

**UNIVERSIDAD NACIONAL AGRARIA LA MOLINA**

**FACULTAD DE INGENIERÍA AGRÍCOLA**



**“IDENTIFICACIÓN DE ESTADOS CONSTRUCTIVOS EN VIVIENDAS  
EN EL VALLE CHILLÓN, EMPLEANDO HERRAMIENTAS DE  
APRENDIZAJE AUTOMÁTICO (MACHINE LEARNING)”**

**TESIS PARA OPTAR EL TÍTULO DE INGENIERO AGRÍCOLA**

**JOSE CARLOS MATTUS ZAPATA**

**LIMA – PERÚ**

**2021**

**UNIVERSIDAD NACIONAL AGRARIA LA MOLINA**

**FACULTAD DE INGENIERÍA AGRÍCOLA**

**“IDENTIFICACIÓN DE ESTADOS CONSTRUCTIVOS EN  
VIVIENDAS EN EL VALLE CHILLÓN, EMPLEANDO  
HERRAMIENTAS DE APRENDIZAJE AUTOMÁTICO  
(MACHINE LEARNING)”**

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE:

**INGENIERO AGRÍCOLA**

Presentado por:

**BACH. JOSE CARLOS MATTUS ZAPATA**

Sustentado y aprobado por el siguiente jurado:

Arq. Victor Eduardo Linares Zaferson  
Presidente

Ing. Saúl Torres Murúa  
Miembro

Ing. Carlos Alberto Bravo Aguilar  
Miembro

Phd. Victor Levingston Peña Guillen  
Asesor

LIMA -PERÚ

2021

# ÍNDICE GENERAL

I.	INTRODUCCIÓN .....	1
1.1.	Objetivos de la investigación.....	4
1.1.1.	Objetivos específicos.....	4
II.	REVISIÓN DE LITERATURA .....	5
2.1.	El proceso de auto-construcción .....	5
2.1.1.	Los estados de construcción progresiva.....	6
2.1.1.1.	Estado A .....	7
2.1.1.2.	Estado B .....	7
2.1.1.3.	Estado C .....	7
2.1.1.4.	Estado D .....	7
2.1.2.	Gestión de la formalidad.....	8
2.2.	La teledetección del crecimiento urbano .....	9
2.2.1.	Resoluciones de sensores remotos .....	10
2.2.2.	Satélite DG Quickbird .....	12
2.3.	Aprendizaje Automático.....	13
2.3.1.	Tipos de Aprendizaje Automático .....	14
2.3.2.	Aprendizaje Profundo (Deep Learning).....	16
2.4.	Redes Neuronales Convolucionales (CNN).....	19
2.4.1.	Estructura .....	19
2.5.	Clasificador Bayesiano .....	23
2.5.1.	Ejemplo estadístico.....	23
2.6.	Matriz de confusión.....	26
2.7.	Coeficiente Kappa .....	28

2.8.	Ejemplo Aprendizaje supervisado .....	30
III.	METODOLOGÍA .....	33
3.1.	Descripción de la zona de estudio.....	34
3.2.	Datos de entrenamiento .....	35
3.2.1.	Selección de Manzanas de muestra.....	35
3.2.2.	Obtención de imágenes satelitales .....	35
3.2.3.	Clasificación manual de los estados constructivos.....	36
3.2.4.	Depuración hasta conseguir lotes representativos.....	38
3.3.	Codificación de modelos .....	38
3.3.1.	Clasificador CNN .....	39
3.3.2.	Clasificador Bayesiano .....	41
3.4.	Elaboración de matrices de confusión .....	42
3.5.	Ingreso del coeficiente de Kappa.....	44
IV.	RESULTADOS Y DISCUSIÓN .....	45
4.1.	Imágenes de entrenamiento .....	45
4.1.1.	Obtención de imágenes satelitales .....	45
4.1.2.	Clasificación Manual según estados constructivos .....	48
4.1.3.	Depuración de lotes .....	52
4.2.	Proceso de clasificación supervisada-algoritmo .....	53
4.2.1.	Modelo de Redes Neuronales Convolucionales (CNN).....	53
4.2.2.	Modelo Bayesiano (NB).....	60
4.3.	Comparación en exactitudes, pérdidas, tiempos. ....	64
4.4.	Paquete de validación.....	66

4.4.1.	Matriz de confusión .....	66
4.4.2.	Coeficiente de Kappa.....	69
4.5.	Discusión.....	70
V.	CONCLUSIONES .....	73
VI.	RECOMENDACIONES .....	74
VII.	BIBLIOGRAFÍA.....	75
VIII.	ANEXOS .....	80
8.1.	Cuadros de Lotes individuales.....	80
8.2.	Código para Modelo CNN.....	87
8.2.1.	Matriz de confusión Modelo CNN .....	90
8.3.	Modelo Bayesiano .....	92
8.3.1.	Matriz de confusión Modelo Bayesiano .....	94

## ÍNDICE DE TABLAS

Tabla 1: Recopilado de presupuestos aprobados para partida de interés .....	9
Tabla 2: Matriz de confusión para clasificación de dos variables .....	26
Tabla 3: Matriz ejemplo para cálculo de Coeficiente de Kappa .....	28
Tabla 4: Valores obtenidos para la precisión total de los ocho modelos basados en CNN transferidos.....	31
Tabla 5: Categorías de consolidación .....	36
Tabla 6: Librerías utilizadas según modelos.....	39
Tabla 7: Coordenadas de referenciación para manzanas estudiadas .....	46
Tabla 8: Detalles de las imágenes obtenidas .....	47
Tabla 9: Datos de imágenes georreferenciadas para el estudio .....	47
Tabla 10: Clasificación obtenida del proceso de clasificación manual .....	48
Tabla 11 Detalles individuales de lotes correspondientes a la manzana n°02.....	50
Tabla 12: Resumen de datos estadísticos del proceso de lotización de manzanas estudiadas .....	51
Tabla 13: Lotes resultantes a ser utilizados en modelos IA .....	52
Tabla 14: Cantidad de imágenes según finalidad .....	53
Tabla 15: Parámetros de ingreso para imágenes e iteraciones .....	54
Tabla 16: parámetros definidos para la relación con carpetas de data .....	55
Tabla 17: Parámetros detallados correspondientes a la estructura del modelo.....	55
Tabla 18: Parámetros de finalización para la estructura .....	57
Tabla 19: Parámetros de modificación para las imágenes .....	58
Tabla 20: Parámetros finales requeridos para las iteraciones .....	59
Tabla 21: Formato de imágenes .....	61
Tabla 22: Ingreso de imágenes .....	62
Tabla 23: Parámetros empleados en la función train_test_split .....	62
Tabla 24: Parámetros de ajuste del clasificador.....	62
Tabla 25: Parámetros requeridos para la matriz de confusión, CNN .....	67
Tabla 26: Parámetros requeridos para la matriz de confusión, Bayesiano .....	68

Tabla 27: Resumen de resultados en ambos modelos.....	71
---	----

## ÍNDICE DE FIGURAS

Figura n.º 1 Resolución espacial-Diferenciación de tamaño de píxeles.....	11
Figura n.º 2 Resolución espectral del satélite Landsat en el visible e infrarrojo reflejado .....	12
Figura n.º 3 Modelo de neurona estándar .....	14
Figura n.º 4 esquema simplificado de aprendizaje supervisado.....	15
Figura n.º 5 detalles de funciones de capas ocultas iniciales .....	17
Figura n.º 6 Esquema simplificado del Perceptrón Multicapa.....	18
Figura n.º 7Proceso de predicción por Perceptrón Multicapa.....	19
Figura n.º 8Ejemplo de estructura de una Red Neuronal Convolutiva (CNN).....	20
Figura n.º 9 Detalles de capa convolutiva.....	21
Figura n.º 10 Introducción de la no-linealidad al modelo.....	21
Figura n.º 11Funciones de las capas CNN .....	22
Figura n.º 12 Secuencia de cálculo utilizando el Algoritmo de Bayes .....	25
Figura n.º 13 Valoración del coeficiente de Kappa .....	29
Figura n.º 14 Área de estudio: sección noreste de Cloverdale .....	31
Figura n.º 15 Codificación de colores para caso estudiado .....	32
Figura n.º 16 Flujograma de la metodología .....	33
Figura n.º 17 Procedimiento para obtención de Imágenes Georeferenciadas .....	35
Figura n.º 18 Proceso para identificación de estados de consolidación .....	37
Figura n.º 19 Representación de matriz de confusión para problema de clasificación de tres clases .....	44
Figura n.º 20 Ubicación espacial de manzanas seleccionadas .....	45
Figura n.º 21 Estados obtenidos manualmente y representación en vista de perfil.....	49
Figura n.º 22 Resultado del proceso de lotización y clasificación.....	49
Figura n.º 23Depuración de lotes no-representativos .....	52
Figura n.º 24 Organización de carpetas de trabajo. Modelo CNN.....	53
Figura n.º 25 Secuencia de importación de librerías, modelo CNN .....	54

Figura n.º 26 Secuencia de capas y neuronas utilizadas para el modelo CNN .....	55
Figura n.º 27 Proceso de compilación del modelo; los parámetros a utilizar en cada ítem.....	57
Figura n.º 28 Secuencia de generadores de datos .....	58
Figura n.º 29 Cantidad de imágenes encontradas para entrenamiento, validación y evaluación.....	59
Figura n.º 30 Proceso de evaluación del modelo calibrado .....	59
Figura n.º 31 Organización de carpetas de trabajo. Modelo NB.....	60
Figura n.º 32 Secuencia de importación de librerías, modelo NB .....	61
Figura n.º 33 Exactitud obtenida para la data de evaluación. Modelo Bayesiano.....	63
Figura n.º 34 Gráfica de porcentaje de exactitud Vs progreso a través de las iteraciones .....	64
Figura n.º 35 Gráfica de Pérdidas Vs progreso a través de las iteraciones .....	65
Figura n.º 36 Comparación de tiempos empleados por los modelos utilizados .....	66
Figura n.º 37 Matriz de confusión resultante del modelo CNN.....	68
Figura n.º 38 Matriz de confusión resultante del modelo Bayesiano.....	69



***Dedicatoria***

*A mis padres y hermano.*

## RESUMEN

La presente investigación profundiza en la aplicación del aprendizaje automático por medio de la clasificación supervisada a escala de vivienda. Este estudio emplea dos modelos de Aprendizaje Automático (Redes Neuronales Convolucionales y Clasificador Bayesiano) para la identificación de estados constructivos durante el proceso de auto construcción. Los ejemplos tomados en cuenta se localizan en el distrito de Carabayllo, Lima. La investigación se propone adaptar dos algoritmos de Aprendizaje Automático y comparar su eficiencia. Para alcanzar esos objetivos, se realiza primero, la recolección de imágenes del lugar de estudio que conformarán la base de datos del estudio; en segundo lugar, se adapta el código de los modelos para el reconocimiento visual automatizado de los estados constructivos. En el primer proceso se eligen puntos de referencia de las manzanas seleccionadas; a partir de estos se extraerán imágenes de alta resolución de libre disponibilidad. Seguidamente se realiza un pretratamiento de las imágenes para la digitalización de los lotes individuales. Posteriormente se clasificaron según estados constructivos, siguiendo metodologías validadas en diferentes estudios. El resultado fue un total de 117 imágenes clasificadas y georreferenciadas. En el segundo proceso, adaptan los códigos de cada modelo, considerando las rutas de ingreso a la base de datos, el formato de ingreso de las imágenes, la estructura, sus parámetros. Finalmente, se realiza la validación empleando dos diferentes métodos. Los resultados muestran una fuerza de concordancia “moderada” por parte del Clasificador Bayesiano, superando así en distintos aspectos al modelo de Red Neuronal Convolutiva.

### **Palabras Clave:**

Aprendizaje automático, Autoconstrucción, Clasificador Bayesiano, Red Neuronal Convolutiva, CNN.

## ABSTRACT

This research deepens in the application of machine learning through the supervised classification in a house level scale. Here, we proposed the usage of two Machine Learning models (Convolutional Neural Networks and Naïve Bayes Classifier) for the identification of constructive states during the self-construction process. The examples are located in the Carabayllo district, Lima province. The research aims to adapt two algorithms of Machine Learning and compare their efficiency. To achieve this objective, we first gather free-usage images of the area of interest, conforming the basic data of the study. Following, we adapt the models codes for the computer vision task aiming the constructive states. In the first process, we select reference points for the targeted blocks, using them we will extract free-high resolution images. After that, the constructive states were classified manually, following validated methods in various researches. Resulting in a total of 117 georeferenced images. The second process involves the adaptation of each model's code, taking into account the paths of the basic data, size of images, structure and parameters. Finally, a validation is performed using two different methods. Resulting in a “moderate” strength of agreement in the case of Naïve Bayes classifier, outperforming the Convolutional Neural Network in several bearings.

**Keywords:** Machine Learning, self-construction, Naïve Bayes Classifier, Convolutional Neural Network, CNN.

## I. INTRODUCCIÓN

Durante largo tiempo se viene trabajando para lograr un país más competitivo, eficiente y articulado; en tal sentido, desde el Estado se han abordado estos objetivos de diferentes formas para que, integrando el desarrollo y utilización de la tecnología y sus herramientas, se puedan agilizar los diferentes procesos administrativos y de obtención de información en las diferentes instituciones y escalas que conforman el sistema de administración del Estado Peruano. Específicamente, en la dimensión de la gestión urbana, se busca la sistematización y la fluidez en el uso de la gran cantidad de información que debe manejar cada entidad en los tres niveles de gobierno.

Estos esfuerzos han venido desarrollándose a nivel del Ministerio de Vivienda, Construcción y Saneamiento. Un resultado es la creación de la plataforma en internet, denominada Sistema de Información para la Planificación Urbana - GEOPLAN. Esta plataforma brinda información cartográfica de consulta referida al ámbito de trabajo del ministerio. Esta información se alimenta de bases de datos provenientes de fuentes que tienen un carácter temporal estático. Visto que el actual proceso de urbanización viene desarrollándose de forma acelerada, especialmente en ciudades de la costa, la información presentada y los métodos empleados requieren ser complementados con monitoreo más dinámico del cambio. De este modo, a través de la presente investigación, se propone una herramienta de ayuda para la actualización de dicha información, enfocándose en la identificación de estados constructivos en las viviendas, mediante la herramienta de aprendizaje automático. Se explorará y evaluará el empleo de esta herramienta para la automatización tanto en la generación de nuevos datos, como en la actualización de la base de datos existente. De este modo se estima colaborar con el órgano competente de la formalización de predios y de manera más abierta a diversas gestiones que requieran dicha información, en el ámbito municipal, distrital y regional. Complementariamente, se cuantificará la cantidad de recursos que significa para el Estado la actualización manual de bases de datos de urbanización a

escala de manzana. “La provisión de vivienda auto gestionada y auto-producida es conocida fuera del contexto formal en Latinoamérica, como el proceso de auto-construcción de vivienda” (Alfaro Malatesta, 2006). En el Perú desde los años 50 se establecieron políticas respecto a la auto construcción de viviendas, admitiéndola y apoyándola por los gobiernos de turno, se convirtió en una posibilidad de acceso a la vivienda para la población migrante. Adicionalmente, podemos apreciar la relación entre las barriadas y programas diversos de apoyo financiero, de este modo la auto construcción se establece como medio promovido por algunas autoridades para lograr una vivienda de baja inversión inicial, con crecimiento progresivo, de acuerdo a las posibilidades económicas de la población, se convierten en la representación de una mejora sostenida en su calidad de vida y sustentada en los hábitos cooperativos de los migrantes andinos (Acevedo, Schreier & Seinfeld, 2018). En este sentido, la auto provisión de viviendas se plantea como una alternativa viable frente a la incapacidad de acceso a la oferta formal de mercado, o frente a la limitada oferta del parque habitacional de viviendas sociales, de que disponen los gobiernos, para resolver el problema de los “sin casa”, para cuantificar el problema Ccanre, 2016, nos detalla que, “en 1990 el déficit era de 745,079 unidades de vivienda, en 1995 había subido a 892,744 viviendas y, en 1999, había cruzado la barrera del millón de unidades de déficit”.

“La vivienda informal es un tipo distintivo de mercado donde la accesibilidad se acumula en base de la ausencia de planificación y regulación formal” (Baross, 1990; Dowall, 1991). La urbanización metropolitana se da de manera informal debido a diferentes lógicas regulatorias, que se dan particularmente en los bordes rurales-urbanos en las afueras de las ciudades, como por ejemplo en México: la privatización de las irrigaciones, en india, la tierra no “planeada” en las afueras rurales de Calcuta, o en Egipto con las leyes de herencia que han creado parcelas agrícolas delgadas, lineales y en última instancia no cultivables.

Precisamente, el desconocimiento de la tierra no “planeada”, no es un evento reciente, la ausencia de información digital sobre procesos constructivos se da incluso en países desarrollados como Inglaterra, donde su larga cultura de investigación se ve mellada por esta carencia (Rosser et al., 2019). El Perú es un caso especial, donde las barriadas y las viviendas informales han sido un fenómeno ampliamente estudiado (De Soto, Ghersi & Ghibellini, 1986; Driant, 1991; C. Cockburn, 2018; Ccanre, 2016), del cual urge la necesidad de llevar un control y registro del proceso de auto construcción. En esta misma línea de acción se crea COFOPRI en el año 1996, inicialmente con el propósito de formalizar las propiedades informales, con el pasar de los años, la inserción del gobierno tecnológico y la utilización de

herramientas como las computadoras, internet y diferentes softwares (como los de gestión o los relacionados a sistemas de información geográfico). Las actividades de COFOPRI se embarcaron en una nueva ruta de actualización de la información, donde no sólo bastaría la formalización, el registro y cuantificación de los nuevos lotes, sino se debía consolidar en una base de datos actualizada año a año, geo referenciada y disponible en todo momento, con el fin de reportar los avances al gobierno central y articular una mejor gestión entre municipalidades, COFOPRI y en general, cualquier otra institución que lo requiriera.

En las dos últimas décadas y gracias al avance de las tecnologías de procesamiento de información aparece una herramienta computacional denominada aprendizaje automático (Machine Learning). La tecnología de aprendizaje automático potencia muchos aspectos de la sociedad moderna: desde búsquedas en la web hasta filtrado de contenido en redes sociales y recomendaciones en sitios web de comercio electrónico. Los sistemas de aprendizaje automático también se utilizan para identificar objetos en imágenes, transcribir la voz en texto, relacionar noticias, publicaciones o productos con los intereses de los usuarios, y seleccionar resultados relevantes de búsqueda. Cada vez más, estas aplicaciones hacen uso de una clase de técnicas llamadas aprendizaje profundo o Deep Learning. (LeCun, Bengio & Hinton, 2020).

Dentro de las tecnologías de aprendizaje automático la estrategia típica para identificar objetos urbanos es la clasificación de imágenes satelitales, que segrega una imagen con categorías predefinidas como por ejemplo edificios, parques, vehículos, entre otros. Una "imagen" de uso urbano es una unidad que contiene varias características y demuestra un escenario único como residencia, carretera, parques, conjuntos habitacionales, asentamientos humanos. La técnica realiza la extracción de características principales que se presentan en las imágenes originales. Las características del objeto extraído se establecen en dos niveles jerárquicos: bajo y alto, de acuerdo a la complejidad. Las características de bajo nivel contienen detalles menores de las imágenes como líneas, puntos, bordes de curvas, degradados y esquinas. Las características de alto nivel se basan en características de bajo nivel que forman formas u objetos más grandes. Las características arriba descritas permiten automatizar la identificación de los objetos en las imágenes, y de este modo, facilitar la gestión de los espacios urbanos que presentan cambio permanente.

## **1.1. Objetivos de la investigación**

La presente investigación abarca se centra en la aplicación del aprendizaje automático por medio de la clasificación supervisada a escala de vivienda. De este modo, la investigación tiene como objetivo principal el desarrollo y la comparación de dos modelos de inteligencia artificial para la clasificación supervisada de imágenes de viviendas en diferentes etapas constructivas de la zona de Carabayllo, Lima, Perú. Los métodos empleados comprenden dos enfoques de aprendizaje automático: las Redes Neuronales Convolucionales y el Clasificador Naive Bayes (estadística Bayesiana).

### **1.1.1. Objetivos específicos**

- Adecuación de dos diferentes modelos de Inteligencia Artificial (IA): CNN (Redes Neuronales Convolucionales) y Clasificador Naive Bayes (estadística Bayesiana).
- Evaluación de la precisión de los dos modelos mencionados para el caso de las viviendas ubicadas en el distrito de Carabayllo, Lima, Perú.

## **II. REVISIÓN DE LITERATURA**

La presente sección se aproxima al problema de estudio a través de tres áreas teóricas y metodológicas que se consideran esenciales para la comprensión de este estudio: el proceso auto constructivo, herramientas de teledetección, y el aprendizaje automático. Iniciando por los conceptos básicos del proceso auto constructivo y de cómo a través del tiempo las poblaciones migrantes han encontrado en ello, la solución al evidente problema de carencia de vivienda, bien por la inexistencia de oferta o bien por los elevados costos que resultan insolubles para una gran porción de los estratos de bajos y moderados recursos en un país en desarrollo. Seguidamente, se abordará algunas de las herramientas de teledetección que se pueden utilizar para la generación y el tratamiento de la información requerida, como son las imágenes satelitales. Como tercer punto, se desarrollarán los conceptos para la comprensión del uso y empleo del aprendizaje automático (Machine Learning) para los fines de reconocimiento de estados constructivos en las viviendas, siendo este punto el más extenso, por el detalle que se necesita para la estructura de los modelos que se utilizarán, así como ejemplos para su comprensión.

### **2.1. El proceso de auto-construcción**

Para entrar en contexto, se desarrollarán las diferentes definiciones que se han dado, a lo largo del tiempo, al proceso de auto construcción; seguidamente, se abordará la gestión de la formalización a las que se han visto afectas estas agrupaciones de viviendas, productos del proceso auto-constructivo. El estudio de ambos tópicos , converge en destacar la posición que ocupa el proceso, como parte importante de la economía, tanto del porcentaje de PBI, como componente del presupuesto anual que el gobierno central asigna a instituciones responsables de la formalización.

La auto construcción se puede definir como el “Conjunto de procedimientos constructivos y organizativos orientados a la intervención y transformación directa del hábitat



residencial por parte de sus habitantes, de acuerdo a sus propias necesidades, intereses y recursos” (INVI, 2005). También se entiende, en sentido estricto, como las formas de edificación que se realizan mediante la inversión directa de trabajo por los propios usuarios de la vivienda. Bajo estas condiciones sólo es posible aceptar un nivel técnico elemental, por lo regular de índole artesanal. Este proceso también se presenta en la edificación de viviendas provisionales, aunque ya dentro de condicionantes económicas más estrictas que reducen el control del usuario sobre los insumos (Secretaría de Asentamientos Humanos y Obras Públicas, 1978).

Acevedo, Schreier & Seinfeld, 2018, mencionan que la auto-construcción comprende “una modalidad de construcción de la vivienda a cargo de los propios pobladores, junto con la barriada se convierten en una solución inmediata para el Estado que no involucra costos significativos y aprovecha la motivación personal de los pobladores de tener una vivienda propia”.

En términos más actuales, esta actividad “responde a la necesidad de un dormitorio más o una ampliación de la vivienda” (Trillo, 2018). Para entender la magnitud que representan estos agrupamientos, De Soto, Ghersi & Ghibellini, 1986, explican que, “para 1982, estos asentamientos informales son el 42.6% del total de viviendas en Lima, en distritos como Comas, Independencia la totalidad de viviendas son informales, mientras Carabayllo y el Agustino, lo son mayoritariamente”, para la época esto se traducía en 23 712 millones de dólares, es decir el 97.9% del total invertido como país en vivienda. Para el año 2009, la auto construcción representaría “el 3.74% de la producción nacional, un total de S/. 12 358 millones y propiamente en el mercado de materiales y acabados de la construcción, representaría entre el 50% y 60 %” (Mendoza, 2010).

### **2.1.1. Los estados de construcción progresiva**

A diferencia de los procesos auto constructivos en otros países, en el Perú se desarrollaban de manera ordenada, teniendo el espacio necesario, Fernandez (2015) nos explica que, “los bloques tenían tamaños regulares, las calles con sección de ancho convencional y se había reservado espacio para futuro equipamiento urbano como parques, casetas de seguridad, lozas deportivas, entre otros; se tenía una mejor organización que la observada en tugurios centrales”. Las familias tenían libertad para construir de acuerdo a sus necesidades, gustos y medios económicos. Turner (1968) llegó a la conclusión que, en las barriadas peruanas, los

residentes habían seguido la estrategia de vivienda más adecuada a sus circunstancias de vida: el desarrollo progresivo.

Tokeshi et al. (2005), estudiaron e hicieron el seguimiento cuidadoso de los procesos de consolidación de la vivienda auto-construida, categorizando los diferentes estados constructivos, en el sur de Lima; este estudio estableció cuatro (04) estados con el fin de analizar la densificación habitacional.

#### **2.1.1.1. Estado A**

No existe estructura definitiva, los materiales son fácilmente maleables con el tiempo y su desgaste es visible en corto plazo, el valor de la propiedad es nulo o reducido; usualmente presentan la utilización de calaminas, plástico, pallets o cualquier material de transporte sencillo.

#### **2.1.1.2. Estado B**

Vivienda incipiente. Denota una intención estructural singular, se ha erigido a través de dinero de uno o más personas. Los elementos estructurales básicos como cimentación o ciertos muros, existen, siendo parte de la intención a futuro. Resultará sencillo para la misma persona o un tercero hacer modificaciones a la estructura existente de manera que puedan habitar más personas.

#### **2.1.1.3. Estado C**

Vivienda de consolidación media. La intención de construcción se encuentra completamente definida. Todo el primer piso se encuentra terminado y con capacidad de albergar personas, sin embargo, se requiere adicionales, para que una persona o grupo adicional de personas ingresen a vivir en el mismo o en el espacio superior; la remodelación no significa un costo excesivo, dependiendo de la relación costo-beneficio.

#### **2.1.1.4. Estado D**

Vivienda Consolidada. La estructura del primer piso se encuentra terminada y posee detalles, el segundo es habitable con menor o igual nivel de detalles. La independización de ambas unidades inmobiliarias es posible bajo ciertos parámetros (usualmente, adición de una entrada independiente a cada unidad o el reforzamiento de la estructura). Las nuevas construcciones están restringidas a los cimientos y la calidad estructural de la primera planta.

### **2.1.2. Gestión de la formalidad**

La informalidad de la propiedad residencial urbana es muy difundida en el Perú; las grandes ciudades se han erigido sobre la base de ocupaciones e invasiones a terrenos de propiedad privada y estatal. Incluso en el caso de habilitaciones urbanas actualmente renombradas por el poder adquisitivo de sus habitantes como es el caso de Lince (1921), Jesús María (1923), Magdalena Vieja (1924) y San Isidro (1926), en algún momento fueron áreas que se mantenían al margen de la formalidad, estrictamente por no cumplir con los requisitos que, en ese entonces, la ley solicitaba (De Soto, Ghersi & Ghibellini, 1986).

Asimismo, de acuerdo a estos autores los hitos históricos del crecimiento urbano en el sector informal en Lima, incluyen los procesos y actuaciones previas a la invasión de los terrenos, principalmente estatales. Entre estos eventos se incluyen algunas invasiones consentidas, como lo sucedido en Huaycán. Aquello ocurrió ante la falta de celeridad y oportunidad, por parte de los funcionarios responsables en la formalización de la propiedad del terreno. De este modo no hubo alternativas para las autoridades de ese entonces, el alcalde de Lima, Alfonso Barrantes Lingán y del Ministro de Vivienda, Javier Velarde Aspíllaga, que consentir dicha invasión. En la misma línea de acción, se tienen las amnistías a invasores ocurridas en Lima en los años 1993, 1996, y 1999 (C. Cockburn, 2018).

Influenciado por los trabajos de De Soto, a partir de 1996, el enfoque del Estado se orienta hacia la formalización de esta actividad, con la creación de la Comisión de Formalización de la Propiedad Informal - COFOPRI.

A partir de entonces cambiarían las políticas, las cuales serían orientadas a la titulación en grandes cantidades de terrenos. Para este punto se asistirían del parcial orden con el que se erigieron algunas barriadas “entre los años 1955 y 1970 se caracterizan siempre por su trazo regular y una cierta jerarquía de espacio” (Driant, 1991). El sistema legal de reconocimiento de propiedad, a través de la entrega de títulos de propiedad registrados, supuso un cambio cualitativo y cuantitativo en materia de formalización. En primer lugar, porque se trata de títulos que, al momento de su entrega al ciudadano, ya constan inscritos en los registros públicos (antes las autoridades entregaban títulos sin registrar y, en muchos casos, más de un título sobre un mismo predio) (Quilcate Tirado, 2019). Además, estos títulos cuentan con la seguridad de haberse adecuado a sus antecedentes registrales y, al estar inscritos, tienen la protección para oponer el derecho frente a cualquier otra persona que se considere propietaria; es decir, ante concurrencia de acreedores se prefiere al propietario con derecho inscrito (art. 1135 del Código Civil).

Si bien es cierto, “el Estado ha logrado formalizar a un gran volumen de predios a través de sus organismos especializados y con facultades para ello, llámense municipalidades o a través de la agencia de formalización de la propiedad urbana en el país” (COFOPRI, 2018). Complementariamente, en el año 2006 se crearía el Proyecto de Consolidación de la Propiedad Informal (PCDPI), con la asistencia de la Superintendencia Nacional de Registros Públicos (SUNARP), C. Cockburn, 2018 nos explica los objetivos de dicho proyecto con cinco (05) componentes esenciales:

- a) Servicio de formalización y catastro
- b) Servicios registrales para estandarizar y digitalizar la información
- c) Propuestas legales para afianzar la propiedad del inmueble
- d) Difusión de los beneficios de la formalización para que la población formalizada por COFOPRI permanezca en la formalidad
- e) Monitoreo y evaluación

Teniendo en cuenta los dos primeros puntos de este proyecto, COFOPRI asigna un presupuesto específicamente a la generación y actualización de base de datos catastrales, detallados en la Tabla N° 1.

Tabla 1: Recopilado de presupuestos aprobados para partida de interés

Partida	Año	Presupuesto Aprobado (S/.)
Generación de base de datos catastral predial urbana	2017	2 571 913.00
Generación de base de datos catastral predial urbana	2018	1 611 038.00
Generación de base de datos catastral predial urbana	2020	199 583.00

Fuente: COFOPRI, 2016, 2017 y 2019.

## 2.2. La teledetección del crecimiento urbano

En la presente sección se explicarán conceptos fundamentales para el desarrollo de la herramienta empleada en la presente investigación. Siendo las imágenes satelitales uno de los dos componentes principales.

Se entiende como teledetección a la “técnica de obtener información a distancia de objetos sin que exista un contacto material” (Sarría, 2003). La información obtenida no se limita, como el ojo humano a la región espectral del visible; de este modo los sensores que se

encuentran fuera de órbita son multi-espectrales e incluso hiper-espectrales, ello significa que son capaces de “registrar el comportamiento de los objetos de la superficie terrestre en diversas longitudes de onda o bandas del espectro electromagnético, desde la región del visible hasta las distintas bandas del infrarrojo (próximo, medio y térmico) y de las microondas” (Martínez Vega & Martín Isabel, 2010). Teniendo en cuenta su naturaleza inalámbrica para la obtención de la información, son necesarios dos componentes:

- Sensor

Son equipos que convierten la radiación electromagnética en información. Dentro de los mismos se pueden distinguir dos tipos:

- a) Sensores activos: generan su propia radiación y sus instrumentos están adecuados para recibir las ondas reflejadas por los objetos.
- b) Sensores pasivos: reciben radiación emitida o reflejada por la tierra, dentro de los cuales tenemos “sensores fotográficos, óptico-electrónicos que combinan una óptica similar a la fotográfica y un sistema de detección electrónica (detectores de barrido y empuje), espectrómetros de imagen, y de antena” (Sarría, 2003).

- Plataforma

Comprende la estructura física que tiene por objeto transportar los sensores a lo largo de un recorrido preestablecido, globos aerostáticos meteorológicos, drones para fotogrametría, satélites (LANSAT, NOAA, SPOT, etc.).

### **2.2.1. Resoluciones de sensores remotos**

a) Resolución espacial

Es la distancia que corresponde a la unidad mínima de información incluida en la imagen (píxel). Así, a menor tamaño del píxel mayor será la resolución espacial, lo que implica que el sensor obtendrá más detalle de los objetos (Bravo, 2018).

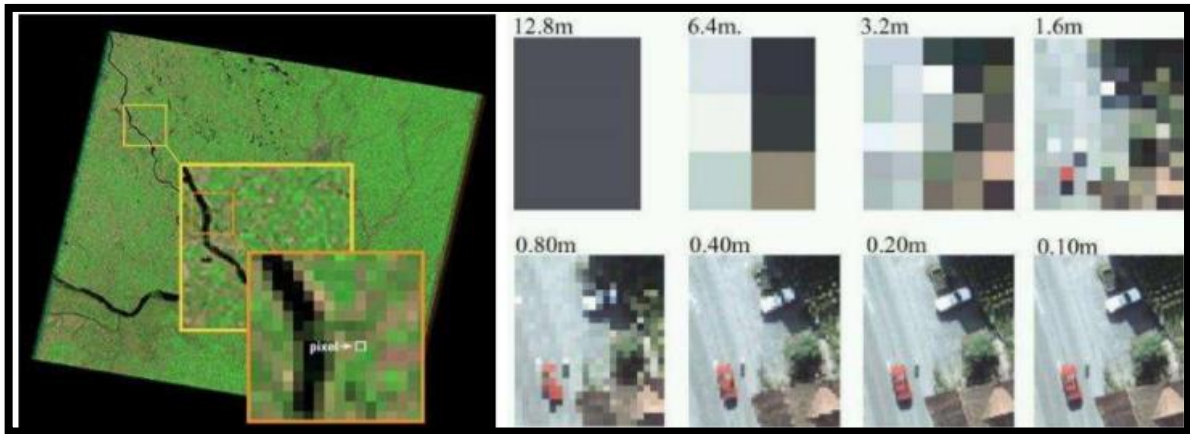


Figura n.º 1 Resolución espacial-Diferenciación de tamaño de píxeles.

Fuente: Bravo, 2018

#### b) Resolución espectral

Esta referido al ancho y número de bandas espectrales que registra el sensor. Sobre la base de este concepto es ampliamente conocido que, a mayor número de bandas, se podrá tener un mayor detalle en las firmas espectrales de diferentes objetos. Asimismo, para la selección del número de bandas se tendrá en cuenta el propósito para el cual se ha diseñado el sensor, como nos explica Benardi, 2009, un sensor destinado a fines meteorológicos tiene que contar con una banda en el visible porque no existen diferencias cromáticas en las nubes, dos en el térmico que le permitan conocer la temperatura de dichas nubes y otra en el infrarrojo medio para observar el contenido de humedad en la atmósfera.

Explicado de otra forma, mediante la Figura n.º 2, donde se visualizan las seis bandas pertenecientes al satélite Landsat (1 al 5 y 7), situadas en los rangos visible e infrarrojo reflejado. “Puesto que la curva situada de fondo es el espectro de emisión del sol, el tamaño de cada una de las barras indica la cantidad de energía que llega a la Tierra en cada una de las bandas” (Sarría, 2003).

#### c) Resolución radiométrica

Es la cantidad de energía que puede captar un sensor, permitiendo obtener el grado de contraste. En el caso de los sistemas fotográficos se la puede definir como la cantidad de tonos de grises que pueden registrar los mismos. Hay que destacar que cuanto mayor sea la precisión radiométrica, mejor será la interpretación de una imagen (Benardi, 2009).

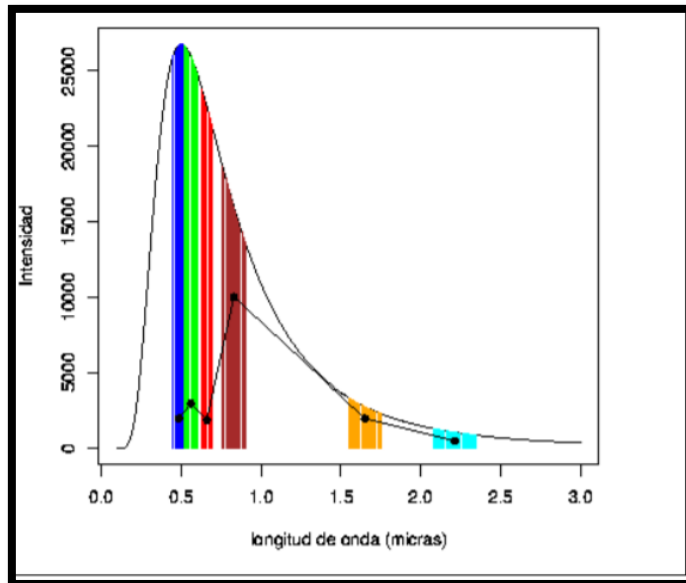


Figura n.º 2 Resolución espectral del satélite Landsat en el visible e infrarrojo reflejado

Fuente: Sarría, 2003

#### d) Resolución temporal

Esta referido al lapso de tiempo en la cual plataforma y sensor orbitan el mismo punto, siendo capaces de obtener, nuevamente, información del mismo. “El ciclo de cobertura está en función de las características orbitales de la plataforma (altura, velocidad, inclinación), así como del diseño del sensor” (Bravo, 2018).

Este autor también explica que, pueden existir tres diferentes tipos de resoluciones según la cantidad de días que demora en completar el ciclo, alta resolución que varía de 1 a 3 días, de 4 a 16 y mayor a 16 días para media y baja resolución, respectivamente.

#### 2.2.2. Satélite DG Quickbird

Es un satélite comercial perteneciente a la empresa DigitalGlobe, lanzado en octubre del 2001; fue durante mucho tiempo el satélite con la mejor resolución espacial disponible en el mercado. Posee tres tipos de productos según el nivel tratamiento que poseen las imágenes:

- Nivel básico, para usuarios experimentados en la materia que poseen el conocimiento y las herramientas para el procesamiento de información sin procesar;
- Nivel estándar, para usuarios cuya área de estudio es reducida y poseen el conocimiento para sacar la mayor cantidad de información en imágenes tratadas parcialmente; y finalmente
- Nivel detallado, que comprende imágenes orto rectificadas, donde se elimina la varianza topográfica, estas imágenes pueden ser cargadas directamente a los softwares de

Sistemas de Información Geográfica; el detalle adicional para estas imágenes es que el usuario debe proveer un modelo de elevación digital (DEM) y puntos de control en el área de interés (GCPs).

A continuación, se presentan las características propias de este satélite (Fuente: Chethan, 2011):

- Resolución espacial: 0.60 m. pancromática, 2.40m. multi espectral
- Bandas espectrales: Pancromático, azul, rojo, verde, NIR
- Capacidad de colección: 16.5 km.\*16.5 km. para múltiples objetivos, 16.5km.\*115km. Para franja larga
- Mínima área de solicitud: 25 km.<sup>2</sup>
- Precisión horizontal: +/- 23m., +/- 23m. y +/- 12 m. para imágenes básicas, estándar y orto rectificadas, respectivamente.

### **2.3. Aprendizaje Automático**

El aprendizaje automático o “Machine Learning” es una forma de Inteligencia Artificial (IA) que permite, a un sistema computacional, aprender de los datos en lugar de hacerlo mediante la programación explícita.

También se le puede definir como “el conjunto de métodos que puede detectar patrones automáticamente en un conjunto de datos y usarlos para predecir datos futuros, o llevar a cabo otro tipo de decisiones en entornos de incertidumbre” (Murphy, 2012).

El aprendizaje automático utiliza una variedad de algoritmos que calculan iterativamente los datos para proveer resultados a partir de categorías previamente identificadas, con los cuales se entrena al algoritmo. De este modo, a medida que los algoritmos reciban nuevos datos de entrenamiento, es posible producir modelos más precisos basados en esos datos. El modelo de aprendizaje automático modifica los parámetros de su algoritmo conforme se va iterando los datos. Después del entrenamiento, cuando se proporcionen nuevos datos de entrada, el modelo proveerá una predicción de salida (Hurwitz & Kirsch, 2018).

Un sistema básico que representa este tipo de sistemas es el modelo de Perceptrón Simple; esta es una analogía de la unidad biológica mínima conocida como neurona. El Perceptrón está formado por un conjunto de diferentes capas y funciones que permiten la operación del modelo de aprendizaje automático.



El Perceptrón Simple (Figura n.º3) fue introducido por Rosenblat (1959) y es un modelo unidireccional compuesto por dos capas de neuronas, una de entrada ( $x_i$ ) y otra de salida ( $y_i$ ); este modelo puede también representarse como una ecuación lineal (Figura n.º 3), donde las variables independientes  $w_{ij}$  denotan los ponderados para cada parámetro (Aguilar, 1999).

$$y_i = f\left(\sum w_{ij} * x_j - \theta_i\right)$$

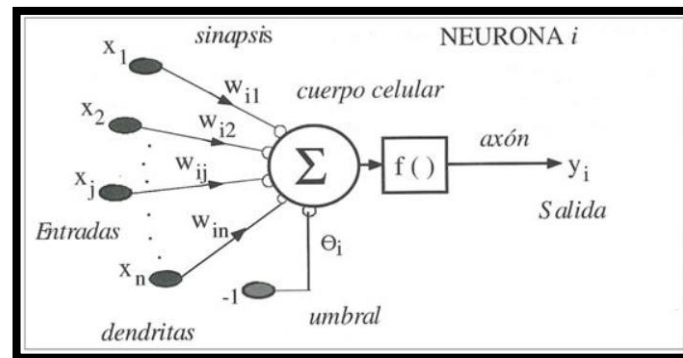


Figura n.º 3 Modelo de neurona estándar

### 2.3.1. Tipos de Aprendizaje Automático

El aprendizaje automático se divide generalmente en dos tipos, aprendizaje supervisado y aprendizaje no supervisado. En el enfoque de aprendizaje predictivo o supervisado, el objetivo es aprender una trayectoria de las entradas “x” a las salidas “y”, dado un conjunto etiquetado de pares entrada-salida.

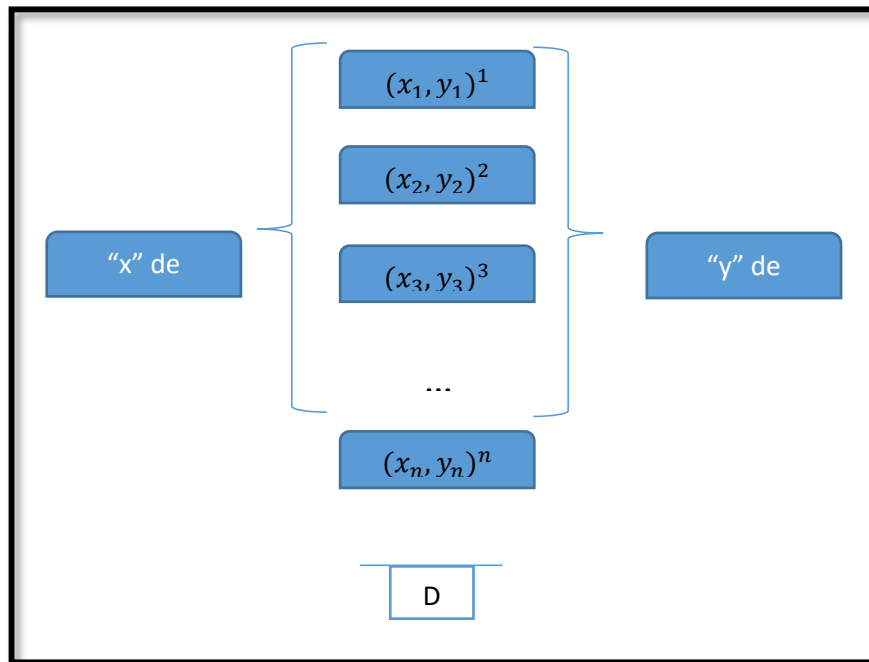


Figura n.º 4 esquema simplificado de aprendizaje supervisado

En la Figura n.º 4, "D" se denomina conjunto de entrenamiento, y "N" es el número de ejemplos de entrenamiento.

Cada entrada de entrenamiento  $x_i$  es un vector de números tridimensionales, que representa, por ejemplo, la forma, el color, la textura de un edificio visto en una imagen RGB de alta resolución. A estos se denominan características, atributos o covariables.

De forma similar, la forma de la variable de salida o respuesta puede ser en principio cualquier valor pre definido, pero la mayoría de los métodos asumen que  $y_i$  es una variable categórica o nominal de un conjunto finito,  $y_i \in \{1, \dots, C\}$  (como masculino o femenino), o que  $y_i$  es un escalar de valor real (como el nivel de ingresos). Cuando  $y_i$  es categórico, el problema se conoce como clasificación o reconocimiento de patrones, y cuando  $y_i$  tiene un valor real, el problema se conoce como regresión.

El segundo tipo principal de aprendizaje automático es el enfoque de aprendizaje descriptivo o no supervisado. Aquí solo se nos dan entradas:

$$D = \{(x_i)\}_{i=1}^N$$

Y el objetivo es encontrar de forma automática "patrones interesantes" en los datos. A esto se le conoce como descubrimiento del conocimiento. Este es un problema mucho menos

definido, ya que no se nos dice qué tipos de patrones hay que buscar, y no hay una métrica de error evidente que usar (a diferencia del aprendizaje supervisado, donde podemos comparar nuestra predicción de “y” para una categoría “x” dada), (Murphy 2012).

### **2.3.2. Aprendizaje Profundo (Deep Learning)**

El aprendizaje profundo es en realidad un subconjunto del aprendizaje automático supervisado. Técnicamente es aprendizaje automático y funciona de la misma manera, pero tiene diferentes capacidades, (Sharma, 2019).

Los métodos de aprendizaje profundo son métodos de representación con múltiples niveles de representación, obtenidos mediante la composición de módulos simples (Ver Figura n.º05), pero no lineales que transforman la representación en un nivel (comenzando con la entrada de datos en bruto) en una representación en un nivel superior, ligeramente más abstracto.

Con la composición de suficientes transformaciones de este tipo, se pueden aprender funciones muy complejas. Para las tareas de clasificación, las capas más altas de representación (capas ocultas, Figura n.º05) amplifican aspectos de la entrada que son importantes para la discriminación y además suprimen las variaciones irrelevantes. Una imagen, por ejemplo (Ver, “x” entrada, Figura n.º03), viene en forma de una matriz de valores de píxeles, y las características aprendidas en la primera capa de representación representan típicamente la presencia o ausencia de bordes en orientaciones y ubicaciones particulares en la imagen. La segunda capa típicamente detecta arreglos particulares de bordes, independientemente de pequeñas variaciones en las posiciones de los bordes. La tercera capa puede ensamblar arreglos en combinaciones más grandes que correspondan a partes de objetos familiares, y las capas subsiguientes detectarían objetos como combinaciones de estas partes. La clave del aprendizaje profundo es que estas capas de características no están diseñadas por ingenieros humanos: se modifican los valores de ponderación mediante un procedimiento de aprendizaje de propósito general llevado a cabo por el algoritmo. (Yann LeCun y Bengio, 2015)

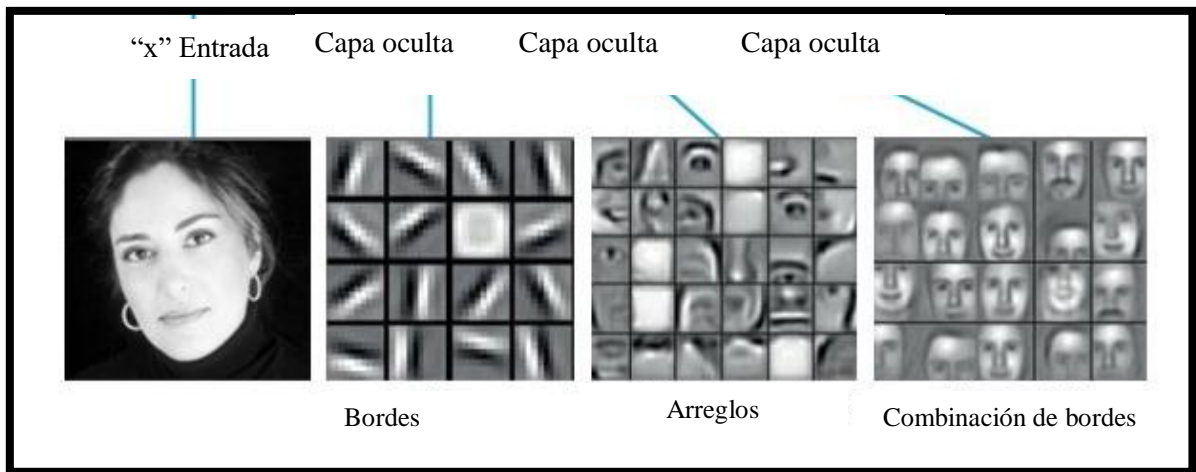


Figura n.º 5 detalles de funciones de capas ocultas iniciales

Fuente: Adaptación Carr 2018.

Para hacer esta representación utilizaremos un esquema representativo de un Perceptrón Multicapa. Este caso corresponde a un arreglo (a través de múltiples capas) de varios Perceptrones Simples.

Como se muestra en la figura n.º06, la arquitectura de Perceptrón Multicapa se caracteriza porque tiene sus neuronas agrupadas en capas de diferentes niveles. Cada una de las capas está formada por un conjunto de neuronas y se distinguen tres tipos de capas diferentes: la capa de entrada, las capas ocultas y la capa de salida.

Este es en la actualidad una de las arquitecturas más utilizadas en la resolución de problemas de identificación automática. Esto es debido, fundamentalmente, a su capacidad como aproximador universal (iterativo), así como a su uso didáctico y aplicabilidad.

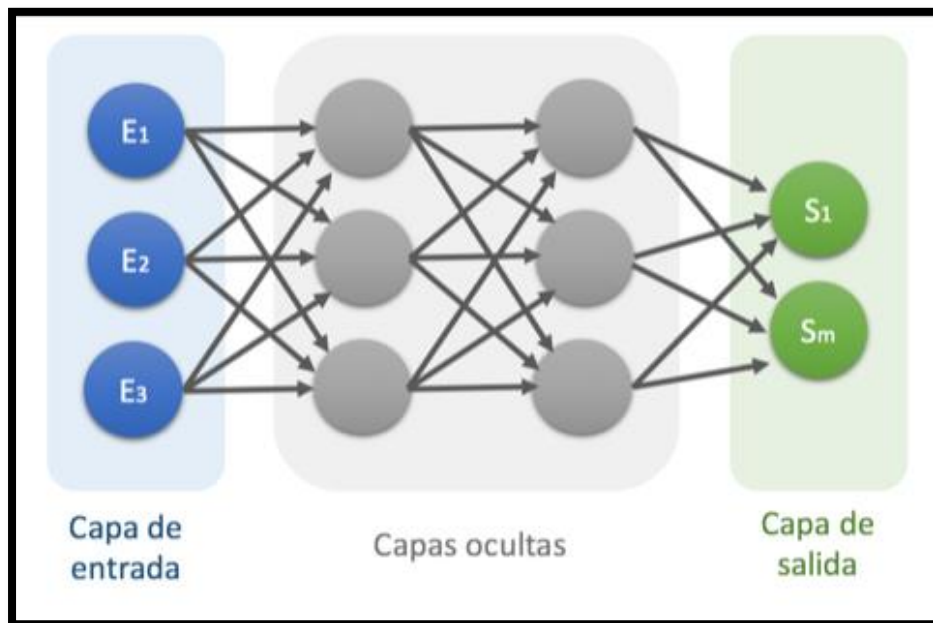


Figura n.º 6 Esquema simplificado del Perceptrón Multicapa

Fuente: DiegoCalvo, 2018

Las neuronas de la capa de entrada (Figura n.º06) no actúan como neuronas propiamente dichas, sino que se encargan únicamente de recibir las señales o patrones del exterior y propagar dichas señales a todas las neuronas de la siguiente capa. La última capa actúa como salida de la red, proporcionando al exterior la respuesta de la red para cada uno de los patrones específicos de la imagen de entrada. Las neuronas de las capas ocultas realizan un procesamiento no lineal de los patrones calculados (Perez Valls, 2013).

Cabe señalar que este proceso de aprendizaje automático tiene un grado de precisión, que se refiere al número de veces que el algoritmo identifica apropiadamente el objeto de estudio.

Para alcanzar un alto grado de precisión es conocido que la cantidad de data de entrenamiento y validación juegan un papel fundamental, llegando incluso a la necesidad de cientos o miles de datos de entrada por cada clase (Brownlee, 2017), por otro lado, Prakash, 2018, nos explica que conforme se ingresen más datos se reducirá la posibilidad de “*overfitting*” o sobre ajuste en el modelo, esto problema quiere decir, que el modelo memorizará cada uno de los datos, no generará una tendencia que represente al conjunto. En ciertos casos para suplir la falencia de la falta de información de entrada se utilizan técnicas de modificación de imágenes conocidas como “*data augmentation*”.

## 2.4. Redes Neuronales Convolucionales (CNN)

Un CNN es una red neural multicapa que está inspirado biológicamente en el córtex visual de los animales, como en una red profunda, las capas anteriores reconocen porciones (como límites), y las capas posteriores recombinan esas funciones en atributos de mayor nivel de entrada, Figura n.º06, (Jones 2017).

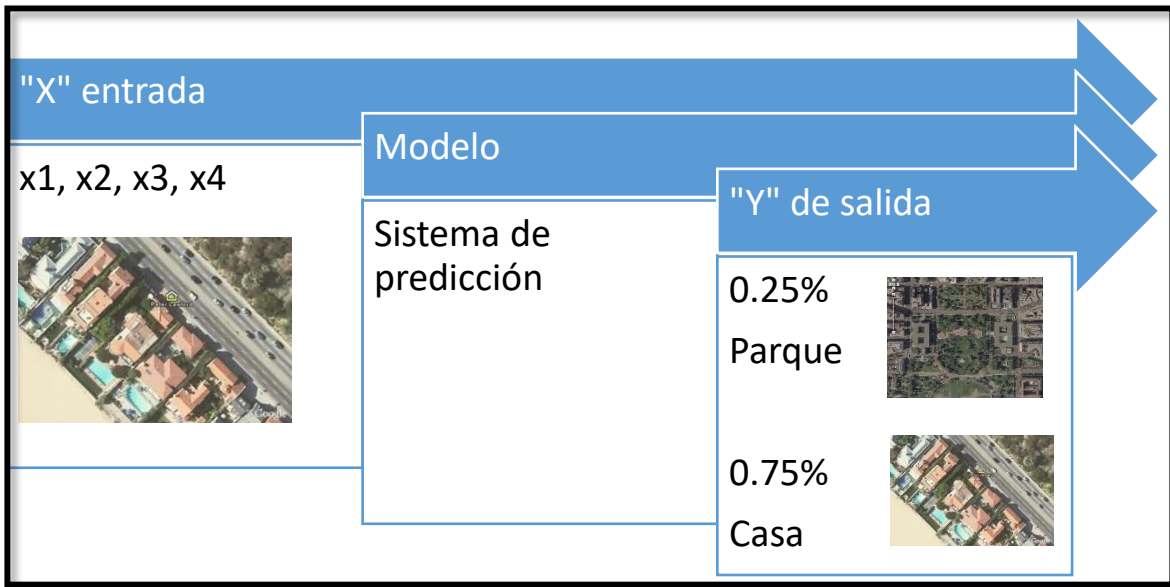


Figura n.º 7 Proceso de predicción por Perceptrón Multicapa

### 2.4.1. Estructura

El método CNN es una secuencia de capas de procesamiento donde cada capa aprende la representación de datos desde niveles bajos hacia niveles altos. Dichas funciones proporcionan información que puede sintetizarse en etapas posteriores para detectar funciones de nivel superior, lo que hace que la CNN funcione como un extractor automático de funciones.

Muchos enfoques modernos en el campo de la visión por computadora explotan la CNN para extraer características singulares que dependen solo de pequeñas sub-regiones de la imagen. Una CNN básica puede incluir cuatro tipos principales de capas (Ver Figura n.º09): capa convolucional (Conv), capa de unidad lineal rectificadora (ReLU), capa de agrupación (Pool)

y capa completamente conectada (FC) (Cao, Dragičević & Li, 2019). Una estructura de ejemplo de CNN se ilustra en la siguiente figura:

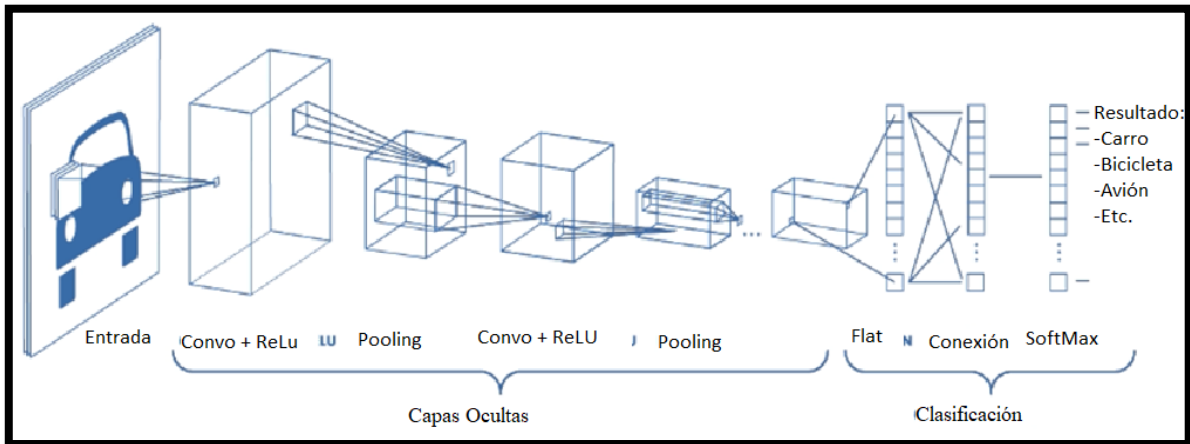


Figura n.º 8 Ejemplo de estructura de una Red Neuronal Convolutiva (CNN)

Fuente: Patel & Pingel, 2017

Una imagen de ingreso de datos está organizada como una matriz de valores de píxeles. Dada una imagen de entrada aleatoria, la CNN reconoce la clase a la que pertenece y los valores de probabilidad de que la entrada pertenezca a cada clase. Hay cuatro tipos básicos de capas que forman una CNN, sus funciones se detallan a continuación:

- a) La capa convolutiva realiza la operación lineal haciendo multiplicaciones y sumas de elementos en cada sub-entrada, utilizando un grupo de matrices de peso denominadas filtros de características. Los filtros de características aprenden respectiva ponderación a partir de los datos de entrenamiento; entonces cada filtro puede detectar la existencia de una característica específica del objeto en la imagen. Las salidas se llaman mapas de características (Ver Figura n.º09). Un mapa de características no solo registra los valores de operación sino también la ubicación espacial relativa de estos valores. En el mapa de características, un valor de salida más alto indica la existencia potencial de la característica correspondiente en su ubicación relativa

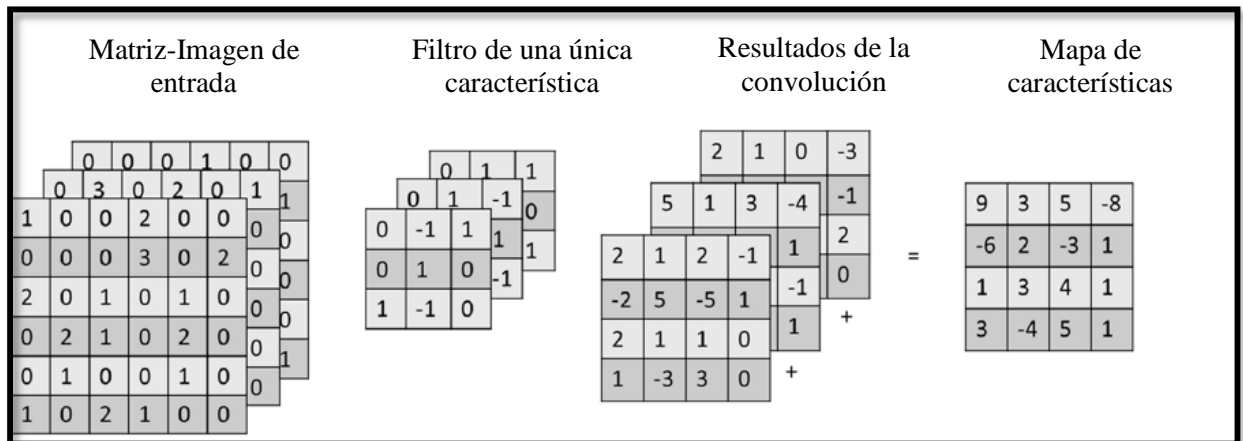


Figura n.º 9 Detalles de capa convolucional.

Fuente: Jones 2017

b) La capa de unidades lineales rectificadas (ReLU), realiza una función de filtro no lineal (como tanh y sigmoide, Figura n.º10) en la entrada. La introducción de la no linealidad en el sistema puede mejorar la eficiencia computacional sin perder mucha precisión. Cuando se realiza una operación de umbral:  $f(x) = \max(0, x)$ , donde  $f$  es una función no lineal aplicada a todos los valores  $x$  en las matrices, los valores negativos de las matrices de entrada se rectifican como 0 manteniendo el tamaño del volumen de entrada sin cambios.

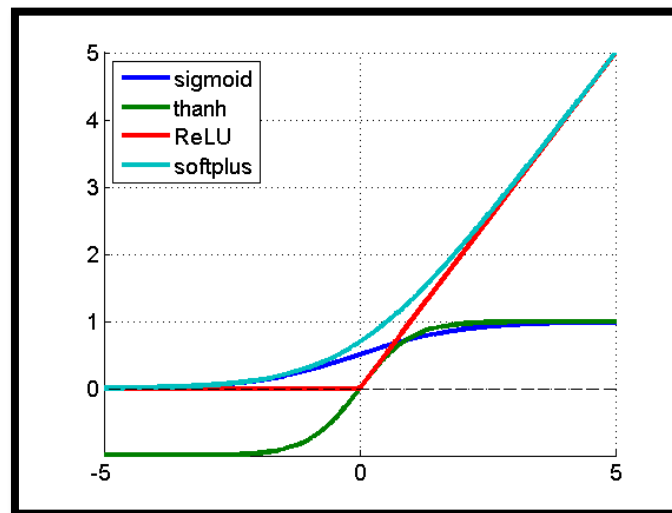


Figura n.º 10 Introducción de la no-linealidad al modelo

La función ReLU, ajusta diferentes funciones sin perder exactitud significativa y cambia valores negativos (Tanh) a 0, en caso se presenten.

Fuente: V. 2013



- c) La capa de agrupación (Max pooling) es una capa de muestreo descendente, cuyo objetivo es disminuir el tamaño de los mapas de características y reducir el costo computacional, en otras palabras, la necesidad de un hardware de mejor performance y costo. Una capa de agrupación máxima mantiene solo el número máximo de cada submatriz de pixels ( $2 \times 2$ ) como el elemento de la matriz de salida. La salida omite características sin importancia, mientras que mantiene la ubicación relativa de las características
- d) La capa totalmente conectada multiplica la entrada por una matriz de ponderación y luego agrega un vector de polarización a la salida, un vector N-dimensional (Figura n.º 11, d). La salida da la probabilidad de que la imagen de entrada pertenezca a cada clase, donde N es el número de la categoría en una tarea de clasificación.

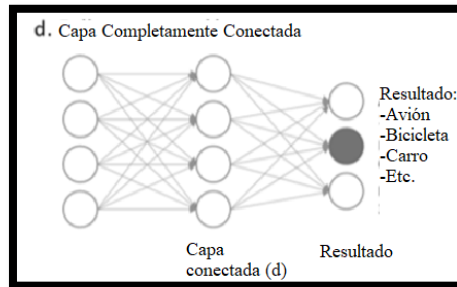


Figura n.º 11 Funciones de las capas CNN

Fuente: Tim Jones, 2017

e) Softmax

La función softmax, o función exponencial normalizada, es una generalización de la Función logística. Se emplea para "comprimir" un vector K-dimensional,  $\{z\}$ , de valores reales arbitrarios en un vector K-dimensional  $\{z\}$ , de valores reales en el rango  $[0, 1]$ . La función está dada por:

$$p(y = j|x) = \frac{e^{z_j}}{\sum_{j=1}^K e^{z_j}}$$

Donde:

$z_i$ : Son los inputs de la función softmax, y pueden tomar cualquier valor positivo, negativo y cero. Por ejemplo, una red neuronal puede tener un vector de salida  $(-0.15, 5.14, 6.10)$ , lo que no es un valor válido para una distribución probabilística, es por esto que es necesario la función softmax.

$E^{z_i}$ : La función exponencial estándar es aplicada para cada vector input. Esto devuelve un valor positivo por encima de 0, que será muy pequeño si es que el valor era negativo, y cercano a 1 si el valor era positivo y alto.

$\sum_{j=1}^K e^{z_j}$ : Es la normalización de los términos, asegurando que todos los outputs de la función (p) sumarán 1 y estarán en el rango [0,1], constituyendo una distribución probabilística válida.

K: se refiere al número de clases en un clasificador multiclase. (Wood, 2021)

Dicho de otra forma, la función logística genera un decimal entre 0.0 y 1.0. Por ejemplo, si un clasificador de correo electrónico tiene un resultado de regresión logística de 0.8, hay un 80% de probabilidad de que un correo electrónico sea spam y un 20 % de que no lo sea.

El softmax es el empleo específico de la fórmula para las clases múltiples.

## 2.5. Clasificador Bayesiano

Según Hernandez y Marín (2011), es un clasificador probabilístico basado en el teorema de Bayes. Este teorema considera que los atributos son independientes dada la hipótesis, por lo que la probabilidad de la hipótesis dada la evidencia puede estimarse como:

$$P(H|E_1, E_2, \dots, E_n) = \frac{P(H) \cdot P(E_1|H) \cdot P(E_2|H) \dots P(E_n|H)}{P(E)}$$

Donde:

$P(H|E_1, E_2, \dots, E_n)$  = Probabilidad (posterior) de que dado los atributos E, esta pertenezca a la clase H.

$P(E_1|H)$  = Probabilidad de que se dé E dado H. también se define como la esperanza.

$P(H)$  = Probabilidad a priori de la clase H

$P(E)$  = Probabilidad a priori de E, se obtiene de los datos iniciales.

Estos modelos son llamados algoritmos “Naive”, o inocentes en español. En ellos se asume que las variables (atributos) son independientes entre sí. En otras palabras, que la presencia de una cierta característica en un conjunto de datos no está en absoluto relacionada con la presencia de otra característica (Roman, 2019).

### 2.5.1. Ejemplo estadístico

Para ilustrar lo expresado en la sección anterior se planteará un ejemplo para dos clases, representado en un grupo de trabajo de oficina, considerando dos compañeros: A y B.

Sabiendo que:

- A se presenta 3 días por semana

- B se presenta 1 días por semana

Esta se considera la información “anterior”

En presencia nuestra, vemos pasar a alguien rápidamente, de manera que no tenemos la certeza si es A o B.

Dado la información “anterior” y asumiendo que sólo se trabajan 4 días a la semana, las probabilidades de que la personas vista sea A o B, son:

- $P_A = \frac{3}{4} = 0.75$
- $P_B = \frac{1}{4} = 0.25$

Cuando se ve a la persona pasar, vemos que llevaba una prenda roja. También se sabe que:

- A viste de rojo 2 veces por semana
- B viste de rojo 3 veces por semana

Por ende, para cada semana de trabajo, de cinco días, se calcula lo siguiente:

- Probabilidad que, A vista de rojo,  $P(\text{Rojo}|A) = \frac{2}{5} = 0.4$
- Probabilidad que, B visa de rojo,  $P(\text{Rojo}|B) = \frac{3}{5} = 0.6$

A partir de esta información se calculará, usando el teorema de Bayes, si se vio pasar a A o B. Esta nueva probabilidad se denomina “posterior”.

Inicialmente conocíamos las probabilidades  $P(A)$  y  $P(B)$ , y después inferíamos las probabilidades de  $P(\text{Rojo}|A)$  y  $P(\text{Rojo}|B)$ .

Se calculará la probabilidad de que A sea la persona vista en rojo.

$$P(A|\text{Rojo}) = \frac{P(\text{Rojo}|A) * P(A)}{P(R)} = \frac{0.4 * 0.75}{0.3 + 0.15} = 0.67$$

De igual manera se calcula la probabilidad para  $P(B|\text{Rojo})$ , el ejemplo, de manera resumida se muestra en la Figura n°12.

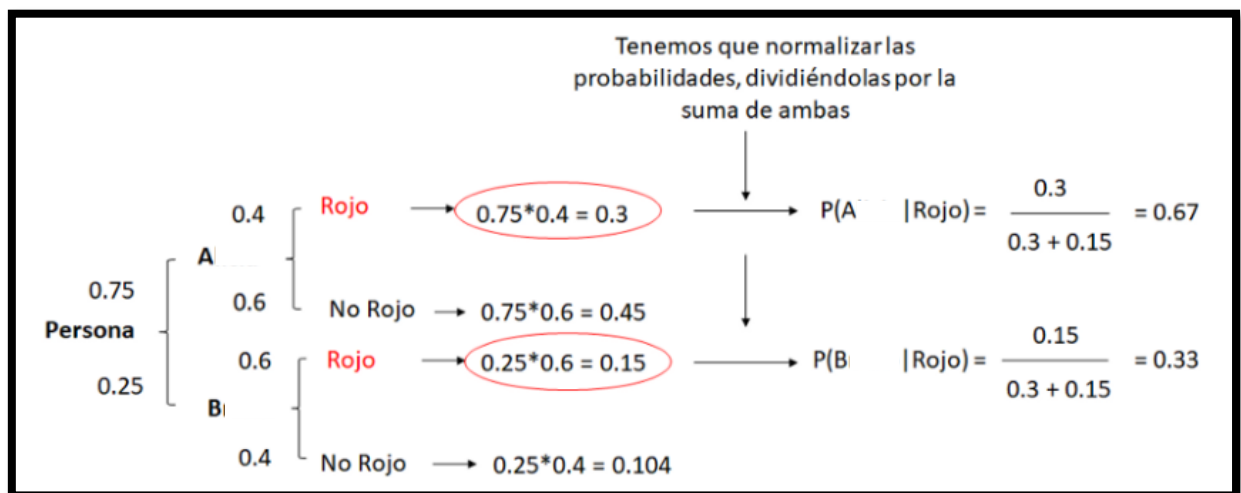


Figura n.º 12 Secuencia de cálculo utilizando el Algoritmo de Bayes

Fuente: Roman, 2019

## 2.5.2. Ventajas y desventajas del algoritmo Bayesiano

### a) Ventajas

- Un manera fácil y rápida de predecir clases, para problemas de clasificación binarios y multiclase.
- Para casos con pocos datos de entrenamiento, puede alcanzar mejores resultados que otros algoritmos de clasificación, debido a que tiene poca tendencia a sobre ajustar datos.
- La predicción para nueva data es rápida. Primero calculando la probabilidad posterior para cada clase. Luego aplicando MAP (Estimación Máxima Posterior, por sus siglas en inglés), (Gahukar, 2018).
- Debido al ajuste lineal, la velocidad de ejecución y el bajo requerimiento computacional (tanto de CPU como de GPU), el algoritmo es una solución viable para casos con gran cantidad de información, que serían excesivamente demandantes para otros algoritmos de clasificación.
- El desacoplamiento de las distribuciones de características condicionales de clase significa que cada distribución puede ser estimada independientemente como si tuviera una sola dimensión. Esto ayuda con problemas derivados de la dimensionalidad y mejora el rendimiento (Sunil, 2017).

## b) Desventajas

- Aunque son unos clasificadores bastante buenos, “los algoritmos Naive Bayes son conocidos por ser pobres estimadores. Por ello, no se deben tomar muy en serio las probabilidades que se obtienen” (Gahukar, 2018). Catanzarite, 2018, nos indica que esto se debe a que se asume que los atributos son independientes entre sí, cuando en la realidad, no se presentan de esta forma; se explica que utilizando la Estimación Máxima Posterior (MAP), así como una muestra para el entrenamiento del modelo que sea representativa de la población general, el modelo podrá corregir este defecto.
- La presunción de independencia Naive muy probablemente no reflejará cómo son los datos en el mundo real.
- Cuando el conjunto de datos de prueba tiene una característica que no ha sido observada en el conjunto de entrenamiento, el modelo le asignará una probabilidad de cero y será inútil realizar predicciones. Uno de los principales métodos para evitar esto, es la técnica de suavizado, siendo la estimación de Laplace una de las más populares (Sunil, 2017).

## 2.6. Matriz de confusión

Teniendo en cuenta que tan importante como la data y el modelo, debe ser una correcta forma de evaluar los resultados obtenidos, se seleccionaron dos métricas ampliamente utilizadas para este fin, explicadas en los puntos 2.7 y 2.8: La matriz de confusión y el coeficiente de Kappa

La matriz de confusión es un resumen de la precisión de los resultados de predicción sobre un problema de clasificación.

El número de predicciones correctas e incorrectas se resume con los valores de conteo y se desglosa por cada clase. Esta matriz nos da una idea no solo de los errores que comete un clasificador, sino de los tipos de errores que se cometen.

Tabla 2: Matriz de confusión para clasificación de dos variables

		Clase 1	Clase 2
		Predicho	Predicho
Clase 1	Actual	TP	FN
Clase 2	Actual	FP	TN

Aquí se tiene:

Clase 1: Positivo (Ejem. Una manzana)

Clase 2: Negativo (Ejem. No es una manzana)

TP: La observación es positiva, y se predice que será positiva.

FN: La observación es positiva, pero se predice negativa.

TN: La observación es negativa, y se predice que será negativa

FP: La observación es negativa, pero se predice positiva.

a) Exactitud

$$\text{Exactitud} = \frac{TP + TN}{TP + TN + FP + FN}$$

Sin embargo, hay problemas con la precisión. Asume costos iguales para ambos tipos de errores. Una precisión del 99% puede ser excelente, buena, mediocre, mala o terrible según el problema.

b) Recuperación

La recuperación puede definirse como la relación entre el número total de ejemplos positivos clasificados correctamente y la división con el número total de ejemplos positivos. La recuperación alta indica que la clase se reconoce correctamente (número pequeño de FN).

$$\text{Recuperación} = \frac{TP}{TP + FN}$$

c) Precisión

Para obtener el valor de la precisión, dividimos el número total de ejemplos positivos clasificados correctamente por el número total de ejemplos positivos previstos. Alta precisión indica que un ejemplo etiquetado como positivo es efectivamente positivo (pequeño número de FP).

$$\text{Precisión} = \frac{TP}{TP + FP}$$

Alta recuperación, baja precisión: Esto significa que la mayoría de los ejemplos positivos se reconocen correctamente (FN bajo) pero hay muchos falsos positivos.

Recuerdo bajo, alta precisión: Esto muestra que faltamos muchos ejemplos positivos (FN alto), pero aquellos que pronosticamos como positivos son ciertamente positivos (FP bajo).

## 2.7. Coeficiente Kappa

El Coeficiente Kappa de Cohen “mide el grado de concordancia de las evaluaciones nominales u ordinales realizadas por múltiples evaluadores cuando se evalúan las mismas muestras” (Minitab, 2019).

Dicho de una manera más sencilla, es una medida cuantitativa de la confiabilidad que tienen dos estimadores que evalúan la misma data, corrigiendo la probabilidad que los estimadores coincidan por azar.

El valor del coeficiente Kappa puede variar de -1 a 1. Un valor de 0 indica que hay completo azar en la concordancia entre ambos estimadores, mientras que un valor de 1, indica que hay una concordancia completa entre ambos estimadores, sin embargo, si el valor se reduce por debajo de 0, indica que la concordancia entre ambos estimadores es menor que el puro azar.

Tabla 3: Matriz ejemplo para cálculo de Coeficiente de Kappa

		Estimador 2	
		Correcto	Incorrecto
Estimador 1	Correcto	A	B
	Incorrecto	C	D

Donde:

A: El total de data que ambos estimadores clasificaron como “Correcto”. Los estimadores concuerdan.

B: El total de data que el estimador 1 clasifico como “Incorrecto”, pero el estimador 2 clasifico como “Correcto”. Esto es un desacuerdo.

C: El total de data que el estimador 2 clasifica como “Incorrecto”, pero el estimador 1 clasifica como “Correcto”. También es un desacuerdo.

D: El total de dada que ambos estimadores clasifican como “Incorrecto”. Los estimadores concuerdan. (Pikes, 2020)

Para trabajar con el Coeficiente de Kappa, primero se debe calcular la probabilidad de concordancia (diagonal de la Tabla n°03).

$$P_0 = \frac{\text{Totl de concordancias}}{\text{Total de data}}$$

El siguiente paso será la probabilidad de que concuerden por puro azar, utilizando la Tabla n°03, el valor se calcula como el total de veces que el estimador 1 clasifica como “Correcto” dividido entre el total de data, multiplicado por, el total de veces que el estimador 2 clasifico

como “Correcto” dividido entre el total de data, a esta multiplicación se agrega el total de veces que el estimador 1 clasificó como “Incorrecto” multiplicado por el total de veces que el estimador 2 clasificó como “Incorrecto”. Debido a que esta información es difícil de comprender sólo en texto se ha agregado las dos ecuaciones a continuación para un mayor entendimiento.

$$P_{(Correcto)} = \left( \frac{A + B}{A + B + C + D} \right) * \left( \frac{A + C}{A + B + C + D} \right)$$

$$P_{(Incorrecto)} = \left( \frac{C + D}{A + B + C + D} \right) * \left( \frac{B + D}{A + B + C + D} \right)$$

Ambas ecuaciones se consolidan en:

$$P_e = P_{Correcto} + P_{Incorrecto}$$

Finalmente, la fórmula de Kappa de Cohen es la probabilidad de concordancia, quitando la probabilidad de concordancia por puro azar, dividido entre 1 menos la probabilidad de concordancia por puro azar, esto se resume en la siguiente fórmula:

$$K = \frac{P_0 - P_e}{1 - P_e}$$

En la Figura n°13 se muestran la correspondencia entre los valores obtenidos y la calidad de concordancia, estudiado por Landis y Koch, 1977; se muestran las expresiones originales entre paréntesis.

<b>Coefficiente kappa</b>	<b>Fuerza de la concordancia</b>
0,00	Pobre ( <i>Poor</i> )
0,01 - 0,20	Leve ( <i>Slight</i> )
0,21 - 0,40	Aceptable ( <i>Fair</i> )
0,41 - 0,60	Moderada ( <i>Moderate</i> )
0,61 - 0,80	Considerable ( <i>Substantial</i> )
0,81 - 1,00	Casi perfecta ( <i>Almost perfect</i> )

Figura n.º 13 Valoración del coeficiente de Kappa

Fuente: Landis & Koch, 1977



## **2.8. Ejemplo Aprendizaje supervisado**

Como ejemplo de aprendizaje supervisado utilizaremos el caso estudiado por Cao, Dragičević & Li, 2019. El objetivo principal es evaluar los modelos de clasificación CNN y detectar cambios en el uso de la tierra a lo largo del tiempo.

En este trabajo, el área de estudio es una sección de la comunidad de Cloverdale, en la parte noreste de la Ciudad de Surrey, Canadá (Figura n.º14). La comunidad de Cloverdale se ha transformado a lo largo de los años de una comunidad agrícola rural a una pequeña ciudad próspera, atractiva para familias jóvenes, y es conocida por sus áreas residenciales de rápido desarrollo; por lo tanto, se eligió como el área adecuada para detectar el cambio de uso de suelo a través de la utilización conjunta de imágenes RGB y métodos de aprendizaje profundo. La base de datos de la ciudad de Surrey proporciona ortofotos desde 2004 a 2017 (resolución de 10 cm), desde la cual se implementó la clasificación de uso de suelo. Debido a que cada ortofoto era grande, demoró 4 horas en clasificarse, solo las imágenes seleccionadas de buena calidad se procesaron por razones de simplicidad. En particular, las ortofotos para los años 2004, 2006, 2011, 2013, 2015 y 2017 fueron consideradas para documentar el cambio de uso de suelo en los últimos 12 años. Las ortofotos digitales cubren una sección de la comunidad de Cloverdale con un tamaño de 2.0 km × 3.6 km con 10 cm de resolución espacial. Los datos de uso del suelo de 2011 del Metro Vancouver Open Data Catalog y de DMTI Spatial Inc. también se usaron como referencias.

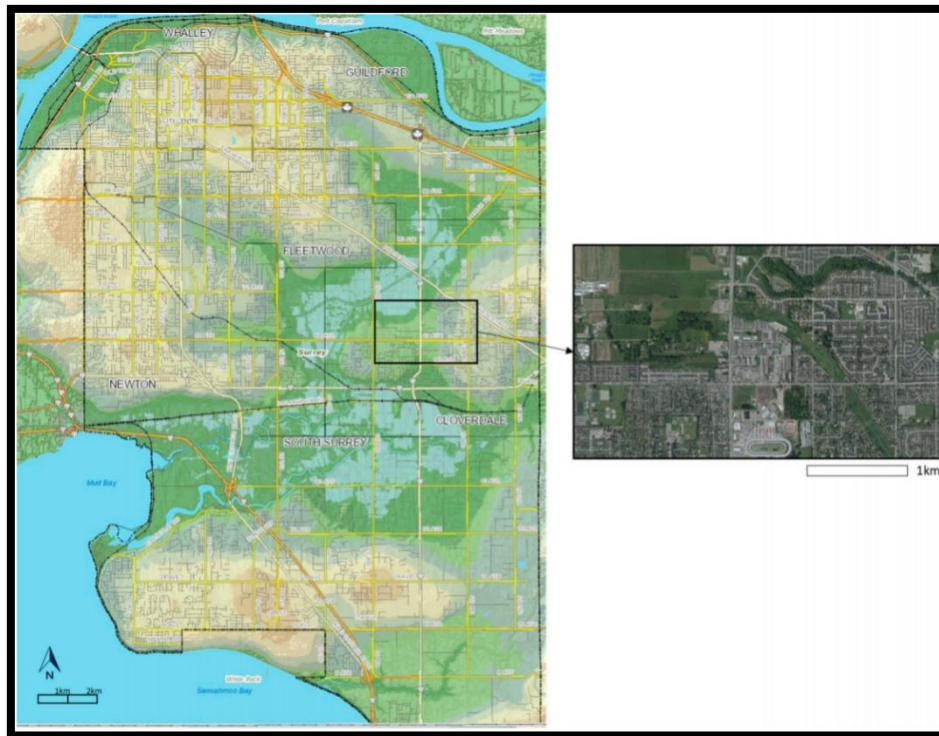


Figura n.º 14 Área de estudio: sección noreste de Cloverdale

Comunidad de la Ciudad de Surrey, Canadá

Fuente: "Maps & COSMOS", 2020

Para este caso se utilizaron diferentes modelos CNN cuyos valores de precisión se obtuvieron a continuación:

Tabla 4: Valores obtenidos para la precisión total de los ocho modelos basados en CNN transferidos. Aquí los conjuntos de datos de entrenamiento y prueba se mantuvieron con valores muy cercanos para cada método.

Models	Total Accuracy
a. AlexNet, full-trained	95.80%
b. GoogLeNet, full-trained	95.06%
c. AlexNet-'fc7'-SVM	95.68%
d. AlexNet-'fc6'-SVM	97.11%
e. GoogLeNet-'loss classifier'-SVM	97.47%
f. GoogLeNet-'pool5-drop'-SVM	98.14%
g. VGGNet-'fc6'-SVM	95.93%
h. VGGNet-'fc7'-SVM	96.99%

Fuente: Cao, Dragičević & Li, 2019

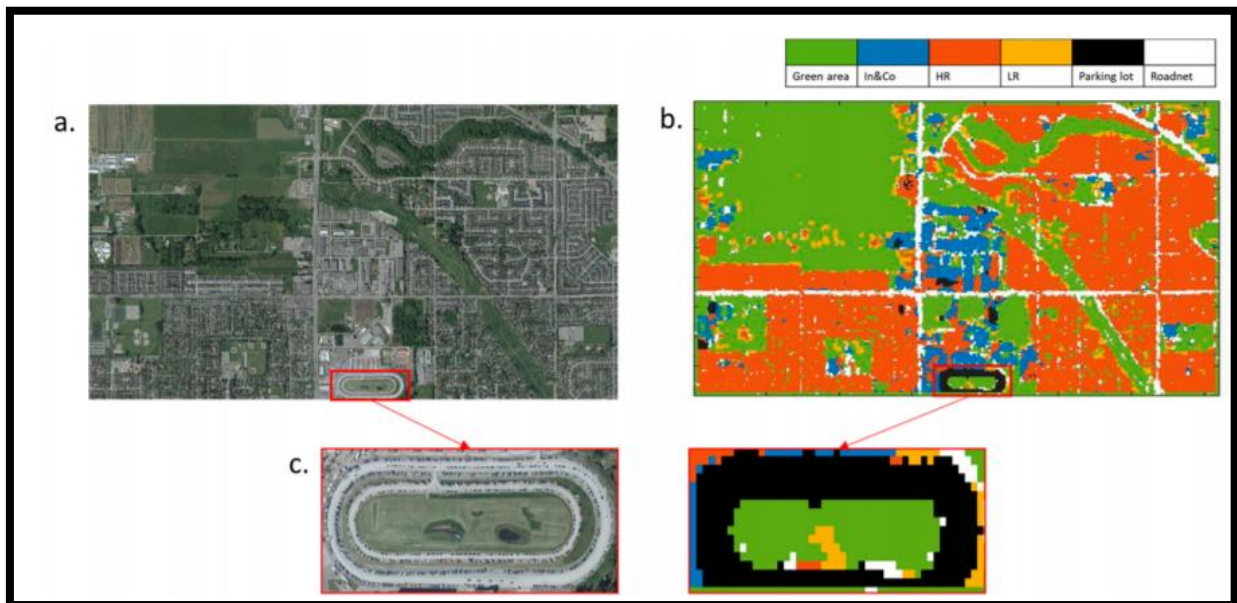


Figura n.º 15 Codificación de colores para caso estudiado

(a) La imagen de ortofoto del área de estudio con (b) el mapa digital de uso del suelo basado en los resultados de clasificación para el año 2017. (1) Área verde (GA), color verde; (2) áreas industriales y comerciales (In & Co), color azul; (3) áreas residenciales de alta densidad (HR), color rojo; (4) áreas residenciales de baja densidad (LR), color amarillo; (5) estacionamiento (PL), color negro; (6) Vías (RN), color blanco. (c) Imagen detallada de la zona con la pista de campo llena de autos.

Fuente: Cao, Dragičević & Li, 2019

### III. METODOLOGÍA

En el desarrollo del tercer componente de la presente investigación considera tres puntos fundamentales, la obtención de las imágenes que conformarán la base de datos; la programación de los dos algoritmos mencionados: CNN y Clasificador Bayesiano; y finalmente, las métricas de validación de los algoritmos. Para la primera parte se utilizarán los softwares de Google Earth Pro y ArcGis 10.3, herramienta de sistema de información geográfico, para los dos últimos casos se emplea el uso del lenguaje de programación Python. El lenguaje Python se está estableciendo como uno de los lenguajes más populares para la computación científica. Gracias a su naturaleza interactiva de alto nivel y su ecosistema de bibliotecas científicas en proceso de maduración, es una opción atractiva para el desarrollo algorítmico y el análisis de datos exploratorios (Dubois, Oliphant, Pérez & Granger, 2007).

El flujograma propuesto para el desarrollo de la presente investigación se muestra la siguiente figura (n.º 16):

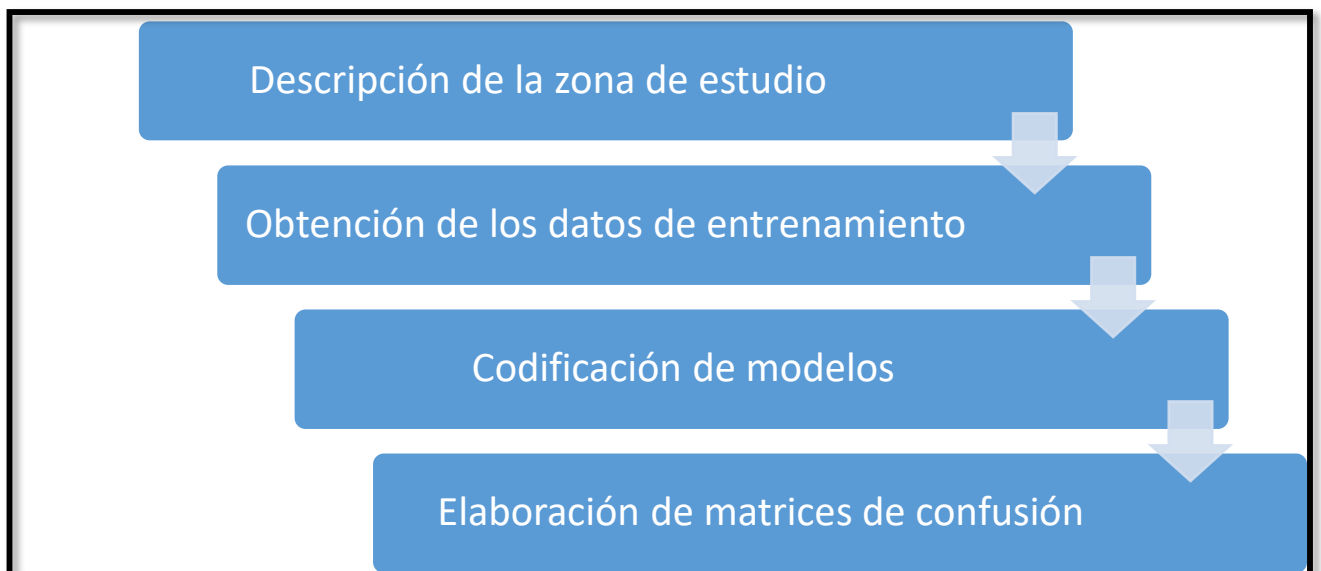


Figura n.º 16 Flujograma de la metodología

Las imágenes satelitales de alta resolución en el espectro visual (RGB) tienen la capacidad de proveer los datos para categorizar los estados constructivos de edificaciones del distrito de Carabaylo, Lima. La fuente considerada de obtención de imágenes es la plataforma Google Earth Pro, la cual tiene la ventaja de proveer imágenes de alta resolución de libre disponibilidad.

### **3.1. Descripción de la zona de estudio**

La zona de estudio, corresponde al distrito de Carabaylo ubicados en la provincia de Lima, región Lima. Este es uno de los 43 distritos de la provincia de Lima y uno de los ocho que conforman Lima Norte, colinda de la siguiente manera:

- al norte, Santa Rosa de Quives, provincia de Canta
- al sur, con el distrito de Comas
- al este, con el distrito de San Juan de Lurigancho
- y por el oeste, con los distritos de Puente Piedra y Ancón

“Actualmente el distrito posee una extensión de 416.00 km<sup>2</sup>” (Distrito.pe, 2020). A lo largo de su historia ha pasado por diferentes modelos de tenencia de tierras desde el modelo de hacienda hasta el modelo inmobiliario, pasando por los modelos cooperativo y parcelario. En la actualidad se encuentra en un acelerado proceso de urbanización, lo que ha significado una pérdida sustancial de tierras agrícolas periurbanas, con repercusiones en el estilo de vida de sus habitantes (Chavarria, 2017), dentro del proceso de urbanización, existe un porcentaje el cual decide hacerlo por la vía formal accediendo a licencias de edificación para viviendas unifamiliares con un total de 1120 para el año 2013 en el distrito estudiado, asimismo, para cerca de 2 millones y medio de viviendas existentes en Lima Metropolitana el 76,5% representan casas independientes; un 84.2% de material noble, 7.3% de madera y 5.0% de adobe (INEI, 2014).

La superficie del distrito se ubica en un rango de altitudes entre 140 a 500 metros sobre el nivel del mar. Se procederá a coleccionar imágenes de este lugar teniendo en cuenta su desarrollo urbano a través del tiempo.

### 3.2. Datos de entrenamiento

#### 3.2.1. Selección de Manzanas de muestra

Para esta sección, la metodología a utilizar será la metodología propuesta por Peña & Murakami, 2017, donde las condiciones para la selección de manzanas se basarán inicialmente en tres características:

- Localización de una franja urbana, que haya sido terreno agrícola en un pasado cercano
- Disponibilidad de una imagen satelital de alta resolución
- Presencia de estados tempranos de residencia para el año 2014.

Las manzanas representan la mayor unidad de medida usada en este estudio, seguido inmediatamente por los lotes individuales, cada uno representa un predio diferente el cuál se clasificará posteriormente. Para el presente estudio se seleccionaron un total de 10 manzanas con la ayuda del software Google Earth Pro y ArcMp 10.3.

#### 3.2.2. Obtención de imágenes satelitales

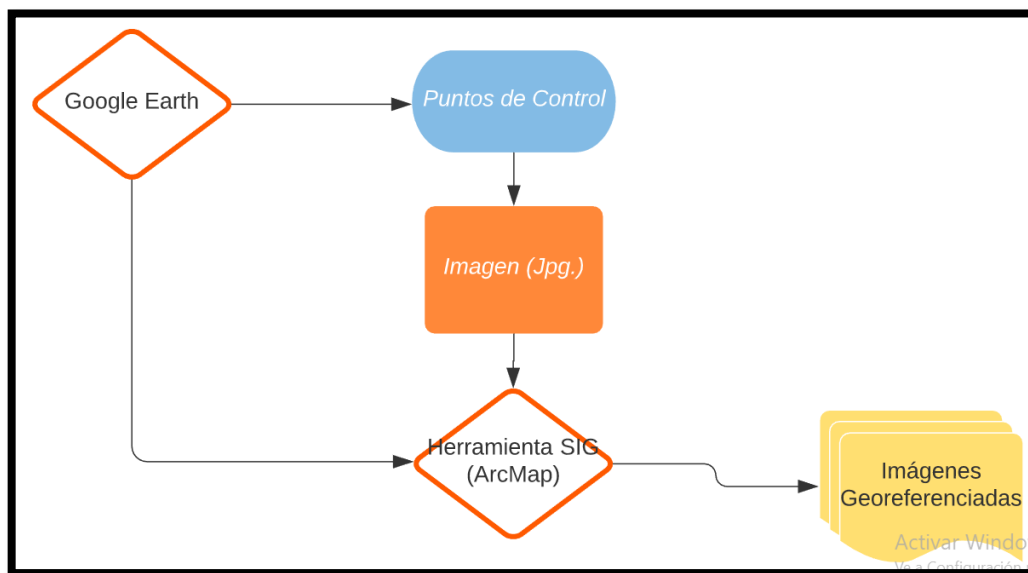


Figura n.º 17 Procedimiento para obtención de Imágenes Georeferenciadas

##### a) Puntos de control

Utilizando el software Google Earth Pro, se denotarán los vértices de las manzanas a seleccionar, cuatro por cada manzana haciendo un total de 40 vértices, que fueron identificados usando la letra M y los números correlativos del 1 al 40. Cada diferente grupo de vértices se guardará como extensión Kmz.

##### b) Imagen (Jpg.)

Teniendo las manzanas delimitadas por sus vértices y las imágenes satelitales identificadas, se procederá a guardar las imágenes en formato Jpg. según las diferentes manzanas. Altura

de ojo referencial en el rango de 300 a 560 metros, el rango superior considera situaciones donde se pueda llegar a incluir dos (02) manzanas dentro de la misma imagen.

En cuanto a la calidad de imágenes, se buscó la mayor resolución espacial disponible en esta plataforma para el año en estudio, 2014, es así que se utilizó las imágenes provistas por el satélite DigitalGlobe Quickbird, “de resolución 65 cm. Para imágenes pancromáticas y 2.62 m. para imágenes RGB” (Chethan, 2011), la ventaja de usar este satélite es la disponibilidad de imágenes PanSharpened, generadas a partir de la fusión de sus productos pancromáticos y multiespectrales, obteniendo la calidad espectral del producto multiespectral y la alta precisión del producto pancromático.

### c) Herramienta SIG ArcMap

Una vez obtenidas las imágenes satelitales de las manzanas, se procederá a utilizar el software ArcMap 10.6 para el producto final de esta sección.

Comenzando por la importación y conversión de los diferentes grupos de vértices según manzanas, de formato Kmz. a formato shape, asimismo, se importará la imagen seleccionada de la manzana 01, seguidamente, se activará y utilizará la herramienta “Georeferencing”, disponible en la barra de herramientas del software, tomando como puntos de control los importados anteriormente.

Se procede de igual manera para las manzanas de la 02 a la 10, con lo que se obtiene el producto final de esta sección, imágenes de alta resolución georreferenciadas en WGS84.

### 3.2.3. Clasificación manual de los estados constructivos

Para esta sección se continuará en el mismo software. La cantidad de lotes dentro de cada manzana varía en cantidad y en estado constructivo, la clasificación de estos estados constructivos se da de manera clara gracias a la tipología presentada por Tokeshi et al. (2005), quien estudio el proceso de auto construcción en lima.

Tabla 5: Categorías de consolidación

Clase	Nombre	Descripción
A	Precario	Suelo llano o refugio provisional
B	Incipiente	Estructura básica con techo de calamina
C	Primer piso consolidado	Primer piso terminado, techo aligerado
D	Segundo piso consolidado o más	Dos o más pisos terminados, techo aligerado

Fuente: Tokeshi et al. (2005)

Seguendo el proceso propuesto por Peña & Murakami, 2017, la clasificación se da mediante un protocolo de interpretación visual, el cual ha sido diseñado para permitir una correcta identificación del estado de consolidación de los lotes (Figura n.º 18).

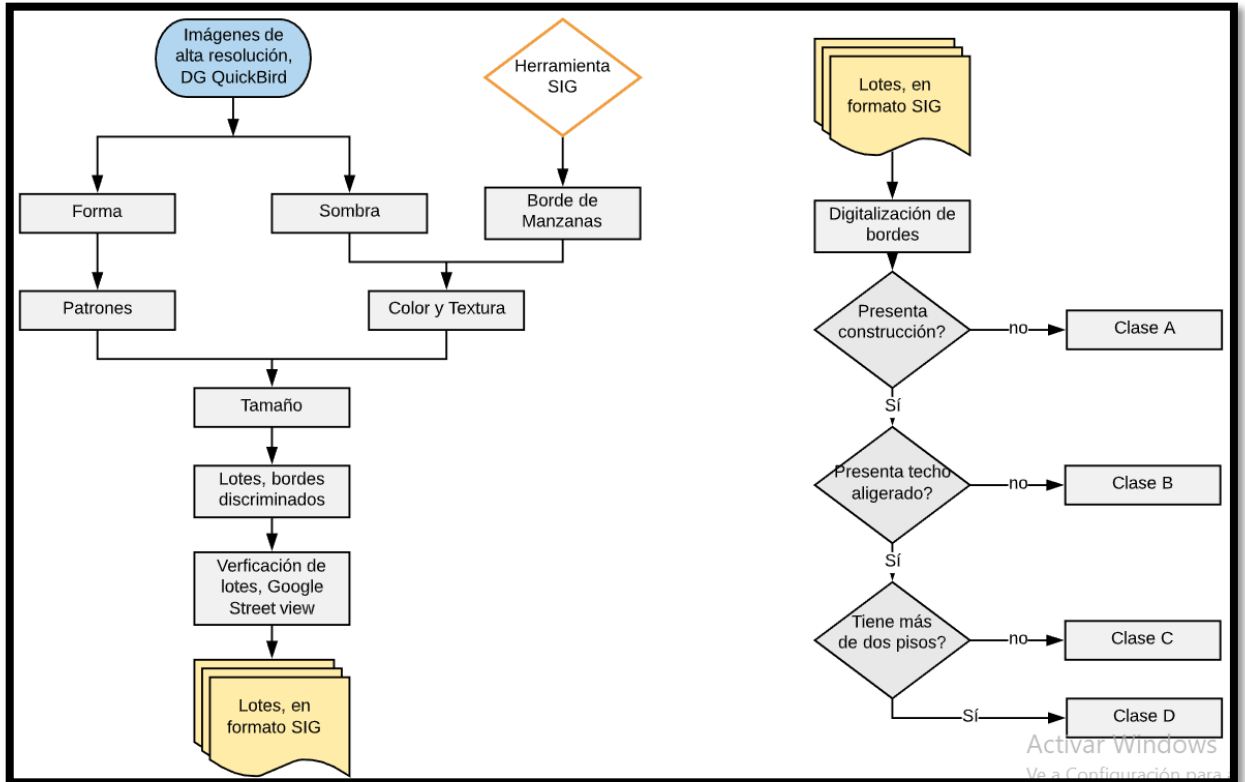


Figura n.º 18 Proceso para identificación de estados de consolidación

Fuente: Peña & Murakami, 2017

A partir del proceso de digitalización de bordes, se obtiene una base de datos espacial cuya tabla de atributos denota la manzana, el número de lote correspondiente y el estado constructivo al cual pertenecen.

Finalmente se procede a cortar los lotes, mediante la herramienta “Extracción por máscara”, para obtener una imagen independiente de cada predio, los cuáles serán almacenados en carpetas diferentes, según sus estados constructivos, catalogados comenzando con la letra correspondiente a su estado y números correlativos del 1 en adelante; se especificará las dimensiones de salida en 64 píxeles de altura por 64 píxeles de ancho.



### **3.2.4. Depuración hasta conseguir lotes representativos**

En esta sección se propone la metodología para, una vez obtenidos las imágenes de lotes independientes según clasificación, mediante un proceso de interpretación visual, propuesto por la Universidad de León, 2006, donde se plantea tres (03) fases:

#### a) Detección, reconocimiento e identificación

Se identificará claramente el lote según su clasificación, como segundo paso se reconoce el objeto familiar sobre la base de su forma, tamaño y otras propiedades visibles.

#### b) Análisis

En esta sección se analizará la imagen determinada para reconocer los siguientes factores propuestos:

- Alteración de la calidad de la imagen
- Tamaños de lotes extremadamente delgados
- Texturas indefinidas

#### c) Clasificación

Finalmente, en esta sección se clasificará la imagen como representativa del estado constructivo o no, siendo el objetivo la desestimación de muestras no representativas para la base de datos de ambos modelos.

### **3.3. Codificación de modelos**

En esta sección se codificarán los modelos de tipo CNN y Bayesiano, comenzando desde la importación de las librerías necesarias, pasando por cada una de las etapas que la estructura de cada modelo requiere.

Seguidamente se procederá al aumento de la base de datos con las imágenes existentes, mediante algoritmos que rotan la orientación de las imágenes, o las varían de diferentes formas, de manera que se crea una base más amplia a partir de las imágenes seccionadas en la etapa previa.

Tabla 6: Librerías utilizadas según modelos

		Modelo	
n°	CNN	n°	Bayesiano
1	Numpy	1	Numpy.
2	Matplotlib-pyplot	2	Glob.
3	Itertools	3	Matplotlib-pyplot.
4	Keras	4	Itertools.
5	Keras.models-Sequential	5	PIL-Image.
6	Keras.layers-Activation	6	Random.
7	Keras.layers.core	7	Sklearn.naive_bayes.
7.1	Conv2D	8	Sklearn.model_selection.
7.2	MAxpool2D	9	Sklearn.decomposition.
7.3	Dropout	10	Sklearn.metrics.
7.4	Flatten	11	sklearn.metrics-cohen_kappa_score
7.5	Dense	12	Sklearn.metrics-Confusion_matrix
8	Keras.layers.normalization-BatchNormalization	13	Keras.preprocessing.image
9	Keras.metrics-Categorical_crossentropy		
10	Sklearn.metrics-Confusion_matrix		
11	sklearn.metrics-cohen_kappa_score		

### 3.3.1. Clasificador CNN

- Numpy

Es la biblioteca central para fines científicos en Python. Provee arreglos de objetos (matrices) multidimensionales de alta performance y herramientas para trabajar con estos arreglos. Un arreglo Numpy es una grilla de valores, todos del mismo tipo.

- Matplotlib-pyplot

Esencialmente provee una interfaz similar a la de MATLAB para graficar información.

- Itertools

Es una biblioteca estándar de Python que incorpora funciones que devuelven objetos iterables, es decir, estructuras de datos basadas en elementos que pueden recorrerse secuencialmente y que pueden utilizarse en procesos repetitivos.

- Keras

Es una Interfaz Específicas (APIs en adelante) diseñada para el aprendizaje profundo (Deep Learning). Keras utiliza las mejores prácticas para reducir la carga cognitiva: ofrece APIs consistente y fáciles de entender, minimiza el número de acciones que realiza el usuario para los casos más comunes de aplicación, de igual manera, provee mensajes de error claros y procesables. Posee vasta documentación y guías de usuarios.

- Keras.models-Sequential

Esta función es la encargada de agrupar diferentes capas en una secuencia lineal. Marca el inicio de la utilización de capas, para este caso se trabajará con las capas de Activation, Dense, Flatten y BatchNormalization.

- `Keras.layers-Activation`

Esta biblioteca según la función escogida (ReLU, Softmax, etc.), tiene la función de determinar si el output de la neurona anterior, es relevante para el modelo o no; también ayuda a normalizar el input que recibe, modificando el rango de valores entre 0 y 1 o entre -1 y 1 dependiendo de la función seleccionada. Se adjunta específicamente en la capa Conv2D.

- `Keras.layers.core`

En esta sección se presentan las capas a utilizar en el modelo de manera secuencial

a) Conv2D

Es una capa de convolución 2D (la primera del modelo), una matriz o máscara que puede ser utilizada para difuminar, afinar, detectar bordes de las imágenes de entrada y así obtener un producto de salida.

b) Maxpool2D

Función que permite el proceso de discretización en base a inputs, dicho de otra forma, reducir las dimensiones de las matrices de entrada.

c) Dropout

Esta función permite que algunas de las neuronas presentes en la red, sean temporalmente desactivadas o ignoradas. Se aplica en la fase de entrenamiento para reducir los efectos del sobre ajuste.

d) Flatten

Permite linealizar las matrices, los formatos de imágenes dados una altura y ancho de 2x2x3 bandas (RGB), al utilizar esta función tendrá un producto final de forma 12x1.

e) Dense

En las hidden, son la cantidad de neuronas que tiene una capa en específico

Permite la correcta unión de las neuronas en una secuencia ordenada y lineal.

- `Keras.layers.normalization-BatchNormalization`

Normaliza los datos de la capa previa; aplica una transformación que mantiene la media cercana a cero y la desviación estándar cerca de 1.

- `Keras.optimizers-Adam`

Es un tipo de optimizador disponible en las bibliotecas de Keras, basado en el teorema de Adams. Es uno de los dos requerimientos para compilar un modelo en keras.

- Keras.metrics- Categorical\_crossentropy

Es el segundo requerimiento para compilar modelos en keras. Se utiliza para problemas que posean dos o más clases. Es la forma de calificar los errores que se han obtenido una vez compilado el modelo, de manera que usando el optimizador seleccionado (Adam), se puedan mejorar los parámetros de la red neuronal.

- Sklearn.metrics-Confusion\_matrix

Es la biblioteca que permite la implementación de la matriz de confusión en el código.

### **3.3.2. Clasificador Bayesiano**

- PIL-Image

Provee a la interface con la cual se esté trabajando el lenguaje Python la capacidad de editar imágenes. La biblioteca también provee ciertas funciones de fábrica como, la posibilidad de cargar imágenes y crear nuevas imágenes.

- Numpy

Es la biblioteca central para fines científicos en Python. Provee arreglos de objetos (matrices) multidimensionales de alta performance y herramientas para trabajar con estos arreglos. Un arreglo Numpy es una grilla de valores, todos del mismo tipo.

- Glob

Utiliza APIs para buscar nombres de archivos. Es útil en cualquier situación donde un programa o código necesite buscar una lista de archivos en el sistema de archivos con nombres que coinciden con un patrón.

- Matplotlib-pyplot

Esencialmente provee una interfaz similar a la de MATLAB para graficar información.

- Itertools

Es una biblioteca estándar de Python que incorpora funciones que devuelven objetos iterables, es decir, estructuras de datos basadas en elementos que pueden recorrerse secuencialmente y que pueden utilizarse en procesos repetitivos

- `Sklearn.naive_bayes`

Esta biblioteca implementa los algoritmos Bayesianos ingenuos. Estos son métodos de aprendizaje supervisado, basados en el teorema de Bayes, donde se asume la independencia de características.

- `Sklearn.decomposition`

Esta biblioteca incluye algoritmos para la descomposición de matrices, entre otros, incluye ACP (Análisis de Componentes Principales), FMN (Factorización de Matrices no Negativas) o ACI (Análisis de Componentes Independientes). La mayoría de los algoritmos de esta biblioteca se pueden considerar como técnicas de reducción dimensional.

- `Sklearn.metrics`

Existen 3 diferentes APIs para evaluar la calidad de las predicciones de un modelo:

- a) Puntaje del método estimador: los estimadores tienen un método de *puntuación* dando un criterio predeterminado de evaluación para el problema que están diseñados a resolver. Esto no se discute aquí, sino en la documentación de cada estimador.

Incluye funciones de puntuación, métricas de performance y métricas de comparación por parejas.

- b) Parámetros de puntuación: Herramientas para evaluación de modelos utilizando la validación cruzada (Evita el sobre ajuste, separa de manera aleatoria la data disponible)
- c) Funciones de métricas: el módulo de métricas implementa funciones de calificación de errores para propósitos específicos. Estas métricas están detalladas en documentación propia del `scikit.learn`.

- `Sklearn.metrics-Confusion_matrix`

Es la biblioteca que permite la implementación de la matriz de confusión en el código.

### **3.4. Elaboración de matrices de confusión**

Es la etapa final del proyecto donde se codificará las matrices de confusión para ambos modelos elaborados, determinando la precisión de cada uno, este paso permitirá discutir las ventajas y desventajas en el empleo de cada uno de los algoritmos utilizados.

Dado que `Sckit.learn` es una interfaz específica para los procesos de aprendizaje automático, las funciones se encuentran en su gran mayoría ya desarrolladas, este es el caso de la función `Sk.learn.metrics-Confusión_Matrix`.

En una matriz cuadrada, de “a” por “a” entradas (Véase Figura n.º19), los elementos diagonales representan el número de puntos cuya etiqueta predicha es igual a la etiqueta real, mientras que, los elementos fuera de la diagonal representan los que han sido predichos

erróneamente. Mientras los valores de la diagonal sean más altos, la matriz de confusión representa un mejor performance del modelo (Buitinck et al., 2013). En este caso se trabaja una matriz multi clase.

En esta sección se tomarán los productos de los modelos codificados anteriormente, para trabajar en la obtención de los parámetros que esta función nos solicita:

a) Y\_verdadero

Corresponde a las categorías verdaderas de las muestras que fueron puestas a prueba en el modelo.

b) Y\_predicho

Corresponde a las categorías que fueron predichas por el modelo, de las muestras que fueron puestas a prueba.

c) Ponderación de muestras

Este valor corresponde al peso que fue asignado a cada neurona para la etapa de entrenamiento. Este parámetro usualmente no es utilizado, se desactiva en el momento de graficar la matriz, debido a la gran cantidad de neuronas que el modelo utiliza.

d) Categorías o clases

Dado que los dos primeros son arreglos que sirven esencialmente para la comparación y producción de los valores internos de la matriz, se deberá proveer a la matriz de confusión de las categorías las cuales está poniendo en la representación visual (Clase “A”, Clase “B”, Clase “C”, etc.)

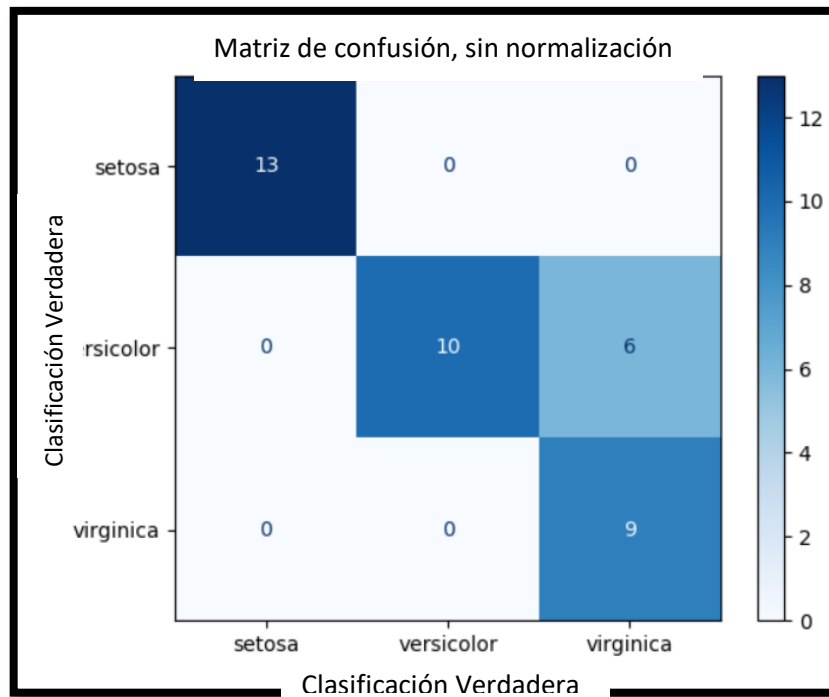


Figura n.º 19 Representación de matriz de confusión para problema de clasificación de tres clases

Fuente: Buitinck et al., 2013

### 3.5. Ingreso del coeficiente de Kappa

Con el fin de tener una segunda métrica para evaluar la precisión de nuestros modelos, se utilizará el coeficiente de Kappa, agregando otra medida para evaluar ventajas y desventajas de los modelos utilizados.

Para este caso se utilizará la librería propia de `sk.learn`, utilizado por poseer códigos simples y eficientes para la predicción y el análisis de data, a este librería se le pedirá importar el coeficiente Kappa de Cohen mediante el comando:

```
>>from sklearn.metrics import cohen_kappa_score
```

## IV. RESULTADOS Y DISCUSIÓN

### 4.1. Imágenes de entrenamiento

#### 4.1.1. Obtención de imágenes satelitales

La zona de estudio fue el distrito de Carabayllo, provincia de Lima, región Lima. Se trabajó un total de 10 manzanas seleccionadas en la zona sur-oeste del distrito con un área global de 15.92 km<sup>2</sup> (Área distrital, Figura n.º20)

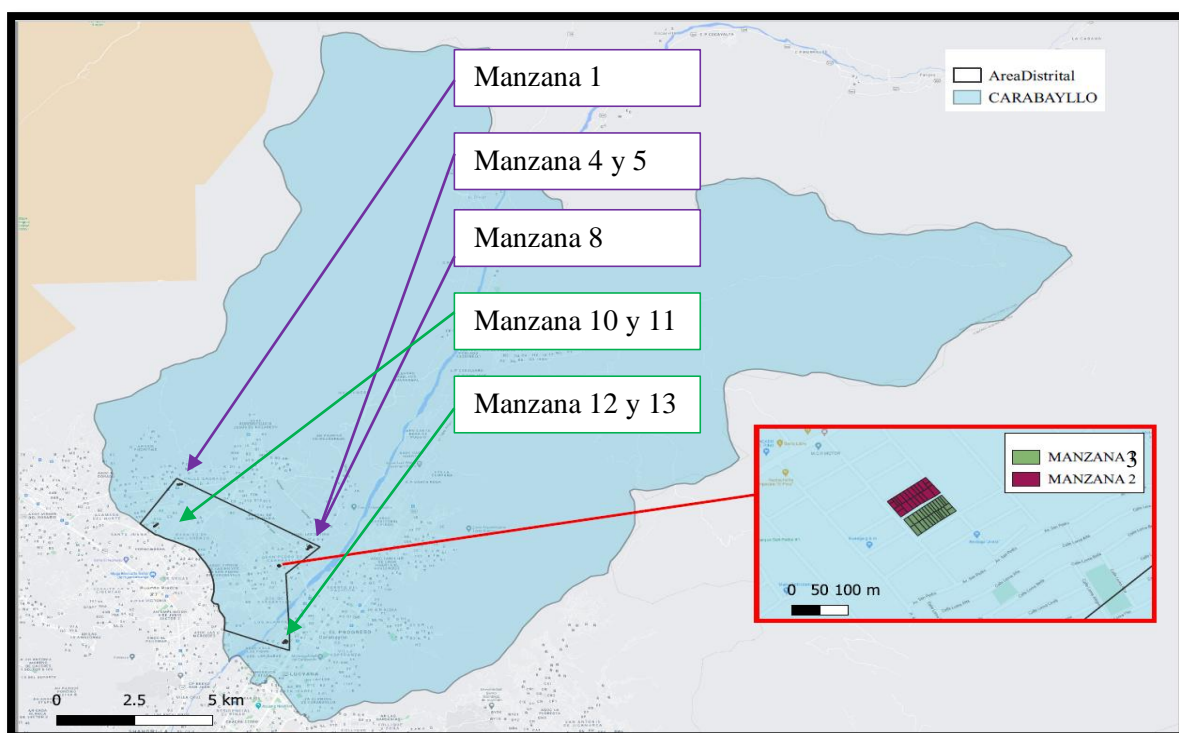


Figura n.º 20 Ubicación espacial de manzanas seleccionadas

Esta área comprende la totalidad de las manzanas a trabajar tomando en cuenta las características propuestas por Peña y Murakami, 2017; la altura referencial se encontró entre los 184 y 220 m.s.n.m. La totalidad del proyecto fue trabajado en el sistema Datum WGS 1984, coordenadas en Universal Transversal Mercator (UTM), zona 18 Sur correspondiente a la zona central de las tres que abarcan todo el litoral peruano.



a) Cuadro de puntos de control

Utilizando el software Google Earth Pro, se ubicaron las manzanas seleccionadas, denotando para cada una de ellas cuatro (04) vértices, teniendo la ocurrencia que, en ciertas manzanas colindantes, cabía la posibilidad de ubicar dos (02) manzanas dentro de un mismo conjunto de vértices.

Se obtuvieron un total de 24 vértices, los cuales fueron identificados utilizando la letra M y los números consecutivos desde el 01 al 24, la totalidad de puntos fueron guardados en un archivo de extensión Marcadores de Keyhole comprimidos (kmz).

Tabla 7: Coordenadas de referenciación para manzanas estudiadas

Punto	Manzana	Coordenadas	
		Norte (m.)	Este (m.)
M1	1	274786.055	8691400.215
M2	1	274607.111	8691385.917
M3	1	274601.799	8691284.287
M4	1	274802.165	8691289.030
M5	2-3	277959.019	8688402.247
M6	2-3	277772.803	8688407.635
M7	2-3	277780.715	8688313.137
M8	2-3	277957.095	8688310.118
M9	4-5	278949.001	8689117.113
M10	4-5	278762.969	8689109.396
M11	4-5	278773.976	8689004.252
M12	4-5	278952.009	8689000.103
M13	6	278951.929	8689000.129
M14	6	278774.201	8689004.103
M15	6	278760.895	8688911.023
M16	6	278922.938	8688908.964
M17	7-8	274018.109	8689915.315
M18	7-8	273842.150	8689920.858
M19	7-8	273830.969	8689826.256
M20	7-8	273989.990	8689804.358
M21	9-10	278091.919	8685641.288
M22	9-10	277967.015	8685484.099
M23	9-10	278036.928	8685425.168
M24	9-10	278180.925	8685572.173

## b) Imágenes

Basado en los puntos se obtuvieron las imágenes individuales, un total de 06 imágenes RGB de alta resolución.

Tabla 8: Detalles de las imágenes obtenidas

N° de imagen	Manzana	Código de Imagen	Fecha De Imagen	Urbanización
1	1	MANZANA 1	Feb-14	Valle Sagrado
2	2-3	MANZANA 2 Y 3	Feb-14	San Pedro de Carabayllo
3	4-5	MANZANA 4 Y 5	Mar-14	Asociación Las Dalias
4	8	MANZANA 8	Feb-14	Asociación Las Dalias
5	10-11	MANZANA 10 Y 11	Mar-14	Paraíso de San Lorenzo
6	12-13	MANZANA 12 Y 13	Feb-14	Las Casuarinas Etapa 3

## c) Georreferenciación

Para iniciar esta sección, se trabajará con el archivo de puntos (. Kmz), en el software ArcMap 10.6, se importará y convertirá a formato Shape (. Shp), para a continuación referenciar cada imagen con la herramienta “Georeferencing”.

Tabla 9: Datos de imágenes georreferenciadas para el estudio

Manzana	Código de Imagen	Coordenadas de centroide		Resolución de pixel		
		Norte (m.)	Este (m.)	Ancho (m.)	Largo (m.)	N° de bandas
a	(.JPG)					
1	MANZANA 1	274689.76	7691344.35	0.65	0.65	3
2	MANZANA 2 Y 3	277858.68	8688359.52	0.63	0.63	3
3	MANZANA 2 Y 3	277858.68	8688359.52	0.65	0.65	3
4	MANZANA 4 Y 5	278853.02	8689067.95	0.63	0.63	3
5	MANZANA 4 Y 5	278853.02	8689067.95	0.65	0.65	3
8	MANZANA 8	278848.80	8688966.50	0.65	0.65	3
10	MANZANA 10 Y	273915.21	8689869.15	0.65	0.65	3
	11					
11	MANZANA 10 Y	273915.21	8689869.15	0.65	0.65	3
	11					
12	MANZANA 12 Y	278072.29	8685541.49	0.65	0.65	3
	13					
13	MANZANA 12 Y	278072.29	8685541.49	0.65	0.65	3
	13					

#### 4.1.2. Clasificación Manual según estados constructivos

Producto del proceso de digitalización de bordes dentro de cada manzana, se pudo extraer de las imágenes (.jpg) correspondientes a cada lote, según su respectiva clasificación (Tabla n°10).

Tabla 10: Clasificación obtenida del proceso de clasificación manual

Manzana	Clasificación				Total
	A	B	C	D	
1	9	30	9	2	50
2	7	9	6	0	22
3	7	11	5	3	26
4	10	8	7	0	25
5	13	15	7	1	36
8	7	9	12	0	28
10	7	17	2	2	28
11	7	16	2	1	26
12	8	9	17	12	46
13	7	4	9	11	31
Total	82	128	76	32	318

Asimismo, producto de la lotización de imágenes y sus respectivos límites, se pudo obtener el detalle de sus áreas y perímetros (Tabla n°11).

Se obtuvo un total de 318 lotes, distribuidos en 82, 128, 76 y 32, de la A a la D, respectivamente; la totalidad de lotes abarcan un total de 4.70 hectáreas.

Para ilustrar los estados constructivos clasificados se muestra la imagen n° 21, donde a la izquierda se puede apreciar los cuatro estados constructivos obtenidos del proceso de clasificación manual (A11; B8; C10 y D14) y en las imágenes de la derecha se pueden observar representaciones de dichos estados constructivos.

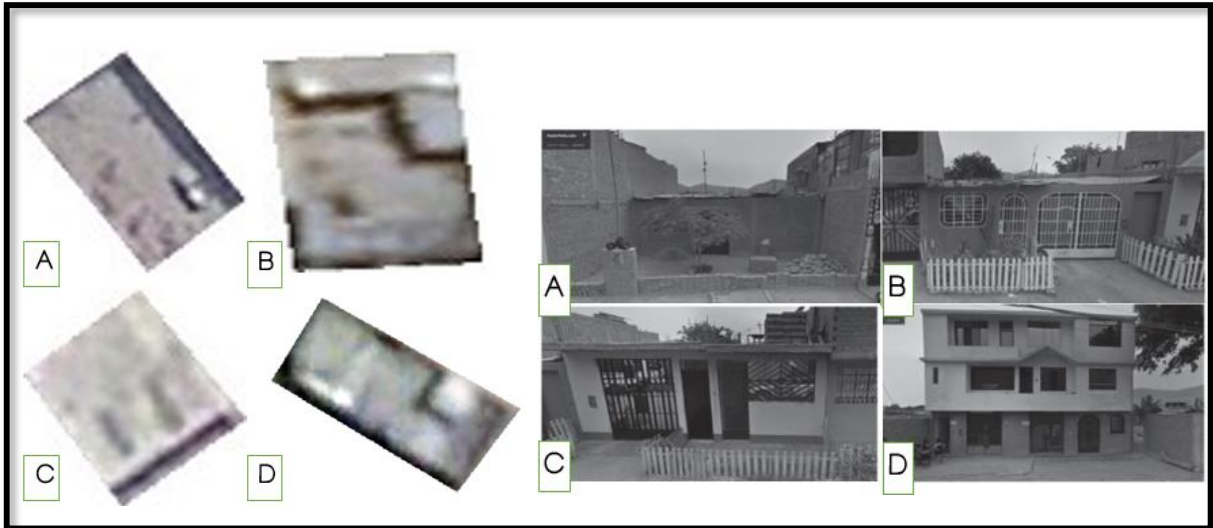


Figura n.º 21 Estados obtenidos manualmente y representación en vista de perfil.

Para introducir una muestra gráfica del proceso de lotización se visualiza la Figura n.º22, donde se puede apreciar las manzanas seccionadas según la clasificación hecha, la manzana n.º02 posee un total de 22 lotes, mientras que la manzana n.º03 posee 26 lotes. El detalle de los lotes donde se incluyen coordenadas norte y este de centroide, área, perímetro y clasificación individual se muestra en el Anexos N.º8.1



Figura n.º 22 Resultado del proceso de lotización y clasificación

Ejemplo en Manzanas n.º02 y 03.

Para el caso de la manzana n°02, se detallan sus parámetros geométricos en la tabla n° 11.

Tabla 11 Detalles individuales de lotes correspondientes a la manzana n°02

Lote	CLASS	Coordenadas de Centroides		Área	Perímetro
		Xcoord	YCoord		
		(m.)	(m.)	(m2)	(m.)
1	B	277814.36	8688338.63	64.77	32.66
2	B	277823.94	8688346.96	93.23	48.20
3	B	277827.66	8688350.40	84.96	47.41
4	B	277832.99	8688353.58	164.31	55.75
5	A	277839.95	8688356.36	167.75	56.36
6	A	277846.57	8688360.59	151.32	54.96
7	B	277853.08	8688361.12	167.10	56.71
8	B	277858.95	8688361.65	128.43	52.91
9	C	277864.77	8688363.37	148.08	54.96
10	B	277871.54	8688365.88	190.41	59.12
11	C	277876.65	8688366.67	105.81	41.25
12	C	277882.50	8688370.38	85.49	38.53
13	A	277892.16	8688371.70	233.61	65.96
14	A	277881.50	8688376.46	325.73	73.23
15	C	277872.09	8688376.86	133.84	53.51
16	C	277865.80	8688381.49	163.53	56.00
17	A	277859.14	8688384.14	150.51	54.49
18	C	277852.38	8688385.86	178.09	57.72
19	A	277843.56	8688389.69	257.63	66.03
20	A	277835.24	8688394.59	176.02	53.49
21	B	277828.87	8688395.78	131.96	47.27
22	B	277820.18	8688403.98	119.30	44.16

Tabla 12: Resumen de datos estadísticos del proceso de lotización de manzanas estudiadas

Manzana	Área					Perímetro				
	Max	Min	Desv. Estándar	Media	Moda	Max	Min	Desv. Estándar	Media	Moda
1	216.72	20.22	59.59	134.94	-	62.67	18.72	10.73	51.20	-
2	325.73	64.77	60.04	155.54	-	73.23	32.66	9.22	53.21	-
3	231.04	69.23	37.00	134.11	-	64.72	45.88	4.39	53.56	-
4	266.92	117.76	30.90	146.11	-	67.18	52.17	3.05	55.32	-
5	217.08	119.79	23.95	140.00	-	61.26	50.03	2.75	52.78	-
8	243.01	70.81	28.40	128.48	-	65.75	36.10	5.33	53.37	-
10	242.04	92.27	27.13	132.99	-	65.37	39.00	4.77	53.77	-
11	212.76	88.66	26.06	121.74	-	59.72	42.53	3.68	48.03	-
12	563.96	132.21	78.43	186.77	-	95.09	47.35	7.13	65.19	-
13	301.96	114.09	57.08	179.48	-	75.17	56.28	4.96	64.08	-

Es así como el producto final de esta sección son imágenes (.jpg) individuales georreferenciadas, clasificadas según su estado constructivo.

#### 4.1.3. Depuración de lotes

Del total de la base de datos inicial se obtuvo un conjunto de imágenes representativas de cada estado constructivo, cuantificados en la Tabla n°13, esta será la base de datos que utilizarán los modelos de inteligencia artificial para su entrenamiento y evaluación.

Tabla 13: Lotes resultantes a ser utilizados en modelos IA

Clasificación	Cantidad
A	30
B	30
C	30
D	27
Total	117

Para ilustrar la metodología descrita por la Universidad de Leon (2006) aplicado a la investigación para la depuración de lotes se muestra la Figura n°23. Aquí se muestra lo expresado en el inciso b) de la metodología en el siguiente orden:

- Alteración de la imagen por aparición de sombras. Imagen de la izquierda, sección resaltada en rojo.
- Alteración de la calidad de la imagen. La imagen de la derecha muestra secciones de píxeles de mayor tamaño, reduciendo la cantidad de información disponible para ambos modelos.
- Tamaños de lote extremadamente delgado. Imagen central.



Figura n.º 23 Depuración de lotes no-representativos

## 4.2. Proceso de clasificación supervisada-algoritmo

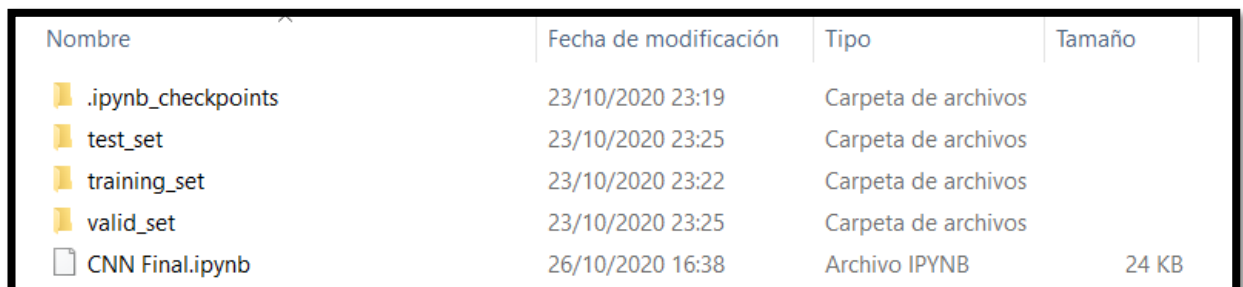
Para este proceso se ha escogido el software Jupyter, un proyecto de código abierto creado en 2014 que evolucionó para ser uno de los principales soportes interactivos en estudios de data científica (Data science) y computación científica (Scientific computing) es compatible con muchos lenguajes de programación, dentro de los que se incluye el utilizado en esta tesis, Python.

### 4.2.1. Modelo de Redes Neuronales Convolucionales (CNN)

Previamente a la codificación del modelo será necesaria la organización de las carpetas, estas contendrán las imágenes obtenidas en el acápite 4.1, en la misma carpeta se alojará el modelo, que para este caso ha sido nombrado como “CNN Final”; como medida de contingencia, Jupyter crea una carpeta automáticamente donde irá guardando periódicamente los avances en el proyecto, para evitar pérdidas indeseadas, el nombre no es elegible por lo que en la Figura n.º24, se muestra cómo “.ipynb\_Checkpoints”. En la misma Figura, se muestran las tres carpetas restantes que contendrán las imágenes de entrenamiento, separadas correspondientemente. Dentro de cada una de estas carpetas se crearán carpetas con los estados constructivos de las imágenes, detalle en la tabla n.º14.

Tabla 14: Cantidad de imágenes según finalidad

Nombre de Carpeta	Descripción	Cantidad de imágenes
training_set	Imágenes de entrenamiento	77
test_set	Imágenes de evaluación	20
valid_set	Imágenes de validación	20
	Total	117



Nombre	Fecha de modificación	Tipo	Tamaño
.ipynb_checkpoints	23/10/2020 23:19	Carpeta de archivos	
test_set	23/10/2020 23:25	Carpeta de archivos	
training_set	23/10/2020 23:22	Carpeta de archivos	
valid_set	23/10/2020 23:25	Carpeta de archivos	
CNN Final.ipynb	26/10/2020 16:38	Archivo IPYNB	24 KB

Figura n.º 24 Organización de carpetas de trabajo. Modelo CNN



Para iniciar con la codificación se importarán las librerías necesarias para la creación del modelo de Red Neuronal Convolutiva (CNN), un parcial de 8 librerías, posteriormente se importarán las necesarias para la matriz de confusión y el coeficiente de Kappa.

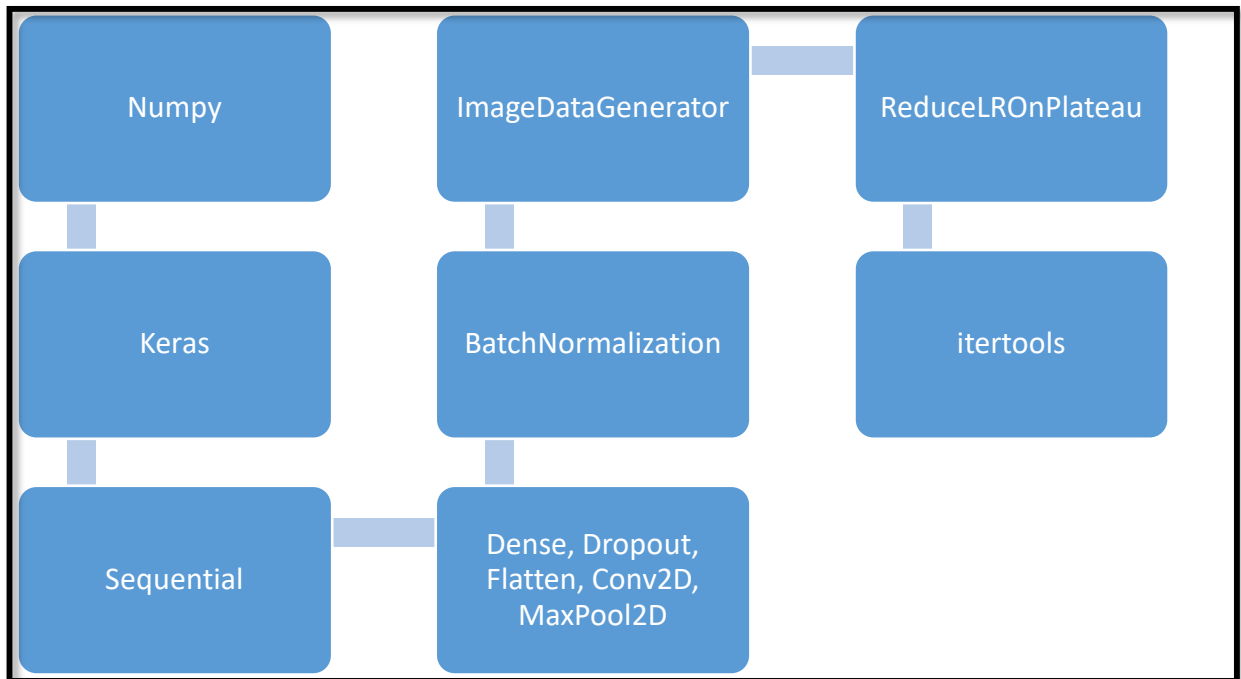


Figura n.º 25 Secuencia de importación de librerías, modelo CNN

Seguidamente se definirán los parámetros iniciales que el modelo solicita. Dichos parámetros se muestran en la tabla n.º15. Adicionalmente, se agrega el formato con el que ingresará la imagen: Ancho, Largo y tres (03) bandas, correspondientes a los colores de las imágenes, rojo, verde y azul.

Tabla 15: Parámetros de ingreso para imágenes e iteraciones

Parámetro	Descripción	Magnitud	Unidad
img_width	Ancho de imagen	224	Pixeles
img_height	Alto de imagen	224	Pixeles
batch_size	Tamaño de lote	1	-
num_classes	Cantidad de clases	4	-
epochs	nº de Iteraciones	5	-

En un punto aparte, se necesitará conectar las carpetas de trabajo creadas previamente, de manera que se puedan relacionar al utilizar los nombres de los parámetros definidos a continuación (Tabla n.º16):

Tabla 16: parámetros definidos para la relación con carpetas de data

Parámetro	Descripción	Nombre de Carpeta
train_data_dir	Carpeta de imágenes de entrenamiento	'training_set'
test_data_dir	Carpeta de imágenes de evaluación	'test_set'
valid_data_dir	Carpeta de imágenes de validación	'valid_set'

Seguidamente se ingresará la estructura del modelo (Figura n.º26), comenzando por especificar que es un modelo secuencial, luego cada designación “.add”, será una nueva pieza en la estructura, teniendo esto en cuenta, se tiene un total de tres (03) capas, cinco (05) neuronas, dos (02) reductores de características y tres (03) reductores de aprendizaje, dichos parámetros se detallan en la Tabla n.º17.

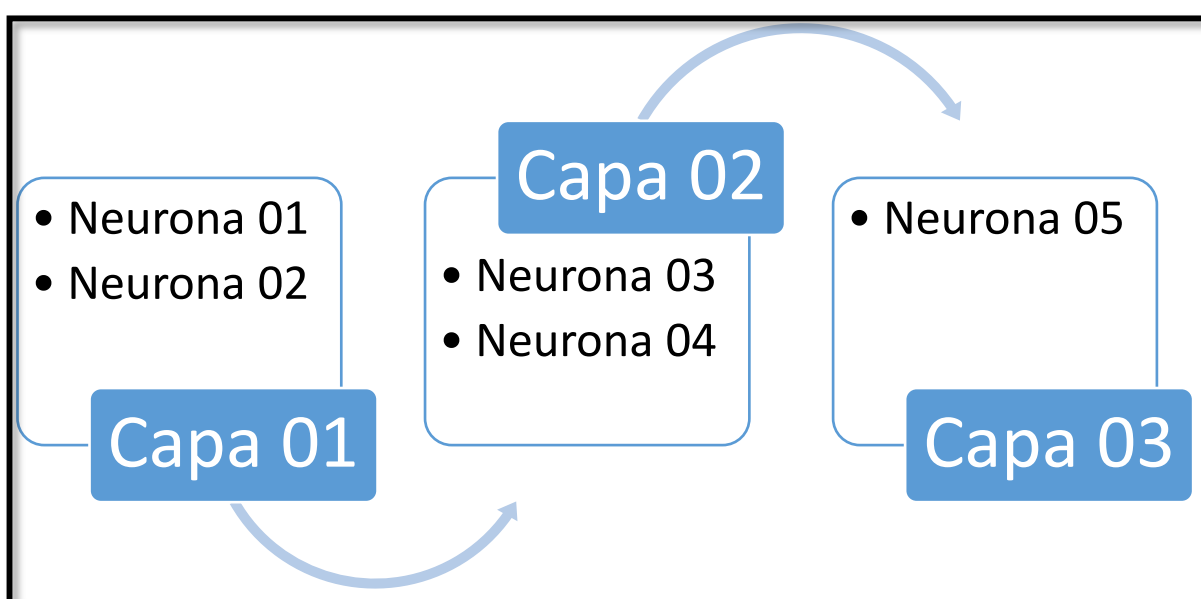


Figura n.º 26 Secuencia de capas y neuronas utilizadas para el modelo CNN

Tabla 17: Parámetros detallados correspondientes a la estructura del modelo

Capa	Neurona	Parámetro	Descripción	Valor	
1	1	Filtros	Cantidad de filtros	32	
		kernel_size	Tamaño de los filtros	3x3	
		Activation	Método de activación	Relu	
			kernel_initializer	Valores iniciales de la neurona	he_normal
	2	Filtros	Cantidad de filtros	32	
		kernel_size	Tamaño de los filtros	3x3	
		Activation	Método de activación	Relu	
		kernel_initializer	Valores iniciales de la neurona	he_normal	
	-	MaxPool2D	Reductor de características	2x2	

	-	Dropout	Inhibidor de aprendizaje	0.2
2	3	Filtros	Cantidad de filtros	64
		kernel_size	Tamaño de los filtros	3x3
		Activation	Método de activación	Relu
		Padding	Los inputs y outputs tendrán el mismo formato de imagen	Same
		kernel_initializer	Valores iniciales de la neurona	he_normal
	4	Filtros	Cantidad de filtros	64
		kernel_size	Tamaño de los filtros	3x3
		Activation	Método de activación	Relu
		Padding	Los inputs y outputs tendrán el mismo formato de imagen	Same
		kernel_initializer	Valores iniciales de la neurona	he_normal
-	MaxPool2D	Reductor de características	2x2	
-	Dropout	Inhibidor de aprendizaje	0.25	
3	5	Filtros	Cantidad de filtros	128
		kernel_size	Tamaño de los filtros	3x3
		Activation	Método de activación	Relu
		Padding	Los inputs y outputs tendrán el mismo formato de imagen	Same
		kernel_initializer	Valores iniciales de la neurona	he_normal
	-	Dropout	Inhibidor de aprendizaje	0.25

Para la siguiente sección, se compilará la estructura definida para el modelo, de manera que se compruebe su validez; también se agregará el factor de reducción de aprendizaje, esto con el fin de reducir la posibilidad de errores por sobre ajuste; en la Figura n.º27, se muestran ambos procesos, en la tabla n.º18 los parámetros utilizados.

Compilación	Reducción de aprendizaje
<input type="checkbox"/>	<input type="checkbox"/>
<input type="checkbox"/> Loss	<input type="checkbox"/> Monitor
<input type="checkbox"/> Optimizador	<input type="checkbox"/> Verbose
<input type="checkbox"/> Metrics	<input type="checkbox"/> Factor
	<input type="checkbox"/> Paciencia
	<input type="checkbox"/> Mínimo de reducción

Figura n.º 27 Proceso de compilación del modelo; los parámetros a utilizar en cada ítem

Tabla 18: Parámetros de finalización para la estructura

	Parámetro	Descripción	Valor
Compilación	loss	Pérdida de aprendizaje	binary_crossentropy
	optimizer	Optimizador	RMSprop()
	metrics	Exactitud	accuracy
Reducción de aprendizaje	Monitor	Exactitud para la validación	val_acc
	patience	Cantidad de iteraciones sin cambios antes aplicar el factor	3
	verbose	Visualización de iteraciones	Si
	factor	Factor de reducción	0.5
	min_lr	margen mínimo	0.0001

Para continuar, es necesario trabajar sobre las imágenes que se utilizarán, modificarlas de manera que obtengamos una base de datos más amplia. Es por esto que se utilizan diferentes comandos para rotar, ampliar, voltear, en general alterar de diferentes formas las imágenes existentes. Dichos detalles se muestran en la Tabla n.º19.

Tabla 19: Parámetros de modificación para las imágenes

Parámetro	Descripción	Valor
rescale	Normalizar colores	1/255
featurewise_center	Modificar la media general a 0	No
samplewise_center	Modificar la media individual a 0	No
featurewise_std_normalization	Dividir data entre desviación estándar general	No
samplewise_std_normalization	Dividir data individual entre desviación estándar individual	No
zca_whitening	Blanqueamiento de las imágenes	No
rotation_range	Rotar imágenes	15
zoom_range	Acercamiento a las imágenes	0.1
width_shift_range	Cambiar el ancho de las imágenes	0.1
height_shift_range	Cambiar el alto de las imágenes	0.1
horizontal_flip	Voltear horizontalmente las imágenes	No
vertical_flip	Voltear verticalmente las imágenes	No

A continuación, se definirán los generadores de datos (Figura n.º28), dicho de otra forma, se codifica una función que aplique todos los parámetros previamente definidos a cada una de las imágenes que se vayan a utilizar. Esto es necesario para los datos de entrenamiento, validación y evaluación.

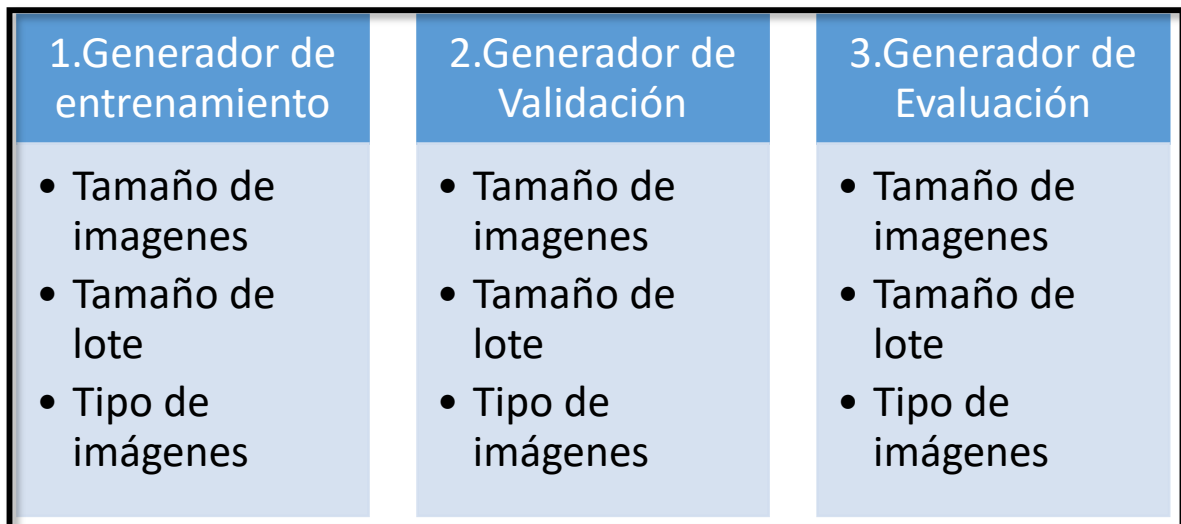


Figura n.º 28 Secuencia de generadores de datos

Para los tres generadores, el código nos muestra sus resultados. Para datos de entrenamiento se encuentran 77 imágenes pertenecientes a cuatro (04) clases, mientras que para validación y evaluación se encuentran 20 imágenes pertenecientes a (04) cuatro clases.

```
Found 77 images belonging to 4 classes.
Found 20 images belonging to 4 classes.
Found 20 images belonging to 4 classes.
```

Figura n.º 29 Cantidad de imágenes encontradas para entrenamiento, validación y evaluación

Finalmente se ejecuta las iteraciones propuestas, tanto para los datos de entrenamientos como para los datos de validación. Los detalles se muestran en la Tabla n.º20.

Tabla 20: Parámetros finales requeridos para las iteraciones

Parámetro	Descripción	Valor
steps_per_epoch	Pasos por iteración	77
epochs	Iteraciones	Epochs
verbose	Visualización	Si
validation_steps	Pasos de la validación	20
callback	Integrar el reductor de aprendizaje	learning_rate_reduction

Ahora, se evaluará el modelo sobre la data de evaluación, para esto es necesario resetear el generador para la data a utilizar.

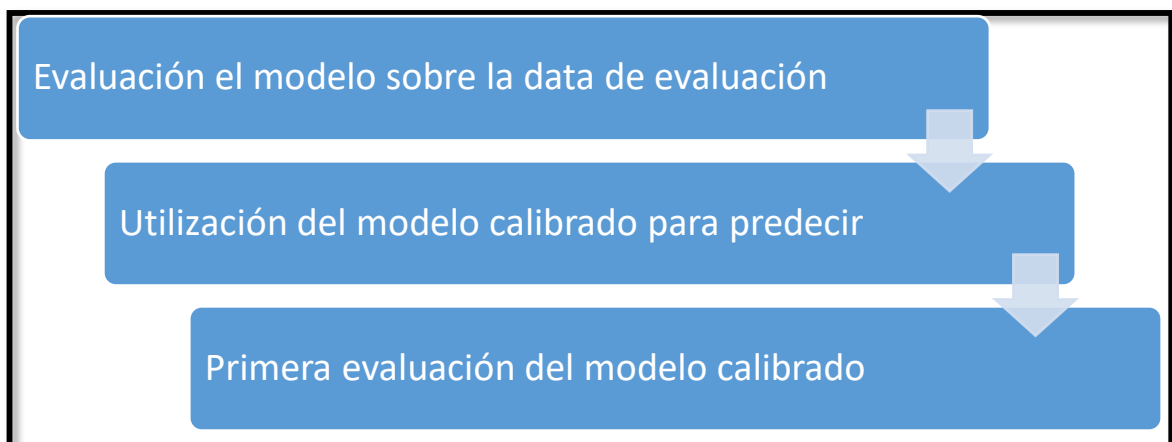


Figura n.º 30 Proceso de evaluación del modelo calibrado

Luego, se utilizará el generador para predecir las clases de las 20 imágenes de la data de prueba. Cuyo resultado se presenta a continuación:

[0 2 1 3 1 3 3 0 0 2 3 0 2 1 1 2 0 1 2 3]

Donde:

- Cero: Estado A
- Uno: Estado B
- Dos: Estado C
- Tres: Estado D

Finalmente se calculará un primer valor de evaluación del modelo, calculadas sobre los 20 pasos que incluyen toda nuestra data de evaluación. El primer valor se refiere a las pérdidas de aprendizaje y el segundo valor a la exactitud del modelo

[5.809216010456254, 0.625]

#### 4.2.2. Modelo Bayesiano (NB)

Como primer paso del proceso de creación del código es necesario crear las carpetas donde se alojarán las de imágenes de entrenamiento a trabajar, obtenidas en el acápite 4.1, dentro de la misma se alojará el proyecto el cual ha sido nombrado “TesisNaiveBayes”. Estas carpetas fueron nombradas según su estado constructivo. El software, a manera de contingencia, crea una carpeta automáticamente donde irá guardando periódicamente los avances en el proyecto, para evitar pérdidas indeseadas, el nombre no es elegible por lo que en la Figura n.º30, se muestra cómo “.ipynb\_Checkpoints”.

Nombre	Fecha de modificación	Tipo	Tamaño
.ipynb_checkpoints	21/07/2020 11:43	Carpeta de archivos	
A	12/07/2020 12:40	Carpeta de archivos	
B	12/07/2020 12:40	Carpeta de archivos	
C	12/07/2020 12:40	Carpeta de archivos	
D	21/10/2020 18:35	Carpeta de archivos	
TesisNaiveBayes.ipynb	21/10/2020 17:58	Archivo IPYNB	31 KB

Figura n.º 31 Organización de carpetas de trabajo. Modelo NB.

Para el inicio de la codificación se importa las librerías necesarias para la generación y el desarrollo del modelo Bayesiano; un parcial de diez (10) librerías, para la sección de matriz de confusión se importarán las restantes.

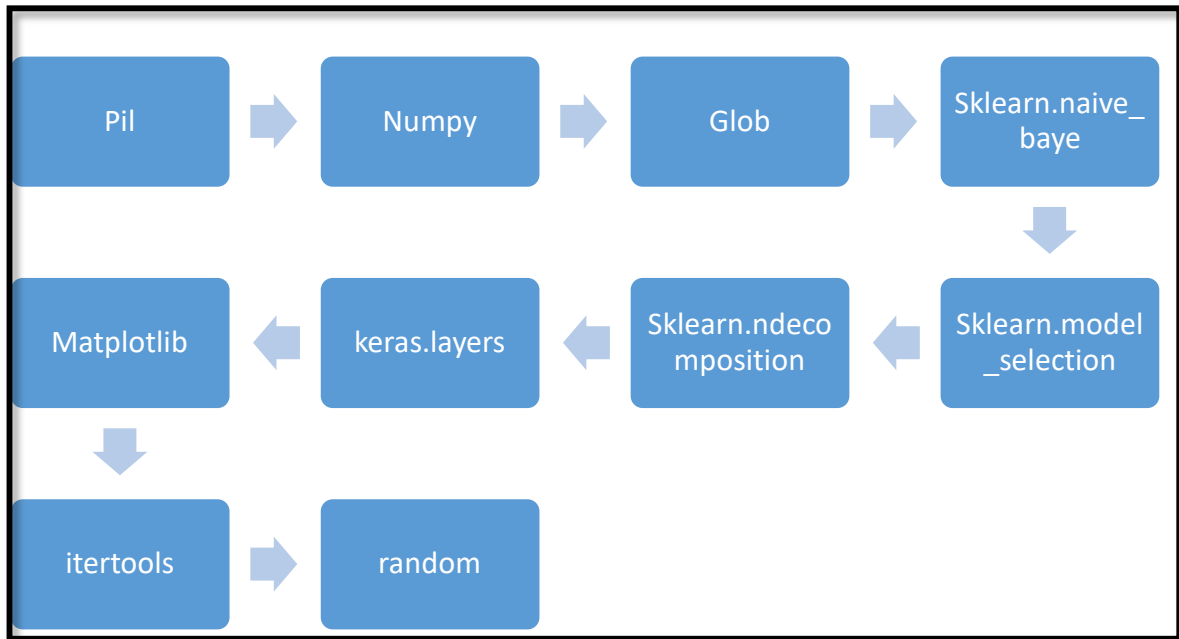


Figura n.º 32 Secuencia de importación de librerías, modelo NB

A continuación, se definirá el formato de las imágenes: comando Def y el arreglo que tendrán dichas imágenes, np.asarray. donde se indica re estructurar las imágenes a un nuevo arreglo de alto y ancho de 96x96, respectivamente; data.Ravel, indica al código que el producto final no será un arreglo 96x96, sino un arreglo lineal de 1x9216; este formato se utilizará en la siguiente sección para cargar las imágenes al código.

Tabla 21: Formato de imágenes

Parámetro	Descripción	Valor
img	Lectura de la imagen en la dirección a especificar	Image.open(infilename)
Data1	Re escalado de la imagen de entrada	96x96
Data2	Conversión a matriz (1*9216)	Ravel

Seguidamente, se indicará con el comando “folders=” las carpetas de las cuales deberá escoger la información, en nuestro caso las carpetas correspondientes a los estados constructivos. La función Glob buscará los archivos en las carpetas seleccionadas,



adicionalmente se indica que sólo ha de seleccionar los archivos en extensión “.jpg”, correspondiente a imágenes.

Cabe mencionar que “X” denota las imágenes, e “Y” las clasificaciones (A, B, C, D) de dichas imágenes.

Tabla 22: Ingreso de imágenes

Parámetro	Descripción	Valor
Folders	Carpetas a relacionar	A', 'B', 'C', 'D'
img	Cargar imágenes	load_image(infile)
X.append	Arreglar imágenes secuencialmente	X.append(img)
Y.append	Arreglar clasificaciones secuencialmente	Y.append(folder)

Como cuarto punto se separará la data en cuatro sub categorías para el entrenamiento y la evaluación de nuestro modelo. Este será una de las funciones medulares del modelo.

Comando:

```
>>X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=30)
```

Tabla 23: Parámetros empleados en la función train\_test\_split

Parámetros	Descripción
test_size	Cantidad de datos de evaluación
X_train	Datos de entrenamiento
X_test	Datos de prueba
Y_train	Clasificación real del entrenamiento
Y_test	Clasificación real de la data a evaluar

Finalmente se escoge un nombre para el clasificador, y se especifica que se utilizará la función de distribución normal (Gaussiana) para el teorema de Ingenuidad de Bayes (NB), a continuación, se ajustarán los datos de entrenamiento al clasificador usando “. fit”.

Tabla 24: Parámetros de ajuste del clasificador

Parámetro	Descripción	Valor
clf	Selección del clasificador	GaussianNB()
clf.fit	Ajustar el clasificador a cierta data	clf.fit(X_train, Y_train)
print	Mostrar los resultados deseados	"Predicted labels on test set:\n"
print	Mostrar los resultados deseados	clf. predict(X_test)
print	Mostrar los resultados deseados	\nAccuracy = ' + str(int(clf.score(X_test, Y_test)*100)) + '%')

Las tres últimas filas de la tabla n.º24 mostrarán la información de salida que solicitamos, las predicciones que hace el modelo sobre las 30 imágenes de prueba, y la exactitud que se ha calculado a partir de la diferencia entre las predicciones y la clasificación real.

Clasificaciones Predichas en la Data de evaluación:

```
['B' 'A' 'A' 'B' 'C' 'B' 'A' 'D' 'D' 'B' 'B' 'B' 'B' 'A' 'B' 'A' 'A' 'B'  
'A' 'B' 'B' 'C' 'A' 'B' 'B' 'B' 'C' 'B' 'B' 'C']
```

Accuracy = 63%

Figura n.º 33 Exactitud obtenida para la data de evaluación. Modelo Bayesiano

### 4.3. Comparación en exactitudes, pérdidas, tiempos.

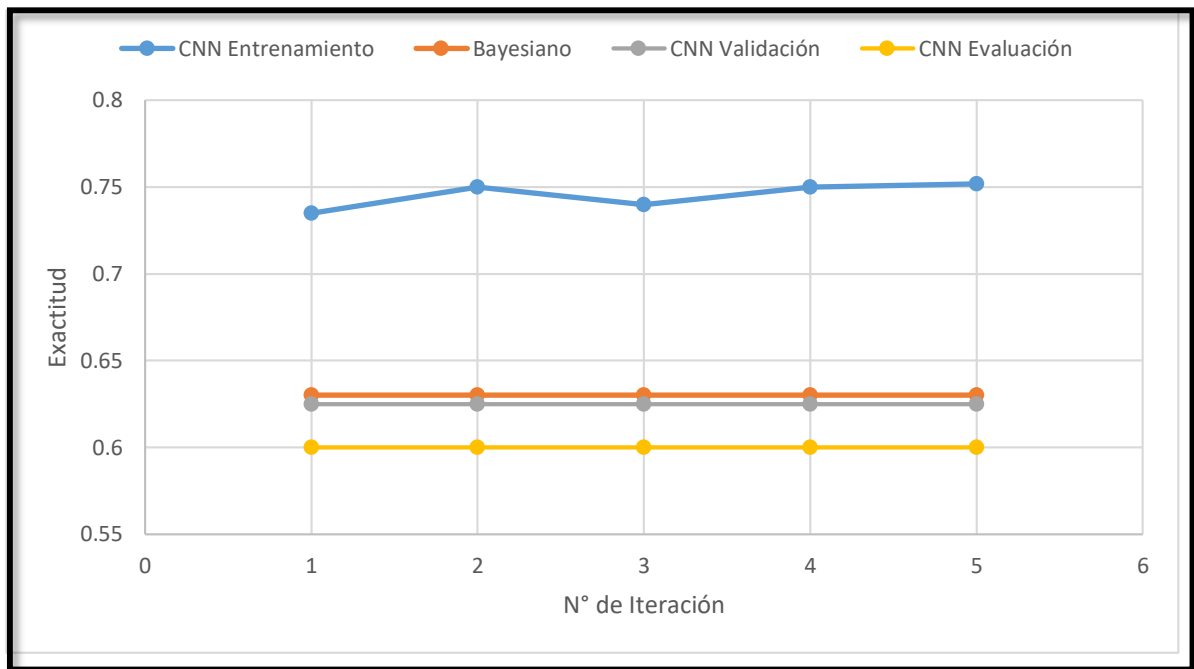


Figura n.º 34 Gráfica de porcentaje de exactitud Vs progreso a través de las iteraciones

Como se puede observar en la Figura n.º34, para el modelo CNN, consta de tres diferentes exactitudes: la de entrenamiento que alcanza un valor máximo de 75.2 % a lo largo de las cinco iteraciones que se han planteado; las exactitudes de validación y evaluación se mantienen constantes con valores 62.5% y 60%, respectivamente. De igual forma, el modelo bayesiano resulta en una única exactitud, 63%.

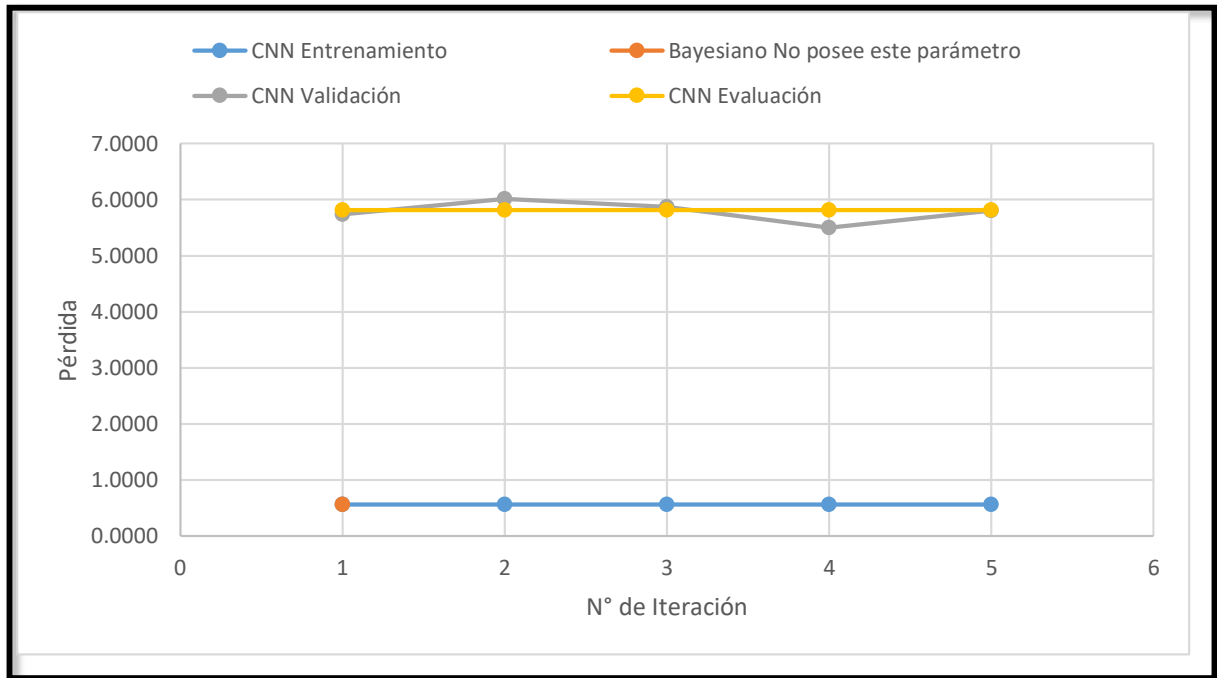


Figura n.º 35 Gráfica de Pérdidas Vs progreso a través de las iteraciones

Como se puede observar, para el gráfico de pérdidas, los valores del entrenamiento se obtienen como máximo hasta 0.5631 puntos; para el caso de validación y evaluación presentan cierto grado de paralelismo, obteniéndose 6.0113 y 5.8092 como máximo, respectivamente. Al ser un modelo más simple, el modelo bayesiano no utiliza parámetros de pérdida de aprendizaje, por lo que no se representa en el gráfico.

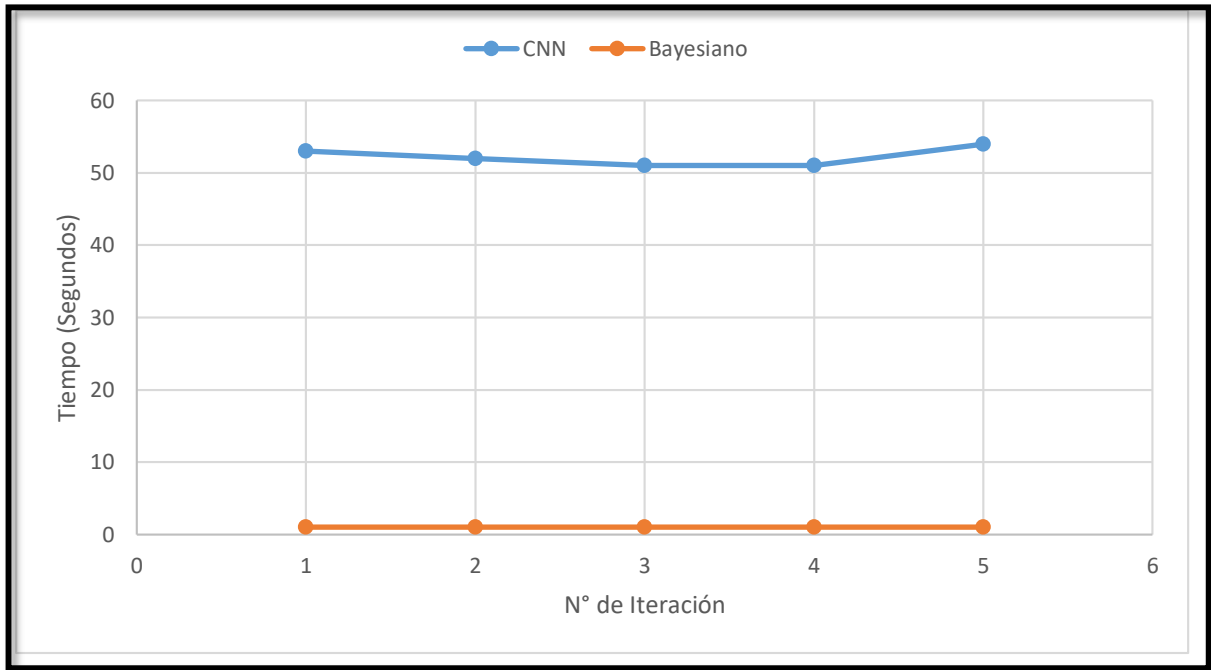


Figura n.º 36 Comparación de tiempos empleados por los modelos utilizados

Como se observa en la Figura n.º36, un valor especialmente importante para este estudio es la cantidad de tiempo empleado por cada modelo para poder iterar y procesar la información que se le ha entregado. Para el modelo CNN varía entre 51 a 54 segundos por cada iteración, haciendo un tiempo total de 04 minutos y 20 segundos; por otro lado, el modelo bayesiano utiliza menos de un segundo como tiempo total.

#### 4.4. Paquete de validación

En esta sección se trabajará con las métricas de evaluación seleccionadas para ambos modelos.

##### 4.4.1. Matriz de confusión

###### a) Red Neuronal Convolutiva

Como primer paso es necesario la librería que nos permitirá crear el gráfico de doble entrada, al ser un código, se debe definir la totalidad de sus parámetros como color, tamaño de etiquetas, rotación de texto, tamaño de texto, entre otros; los parámetros necesarios se especifican en la Tabla n.º25.

Tabla 25: Parámetros requeridos para la matriz de confusión, CNN

Parámetro	Descripción	Valor
normalize	Normalización	No
title	Título del gráfico	Confusion Matrix
cmap	Color de la digitalización	plt.cm.Blues
plt.xticks	Etiquetas Eje X	classes
plt.yticks	Etiquetas Eje Y	classes
Rotation	Rotación de las etiquetas	45°

A continuación, es necesario definir los argumentos de la matriz de confusión, se importará la librería necesaria y se tendrán dos argumentos: las clases originales (Clasificaciones correctas), y las predicciones hechas por nuestro modelo CNN.

De la librería Sklearn.metrics:

- Se importa “confusión\_matrix”

Aquí, se traen a acotación:

- a) Classes
- b) y\_pred

Finalmente, se definen las clasificaciones y se conecta con la data a utilizar, ya explicada en el párrafo anterior.

Etiquetas:

- 'A' , 'B' , 'C' , 'D'

Dibujar la matriz de confusion

- Parámetros definidos
- Etiquetas pre definidas

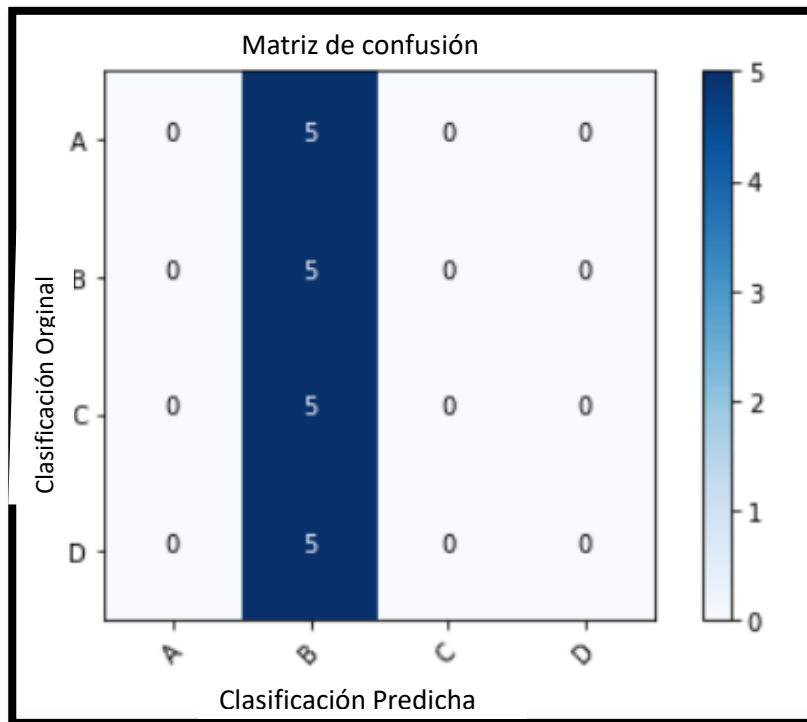


Figura n.º 37 Matriz de confusión resultante del modelo CNN

Como se puede observar en la Figura n.º37, la matriz de confusión dista mucho de lo deseado para un modelo ideal; el total de predicciones se han dado en el estado constructivo B (Incipiente), correspondiendo sólo a un porcentaje de exactitud del 25%.

#### b) Modelo Bayesiano

Como primer paso es necesario la librería que nos permitirá crear el gráfico de doble entrada, al ser un código, se debe definir la totalidad de sus parámetros como color, tamaño de etiquetas, rotación de texto, tamaño de texto, entre otros. Los parámetros necesarios se especifican en la Tabla nº26.

Tabla 26: Parámetros requeridos para la matriz de confusión, Bayesiano

Parámetro	Descripción	Valor
normalize	Normalización	No
title	Título del gráfico	Confusion Matrix
cmap	Color de la digitalización	plt.cm.Blues
plt.xticks	Etiquetas Eje X	classes
plt.yticks	Etiquetas Eje Y	classes
Rotation	Rotación de las etiquetas	45°

A continuación, es necesario definir los argumentos de la matriz de confusión, se importará la librería necesaria y se tendrán dos argumentos: las clases originales (Clasificaciones correctas), y las predicciones hechas por nuestro modelo Bayesiano.

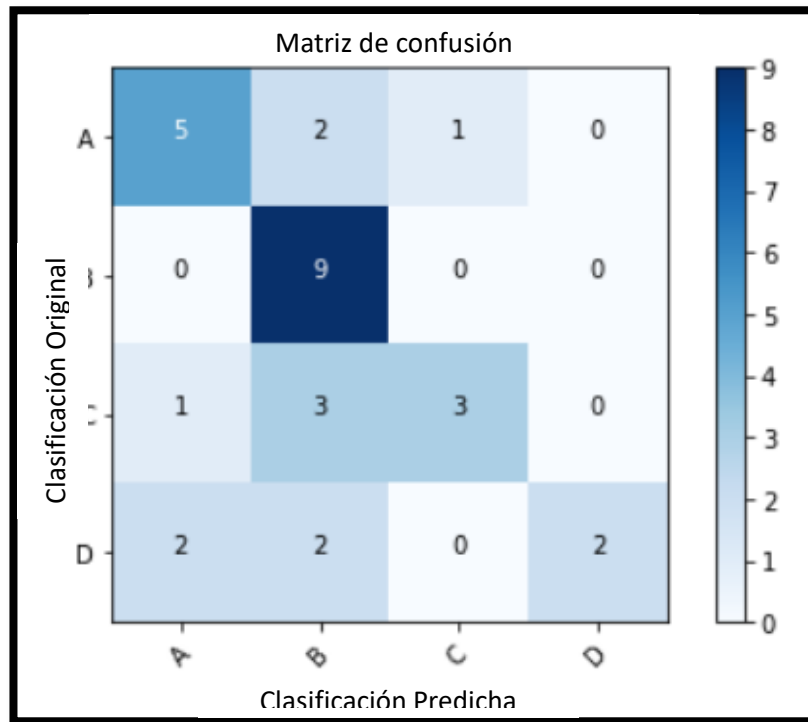


Figura n.º 38 Matriz de confusión resultante del modelo Bayesiano

Como se puede observar en la Figura n.º38, la diagonal se muestra con una mayor cantidad de concordancia entre las clases predichas y las clases originales, un total de 19 predicciones correctas, sobre una base de evaluación de 30 imágenes, cuya exactitud, resulta en 63%, concordante con el resultado mostrado en el código, en el acápite 4.2.2, Figura nº32.

#### 4.4.2. Coeficiente de Kappa

a) Red Neuronal Convolutiva

Aquí, se importa la librería correspondiente y los argumentos necesarios: las clases originales (Clasificaciones correctas), y las predicciones hechas por nuestro modelo CNN.

Se tiene de las métricas de Sklearn:

- `Cohen_kappa_score`

Cuyo resultado es:

[0.00]



#### b) Modelo Bayesiano

Aquí, se importa la librería correspondiente y los argumentos necesarios: las clases originales (Clasificaciones correctas), y las predicciones hechas por nuestro modelo Bayesiano.

Se tiene de las métricas de Sklearn:

- Cohen\_kappa\_score

Para cuyo resultado se obtiene:

[0.4375000000000001]

Como se puede observar, el coeficiente de Kappa, para el modelo CNN, resulta un valor de 0.00 lo que significa un completo azar entre las clases predichas y las clases originales; por otro lado, el modelo Bayesiano, da un resultado de 0.4375.

#### 4.5. Discusión

- El método de aprendizaje automático requiere imágenes para las etapas de entrenamiento, validación y predicción, para ello se realizó un pretratamiento de la información de libre acceso disponible en Google Earth, de lo cual se obtuvieron 318 imágenes de lotes individuales correspondientes a casas en las cuatro etapas constructivas categorizadas.

##### a) De las imágenes de entrenamiento

- De las imágenes correspondientes a los 318 lotes individuales, pertenecientes a diez manzanas, se escogieron 117 imágenes (36.79 % del total). Esta depuración considero limitaciones como el tamaño del pixel, la aparición de sombras en días soleados, nubosidad, entre otros factores que no correspondían a la estructura física de la edificación.

##### b) Parámetros utilizados

- La adecuación de los modelos Bayesiano y CNN, para la tarea de reconocimiento visual computacional de los cuatro estados constructivos, emplea dos estructuras de procesamiento diferentes. La estadística bayesiana trabaja bajo la inferencia de que todos los atributos o variables de la data son independientes entre sí, reduciendo los parámetros a características básicas como reconocimiento de bordes, tamaños de lotes y texturas, por otro lado, el modelo CNN posee una mayor cantidad de parámetros, los cuales fueron modificados sin lograr obtener mejores resultados.

- El modelo CNN registra diferentes niveles de procesamiento, para el presente estudio fueron estructurados en cinco (05) niveles de convoluciones y rectificaciones lineales, donde los primeros filtros se encargan de la diferenciación de características como bordes, continuando en la tercera y cuarta capa donde se utilizan los bordes obtenidos para realizar la conformación de estructuras más complejas como bordes unidos, agregado de texturas colindantes con el borde y el análisis de los colores, finalmente, en el último nivel, se realiza la conformación de imágenes con todas las características previamente definidas.
  - Uno de los motivos por la cual el modelo bayesiano tiene un buen performance se puede señalar como la presunción bajo la cual trabaja el modelo donde las variables son independientes entre sí, en concordancia con lo dicho por Roman, 2019, la presencia de una cierta característica (tamaño de lote, textura) en un conjunto de datos no está en absoluto relacionada con la presencia de otra característica.
- c) De la data a utilizarse
- El modelo CNN utiliza diferentes parámetros como cantidad y tamaño de filtros, cantidad de iteraciones, ratio de aprendizaje, entre otros; junto a los requerimientos computacionales y la data necesaria pueden significar un factor importante al momento de seleccionarlo para un trabajo específico. La información ideal para la data de ingreso se podría considerar como, imágenes de alta resolución espacial (0.10 metros, Cao, Dragičević & Li, 2019), tomadas por la fuente durante un solo día y una clasificación de estados constructivos validada por un experto, siendo otra necesidad, una gran cantidad de data de entrenamiento, superior a las mil muestras por clase (Brownlee, 2017).

Tabla 27: Resumen de resultados en ambos modelos

Clasificador	Exactitud (%)		Coef. Kappa
	Promedio	Max	
CNN	0.6568	0.7520	0.0000
Bayesiano	0.6300		0.4375

- d) Resultados de las exactitudes alcanzadas
- El modelo CNN muestra métricas más cercanas al 100% de exactitud (Tabla n.º27), sin embargo, el resultado del coeficiente de Kappa, indica que, entre los resultados predichos y las clasificaciones originales, no hay mayor coincidencia que el puro azar, restándole validez a la utilización de este modelo, para este trabajo específico de detección de estados constructivo; en comparación, el modelo Bayesiano muestra un resultado de

exactitud menor, sin embargo, existe una concordancia mayor al momento de calcular el coeficiente de Kappa, logrando un valor de 0.4375. Respecto a este valor Landis&Koch, 1977 hicieron una clasificación (Figura n.º13), donde nos indican una fuerza de concordancia “moderada” entre los valores predichos y los originales.

- La exactitud aquí considerada presenta dos acepciones, la primera corresponde a la etapa de evaluación del propio modelo y como segunda acepción se considera al resultado obtenido en la diagonal de la matriz de confusión. La inconsistencia entre ambas, se puede deber al error por sobre ajuste discutido por Prakash, 2018, quien nos indica que debemos tener una gran cantidad de información de entrada para evitar incurrir en este error, el cual no fue el caso del presente estudio donde el total de la data fueron 117 imágenes de entrada.

e) De los tiempos de procesamiento

- El modelo CNN permite un trabajo más refinado en diferentes aspectos (Cantidad de información e iteraciones, parámetros modificables, etc.). Por lo tanto, se debe resaltar la cantidad de tiempo (4 minutos y 20 segundos) que ha empleado para completar las iteraciones en una base de datos relativamente pequeña, 117 imágenes con tan sólo 5 iteraciones, asimismo se debe considerar que este modelo tiene mayores requerimientos computacionales. Por el contrario, el procesamiento del modelo bayesiano demoró menos de un segundo concordando en lo dicho por Roman, 2019 este modelo es “una manera rápida para problemas de clasificación multi clase”.

f) De la matriz de confusión

- Las matrices de confusión resultaron útiles para representar el grado de precisión de los resultados de los modelos en el reconocimiento visual de cada etapa constructiva. Como se observa en la figura n.º34 teniendo en cuenta los valores de exactitud obtenidos por el modelo (Tabla n.º27), la matriz de confusión muestra las predicciones del modelo CNN como todas correspondientes al estado B, significando una precisión del 25%. Para el caso de la Figura n.º33, correspondiente al modelo bayesiano, la diagonal corresponde al 63% obtenido en la evaluación del modelo.

## V. CONCLUSIONES

- a) De las imágenes de entrenamiento y sus aplicaciones
- El presente estudio provee información relevante para las gestiones a nivel municipal y de organismos del estado, en especial, aquellas que deben velar por la formalización de viviendas y a su vez, monitorear el progreso constructivo. Dentro de estos puntos la investigación colabora en los protocolos de “Generación de base de datos catastral predial urbana” específicamente para el caso de COFOPRI, donde pretende reemplazar las visitas de campo por modelos de inteligencia artificial.
  - El proceso de extracción de lotes individuales de las imágenes base se hizo de manera manual, este proceso se puede realizar de manera automática en softwares de sistemas de información geográfica asistidos de la codificación en lenguajes compatibles a ellos, como lo es Python, como: *Parallel Python extraction* (Kimosfo, 2016) y *Scrip extract by mask* (Pantoja, 2016).
- b) De la data a utilizarse
- El presente estudio se puede utilizar como base para trabajos futuros, donde una mayor resolución espacial, con tamaños de píxeles inferiores a 0.60 metros en las imágenes de entrenamiento extraídas, se traduzcan en ambos modelos evaluados en matrices con mayor cantidad de información para una mejor identificación de características, en la sección: determinación de filtros para mapas de características.
- c) De los resultados
- Los modelos de inteligencia artificial que utilizan estructuras neuronales suelen alcanzar valores de exactitud en el rango de 95.06 a 98.14 % (Cao, Dragičević & Li, 2019), en este caso, para una estructura relativamente sencilla el modelo CNN utilizado se acercó a los valores teóricos, pero no logró superar el performance del modelo bayesiano.
  - Para el modelo CNN el rediseño de la estructura permitirá un mejor performance para trabajos futuros, específicamente modificando la cantidad de iteraciones (500 o más), tamaño y cantidad de filtros, estimándose un ideal en 16 capas de filtros con un tamaño de 2x2, cantidad de funciones convolucionales, ReLus (por encima de 10) y max pooling (5), adicionalmente, cambiando el optimizador utilizado.

## **VI. RECOMENDACIONES**

- Para futuros trabajos de clasificación supervisada utilizando algoritmos de aprendizaje automático, es aconsejable evaluar un primer acercamiento utilizando un modelo Bayesiano y, posteriormente, en caso de ser necesario por un pobre performance, proceder a la evaluación de un modelo de Red Neuronal Convolucional, en la medida que sea posible.
- Se recomienda profundizar la investigación en el clasificador bayesiano, en puntos como las capacidades de clasificación o las potencialidades que puede presentar en sinergia con otros clasificadores como Vecinos más cercanos, árboles de decisión entre otros, por los reducidos requerimientos computacionales y la capacidad de realizar aprendizaje automático a partir de la reducida cantidad de datos de entrada.
- Se recomienda extender las bases de datos para fines de reconocimiento de estados constructivos, asimismo, hacerla colaborativa de manera que se pueda ampliar de manera constante bajo la metodología propuesta por Peña y Murakami, 2017.
- Para el caso de la gestión territorial, se recomienda el análisis de las capacidades de las redes neuronales convolucionales, en aspectos sociales tales como, distribución de recursos hidráulicos para la producción eléctrica, análisis de riesgos de desastres naturales, gestión de recursos de agua, análisis sociodemográfico de las poblaciones, así también se propone para aspectos económicos como gestión de costes de proyectos de construcción, análisis de datos y clasificación, predicción de demanda de servicios públicos por zonas.
- Los modelos de Inteligencia Artificial tienen la posibilidad de desarrollarse como herramientas en la gestión del territorio, por la capacidad que poseen sus parámetros de resolver simulaciones de la realidad de alta complejidad, como las existentes en el análisis de capacidades de uso de suelos o la integración de la totalidad de variables que se deben tener en cuenta para un efectivo cuidado del medio ambiente, incluyendo las limitaciones que el territorio posee naturalmente.

## VII. BIBLIOGRAFÍA

- Acevedo, A., Schreier, C., & Seinfeld, C. (2018). PAPEL DEL ESTADO FRENTE A LA AUTO CONSTRUCCIÓN EN EL PERÚ, 1950-1968. *Paideia XXI*, 6(7), 220-223. doi: <https://doi.org/10.31381/paideia.v6i7.1610>
- Alfaro Malatesta, S. (2007). Análisis del proceso de autoconstrucción de la vivienda en Chile. Bases para la ayuda informática para los procesos comunicativos de soporte (Doctoral). Universitat Politècnica de Catalunya. Departament de Projectes d'Enginyeria.
- BERNARDI, C. (2009). Teledetección. Recuperado 15 agosto 2020, from <http://www.essa.ara.mil.ar/cens/MATERIAS%20TERCER%20A%C3%91O/3%C2%B0%20A%C3%91O/09-TELEDETECCION/PRIMER%20CUATRIMESTRE/PRIMER%20CUATRIMESTRE.pdf>
- Bravo, N. (2018). TELEDETECCIÓN ESPACIAL. Recuperado 15 agosto 2020, from [https://acolita.com/wp-content/uploads/2018/01/Teledeteccion\\_espacial\\_ArcGeek.pdf](https://acolita.com/wp-content/uploads/2018/01/Teledeteccion_espacial_ArcGeek.pdf)
- Brownlee, J. (2017). How Much Training Data is Required for Machine Learning?. Recuperado 24 noviembre 2020, from <https://machinelearningmastery.com/much-training-data-required-machine-learning/>
- Calderón Cockburn, J. (2018). POLÍTICA DE FORMALIZACIÓN COMO INSTRUMENTO DE INCLUSIÓN SOCIA. Presentation, Limia.
- Cao, C., Dragičević, S., & Li, S. (2019). Land-Use Change Detection with Convolutional Neural Network Methods. *Environments*, 6(2), 25. doi: 10.3390/environments6020025
- Catanzarite, J. (2020). The Naive Bayes Classifier. Recuperado 20 julio 2020, de <https://towardsdatascience.com/the-naive-bayes-classifier-e92ea9f47523>
- Ccanre, P., 2016. La Tierra Prometida. Las Invasiones A La Zona Agropecuaria De Villa El Salvador Y La Nueva Rinconada En San Juan De Miraflores. Lima, 2000. Licenciatura. Universidad Nacional Mayor de San Marcos.
- Chavarria, Y. (2017). Información geo-referencial y clasificación del caso [Blog]. Recuperado de <https://porlatierra.org/casos/140/georeferencial>

- Chethan, S. (2011). Spatial resolution of Google Earth Imagery [Blog]. Recuperado de <https://gis.stackexchange.com/questions/11395/spatial-resolution-of-google-earth-imagery>
- COFOPRI. (2016). Resolución Directoral N°149-2016-COFOPRI/DE (p. 4). Lima: Ministerio de Economía y Finanzas.
- COFOPRI. (2017). Resolución Directoral N°173-2017-COFOPRI/DE (pp. 3-8). Lima: Ministerio de Economía y Finanzas.
- COFOPRI. (2018). COFOPRI entregó título de propiedad N° 2,575,000 en Puente Piedra. Recuperado de <https://www.gob.pe/institucion/cofopri/noticias/187396-cofopri-entrego-titulo-de-propiedad-n-2-575-000-en-puente-piedra>
- COFOPRI. (2019). Resolución Directoral N°154-2019-COFOPRI/DE (p. 6). Lima: Ministerio de Economía y Finanzas.
- DiegoCalvo. (2018). Perceptrón Multicapa – Red Neuronal [Imagen]. Recuperado de <https://www.diegocalvo.es/perceptronmulticapa/#:~:text=Definici%C3%B3n%20del%20perceptr%C3%B3n%20Multicapa&text=El%20perceptr%C3%B3n%20multicapa%20esta%20compuesto,la%20entrada%20de%20la%20siguiente>. Carr, L. 2018. Multi-dimensional descriptions. 1 ed. s.l., s.e., p.1.
- Distrito.pe. (2020). Carabayllo en la region de Lima - Municipio y municipalidad de Perú - municipalidad Perú - Información municipalidad, ciudades y pueblos de Perú. Recuperado 22 julio 2020, from <https://www.distrito.pe/distrito-carabayllo.html>
- Dubois, P., Oliphant, T., Pérez, F., & Granger, B. (2007). Python: Batteries Included (3rd ed., pp. 20-45). cise.api.org.
- Gahukar, G. (2018). The Naive Bayes Classifier. Recuperado 20 julio 2020, de <https://towardsdatascience.com/the-naive-bayes-classifier-e92ea9f47523>
- Hernandez, J., & Marin, M. (2011). Clasificador Jerárquico de Imágenes utilizando Naive Bayes [Ebook] (1st ed., pp. 1-2). México: Instituto Nacional de Astrofísica Óptica y Electrónica. Recuperado de [https://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/2011/Reporte\\_proyecto\\_modelos.pdf](https://ccc.inaoep.mx/~esucar/Clases-mgp/Proyectos/2011/Reporte_proyecto_modelos.pdf)
- Hurwitz, J., & Kirsch, D. (2018). Machine Learning for dumies. IBM limited edition. (1st ed., pp. 4-17). John Wiley & sons.
- INEI. (2014). Una Mirada a Lima metropolitana (pp. 54-71). Lima: INEI. Recuperado de

[https://www.inei.gob.pe/media/MenuRecursivo/publicaciones\\_digitales/Est/Lib1168/libro.pdf](https://www.inei.gob.pe/media/MenuRecursivo/publicaciones_digitales/Est/Lib1168/libro.pdf)

- Jones, T. (2017). Arquitecturas de aprendizaje profundo. Recuperado 10 noviembre 2020, de <https://developer.ibm.com/es/articles/cc-machine-learning-deep-learning-architectures/>
- Kimosfo. (2016). Parallel python extract by mask ArcGIS. Recuperado 24 noviembre 2020, de <https://stackoverflow.com/questions/38788222/parallel-python-extract-by-mask-arcgis>
- LeCun, Y., Bengio, Y., & Hinton, G. (2020). (PDF) Deep Learning. Recuperado 10 noviembre 2020, de [https://www.researchgate.net/publication/277411157\\_Deep\\_Learning](https://www.researchgate.net/publication/277411157_Deep_Learning)
- Maps & COSMOS. (2020). Recuperado 9 noviembre 2020, de <https://www.surrey.ca/services-payments/online-services/maps-cosmos>
- Martínez Vega, J., & Martín Isabel, P. (2010). Guía Didáctica de Teledetección y Medio Ambiente. Recuperado 15 agosto 2020, de [http://www.aet.org.es/files/guia\\_teledeteccion\\_medio\\_ambiente.pdf](http://www.aet.org.es/files/guia_teledeteccion_medio_ambiente.pdf)
- Minitab. (2019). Estadísticos kappa y coeficientes de Kendall - Minitab. Recuperado 23 October 2020, de <https://support.minitab.com/es-mx/minitab/18/help-and-how-to/quality-and-process-improvement/measurement-system-analysis/supporting-topics/attribute-agreement-analysis/kappa-statistics-and-kendall-s-coefficients/>
- Murphy, K. (2012). Machine Learning A Probabilistic Perspective (1st ed., pp. 149-184). Londres: The MIT press.
- Oliphant, T. (2007). Python for Scientific Computing. Computing In Science & Engineering, 9(3), 10-20. doi: 10.1109/mcse.2007.58
- Pantoja, A. (2016). Recortar archivos Raster-Grid con #Python | El blog de franz. Recuperado 24 noviembre 2020, de <https://acolita.com/recortar-archivos-raster-grid-con-script-python/>
- Patel, S., & Pingel, J. (2017). Introduction to Deep Learning: What Are Convolutional Neural Networks? Video. Recuperado 10 noviembre 2020, de <https://la.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html>



- PEÑA-GUILLEN, V., & MURAKAMI, A. (2017). Dynamics of house state consolidation in Lima Metropolitan area: a cellular automata approach. *Journal of The Japanese Institute Of Landscape Architecture*, 80(5), 657-662. doi: 10.5632/jila.80.657
- Perez Valls, J. (2013). HERRAMIENTA MATLAB PARA LA SELECCION DE ENTRADAS Y PREDICCIÓN NEURONAL DE VALORES DE BOLSA (Grado). Universidad de Sevilla. Sharma, A. 2019. Artificial intelligence vs Machine Learning vs Deep Learning. s.e. Consultado 8 jul. 2019. Disponible en <https://www.geeksforgeeks.org/artificial-intelligence-vs-machine-learning-vs-deep-learning/> (GeeksforGeeks).
- Prakash, J. (2018). Breaking the curse of small datasets in Machine Learning: Part 1. Recuperado 24 noviembre 2020, de <https://towardsdatascience.com/breaking-the-curse-of-small-datasets-in-machine-learning-part-1-36f28b0c044d>
- Quilcate Tirado, J. (2019). Formalización de la propiedad informal y acceso a nueva vivienda. ¿Qué hacer? [Blog]. Recuperado de [https://latinamericantoplawyer.com/2019/07/26/peru-formalizacion-de-la-propiedad-informal-y-acceso-a-nueva-vivienda-que-hacer/#\\_ftn4](https://latinamericantoplawyer.com/2019/07/26/peru-formalizacion-de-la-propiedad-informal-y-acceso-a-nueva-vivienda-que-hacer/#_ftn4)
- Roman, V. (2019). Algoritmos Naive Bayes: Fundamentos e Implementación. Recuperado 18 julio 2020, de <https://medium.com/datos-y-ciencia/algoritmos-naive-bayes-fundamentos-e-implementaci%C3%B3n-4bcb24b307f>
- Rosser, J., Boyd, D., Long, G., Zakhary, S., Mao, Y., & Robinson, D. (2019). Predicting residential building age from map data. *Computers, Environment And Urban Systems*, 73, 56-67. doi: 10.1016/j.compenurbsys.2018.08.004
- Sarria, F. (2003). <https://www.um.es/geograf/sigmur/sigpdf/temario.pdf>. Recuperado 15 agosto 2020, de <https://www.um.es/geograf/sigmur/sigpdf/temario.pdf>
- Secretaría de Asentamientos Humanos y Obras Públicas. (1978). Glosario de términos sobre asentamientos humanos (pp. 20-21). México: Secretaría de Asentamientos Humanos y Obras Públicas.
- Sunil, R. (2017). Learn Naive Bayes Algorithm | Naive Bayes Classifier Examples. Recuperado 20 julio 2020, de <https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/#:~:text=In%20simple%20terms%2C%20a%20Naive,about%203%20inches%20in%20diameter.>

- Tokeshi, J., Noriega, C., Quiñónez, R., Neyra, L., Ginoccio, C., Bermy, G., & Rodriguez, G. (2005). Herramientas de análisis, para la densificación habitacional: Promesha-Cuaderno de análisis 11, 44.
- Universidad de León. (2006). ANÁLISIS VISUAL DE IMÁGENES (pp. 13-15). España.
- V, V. (2013). Rectifier Nonlinearities. Recuperado 10 noviembre 2020, de <https://imiloainf.wordpress.com/2013/11/06/rectifier-nonlinearities/>
- Wood, T. (2021). Softmax Function. DeepAI. Recuperado 1 febrero 2021, de <https://deepai.org/machine-learning-glossary-and-terms/softmax-layer>.

## VIII. ANEXOS

### 8.1. Cuadros de Lotes individuales

#### Manzana 1

Lote	CLASS	X (m.)	Y (m.)	Área (m2)	Perímetro (m.)
1	B	274603.50	8691357.65	82.80	39.80
2	B	274609.48	8691343.95	84.82	40.05
3	C	274615.31	8691345.81	105.83	52.39
4	B	274620.29	8691346.40	104.98	52.21
5	B	274627.83	8691344.68	214.54	62.51
6	B	274637.81	8691345.21	206.65	61.66
7	B	274647.86	8691347.78	214.96	62.31
8	B	274655.28	8691348.10	95.23	50.61
9	B	274660.19	8691346.87	109.21	51.94
10	A	274667.76	8691347.20	204.46	61.11
11	B	274677.73	8691350.56	210.91	61.92
12	A	274684.75	8691351.03	83.98	49.97
13	B	274689.59	8691348.89	119.49	53.57
14	B	274697.50	8691349.68	216.72	62.67
15	B	274707.60	8691352.78	206.68	61.54
16	C	274717.65	8691353.97	208.48	61.27
17	B	274727.58	8691351.11	202.35	60.92
18	B	274737.53	8691351.98	212.04	62.02
19	A	274747.19	8691355.32	191.44	60.10
20	B	274758.11	8691356.03	150.28	49.29
21	A	274758.53	8691354.76	65.80	36.40
22	B	274758.86	8691355.40	64.96	36.03
23	A	274762.78	8691349.60	68.91	34.66
24	B	274763.22	8691349.92	20.22	18.72
25	B	274763.55	8691356.35	35.01	23.67
26	B	274757.45	8691359.68	39.40	25.12
27	B	274756.57	8691370.08	93.83	41.84
28	C	274751.51	8691374.66	86.62	49.69
29	B	274746.81	8691354.45	108.01	51.75
30	B	274741.70	8691361.79	104.22	51.40
31	B	274736.72	8691376.14	100.50	51.05

32	B	274729.42	8691375.40	203.04	60.99
33	D	274719.39	8691382.65	212.92	61.96
34	B	274709.21	8691376.44	208.50	61.31
35	D	274699.33	8691372.59	192.79	59.33
36	B	274691.48	8691371.56	121.38	51.95
37	B	274686.49	8691374.52	76.69	47.57
38	A	274679.41	8691373.33	207.31	60.73
39	A	274669.34	8691369.93	196.64	59.71
40	A	274659.71	8691369.78	190.46	59.06
41	C	274652.25	8691371.11	109.21	50.77
42	C	274646.93	8691370.52	102.34	49.99
43	C	274641.74	8691367.12	104.32	50.13
44	C	274636.86	8691366.23	89.19	48.55
45	A	274632.18	8691369.04	96.72	49.27
46	C	274627.13	8691368.30	103.40	49.91
47	C	274619.59	8691368.45	194.96	59.20
48	B	274611.89	8691367.86	109.07	50.50
49	B	274605.90	8691365.94	128.82	52.62
50	B	274607.49	8691365.49	85.81	38.34

## Manzana 2

Lote	CLASS	Xcoord (m.)	YCoord (m.)	Área (m2)	Perímetro (m.)
1	B	277814.361	8688338.63	64.77	32.66
2	B	277823.942	8688346.96	93.23	48.20
3	B	277827.655	8688350.40	84.96	47.41
4	B	277832.993	8688353.58	164.31	55.75
5	A	277839.949	8688356.36	167.75	56.36
6	A	277846.567	8688360.59	151.32	54.96
7	B	277853.082	8688361.12	167.10	56.71
8	B	277858.951	8688361.65	128.43	52.91
9	C	277864.768	8688363.37	148.08	54.96
10	B	277871.544	8688365.88	190.41	59.12
11	C	277876.647	8688366.67	105.81	41.25
12	C	277882.498	8688370.38	85.49	38.53
13	A	277892.157	8688371.70	233.61	65.96
14	A	277881.496	8688376.46	325.73	73.23
15	C	277872.094	8688376.86	133.84	53.51
16	C	277865.804	8688381.49	163.53	56.00
17	A	277859.137	8688384.14	150.51	54.49
18	C	277852.381	8688385.86	178.09	57.72
19	A	277843.562	8688389.69	257.63	66.03
20	A	277835.235	8688394.59	176.02	53.49
21	B	277828.865	8688395.78	131.96	47.27
22	B	277820.18	8688403.98	119.30	44.16

## Manzana 03

Lote	CLASS	Xcoord (m.)	YCoord (m.)	Área (m2)	Perímetro (m.)
1	A	277847.37	8688308.10	172.89	56.71
2	B	277855.14	8688314.02	202.85	59.94
3	C	277862.43	8688332.54	154.96	54.97
4	C	277868.44	8688334.55	135.97	53.46
5	C	277873.58	8688325.88	116.73	51.99
6	B	277878.25	8688329.26	117.60	51.99
7	D	277883.24	8688342.28	133.99	53.72
8	B	277887.70	8688350.53	95.45	50.15
9	B	277892.24	8688336.99	125.87	53.46
10	B	277897.19	8688340.80	127.76	53.90
11	C	277902.04	8688355.30	121.80	53.34
12	B	277906.00	8688351.72	88.26	50.10
13	B	277916.53	8688362.07	130.15	45.88
14	A	277923.02	8688358.87	134.07	46.58
15	B	277912.09	8688341.54	129.60	53.65
16	A	277907.17	8688338.36	121.18	52.65
17	A	277903.21	8688349.08	69.23	47.59
18	A	277899.98	8688345.77	83.58	49.16
19	D	277895.77	8688324.47	133.92	54.10
20	B	277890.78	8688319.18	119.74	52.75
21	B	277886.22	8688334.39	115.22	52.40
22	C	277881.14	8688330.29	153.06	56.86
23	A	277874.89	8688364.50	172.21	59.05
24	D	277867.58	8688302.80	231.04	64.72
25	A	277859.73	8688318.36	193.99	61.71
26	B	277909.92	8688322.38	105.78	51.78

#### Manzana 04

Lote	CLASS	Xcoord (m.)	YCoord (m.)	Área (m2)	Perímetro (m.)
1	A	278805.84	8689074.84	191.65	59.73
2	B	278811.88	8689069.73	135.66	53.89
3	C	278817.16	8689065.83	136.24	53.98
4	A	278821.87	8689061.48	117.76	52.17
5	C	278827.06	8689057.88	148.41	55.27
6	B	278832.20	8689053.08	127.66	53.25
7	B	278837.26	8689049.18	134.60	53.97
8	C	278842.36	8689044.68	138.74	54.41
9	B	278847.46	8689040.63	130.84	53.67
10	B	278852.25	8689036.73	119.05	52.55
11	A	278857.34	8689032.53	152.43	55.85
12	A	278862.70	8689028.33	133.37	54.03
13	B	278867.81	8689023.83	139.96	54.71
14	C	278854.42	8689007.47	150.03	56.53
15	A	278849.21	8689012.27	138.54	55.34
16	B	278844.05	8689016.47	150.56	56.37

17	A	278838.83	8689020.68	126.73	53.99
18	A	278834.07	8689024.73	134.19	54.60
19	B	278828.91	8689029.08	146.22	55.65
20	A	278821.30	8689035.53	266.92	67.18
21	A	278813.61	8689041.83	146.69	55.39
22	C	278808.58	8689046.18	121.88	52.87
23	C	278803.92	8689050.38	131.00	53.65
24	C	278798.66	8689054.28	142.05	54.66
25	A	278792.46	8689059.68	191.53	59.30

Manzana 05

Lote	CLASS	Xcoord (m.)	YCoord (m.)	Área (m2)	Perímetro (m.)
1	C	278838.26	8689043.51	202.53	59.83
2	C	278844.94	8689038.74	133.78	52.79
3	C	278850.33	8689032.88	138.49	53.26
4	C	278855.70	8689039.60	128.77	52.26
5	C	278860.80	8689025.07	128.05	52.18
6	B	278866.34	8689019.43	145.56	53.97
7	A	278871.92	8689053.70	137.65	53.16
8	B	278877.48	8689058.47	139.69	53.36
9	B	278883.22	8689045.03	146.62	54.07
10	B	278888.75	8689049.58	130.60	52.42
11	D	278893.77	8689070.84	122.47	51.58
12	B	278898.99	8689075.18	138.89	53.26
13	A	278904.41	8689062.38	130.36	52.38
14	B	278909.58	8689066.72	130.72	52.42
15	A	278915.07	8689086.76	139.55	53.32
16	B	278920.56	8689088.84	133.95	52.74
17	A	278927.55	8689082.31	217.08	61.26
18	A	278935.88	8689092.96	189.45	58.60
19	A	278922.97	8689095.49	139.01	52.37
20	A	278915.50	8689079.51	205.44	59.39
21	B	278908.70	8689091.04	122.20	50.46
22	A	278903.43	8689084.28	129.71	51.24
23	B	278897.90	8689105.76	131.29	51.38
24	B	278892.59	8689111.18	124.85	50.66
25	B	278887.08	8689097.08	137.06	51.94
26	B	278881.70	8689101.85	119.79	50.06
27	B	278876.61	8689069.29	124.20	50.51
28	B	278871.11	8689064.77	138.02	51.97
29	B	278865.40	8689078.18	132.92	51.39
30	A	278859.78	8689074.06	132.92	51.36
31	C	278854.27	8689052.18	131.47	51.18
32	A	278848.80	8689047.63	124.18	50.36
33	C	278843.64	8689059.99	121.37	50.03
34	A	278838.21	8689056.30	134.09	51.38
35	A	278832.77	8689035.05	124.52	50.32

36	A	278827.34	8689030.71	132.71	51.18
----	---	-----------	------------	--------	-------

### Manzana 08

Lote	CLASS	Xcoord (m.)	YCoord (m.)	Área (m2)	Perímetro (m.)
1	C	278758.42	8688975.77	89.40	41.08
2	C	278765.18	8688971.21	120.86	54.87
3	A	278769.09	8688948.91	140.28	56.40
4	A	278773.31	8688944.55	125.44	54.78
5	C	278777.37	8688957.71	134.86	55.40
6	C	278781.44	8688953.21	124.59	54.19
7	C	278785.53	8689012.23	122.10	53.70
8	C	278789.19	8689007.57	118.51	53.11
9	C	278793.31	8688937.54	132.17	54.17
10	B	278797.41	8688946.80	117.91	52.54
11	C	278801.08	8688994.18	109.87	51.52
12	B	278805.18	8688989.74	133.31	53.59
13	B	278809.39	8689002.86	121.22	52.13
14	B	278813.82	8688998.52	143.25	54.09
15	A	278769.75	8688976.77	97.30	44.42
16	C	278780.36	8688972.28	70.81	36.10
17	C	278781.03	8688985.35	117.15	53.05
18	B	278784.80	8688981.06	126.76	54.09
19	A	278788.85	8688958.78	123.33	53.89
20	A	278792.88	8688957.04	133.03	54.93
21	A	278796.87	8688967.88	122.57	54.08
22	B	278801.10	8688963.33	137.94	55.64
23	B	278804.87	8688988.73	116.52	53.75
24	C	278808.94	8688982.25	139.99	56.06
25	A	278814.83	8688998.59	243.01	65.75
26	C	278820.88	8688993.89	148.66	57.21
27	B	278825.12	8688966.18	132.36	55.85
28	B	278829.44	8688962.21	154.31	57.98

### Manzana 10

Lote	CLASS	Xcoord (m.)	YCoord (m.)	Área (m2)	Perímetro (m.)
1	A	273840.51	8689803.16	93.39	43.28
2	C	273847.15	8689805.94	151.14	56.90
3	A	273853.99	8689811.89	242.04	65.37
4	B	273860.92	8689818.08	156.81	57.34
5	B	273865.76	8689822.37	118.02	53.67
6	B	273870.20	8689826.58	136.28	55.35
7	B	273874.70	8689830.78	122.47	54.02
8	B	273879.12	8689834.67	126.38	54.36
9	B	273883.48	8689838.56	125.32	54.24

10	B	273887.83	8689842.21	121.61	53.85
11	B	273892.36	8689846.42	134.90	55.08
12	B	273897.12	8689850.79	141.28	55.65
13	D	273902.38	8689855.47	153.74	56.79
14	A	273850.33	8689791.49	107.30	44.41
15	A	273858.74	8689781.49	92.27	39.00
16	C	273861.30	8689790.30	149.25	55.80
17	B	273866.10	8689794.99	120.27	53.05
18	A	273870.42	8689798.40	116.21	52.70
19	A	273875.15	8689802.53	151.01	56.11
20	B	273880.00	8689806.97	117.90	52.98
21	A	273884.53	8689810.94	135.61	54.73
22	B	273889.02	8689815.15	118.44	53.13
23	B	273893.50	8689819.04	129.51	54.25
24	B	273897.93	8689823.08	122.57	53.64
25	B	273902.25	8689826.89	120.49	53.48
26	B	273906.84	8689831.02	136.70	55.08
27	D	273911.47	8689835.39	131.52	54.64
28	B	273916.54	8689839.83	151.38	56.56

Manzana 11

Lote	CLASS	Xcoord (m.)	YCoord (m.)	Área (m2)	Perímetro (m.)
1	A	273922.18	8689898.46	212.76	59.72
2	A	273929.19	8689893.50	131.95	50.71
3	A	273934.14	8689865.54	108.77	48.09
4	B	273938.85	8689860.43	123.66	49.70
5	A	273944.93	8689876.66	177.44	55.66
6	B	273951.31	8689871.10	131.60	50.51
7	B	273956.05	8689847.35	99.29	46.87
8	A	273960.03	8689919.95	99.90	46.91
9	A	273964.65	8689856.37	120.51	49.19
10	B	273969.55	8689851.56	113.29	48.35
11	B	273974.20	8689906.13	120.04	49.08
12	B	273979.16	8689901.62	115.60	48.54
13	B	273984.12	8689914.84	129.24	50.05
14	B	273930.61	8689910.33	88.66	42.53
15	B	273935.09	8689889.14	97.85	43.71
16	A	273940.11	8689883.43	116.48	46.09
17	B	273945.24	8689897.56	100.06	43.98
18	C	273950.18	8689893.50	110.54	45.33
19	C	273956.23	8689868.70	148.66	50.24
20	D	273962.54	8689862.08	117.57	46.26
21	B	273968.33	8689877.72	131.19	48.03
22	B	273974.42	8689873.21	124.71	47.21
23	B	273980.37	8689907.78	130.30	47.93
24	B	273985.70	8689902.67	100.18	44.09
25	B	273990.63	8689887.79	106.61	44.93



26	B	273995.58	8689881.77	108.43	45.17
----	---	-----------	------------	--------	-------

Manzana 12

Lote	CLASS	Xcoord (m.)	YCoord (m.)	Área (m2)	Perímetro (m.)
1	D	277984.49	8685500.43	341.16	78.74
2	A	277989.36	8685508.24	132.21	58.33
3	A	277994.96	8685515.11	273.11	70.35
4	C	278000.81	8685521.98	161.38	61.23
5	C	278005.33	8685526.81	147.67	60.12
6	D	278011.01	8685533.49	277.53	71.31
7	C	278016.65	8685540.30	145.00	60.56
8	A	278020.98	8685544.80	171.35	63.84
9	D	278030.46	8685556.04	563.96	95.09
10	A	278040.92	8685567.29	191.24	67.58
11	A	278046.89	8685572.38	294.95	75.06
12	C	278052.36	8685581.18	169.36	65.12
13	A	278056.33	8685585.68	145.65	63.02
14	D	278060.14	8685590.37	164.10	64.17
15	C	278065.88	8685597.32	298.22	74.48
16	C	278071.71	8685604.26	162.59	64.03
17	C	278075.80	8685608.83	152.07	63.58
18	B	278079.43	8685613.59	158.67	63.49
19	C	278085.80	8685619.61	314.13	76.85
20	C	277999.58	8685487.65	167.62	51.81
21	C	278008.80	8685479.97	139.35	47.35
22	C	278008.52	8685492.01	149.32	65.20
23	C	278012.27	8685496.78	175.43	67.11
24	C	278016.46	8685500.68	138.61	64.36
25	B	278019.98	8685505.24	170.62	66.41
26	D	278024.25	8685510.07	177.43	66.90
27	B	278028.10	8685514.97	149.41	64.66
28	D	278032.18	8685518.93	156.64	64.62
29	D	278035.90	8685523.76	157.51	64.83
30	D	278040.02	8685528.26	164.74	64.19
31	B	278044.12	8685532.49	141.38	62.60
32	D	278047.08	8685537.12	138.94	62.87
33	D	278051.24	8685541.49	180.25	66.51
34	A	278055.51	8685546.39	164.40	65.77
35	A	278060.35	8685550.82	166.34	62.22
36	C	278064.18	8685555.65	132.87	59.79
37	B	278068.21	8685559.75	150.45	61.51
38	C	278071.80	8685564.77	152.54	61.96
39	C	278075.69	8685569.27	143.46	61.53
40	B	278079.58	8685573.77	164.22	63.48
41	B	278082.97	8685579.00	136.64	61.72
42	B	278087.27	8685582.77	157.62	63.41
43	B	278091.07	8685587.73	156.50	62.97

44	D	278095.15	8685592.09	149.59	62.08
45	C	278098.72	8685596.85	149.71	62.72
46	D	278105.07	8685603.27	295.72	73.05

### Manzana 13

Lote	CLASS	Xcoord (m.)	YCoord (m.)	Área (m2)	Perímetro (m.)
1	D	278064.36	8685522.78	156.43	64.10
2	C	278068.00	8685514.31	140.34	62.14
3	D	278071.99	8685490.08	147.31	62.72
4	D	278075.85	8685485.63	164.30	64.19
5	D	278080.16	8685498.44	165.00	64.36
6	C	278084.22	8685493.46	149.52	63.28
7	B	278087.43	8685568.82	134.46	62.21
8	B	278091.84	8685564.37	179.93	65.81
9	D	278097.93	8685579.61	300.66	75.17
10	A	278105.02	8685573.47	301.96	73.54
11	D	278111.14	8685548.71	151.03	61.53
12	C	278115.34	8685538.97	149.13	61.63
13	C	278119.47	8685560.14	139.61	61.54
14	D	278123.25	8685555.91	165.72	63.84
15	C	278127.44	8685523.52	157.27	63.40
16	A	278133.05	8685518.76	271.41	72.58
17	A	278083.50	8685532.30	127.85	56.66
18	A	278086.72	8685528.18	114.09	56.28
19	C	278090.64	8685505.53	142.88	58.84
20	B	278094.64	8685501.08	145.07	59.04
21	D	278098.96	8685514.10	152.07	59.67
22	C	278103.16	8685509.66	140.07	58.68
23	D	278108.80	8685557.28	264.95	69.41
24	C	278116.68	8685552.52	280.83	70.87
25	D	278121.90	8685563.53	163.91	63.60
26	B	278126.08	8685535.37	152.55	62.57
27	A	278132.18	8685530.82	281.54	72.67
28	D	278138.25	8685547.65	155.61	62.09
29	A	278142.35	8685541.51	160.36	62.39
30	A	278146.52	8685507.86	148.14	61.31
31	C	278152.13	8685502.99	260.03	70.31

## 8.2. Código para Modelo CNN

```
>>import numpy as np
>>import keras
>>from keras.models import Sequential
```

```

>>from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
>>from keras.layers.normalization import BatchNormalization
>>from keras.preprocessing.image import ImageDataGenerator
>>from keras.callbacks import ReduceLROnPlateau
>>import itertools

>>img_width, img_height = 224, 224
>>batch_size = 1
>>num_classes = 4
>>epochs = 5
>>input_shape = (img_width, img_height, 3)
>>train_data_dir = 'training_set'
>>test_data_dir = 'test_set'
>>valid_data_dir = 'valid_set'

>>model = Sequential ()
>>model.add(Conv2D(32,kernel_size=(3,3),activation='relu',kernel_initializer='he_normal'
,input_shape=input_shape))
>>model.add(Conv2D(32,kernel_size=(3,3),activation='relu',kernel_initializer='he_normal'
))
>>model.add (MaxPool2D ((2, 2)))
>>model.add (Dropout (0.20))
>>model.add(Conv2D(64,(3,3),activation='relu',padding='same',kernel_initializer='he_nor
mal'))
>>model.add(Conv2D(64,(3,3),activation='relu',padding='same',kernel_initializer='he_nor
mal'))
>>model.add (MaxPool2D (pool_size= (2, 2)))
>>model.add(Dropout(0.25))
>>model.add(Conv2D(128,(3,3),activation='relu',padding='same',kernel_initializer='he_no
rmal'))
>>model.add (Dropout (0.25))

>>model.compile(loss=keras.losses.binary_crossentropy,

```

```

optimizer=keras.optimizers.RMSprop(),
metrics=['accuracy'])
>>learning_rate_reduction = ReduceLROnPlateau(monitor='val_acc',
patience=3,
verbose=1,
factor=0.5,
min_lr=0.0001)

>>datagen = ImageDataGenerator(
rescale=1. / 255,
featurewise_center=False,
samplewise_center=False,
featurewise_std_normalization=False,
samplewise_std_normalization=False,
zca_whitening=False,
rotation_range=15,
zoom_range = 0.1,
width_shift_range=0.1,
height_shift_range=0.1,
horizontal_flip=False
vertical_flip=False,
validation_split=0.1)

>>train_generator = datagen.flow_from_directory(
train_data_dir,
target_size=(img_width, img_height),
batch_size=batch_size,
class_mode='categorical')
>>test_generator = datagen.flow_from_directory(
test_data_dir,
target_size=(img_width, img_height),
batch_size=batch_size,
class_mode='categorical')

```

```

>>valid_set = datagen.flow_from_directory(valid_data_dir,
                                         target_size = (img_width, img_height),
                                         batch_size = batch_size,
                                         class_mode = 'categorical')

>>h=model.fit_generator(
    train_generator,
        steps_per_epoch=77//batch_size,
    epochs=epochs,
        verbose=1,
    validation_data=valid_set,
    validation_steps=20,
        callbacks=[learning_rate_reduction])

>>test_generator.reset()
>>Y_pred = model.predict_generator(test_generator, steps=20)
>>classes = test_generator.classes[test_generator.index_array]
>>y_pred = np.argmax(Y_pred, axis=-1)
>>classes = test_generator.classes[test_generator.index_array]

>>print(classes)

>>scores = model.evaluate_generator(test_generator, steps=20, workers=1)

>>print(scores)

```

### 8.2.1. Matriz de confusion Modelo CNN

```

>>import matplotlib.pyplot as plt

>>def plot_confusion_matrix(cm, classes,
                            normalize=False,

```

```

        title="Confusion matrix",

        cmap=plt.cm.Blues):

plt.imshow(cm, interpolation='nearest',cmap=cmap)

plt.title(title)

plt.colorbar()

tick_marks= np.arange(len(classes))

plt.xticks(tick_marks, classes, rotation=45)

plt.yticks(tick_marks, classes)

    if normalize:

        cm= cm.astype('float')/ cm.sum(axis=1)[:, np.newaxis]

        print("Normalized confusion matrix")

    else:

        print('Confusion matrix, without normalization')

print(cm)

    thresh = cm.max()/2.

for i,j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):

    plt.text(j, i, cm[i, j],

             horizontalalignment="center",

             color="white" if cm[i, j]>thresh else "black" )

```

```
plt.tight_layout()

plt.ylabel('True label')

plt.xlabel('Predicted label')

>>from sklearn.metrics import confusion_matrix

>>cm = confusion_matrix(classes,y_pred)

>>cm_plot_labels = ['A' , 'B' , 'C' , 'D']

>>plot_confusion_matrix(cm, cm_plot_labels, title="Confusion matrix")

>>from sklearn.metrics import cohen_kappa_score

>>cohen_kappa_score(classes, y_pred)
```

### **8.3. Modelo Bayesiano**

```
>>from PIL import Image

>>import numpy as np

>>import glob

>>from sklearn.naive_bayes import *

>>from sklearn.model_selection import *

>>from sklearn.decomposition import PCA

>>from keras.layers import Flatten

>>from matplotlib import pyplot as plt

>>import itertools
```

```

>>import random

>>def load_image (infilename):

    img = Image.open(infilename)

    data = np.asarray(img.resize((96,96)), dtype=np.float64)

    data = data.ravel()

    return data

>>def my_trasform(data, X_t, components):

    return np.dot(data-X_t.mean_, components.T)

>>folders = ['A' , 'B' , 'C' , 'D']

>>X = []

>>Y = []

>>for folder in folders:

    for infile in glob.glob(folder + "\*.jpg"):

        img = load_image(infile)

        X.append(img)

        Y.append(folder)

>>X = np.array(X)

>>Y = np.array(Y)

```



```

>>random.seed(0)

>>X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=30)

>>clf = GaussianNB()

>>clf.fit(X_train, Y_train)

>>print("Predicted labels on test set:\n")

>>print(clf.predict(X_test))

>>print("\nAccuracy = ' + str(int(clf.score(X_test, Y_test)*100)) + '%')

```

### 8.3.1. Matriz de confusión Modelo Bayesiano

```

>>def plot_confusion_matrix(cm, classes,
                            normalize=False,
                            title="Confusion matrix",
                            cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest',cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks= np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm= cm.astype('float')/ cm.sum(axis=2)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

```

```

thresh = cm.max()/2.
for i,j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j],
             horizontalalignment="center",
             color="white" if cm[i, j]>thresh else "black" )
plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')

>>y_true= Y_test
>>y_pred= clf.predict(X_test)
>>cnf_matrix = confusion_matrix(Y_test, clf.predict(X_test),labels=['A' , 'B' , 'C' , 'D'])
>>np.set_printoptions(precision=2)

>>plt.figure()
>>plot_confusion_matrix(cnf_matrix, classes=['A' , 'B' , 'C' , 'D'],
                        title='Confusion matrix, without normalization')

>>from sklearn.metrics import cohen_kappa_score

>>cohen_kappa_score(y_true, y_pred)

```