar la calidad, como estan distribuidas las partes a analizar, fuentes, tamaños ewntre otras serian tareas de este proyecto para esta fase.

- **Preparación de los datos (Data Preparation):** En esta fase, se realiza la limpieza, transformación y selección de los datos para prepararlos para el modelado. Esto incluye el manejo de datos faltantes, la normalización, la codificación de variables categóricas y otras tareas de preparación de datos comunes. En este caso el proyecto me llevaría a realizar análisis de distorsiones, reducción del ruido, luego se realizaría una segmentación de la imagen para aislar partes donde no sea necesario analizar texto y adicional a esto armar subconjuntos de datos para después comparar resultados.
- **Modelado (Modeling):** En esta etapa, se seleccionan y aplican diversas técnicas de modelado para construir modelos predictivos o descriptivos. Se evalúan los modelos y se optimizan para obtener el mejor rendimiento posible. En este caso lo que se ve como una buena opción es la combinación de SVM con OCR de esta manera segmenta las áreas donde se encuentra el texto deseado y con OCR captura la información.
- **Evaluación (Evaluation):** Se evalúan los modelos generados en la fase anterior utilizando métricas y técnicas de validación para medir la precisión y el rendimiento del modelo en un conjunto de datos. La precisión, el recall o el F1-score serían algunas de las apropiadas en nuestro caso.
- **Despliegue (Deployment):** En esta fase, se implementan los resultados del proyecto en un entorno operativo o productivo.

Es importante tener presente que el modelo CRISP-DM es un enfoque cíclico y que nos provee de una metodología que podemos aplicar a cualquier proyecto de esta categoría, con el fin de realizar un monitoreo constante y una mejora continua del mismo.

3.3. TECNICAS DE PROCESAMIENTO DE IMAGEN

3.3.1. Filtrado de Ruido

Este proceso es utilizado para mejorar la calidad y la legibilidad de una imagen, eliminando o reduciendo las perturbaciones no deseadas que pueden estar presentes debido a diversas fuentes sea por su origen, su formato, al imprimir, al escanear o algún proceso que influya en el aumento del ruido de la imagen.

El ruido en una imagen se refiere a las fluctuaciones aleatorias de intensidad de píxeles que pueden distorsionar o degradar la calidad de la imagen. El objetivo del filtrado de ruido es suavizar la imagen sin perder demasiados detalles importantes o por lo menos tratar de buscar la manera de que se mejore la calidad en las partes que vamos a usar en nuestro proyecto. A menudo, el filtrado de ruido se realiza antes de aplicar algoritmos de procesamiento de imágenes adicionales, como la detección de bordes o el reconocimiento de objetos.

Existen diferentes técnicas de filtrado que se utilizan para eliminar el ruido en una imagen. Algunas de las más comunes son:

- Filtro de media (Filtro promedio): Calcula el valor promedio de los píxeles en una ventana local (por ejemplo, una matriz de 3x3 o 5x5) y lo asigna al píxel central de la ventana. Este filtro es eficaz para eliminar el ruido de baja frecuencia.
- Filtro de mediana: Reemplaza el valor de un píxel por la mediana de los valores en una ventana local. Este filtro es útil para eliminar el ruido impulsivo (saltos abruptos en los valores de los píxeles).
- Filtro Gaussiano: Utiliza una distribución de Gauss para ponderar los píxeles en una ventana local y calcular el valor del píxel filtrado. Es eficaz para eliminar el ruido de alta frecuencia.
- Filtro Bilateral: Es una extensión del filtro Gaussiano que también tiene en cuenta la diferencia de intensidades entre los píxeles en la ventana local. Preserva los bordes y detalles de la imagen mientras reduce el ruido.

El proceso de filtrado de ruido funciona al recorrer cada píxel de la imagen y aplicar la operación del filtro seleccionado en una vecindad local del píxel. La vecindad puede ser un cuadrado, un círculo o cualquier otra forma, dependiendo del tipo de filtro utilizado. Al aplicar el filtro en toda la imagen, los píxeles ruidosos se suavizan o se corrigen según el tipo de filtro aplicado, mejorando la calidad de la imagen.

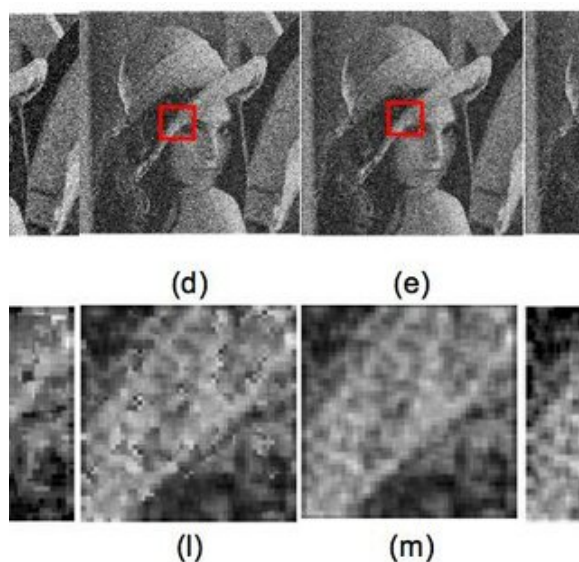


Figura 3.2: Técnica de filtrado aplicada

Es importante tener en cuenta que el filtrado de ruido es una operación de procesamiento de imágenes que se realiza en el dominio espacial y puede ser parte de un preprocesamiento

más amplio antes de aplicar técnicas de análisis más avanzadas. La elección del filtro adecuado depende del tipo de ruido presente y del resultado deseado. En algunos casos, puede ser necesario experimentar con diferentes filtros y parámetros para obtener los mejores resultados para un caso específico.

3.3.2. Mejora de Contraste

Es un proceso utilizado para resaltar las diferencias de intensidad entre los píxeles de una imagen entendiendo que el contraste es la diferencia en los valores de intensidad de los píxeles en determinadas partes de una imagen, entre mayor sea el contraste la gama de valores de intensidad aumenta y las partes claras serán muy diferenciadas de las oscuras si el contraste es muy bajo no es fácil distinguir esos cambios de intensidad por lo que se haría más complejo detectar algún objeto dentro de la misma. El objetivo es mejorar la visualización de detalles y realzar las características relevantes presentes en la imagen. La mejora de contraste no solo afecta la visibilidad de la imagen, sino que también puede hacer que los objetos y patrones sean más fáciles de detectar y analizar.

El proceso de mejora de contraste funciona mediante la redistribución de los niveles de intensidad de los píxeles para que haya una mejor distribución en toda la gama de valores posibles. Esto se puede lograr mediante diversas técnicas de mejora de contraste. Algunas de las técnicas más comunes incluyen:

- **Ecualización del histograma:** Es una técnica ampliamente utilizada para mejorar el contraste de una imagen. El histograma representa la distribución de frecuencias de los diferentes niveles de intensidad en la imagen. La ecualización del histograma redistribuye los valores de intensidad para que el histograma resultante sea más uniforme, lo que aumenta el contraste.
- **Estiramiento de contraste:** Esta técnica mapea los niveles de intensidad originales a un nuevo rango más amplio de valores de intensidad. El objetivo es expandir los valores de intensidad para abarcar todo el rango de niveles disponibles, lo que aumenta el contraste en la imagen.
- **Mejora local de contraste:** En lugar de aplicar técnicas de mejora de contraste en toda la imagen, esta técnica se centra en regiones locales de la imagen. Es especialmente útil cuando diferentes áreas de la imagen tienen diferentes niveles de contraste.

Es importante tener en cuenta que la mejora de contraste puede mejorar la visualización de detalles y características en la imagen, pero también puede acentuar el ruido y artefactos presentes en la imagen. Por lo tanto, es esencial utilizar estas técnicas de manera adecuada y ajustar los parámetros según las necesidades específicas del procesamiento de imágenes que se esté realizando.

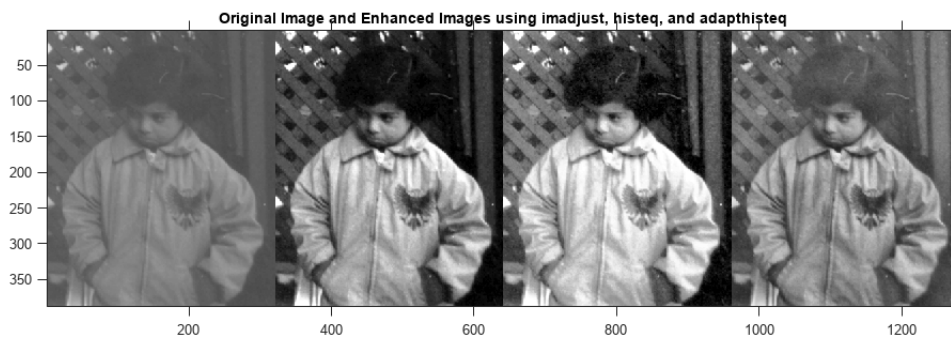


Figura 3.3: Técnica de contraste aplicada

3.3.3. Binarización

Es un proceso mediante el cual se convierte una imagen en una versión de dos colores: blanco y negro (o 0 y 1), donde los píxeles con intensidades por encima de un cierto umbral se asignan al color blanco (o 1) y los píxeles con intensidades por debajo del umbral se asignan al color negro (o 0). En esencia, la binarización divide la imagen en dos niveles de intensidad para resaltar regiones o características de interés.

La binarización es una técnica fundamental y ampliamente utilizada en el procesamiento de imágenes y tiene diversas aplicaciones, como la detección de bordes, la segmentación de objetos, el reconocimiento de caracteres en documentos escaneados y más.

El proceso de binarización funciona de la siguiente manera:

- **Selección del umbral:** El primer paso es determinar el umbral que separará los píxeles en blanco y negro. La elección del umbral puede variar según las características de la imagen y los objetivos del procesamiento. Puedes usar un umbral fijo predefinido o aplicar técnicas de umbralización adaptativa que calculen el umbral localmente para cada región de la imagen.
- **Asignación de píxeles:** Después de seleccionar el umbral, cada píxel en la imagen se compara con el umbral. Si la intensidad del píxel es mayor o igual al umbral, se asigna al color blanco (o 1); de lo contrario, se asigna al color negro (o 0).

El resultado final de la binarización es una imagen binaria, donde los objetos de interés se representan en blanco (o 1) y el fondo en negro (o 0). La imagen binaria puede utilizarse para realizar diversas operaciones de procesamiento de imágenes, como detección de bordes, conteo de objetos, extracción de características y más.

Es importante tener en cuenta que el éxito de la binarización depende en gran medida de la elección adecuada del umbral y del tipo de imagen que se está procesando. En algunas



Figura 3.4: Técnica de binarización aplicada

situaciones, el ruido, la iluminación desigual o la presencia de sombras pueden afectar la binarización, lo que puede requerir técnicas de preprocesamiento adicionales para mejorar los resultados. También existen métodos más avanzados de binarización, como la binarización por umbralización local adaptativa o técnicas basadas en aprendizaje automático, que pueden ser más efectivas en escenarios más complejos.

3.3.4. Corrección de Inclinación

Es un proceso mediante el cual se ajusta automáticamente la orientación de la imagen para alinear objetos o elementos con una orientación específica. La corrección de inclinación se utiliza para enderezar imágenes que puedan estar inclinadas o giradas de manera no deseada y donde claramente hace que sea más difícil identificarlas de manera correcta.

En la práctica se da mucho en casos de capturas de imágenes o escaneados que no quedan rectos en el momento de su generación. En tales casos, la corrección de inclinación es útil para enderezar la imagen y alinear el contenido en una orientación horizontal o vertical, lo que facilita su visualización y procesamiento.

El proceso de corrección de inclinación generalmente sigue los siguientes pasos:

- **Detección de la inclinación:** Se utiliza una técnica de análisis de características para determinar la inclinación presente en la imagen. Esto puede implicar la detección de líneas o bordes en la imagen y el cálculo del ángulo de inclinación.
- **Corrección de la inclinación:** Una vez que se ha detectado la inclinación, se aplica una transformación geométrica a la imagen para enderezarla. En el caso de una inclinación horizontal, la imagen se rotará para alinear los elementos en una posición horizontal; en

el caso de una inclinación vertical, se realizará una rotación para alinear los elementos en posición vertical.

- **Recorte opcional:** Si después de la corrección de inclinación quedan bordes negros o áreas vacías en la imagen, es posible realizar un recorte para eliminar estos espacios no deseados.

Existen varias técnicas y algoritmos para llevar a cabo la corrección de inclinación, incluyen métodos basados en transformaciones geométricas, análisis de líneas y bordes, y técnicas de aprendizaje automático como redes neuronales.

La corrección de inclinación es una etapa importante en el procesamiento de imágenes, especialmente en aplicaciones de OCR (Reconocimiento Óptico de Caracteres) y procesamiento de documentos, donde se busca mejorar la precisión y legibilidad del texto. Al corregir la inclinación, se facilita el reconocimiento de caracteres y la extracción de información de la imagen.

3.4. OCR (Optical Character Recognition)

El Reconocimiento Óptico de Caracteres en español, es una tecnología que se utiliza para convertir imágenes de texto en caracteres de texto que se pueden editar y buscar. En esencia, OCR permite que las computadoras "leen" texto en imágenes, como documentos escaneados, imágenes de cámaras digitales o capturas de pantalla, y lo convierten en texto digital que se puede manipular y procesar.

- **Preprocesamiento:** Este proceso es muy importante pues de él dependerá la calidad con que el OCR reconozca el texto que deseamos extraer para ser manipulado, la idea es aplicar las técnicas que ya fueron descritas en el apartado anterior de este documento hasta que logremos tener la calidad que creamos conveniente. Por lo cual se debe tener muy claro conocimiento de cuáles son las imágenes a trabajar y su estructura independientemente de si son documentos escaneados, imágenes con textos, recortes de pantalla entre otros.
- **Detección de Caracteres:** Una vez que la imagen ha sido preprocesada, el OCR analiza la imagen en busca de patrones que se asemejen a caracteres. Esto puede implicar la identificación de bordes y características que indican la presencia de texto.
- **Reconocimiento de Caracteres:** Una vez que se han detectado las regiones de texto, el OCR identifica qué caracteres específicos están presentes en esas regiones. Esto implica comparar los patrones detectados con una base de datos de patrones de caracteres conocidos.

- **Postprocesamiento:** Después de reconocer los caracteres, es importante realizar un post-procesamiento para corregir errores y mejorar la precisión. Esto puede incluir la corrección de caracteres mal reconocidos y la restauración del formato original.

En algunas ocasiones el uso de Expresiones Regulares y el definir si la extracción se va a manipular por palabras o por todo el texto como uno es relevante e influye en los resultados finales.

Se pueden realizar también guardados en una Base de Datos de estos textos estructurados, para realizar mejoras o modificaciones posteriormente.

OCR es ampliamente utilizado en la digitalización de documentos, la automatización de procesos empresariales, la conversión de archivos en papel a formato digital, el reconocimiento de matrículas de vehículos, la extracción de datos de facturas y muchos otros escenarios donde se necesita extraer información de imágenes que contienen texto.

3.5. Aprendizaje Supervisado

El aprendizaje supervisado es un enfoque dentro del campo del aprendizaje automático en el cual se entrena un modelo utilizando un conjunto de datos etiquetado, es decir, un conjunto de ejemplos de entrada junto con las respuestas deseadas correspondientes. El objetivo es que el modelo aprenda a mapear las entradas a las salidas de acuerdo con las etiquetas proporcionadas en el conjunto de datos de entrenamiento. En otras palabras, el modelo aprende a generalizar a partir de ejemplos conocidos y luego se puede utilizar para predecir o clasificar nuevas entradas en función de su aprendizaje previo.

El proceso de aprendizaje supervisado implica los siguientes pasos básicos:

- **Recopilación de Datos Etiquetados:** Se recopila un conjunto de datos que contiene ejemplos de entrada junto con las salidas correctas o etiquetas correspondientes dependiendo de la necesidad de etiquetado y de identificación, si lo llevamos a el proyecto de extracción de texto lo ideal sería vincular algunos textos a la identificación del patrón de la imagen.
- **Preparación de Datos:** Los datos se preparan para su procesamiento, lo que puede incluir la normalización, limpieza y transformación de características, de esta manera las entradas se convierten en el formato necesario para entrenar entendiendo como entradas las imágenes.
- **División de Datos:** El conjunto de datos se divide en un conjunto de entrenamiento, un conjunto de prueba y un conjunto de validación para evaluar la calidad del modelo. Los porcentajes aunque existen documentos con sugerencias no deben tener un valor puntual para la obtención de buenos resultados.
- **Entrenamiento del Modelo:** Se elige un modelo o varios modelos que sean los más acordes al manejo de la información que vamos a usar y al aprendizaje de la misma, luego

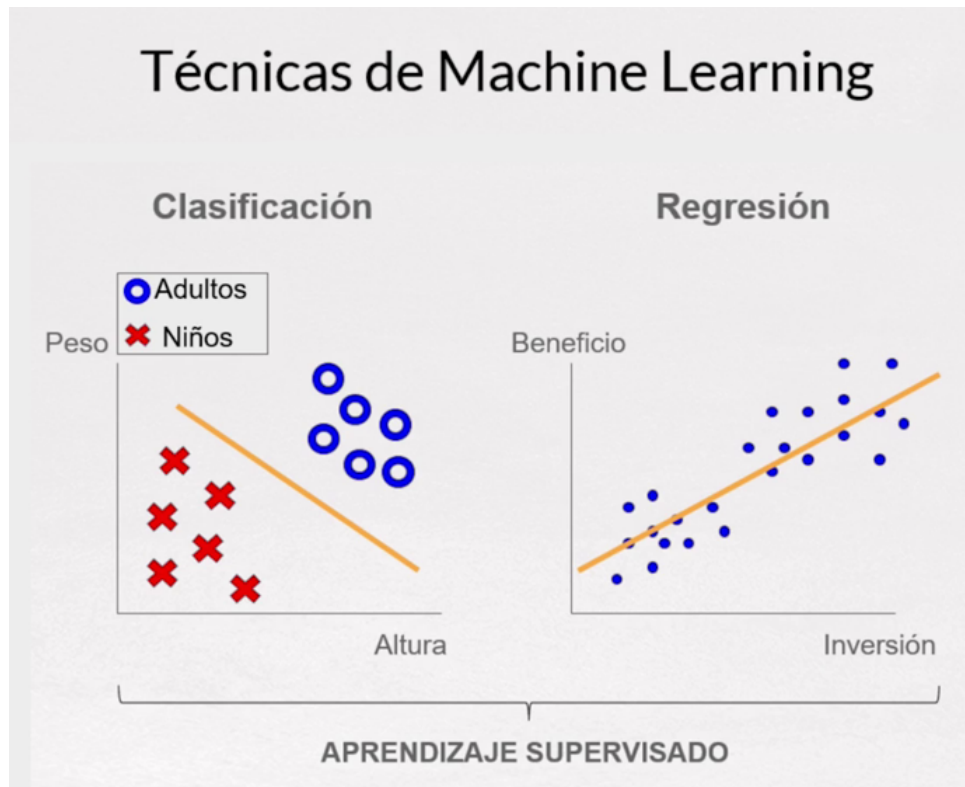


Figura 3.5: Aprendizaje Supervisado

se utiliza el conjunto de entrenamiento para entrenar el modelo, y se ajustan los hiperparametros combinado con el uso de metricas para encontrar el mejor resultado.

- **Evaluación del Modelo:** Se evalúa el modelo utilizando el conjunto de prueba para medir su rendimiento en datos no vistos. Se utilizan métricas como precisión, recall, F1-score, etc.
- **Ajuste y Validación:** Si el rendimiento no es satisfactorio, se pueden realizar ajustes en el modelo y se puede utilizar el conjunto de validación para validar estos cambios.
- **Predicción:** Una vez que el modelo ha sido entrenado y evaluado, se puede utilizar para hacer predicciones en nuevos datos.

El aprendizaje supervisado es una herramienta poderosa cuando se tiene acceso a datos etiquetados de alta calidad. En el caso de la extracción de texto en imágenes, podría usarse para entrenar un modelo que identifique regiones de texto en las imágenes y las convierta en texto legible utilizando OCR.

3.6. Clasificacion Multiclase

La clasificación multiclase en machine learning es una técnica en la que un algoritmo se entrena para asignar una serie de clases a cada instancia de datos, pero en este caso, el

número de clases es mayor a dos. En otras palabras, se trata de asignar una etiqueta a una instancia de datos de entre tres o más categorías posibles en lugar de solo dos.

Por ejemplo, si estás construyendo un modelo para clasificar diferentes tipos de animales como "tigre", "gallo", "pez", estarías realizando una clasificación multiclase, ya que hay más de dos clases posibles.







		True/Actual		
				
Predicted		4	6	3
		1	2	0
		1	2	6

Figura 3.6: Clasificación Multiclase

En el contexto de extraer texto de distintos formatos de cartas, la clasificación multiclase podría ser útil para categorizar los diferentes tipos de cartas en categorías específicas. En nuestro caso, tendríamos cartas de "Crédito", "Subsidio" que al mismo tiempo se dividirían en subgrupos dependiendo de la entidad bancaria. La extracción de texto se realiza después de la clasificación, pero la clasificación multiclase te ayuda a identificar la categoría de la carta antes de procesar su contenido.

Para llevar a cabo la extracción de texto de distintos formatos de cartas se podría manejar de la siguiente forma:

- **Preprocesamiento de Datos:** Reúne un conjunto de cartas en diferentes formatos y categorías aplicando su respectivo preprocesamiento.
- **Clasificación Multiclase:** Entrenar un modelo de clasificación multiclase utilizando algoritmos como SVM, Redes Neuronales, Naive Bayes o Random Forest. Utilizar el conjunto de cartas preprocesadas como datos de entrenamiento y asignándole etiquetas a cada categoría.
- **Extracción de Texto:** Una vez que el modelo ha clasificado las cartas en categorías, puedo extraer el texto de cada carta utilizando técnicas de OCR si las cartas están en formato de imagen, o simplemente procesar el texto si las cartas están en formato de texto plano, también en los casos que sean documentos puedo convertirlos en imágenes dentro del paso de preprocesamiento.
- **Procesamiento Adicional:** Podemos realizar análisis adicionales en el texto extraído, como la identificación de palabras clave o la extracción de información relevante.

- Resultados: Tener los textos finales para terminar de ajustar y utilizar de la forma que el proyecto lo plantee.

En resumen, la clasificación multiclase te ayuda a organizar y categorizar diferentes tipos de cartas antes de realizar la extracción de texto, lo que puede facilitar y mejorar la precisión de la extracción de información específica de cada tipo de carta.

3.7. SVM

SVM (Support Vector Machine) es un algoritmo de aprendizaje supervisado utilizado en machine learning para tareas de clasificación y regresión. En la clasificación, SVM se utiliza para separar y clasificar datos en diferentes categorías o clases, buscando encontrar un hiperplano de separación óptimo entre ellas. En esencia, SVM busca el hiperplano que maximiza el margen entre las clases, lo que permite una mejor generalización a datos no vistos.

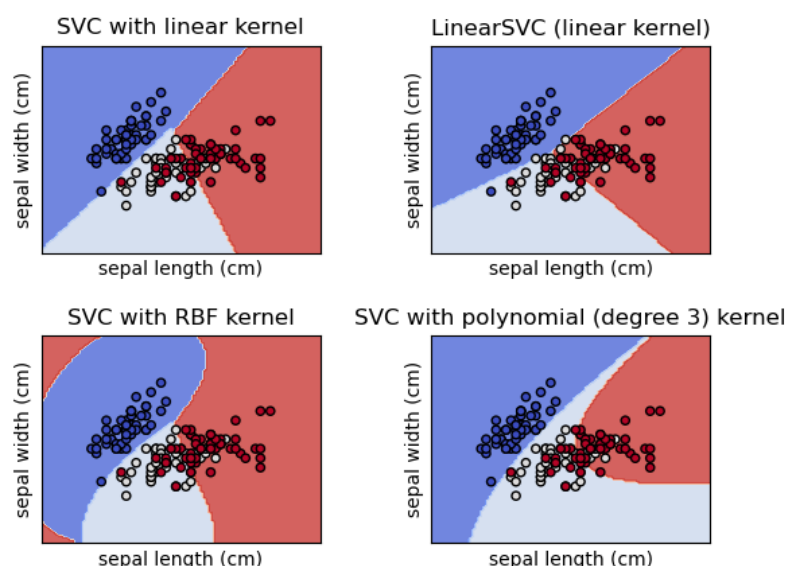


Figura 3.7: SVM

Un modelo SVM crea, implícitamente, un espacio de representación de mayor dimensionalidad en el cual sí podemos separar de forma clara nuestros datos.

Para ajustar la calidad del algoritmo se deben manipular sus hiperparámetros, entre los mas relevantes estan:

- C (parámetro de regularización): El parámetro C controla el equilibrio entre maximizar el margen y minimizar la clasificación incorrecta. Un valor pequeño de C permite un margen más amplio pero puede permitir errores en la clasificación, mientras que un valor grande de C penalizará más los errores de clasificación, lo que puede llevar a un margen más

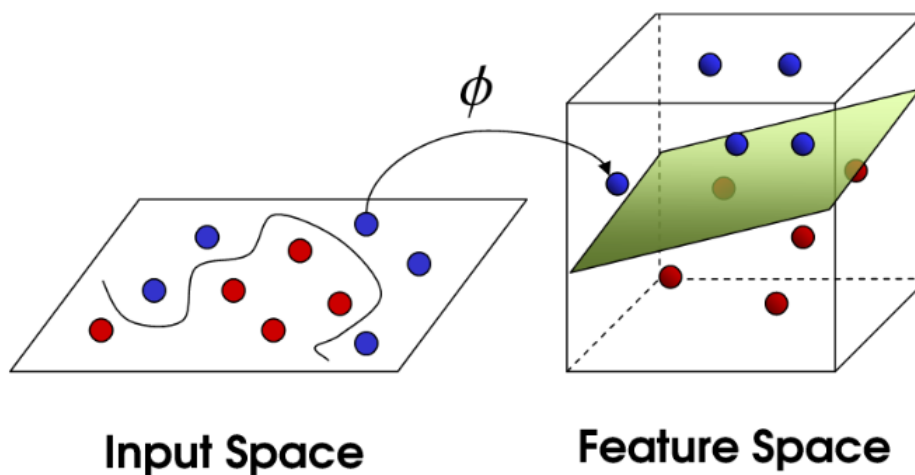


Figura 3.8: SVM Espacio de representación

estrecho. Ajustar C permite encontrar un equilibrio entre la precisión en los datos de entrenamiento y la generalización a nuevos datos.

- **Kernel:** Los kernels determinan cómo se mapean los datos en un espacio de características de mayor dimensión donde es más probable que sean linealmente separables. Algunos kernels comunes son el lineal, el polinomial y el radial (RBF). La elección del kernel depende de la naturaleza de los datos y si es posible encontrar una transformación que haga que los datos sean linealmente separables en un espacio de mayor dimensión.
- **Gamma (para kernels no lineales):** El parámetro gamma afecta el alcance de la influencia de cada ejemplo de entrenamiento en el ajuste del modelo. Un valor pequeño de gamma da como resultado un alcance más amplio, lo que significa que cada ejemplo influye más en el modelo. Un valor grande de gamma da como resultado un alcance más estrecho, lo que hace que el modelo se ajuste más a los datos individuales.
- **Degree (para kernel polinomial):** Este hiperparámetro controla el grado del kernel polinomial. Cuanto mayor sea el grado, más compleja será la frontera de decisión. Sin embargo, un grado muy alto puede llevar a un sobreajuste.
- **Coef0 (para kernel polinomial y sigmoidal):** Este parámetro influye en la influencia de los términos de grado superior en los kernels polinomiales y sigmoidales. Específicamente, controla cómo los términos de grado más alto afectan a los términos de grado más bajo.
- **Shrinking:** La técnica de "shrinking" reduce el número de vectores de soporte utilizados en la construcción del modelo. Esto puede acelerar el proceso de entrenamiento y reducir el uso de memoria, pero puede afectar ligeramente la precisión.
- **Probabilidad:** Habilitar la estimación de probabilidad puede ser útil si deseas obtener estimaciones de las probabilidades de clasificación.

La elección de estos hiperparámetros puede influir en el rendimiento y la capacidad de generalización del modelo SVM. Puedes utilizar técnicas como la búsqueda de hiperparámetros (grid search o random search) y la validación cruzada para encontrar los valores óptimos de estos hiperparámetros para tu problema específico.

Estas técnicas de validación ayudan a visualizar de mejor forma los resultados de estos cambios de hiperparámetros.

En un proyecto de extracción de texto en imágenes, SVM puede ser utilizado para identificar y clasificar regiones que contengan texto y regiones que no lo contengan, se extraen características relevantes de las imágenes para representar las regiones como pueden ser desde palabras hasta características de textura, bordes, colores, etc.

El objetivo es encontrar el hiperplano que mejor separe las regiones de texto de las regiones sin texto en función de las características y así recortar las relevantes y aplicar posteriormente OCR para extraer el texto.

Es importante tener en cuenta que SVM puede ser efectivo para problemas de separación lineal y no lineal, pero el rendimiento dependerá de las características utilizadas y la calidad del conjunto de datos. Además, en combinación con la extracción de texto, SVM ayuda a identificar regiones potenciales de texto antes de aplicar el OCR, lo que puede mejorar la precisión y la eficiencia de la extracción.

3.7.1. TfidfVectorizer

Entendiendo que el algoritmo SVM solo trabaja con datos numéricos ya que como sabemos sus funciones principales se manejan por medio de vectores y matemática lineal, es necesario usar alguna técnica de procesamiento adicional que convierta nuestros textos en números y para ello podemos usar TfidfVectorizer, con esta técnica podemos convertir en representaciones numéricas los textos y de esta manera ser procesada por el SVM.

Esos vectores numéricos se basan en la frecuencia de las palabras (TF) y la inversa de la frecuencia de las mismas (IDF). Para el proyecto que estoy planteando esta forma de análisis es muy óptima ya que las cartas en su naturaleza tienen palabras muy propias de cada entidad y tipo, lo que haría fácilmente diferenciable y exacto el modelo.

3.8. Expresiones Regulares

Las expresiones regulares son patrones de búsqueda utilizados para encontrar y manipular cadenas de texto en varios lenguajes de programación. Las expresiones regulares son una poderosa herramienta para realizar tareas de procesamiento de texto, como búsqueda, extracción, validación y manipulación de patrones de texto.

Las expresiones regulares se componen de caracteres literales (que coinciden exactamente) y metacaracteres (que representan clases de caracteres o patrones). Por ejemplo, el metacaracter `.` coincide con cualquier carácter, y el metacaracter `*` indica que el carácter anterior

puede aparecer cero o más veces. La complejidad del patron dependera de la necesidad de cada caso.

Entendiendo esto las epxresiones regulares son una podera pcion para extraer los textos requeridos despues de tener clasificadas las cartas y extraidos los textos generales de las mismas.

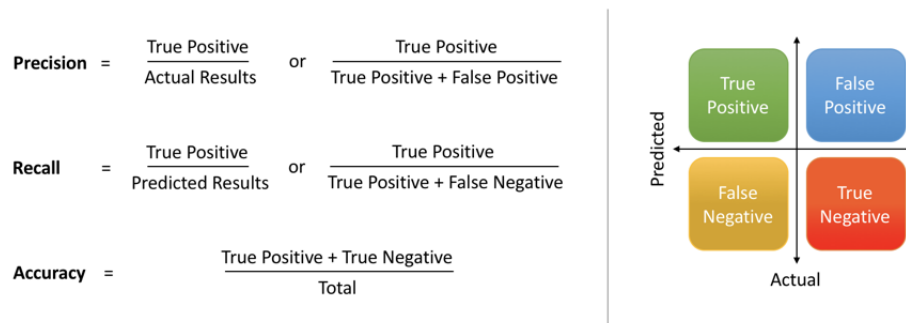
3.9. Matriz de Confusion y Metricas

- **Matriz de Confusión:** La matriz de confusión muestra la cantidad de predicciones correctas e incorrectas para cada clase. A partir de esta matriz, puedes derivar otras métricas como precisión, recall y F1-Score para cada clase.

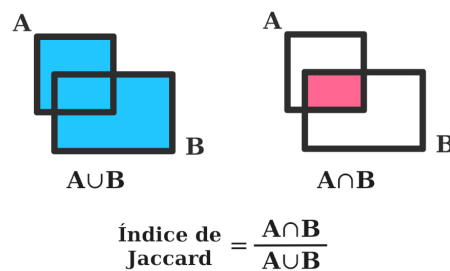
		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Figura 3.9: Matriz Confusion

- **Exactitud (Accuracy):** La exactitud es una medida básica que indica la proporción de predicciones correctas en relación con el total de predicciones. Sin embargo, la exactitud puede ser engañosa si las clases están desequilibradas en el conjunto de datos, lo que es común en la segmentación de texto en imágenes.
- **Precisión y Recall:** La precisión es la proporción de ejemplos positivos correctamente clasificados respecto a todos los ejemplos clasificados como positivos. El recall (también conocido como sensibilidad o tasa de verdaderos positivos) es la proporción de ejemplos positivos correctamente clasificados respecto a todos los ejemplos positivos en el conjunto de datos. En la extracción de texto, un alto recall es importante para asegurarse de que la mayoría del texto real sea capturado.
- **F1-Score:** El F1-Score es la media armónica de precisión y recall. Es una métrica que equilibra tanto los falsos positivos como los falsos negativos, lo que es útil cuando las clases están desequilibradas.

**Figura 3.10: Medidas Matriz Confusion**

- métrica de Jaccard: En el contexto de la extracción de texto en imágenes o la segmentación de objetos en imágenes, la métrica de Jaccard se utiliza para evaluar la superposición y la precisión de la segmentación. Se calcula comparando las áreas superpuestas entre la máscara de segmentación predicha y la máscara de segmentación real, y luego dividiendo esto por la unión de las áreas de ambas máscaras.

**Figura 3.11: Indice Jaccard**

- Métricas de tiempo de ejecución: Además de las métricas de rendimiento, también es importante considerar el tiempo de ejecución del modelo para tareas en tiempo real o aplicaciones en las que la velocidad es crucial.

La elección de las métricas dependerá de la naturaleza de tu problema y de los objetivos específicos de evaluación. Puede ser útil considerar varias métricas en conjunto para obtener una imagen completa del rendimiento de tu modelo SVM en la extracción de texto en imágenes.

Desarrollo del Proyecto

4

4.1. Recoleccion y Generacion de Cartas

Para el planteamiento que tengo del proyecto es necesario conseguir cartas fisicas de Bancos y Subsidios en las cuales se plazme la informacion de la adjudicacion de estos prestamos y subsidios para la adquisicion de vivienda nueva.

Se conseguiran Cartas de cada tipo base (Credito,Subsidio) , entendiendo que serian combinadas entre 3 de Bancos para Creditos (Davivienda, Bancolombia y BCSC) y 3 de Cajas para Subsidios (CAFAM, Compensar y Colsubsidio), con estas haria la base o el insumo inicial para trabajar y a medida que avance en la ejecucion del objetivo generaria una BD mas grande de forma manual con clases equilibradas.

4.2. Entendimiento y Exploracion de los Cartas

Las cartas son cartas de credito y subsidios como ya se ha mencionado y manejan una estructura estandar con algunas variaciones. Las variaciones mas que estructurales son de textos ; pero al mismo tiempo manejan algunos textos y ubciaciones muy concretas que pueden facilitar el reconocimiento de la informacion buscada en las mismas.

La informacion textual que se busca es :

- Tipo: Saber o Determinar si es una Carta de Credito o de Subsidio.
- Cedula y Nombre del Beneficiario: La cedula y el nombre de la persona a la que se le aprobo el Credito o se le asigno el subsidio.
- Entidad: Nombre del Banco o Caja que desembolsara el dinero Asignado.
- Valor: Cantidad de Dinero Asignado o prestado por la entidad.
- Fecha: Fecha en que se genera la carta con el fin de saber que tan antiguo es el dato de asignacion.

Adicional a esto estructuralmente podemos partir las cartas en 3 segmentos:

- **Encabezado:** Es el primer tercio del documento donde se encuentran la mayoría de los campos Cedula y Nombre del Beneficiario, Nombre Entidad normalmente en forma de logo , Fecha.
- **Cuerpo:** Es el segundo tercio del documento donde encontraremos el Tipo, Valor y en algunos casos poco probables la cedula.
- **Extra:** Es el tercer tercio y las demas paginas en caso de existir esta parte de documento no es relevante para el proyecto pues no trae informacion que estemos buscando en ninguno de los casos.

En el etiquetado de las clases lo que se haria es generar un dataset con las cartas y su clases iniciales que serian : Credito Bancolombia, Credito Davivienda, Credito BCSC, Subsidio CAFAM, Subsidio Compensar y Subsidio Colsubsidio. Lo que nos daría inicialmente 6 clases.

4.3. Transformacion y Preprocesamiento de Cartas

En esta seccion mostrare y explicare el codigo fuente y el plan de trabajo que tengo para realizar tareas de preprocesamiento que ayuden al alcance del objetivo

```
1
2  #Conversion y Validacion de Imagenes
3  import magic
4  import cv2
5  from pdf2image import convert_from_path, convert_from_bytes
6
7
8  #Preprocesamiento de Imagenes
9  from PIL import Image, ImageFilter
10 import numpy as np
11
12
13 #Extraccion de Caracteristicas y Limpieza de texto
14 import pytesseract
15 from pytesseract.pytesseract import Output
16 import re
```

Listing 4.1: Librerías que vamos a utilizar

4.3.1. Conversion y Validacion de Imagenes

El proyecto solo abarcara los formatos de imagen mas comunes y los archivos en PDF, entendiendo esto es necesario identificar el tipo de formato con el fin de saber si es PDF para

realizar la tarea de conversion y si no es PDF ni es una Imagen no seguir con el proceso.

Para ello cree una funcion llamada (validaarchivo) que nos ayuda con estas tareas y nos devuleve la imagen.

```

1 def valida_archivo(archivo_ruta):
2     #Verifico el tipo de Archivo
3     mime = magic.Magic()
4     tipo_archivo = mime.from_file(archivo_ruta)
5
6     #Creo dos variables para identificar si es Imagen o PDF el archivo que entra.
7     es_pdf = 'pdf' in tipo_archivo.lower()
8     es_imagen = 'image' in tipo_archivo.lower()
9
10    #Dependiendo si es Imagen hago la tarea de lectura y antes la de
        transformacion en caso de ser PDF
11    if es_pdf:
12        pages = convert_from_path(pdf_path=archivo_ruta, dpi=300, last_page=1)
13        for page in pages:
14            page.save(archivo_ruta, 'PNG')
15            imgCV = cv2.imread(archivo_ruta)
16    elif es_imagen:
17        imgCV = cv2.imread(archivo_ruta)
18    else:
19        imgCV = 0
20
21    #retorno la imagen en caso de no ser ni imagen ni PDF retorno 0
22    return imgCV

```

Listing 4.2: *Funcion de Validacion y Generacion Imagen en Python*

4.3.2. Preprocesamiento de Imagenes

Luego Avanzamos en la mejora de la calidad de la imagen para ello cree una funcion llamada (preprocesadaimagen) esta recibira la imagen y le haremos las modificaciones que creamos convenientes que para este caso son mejora de contraste, umbralizar imagen y aplicar filtro o suavizar imagen.

Para ello trabajaremos con la libreria PILLOW, generamos las matrices para mejorar el contraste sin subirlo demasiado, luego umbralizamos con el fin de resaltar negros y blancos entendiendo que ya la imagen se convirtio en escala de grises y al final filtramos para disminuir el ruido.

```

1 def preprocesada_imagen(imagen):
2
3     # Convertir la matriz en una imagen de Pillow
4     imagen_pillow = Image.fromarray(imagen)
5
6     # Convertir la imagen en un arreglo NumPy

```

```

7     matriz_imagen = np.array(imagen_pillow)
8
9     ##### CONTRASTE #####
10    factor_contraste = 1.2
11    imagen_mejor_contraste = np.clip(matriz_imagen * factor_contraste, 0, 255).
    astype(np.uint8)
12
13    # Convertir la matriz mejorada de nuevo a una imagen de Pillow
14    imagen_pillow_mejor_contraste = Image.fromarray(imagen_mejor_contraste)
15    imagen_gris = imagen_pillow_mejor_contraste.convert('L')
16
17    ##### UMBRALIZAR #####
18    umbral = 220
19    imagen_umbralizada = imagen_gris.point(lambda p: 255 if p > umbral else 0)
20
21    ##### FILTRADO #####
22    imagen_filtrada = imagen_umbralizada.filter(ImageFilter.GaussianBlur(radius
    =1.3))
23
24    # retono la imagen preprocesada
25    return imagen_filtrada

```

Listing 4.3: Funcion para Preprocesar la Imagen en Python

4.3.3. Extraccion de Caracteristicas y Limpieza de texto

```

1 def caracteristicas_imagenes(imagen_preprocesada):
2
3     text = pytesseract.image_to_string(imagen_preprocesada, output_type=Output.
    STRING, lang='spa')
4     texto_limpio = re.sub(r'[!¿?]', '', text)
5     texto_limpio = re.sub(r'\n', ' ', texto_limpio)
6
7     return texto_limpio

```

Listing 4.4: Funcion para Extraer Caracteristicas y Limpiar Textos de la imagen en Python

4.3.4. Creacion Listas con asignacion de Etiquetas

```

1
2 # Extraer caracteristicas y crear una matriz de caracteristicas
3 imagen_rutas = ['RAT_BANCOL.pdf', 'RAT_BANCOL2.pdf', 'INIBANCOLOMBIA.pdf', 'INI-
    BCSC.pdf', 'RAT-BCSC.pdf', 'T2-0808-BANCOLOMBIA.pdf', '03 0503-CARTA INICIAL
    -BCSC.pdf', 'CARTA_DAVIVIENDA.tif', 'CREDITO_DAVIVIENDA_03-0102.tif',
4     'AJSCAFAM.pdf', 'AJSCOLSUBSIDIO.pdf', 'AJSCOMPENSAR.pdf', '
    AJUSTE_CAFAM_T2_1009.pdf', 'APROBCAFAM.pdf', 'APROBCOLSUBSIDIO.pdf', '
    APROBCOMPENSAR.pdf', 'CARTA_AJUSTE_COMPENSAR.pdf', 'CARTA_SUBSIDIO_CAFAM_02
    -1009.pdf', 'PRIM_T1-0704_COMPENSAR.pdf', 'CARTA_SUBSIDIO_COLSUBSIDIO_T3
    -0102.pdf']

```

```

5
6 labels = ['credito', 'credito', 'credito','credito', 'credito', 'credito','
    credito', 'credito', 'credito',
7     'subsidio', 'subsidio', 'subsidio','subsidio', 'subsidio', 'subsidio','
    subsidio', 'subsidio', 'subsidio', 'subsidio', 'subsidio']
8
9 caracteristicas = []
10
11 for path in imagen_rutas:
12     img = valida_archivo(path)
13
14     if type(img) != int:
15
16         imagen_preprocesada = preprocesada_imagen(img)
17         caracteristicas.append(caracteristicas_imagenes(imagen_preprocesada))
18
19 caracteristicas

```

Listing 4.5: *Codigo para Extraer Caracteristicas de una lista de Imagenes ya Etiquetadas en Python*

4.4. Modelado y Entrenamiento con SVM

En esta seccion mostrare y explicare el codigo fuente y el planetamiento que tengo para entrenar el modelo SVM el cual usare para la clasificacion de las cartas, de esta manera detectare el tipo de carta y la entidad de la misma. Luego este modelo sera guardado para un uso posterior en futuras predicciones

```

1
2 #Vectorizacion de caracteristicas
3 from sklearn.model_selection import train_test_split
4 from sklearn.preprocessing import LabelEncoder
5 from sklearn.feature_extraction.text import TfidfVectorizer
6
7
8 #Modelo y Entrenamiento SVM
9 from sklearn.svm import SVC
10
11 #Guardar Modelos y Vectorizadores
12 import joblib

```

Listing 4.6: *Librerias que vamos a utilizar para el Modelado*

Entendiendo que lo que vamos a recibir son caracteristas de tipo String por que es la extraccion del texto como tal y que las etiquetas o labels tambien son de tipo String, se hace necesario vectorizar esos datos para que el SVM los reciba ya que como sabemos la naturaleza de este algoritmo y sus calculos son basados en vectores e hiperplanos. Para este caso

usare `TfidfVectorizer()` el se encargara de convertir el conjunto de textos en vectores numericos, y al mismo tiempo su relevancia en el documento osea que tantas veces sale en el documento.

```
1  codificador = LabelEncoder()
2  etiquetas_numericas = codificador.fit_transform(labels)
3
4  # Crear un objeto TfidfVectorizer
5  vectorizador_tfidf = TfidfVectorizer()
6  # Ajustar y transformar los documentos en vectores TF-IDF
7  vectores_tfidf = vectorizador_tfidf.fit_transform(caracteristicas)
8  # Convertir los vectores TF-IDF a una matriz densa
9  matriz_tfidf = vectores_tfidf.toarray()
```

Listing 4.7: *Codigo fuente para vectorizar las caracteristicas y las etiquetas en Python*

ya teniendo en formatos numericos las caracteristicas y los labels podemos dividir los datos para pruebas y para entrenamiento. Con el fin de entrenar el algoritmo.

```
1  # Dividir los datos en conjuntos de entrenamiento y prueba
2  X_train, X_test, y_train, y_test = train_test_split(matriz_tfidf,
3  etiquetas_numericas, test_size=0.2, random_state=42)
4
5  # Crear un modelo SVM
6  modelo_svm = SVC(kernel='linear', C=1)
7
8  # Entrenar el modelo
9  modelo_svm.fit(X_train, y_train)
```

Listing 4.8: *Codigo fuente para entrenar el modelo en Python*

Cuando el entrenamiento termina guardamos modelo y adicional es necesario guardar el vectorizador con el que se entreno pues al momento de recibir una nueva imagen para predecir es necesario someterla al mismo proceso de vectorizacion con el que se entreno.

```
1
2  modelo = modelo_svm
3
4  # Guardar el modelo en un archivo
5  nombre_archivo = "modelo_entrenado_svm.joblib"
6  joblib.dump(modelo, nombre_archivo)
7
8  nombre_archivo_vectorizador = "vectorizador_tfidf_svm.joblib"
9  joblib.dump(vectorizador_tfidf, nombre_archivo_vectorizador)
```

Listing 4.9: *Funcion para Extraer Caracteristicas y Limpiar Textos de la imagen en Python*

Codigo fuente para cargar el modelo con el vectorizador y realizar una prediccion.

1

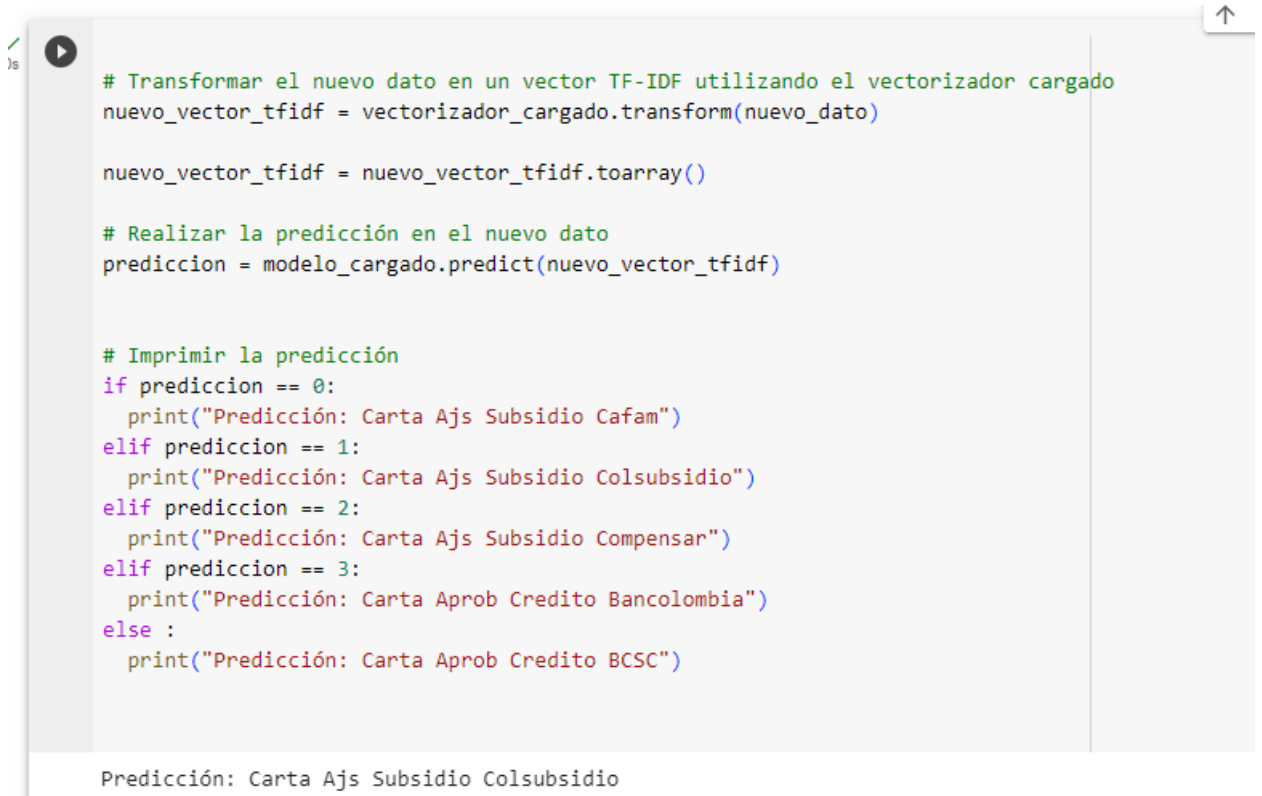

```

2  modelo_cargado = joblib.load(nombre_archivo)
3
4  # Cargar el objeto TfidfVectorizer desde el archivo
5  nombre_archivo_vectorizador = "vectorizador_tfidf_svm.joblib"
6  vectorizador_cargado = joblib.load(nombre_archivo_vectorizador)
7
8  nuevo_dato = []
9
10 #nueva imagen en este caso un archivo que no usamos en el entrenamiento (
    subsidio Pedro Carvajal.pdf)
11 img = valida_archivo('subsidio Pedro Carvajal.pdf')
12 imagen_preprocesada = preprocesada_imagen(img)
13 nuevo_dato.append(caracteristicas_imagenes(imagen_preprocesada))
14
15
16 # Transformar el nuevo dato en un vector TF-IDF utilizando el vectorizador
    cargado
17 nuevo_vector_tfidf = vectorizador_cargado.transform(nuevo_dato)
18 nuevo_vector_tfidf = nuevo_vector_tfidf.toarray()
19
20 # Realizar la prediccion en el nuevo dato
21 prediccion = modelo_cargado.predict(nuevo_vector_tfidf)
22
23 # Imprimir la prediccion
24 if prediccion == 0:
25     print("prediccion: Carta Ajs Subsidio Cafam")
26 elif prediccion == 1:
27     print("prediccion: Carta Ajs Subsidio Colsubsidio")
28 elif prediccion == 2:
29     print("prediccion: Carta Ajs Subsidio Compensar")
30 elif prediccion == 3:
31     print("prediccion: Carta Aprob Credito Bancolombia")
32 else :
33     print("prediccion: Carta Aprob Credito BCSC")

```

Listing 4.10: *Funcion para Extraer Caracteristicas y Limpiar Textos de la imagen en Python*

La Ejecucion del codigo anterior nos da como resultado (Carta Ajs Subsidio Colsubsidio) indicandonos que es una carta de Subsidio de la Caja de compensacion familiar Colsubsidio.



```

# Transformar el nuevo dato en un vector TF-IDF utilizando el vectorizador cargado
nuevo_vector_tfidf = vectorizador_cargado.transform(nuevo_dato)

nuevo_vector_tfidf = nuevo_vector_tfidf.toarray()

# Realizar la predicción en el nuevo dato
prediccion = modelo_cargado.predict(nuevo_vector_tfidf)

# Imprimir la predicción
if prediccion == 0:
    print("Predicción: Carta Ajs Subsidio Cafam")
elif prediccion == 1:
    print("Predicción: Carta Ajs Subsidio Colsubsidio")
elif prediccion == 2:
    print("Predicción: Carta Ajs Subsidio Compensar")
elif prediccion == 3:
    print("Predicción: Carta Aprob Credito Bancolombia")
else :
    print("Predicción: Carta Aprob Credito BCSC")

```

Predicción: Carta Ajs Subsidio Colsubsidio

Figura 4.1: Resultado Prediccion

4.5. Extraccion de Texto

Existen varias maneras que estudie para la extraccion del texto de las cartas pero algunas tienen limitaciones por lo que para el planetamiento del proyecto lo hare de la manera mas sencilla desde el punto de vista de codificacion y adicional dare un sesgo practico teorico de como planteralo con otras maneras y sus obstaculos.

4.5.1. valores IDF del Vectorizador usado en SVM

La primer idea que me surgio fue usar esta calificacion para capturar las palabras menos comunes entendiendo que la estructura de la carta es casi identica donde solo variarian Nombres , Fechas, Cedula y Valores, esto si o si asumiendo que ya esta clasificada por el SVM pero al entrar en detalle no fue optimo, me devolvio mucha informacion que fue ya poco clasificable y desisti de esta forma.

```

1
2 valores_idf = vectorizador_cargado.idf_
3
4 # Obtener las palabras (caracteristicas) que se utilizan en el vectorizador
5 palabras = vectorizador_cargado.get_feature_names_out()
6

```

```

7  # Crear un diccionario que asocie palabras con sus valores IDF
   correspondientes
8  idf_dict = dict(zip(palabras, valores_idf))
9
10 # Imprimir los valores IDF
11 for palabra, idf_valor in idf_dict.items():
12     if(idf_valor) > 3:
13         print(f"Palabra: {palabra}, IDF: {idf_valor}")

```

Listing 4.11: *Codigo Fuente para saber los coeficientes IDF de las palabras en Python*

```

Palabra: 0000201439, IDF: 3.3513752571634776
Palabra: 009, IDF: 3.3513752571634776
Palabra: 00962, IDF: 3.3513752571634776
Palabra: 011, IDF: 3.3513752571634776
Palabra: 015, IDF: 3.3513752571634776
Palabra: 020, IDF: 3.3513752571634776
Palabra: 022, IDF: 3.3513752571634776
Palabra: 032, IDF: 3.3513752571634776
Palabra: 05700002300452214, IDF: 3.3513752571634776
Palabra: 05700002400387526, IDF: 3.3513752571634776
Palabra: 0704, IDF: 3.3513752571634776
Palabra: 09, IDF: 3.3513752571634776
Palabra: 1014226015, IDF: 3.3513752571634776
Palabra: 1022347505, IDF: 3.3513752571634776
Palabra: 103, IDF: 3.3513752571634776
Palabra: 103061760154806, IDF: 3.3513752571634776
Palabra: 103061760157146, IDF: 3.3513752571634776
Palabra: 1032404516, IDF: 3.3513752571634776
Palabra: 1049794815, IDF: 3.3513752571634776
Palabra: 1080x1186, IDF: 3.3513752571634776
Palabra: 1118167934, IDF: 3.3513752571634776
Palabra: 114, IDF: 3.3513752571634776
Palabra: 1140938750, IDF: 3.3513752571634776
Palabra: 115, IDF: 3.3513752571634776
Palabra: 1156358, IDF: 3.3513752571634776
Palabra: 1163675, IDF: 3.3513752571634776
Palabra: 1190, IDF: 3.3513752571634776
Palabra: 148, IDF: 3.3513752571634776
Palabra: 15, IDF: 3.3513752571634776

```

Figura 4.2: *valores IDF del Vectorizador*

4.5.2. Librerías de NPL

Otra de las formas que desisti de usar fue el manejo de librerías como spaCy y NLTK (Natural Language Toolkit) ya que su precision no fue buena, en la mayoría de los caso clasifico entidades erradas y no reconocio otras, adicional en el manejo de las fechas tipo texto no es bueno por lo menos en el idioma español esos fueron los resultados que evidencie.

4.5.3. Expresiones regulares

En esta dinamica la forma es capturar el texto de entrada del documento a Identificar, este documento ya debe estar previamente preprocesado y clasificado por el SVM que ya vimos.

Entendiendo que con la clasificacion ya obtuvimos el tipo y la entidad, tendríamos que extraer ahora la fecha, el numero de Documento y el valor.

```

1
2  import re
3
4  # Texto es la cadena de la nueva carta
5  texto = nuevo_dato[0]

```

```

6 parte_del_texto = texto[:1000] # Recorto el texto pues ya se que en esta
   cantidad de caracteres encuentro lo que necesito.
7 print(parte_del_texto)
8
9 # Defino los patrones para fecha, cedula y valor con base a las
   características de las cartas.
10 fecha_pattern = r'\d{1,2}\s+de\s+[A-Za-z]+\s+de\s+\d{4}'
11 cedula_pattern = r'.{0,50}(?:iden.*|CC|C\C\.)\s*([0-9,]{7,})'
12 valor_pattern = r'\d[0-9,]{7,}'
13
14
15 # Buscar coincidencias de fechas utilizando expresiones regulares
16 fechas_encontradas = re.findall(fecha_pattern, texto)
17 cedula_encontradas = re.findall(cedula_pattern, texto)
18 valor_encontradas = re.findall(valor_pattern, texto)

```

Listing 4.12: Código Fuente para extraer textos puntuales de la carta con ExpReg en Python

```

[ ] for val in valor_encontradas:
    print(val)

```

1057594297
81,800,000

Figura 4.3: Textos Expresiones regulares Cedula/Valor

```

[ ] for fecha in fechas_encontradas:
    print(fecha)

```

03 de noviembre de 2020

Figura 4.4: Textos Expresiones regulares Fecha

4.5.4. Redes Neuronales

Esta sería la forma más compleja de obtener los textos ya que tengo varios obstáculos entre los más relevantes la data de entrenamiento y la generación de coordenadas que es de forma manual.

El set de datos debería tener una estructura como la que propongo donde las características que ya obtuvimos son las mismas, y las etiquetas vienen en un arreglo con tuplas, en cada tupla estaría la coordenada inicial y final del texto a encontrar y en la tercera posición la etiqueta que le daría.

Adicional coloqué un segmento de código de lo que para mí sería el comienzo de ejecución de una red Neuronal para lo que hemos venido trabajando.

```

2
3 # Etiquetas: Listas de tuplas (inicio, fin, categoria) para cada entidad en
  cada carta
4
5 textos_carta = caracteristicas
6
7 etiquetas = [
8     [(27, 47, "fecha"), (36, 51, "valor")],
9     [(27, 47, "fecha"), (36, 51, "valor")],
10    [(27, 49, "fecha"), (50, 60, "valor")],
11    [(27, 51, "fecha"), (45, 57, "valor")],
12
13 ]
14
15 # estructura inicial RedNeuronal
16
17 import tensorflow as tf
18 from tensorflow.keras import layers
19
20 # Tendiarnos una red secuencial
21 model = tf.keras.Sequential([
22     # La entrada seria inicialmente sin tamano osea podria entrar cualquier
    tamano y que son de tipo texto.
23     layers.Input(shape=(None,), dtype=tf.string),
24     # usando la funcion map_fn y una lambda lo que hacemos es simplemente
    etiquetar los texto formando secuencias
25     layers.Lambda(lambda x: tf.map_fn(lambda x: tf.strings.join(['<s> ', x, '
    </s>'], x)), x)),
26     # con la secuencias ya puedo hacer representaciones numericas y limito
    los tokens a 1000
27     layers.TextVectorization(max_tokens=1000, output_mode='int'),
28     # con la capa Embedding ayudamos que la red comprenda de mejor forma el
    texto vectorizado.
29     layers.Embedding(1000, 32),
30     # aca tendriamos una capa recurrente esta capa es bidireccional y con ella
    capturamos patrones ida y vuelta en las secuencias que obtenemos para al
    final devolverla toda completa.
31     layers.Bidirectional(layers.LSTM(64, return_sequences=True)),
32     #colocamos una cpa densa de salida con 2 que son las etiquetas a devolver
    con softmax que es el mas comun
33     layers.Dense(2, activation='softmax')
34 ])
35
36

```

Listing 4.13: *Codigo Fuente para extraer textos puntuales de la carta con ExpReg en Python*

4.6. Evaluacion

4.6.1. Matrices de confusion

4.6.2. Metricas y Resultados

Conclusiones y Limitaciones Futuras

5

Referencias

6

Anexos

7