UNIVERSIDAD UTE

FACULTAD DE CIENCIAS DE LA INGENIERÍA E INDUSTRIAS

CARRERA DE INGENIERÍA INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

DETERMINACIÓN EXPERIMENTAL DEL PIPELINE ÓPTIMO DE TRANSFORMACIÓN Y CLASIFICACIÓN DE IMÁGENES SOBRE EL DATASET PKLOT.

TRABAJO PREVIO A LA OBTENCIÓN DEL TÍTULO

DE INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN

RONALD DANIEL GÓMEZ LARA

DIRECTOR: DR. DIEGO ORDÓÑEZ

Quito, febrero 2020

FORMULARIO DE REGISTRO BIBLIOGRÁFICO
TRABAJO DE TITULACIÓN

DATOS DE CONTACTO	
CÉDULA DE IDENTIDAD:	0920202835
APELLIDO Y NOMRES:	GÓMEZ LARA RONALD DANIEL
DIRECCIÓN:	LOS GERANIOS Oe 3-120, LAS AVELLANAS, PICHINCHA - QUITO
EMAIL:	ronald93_@hotmail.com
TELÉFONO FIJO:	022032807
TELÉFONO MOVIL:	0961412871

DATOS DE LA OBRA	
TÍTULO	DETERMINACIÓN EXPERIMENTAL DEL PIPELINE ÓPTIMO DE TRANSFORMACIÓN Y CLASIFICACIÓN DE IMÁGENES SOBE EL DATASET PKLOT
AUTOR O AUTORES:	GÓMEZ LARA RONALD DANIEL
FECHA DE ENTREGA DEL PROYECTO DE TITULACIÓN:	21 de febrero de 2020
DIRECTOR DEL PROYECTO DE TITULACIÓN:	DIEGO ORDÓÑEZ
PROGRAMA:	PREGRADO [X] POSGRADO [ ]
TÍTULO POR EL QUE OPTA:	INGENIERO EN INFORMÁTICA Y CIENCIAS DE LA COMPUTACIÓN
RESUMEN:	En este trabajo se estableció como objetivo principal encontrar el mejor modelo predictivo procesando el *dataset* PKLot mediante técnicas de *machine learning*, para determinar

	automáticamente y en tiempo real las plazas de parqueo disponibles en un parqueadero institucional. Se siguió una metodología experimental en donde se seleccionó los datos relacionados de acuerdo con el entorno en el que se trabaja, estos datos son manipulados para lograr obtener la solución. En principio, escoger una muestra fue esencial para obtener resultados válidos en cada experimentación realizada. Estos resultados dependieron de las técnicas elegidas, asegurando que el experimento se realizó correctamente y que los resultados obtenidos son un reflejo de lo que es el mundo real, ayudando a la sociedad a mejorar su vida de la mejor manera posible. Para determinar definitivamente con qué algoritmos se trabajó, se utilizó el *dataset* más pequeño, debido a que los algoritmos deberían ser capaces de clasificar los *dataset* en el menor tiempo posible, mostrando como resultado que 16 algoritmos fueron efectivos. Por último, al combinar los 3 mejores filtros no fue mejor que al utilizar un solo filtro, como se observó no se halló una mejora significativa con relación al utilizar un solo filtro. Se tomó la decisión de mantener el filtro gris, una vez que se finalizó la experimentación se obtuvo el mejor *dataset*, el mejor algoritmo y el mejor filtro, como el mismo *dataset* PKLot con filtro Gris se creó el modelo, y por último se integró en el proyecto DAPLP. En conclusión, se obtuvo el mejor modelo predictivo con un 99.41% de clasificación, capaz de procesar cualquier cantidad de datos, determinando

	automáticamente en tiempo real las plazas de parqueo disponibles en una institución.
PALABRAS CLAVES:	Java, Marvin Plugins, Weka, Aprendizaje Automático, Algoritmos de Clasificación.
ABSTRACT:	In this work, the main objective was to find the best predictive model by processing the PKLot dataset using machine learning techniques, in order to determine automatically and in real time the available parking spaces in an institutional parking lot. An experimental methodology was followed where the related data was selected according to the environment in which we work, these data are manipulated to obtain the solution. In principle, choosing a sample was essential to obtain valid results in each experiment carried out. These results depended on the chosen techniques, ensuring that the experiment was performed correctly and that the results obtained are a reflection of what the real world is like, helping society to improve its life in the best possible way. To determine definitively which algorithms were used, the smallest dataset was used, because the algorithms should be able to classify the datasets in the shortest time possible, showing as a result that 16 algorithms were effective. Finally, combining the 3 best filters was not better than using a single filter, as observed there was no significant improvement over using a single filter. The decision was taken to keep the grey filter, once the experiment was finished the best dataset, the best algorithm and the best filter were obtained, as the same PKLot dataset with the grey filter was

	created, and finally it was integrated in the DAPLP project. In conclusion, the best predictive model was obtained with a 99.41% classification, capable of processing any amount of data, automatically determining in real time the available parking spaces in an institution.
KEYWORDS:	Java, Marvin Plugins, Weka, Machine Learning, Classification Algorithms.

Se autoriza la publicación de este Proyecto de Titulación en el Repositorio Digital de la Institución.

f: _____

GÓMEZ LARA RONALD DANIEL

0920202835

DECLARACIÓN Y AUTORIZACIÓN

Yo, **GÓMEZ LARA RONALD DANIEL,** CI 0920202835 autor/a del trabajo de titulación: **DETERMINACIÓN EXPERIMENTAL DEL PIPELINE ÓPTIMO DE TRANSFORMACIÓN Y CLASIFICACIÓN DE IMÁGENES SOBRE EL DATASET PKLOT,** previo a la obtención del título de **Ingeniero en Informática y Ciencias de la Computación** en la Universidad UTE.

1. Declaro tener pleno conocimiento de la obligación que tienen las Instituciones de Educación Superior, de conformidad con el Artículo 144 de la Ley Orgánica de Educación Superior, de entregar a la SENESCYT en formato digital una copia del referido trabajo de titulación de grado para que sea integrado al Sistema Nacional de información de la Educación Superior del Ecuador para su difusión pública respetando los derechos de autor.
2. Autorizo a la BIBLIOTECA de la Universidad UTE a tener una copia del referido trabajo de titulación de grado con el propósito de generar un Repositorio que democratice la información, respetando las políticas de propiedad intelectual vigentes.

Quito, 21 de febrero de 2020

f: _____

GÓMEZ LARA RONALD DANIEL

0920202835

CERTIFICACIÓN DEL TUTOR

En mi calidad de tutor, certifico que el presente trabajo de titulación que lleva por título **DETERMINACIÓN EXPERIMENTAL DEL PIPELINE ÓPTIMO DE TRANSFORMACIÓN Y CLASIFICACIÓN DE IMÁGENES SOBRE EL DATASET PKLOT,** para aspirar al título de **Ingeniero en Informática y Ciencias de la Computación** fue desarrollado por **GÓMEZ LARA RONALD DANIEL,** bajo mi dirección y supervisión, en la Facultad de Ciencias de la Ingeniería e Industrias; y que dicho trabajo cumple con las condiciones requeridas para ser sometido a las evaluación respectiva de acuerdo a la normativa interna de la Universidad UTE.

Diego Ordóñez Camacho

DIRECTOR DEL TRABAJO

C.I. 1710449016

ÍNDICE DE CONTENIDOS

ÍNDICE DE TABLAS

PÁGINA

ÍNDICE DE FIGURAS

PÁGINA

RESUMEN

En este trabajo se estableció como objetivo principal encontrar el mejor modelo predictivo procesando el *dataset* PKLot mediante técnicas de *machine learning*, para determinar automáticamente y en tiempo real las plazas de parqueo disponibles en un parqueadero institucional. Se siguió una metodología experimental en donde se seleccionó los datos relacionados de acuerdo con el entorno en el que se trabaja, estos datos son manipulados para lograr obtener la solución. En principio, escoger una muestra fue esencial para obtener resultados válidos en cada experimentación realizada. Estos resultados dependieron de las técnicas elegidas, asegurando que el experimento se realizó correctamente y que los resultados obtenidos son un reflejo de lo que es el mundo real, ayudando a la sociedad a mejorar su vida de la mejor manera posible. Para determinar definitivamente con qué algoritmos se trabajó, se utilizó el *dataset* más pequeño, debido a que los algoritmos deberían ser capaces de clasificar los *dataset* en el menor tiempo posible, mostrando como resultado que 16 algoritmos fueron efectivos. Por último, al combinar los 3 mejores filtros no fue mejor que al utilizar un solo filtro, como se observó no se halló una mejora significativa con relación al utilizar un solo filtro. Se tomó la decisión de mantener el filtro gris, una vez que se finalizó la experimentación se obtuvo el mejor *dataset*, el mejor algoritmo y el mejor filtro, como el mismo *dataset* PKLot con filtro Gris se creó el modelo, y por último se integró en el proyecto DAPLP. En conclusión, se obtuvo el mejor modelo predictivo con un 99.41% de clasificación, capaz de procesar cualquier cantidad de datos, determinando automáticamente en tiempo real las plazas de parqueo disponibles en una institución.

Palabras clave: Java, Marvin Plugins, Weka, Aprendizaje Automático, Algoritmos de Clasificación.

ABSTRACT

In this work, the main objective was to find the best predictive model by processing the PKLot dataset using machine learning techniques, in order to determine automatically and in real time the available parking spaces in an institutional parking lot. An experimental methodology was followed where the related data was selected according to the environment in which we work, these data are manipulated to obtain the solution. In principle, choosing a sample was essential to obtain valid results in each experiment carried out. These results depended on the chosen techniques, ensuring that the experiment was performed correctly and that the results obtained are a reflection of what the real world is like, helping society to improve its life in the best possible way. To determine definitively which algorithms were used, the smallest dataset was used, because the algorithms should be able to classify the datasets in the shortest time possible, showing as a result that 16 algorithms were effective. Finally, combining the 3 best filters was not better than using a single filter, as observed there was no significant improvement over using a single filter. The decision was taken to keep the grey filter, once the experiment was finished the best dataset, the best algorithm and the best filter were obtained, as the same PKLot dataset with the grey filter was created, and finally it was integrated in the DAPLP project. In conclusion, the best predictive model was obtained with a 99.41% classification, capable of processing any amount of data, automatically determining in real time the available parking spaces in an institution.

Keywords: Java, Marvin Plugins, Weka, Machine Learning, Classification Algorithms.

1. INTRODUCCIÓN

1. INTRODUCCIÓN

Llegar a un lugar determinado a fin de conseguir estacionarse en las limitadas zonas de parqueos disponibles en las grandes ciudades es uno de los grandes problemas que padecen las personas hoy en día, cuando buscan plazas libres de parqueo (Cheng & Qiao, 2016).

El problema se observa en las grandes y pequeñas instituciones públicas o privadas, haciendo que el asunto se agrave, debido a todo el tiempo que le toma a la persona llegar a la institución obteniendo como resultado que no se encuentren plazas de parqueo disponibles a simple vista (Lin, Rivano, & Le Mouel, 2017).

La persona empieza a movilizarse por el parqueadero buscando una plaza de parqueo libre, tomando en consideración los vehículos y personas que circulan por esa zona, evitando en todo lo posible originar un congestionamiento vehicular, que ocasionaría accidentes (Ordóñez, Gómez, & Ávalos, 2018).

Otro efecto sería la pérdida de tiempo cuando se buscan plazas libres en todas las zonas de parqueo disponibles en la institución, en consecuencia, esto podría molestar a los maestros o estudiantes que tienen prisa por llegar a sus actividades diarias (Mancilla, Ibañez, & Castillo, 2014).

Se podría prevenir todo esto si hubiera información disponible, sobre las plazas de parqueo libres, un modo de obtener la información sobre que plazas están disponibles sería a través del internet con el fin de ahorrarnos tiempo e ir directamente al sitio.

Desafortunadamente, en la actualidad no existen sistemas que funcionen de esa manera, no son capaces de realizar esta tarea, los sensores al ingresar a los parqueaderos solo indican que hay plazas disponibles (Taipe, 2018).

Una alternativa para la detección de plazas disponibles sería crear sistemas que tengan acceso a cámaras e implementen metodologías que incluyan algoritmos que clasifiquen las imágenes, teniendo sistemas más seguros y de fácil manejo que puedan conectarse a su dispositivo (Nieto, 2018).

Actualmente, existen soluciones atrayentes que ayudan a solucionar el problema, una de las soluciones es usar el internet como medio de interacción, es una de las tecnologías más calificadas, capaz de conectar al mundo y en unión con la inteligencia artificial, lograr un extenso campo de aplicaciones innovadoras o sistemas inteligentes (Alvear-Puertas, Rosero-Montalvo, Peluffo-Ordóñez, & Pijal-Rojas, 2017).

Las aplicaciones actuales para detectar plazas de parqueos libres utilizan sensores de muy alto costo en instalación y mantenimiento, y dejan de ser de gran interés, en comparación con las soluciones orientadas a la vista, en las cuales se buscar explotar las cámaras de vigilancia, para la detección de

parqueos libre, debido a su bajo costo en instalación y mantenimiento (Masmoudi, Wali, Jamoussi, & Alimi, 2014).

Los sistemas inteligentes existentes son capaces de detectar vehículos capturándolos en múltiples imágenes a través de un servidor, utilizan diferentes tipos de métodos para hallar soluciones que detecten las plazas de parqueo libres (Badii, Nesi, & Paoli, 2018).

Se podría combinar las diferentes metodologías aplicadas, utilizando técnicas de *machine learning* y los filtros de imágenes, con el fin de mejorar los resultados en la detección de parqueos libres.

La solución a este problema es crear un modelo a través del *dataset* PKLot, un conjunto extenso de imágenes de parqueo con situaciones distintas, se usará como base para la experimentación (De Almeida, Oliveira, Britto, Silva, & Koerich, 2015). Utilizando técnicas de programación, aprendizaje automático y un sistema en tiempo real. Para alcanzar como resultado, que a la persona se le pueda indicar e informar cuál es la plaza de parqueo más cercana y disponible (Ávalos, Gómez, Ordónez-Camacho, & Taipe, 2018).

El *dataset* PKLot es un conjunto de ficheros, que se encuentra dividido en imágenes completas y segmentadas, dichas imágenes se encuentran divididas en zona de estacionamiento (PUCPR, UFR04, UFR05), que a su vez se encuentran agrupadas en condiciones climáticas (*Cloudy, Rainy, Sunny*), y por ultimo divididas por la disponibilidad de la plaza (*Empty, Occupied*) (De Almeida et al., 2015).

El *dataset* CNRPark es un conjunto de ficheros, que se encuentra disponible en imágenes segmentadas, las cuales se encuentran agrupadas por el tipo de cámara (A, B) y por último divididas igualmente por la disponibilidad de la plaza (*free, busy*) (Amato, Carrara, Falchi, Gennaro, & Vairo, 2016).

Java es un lenguaje de programación orientado a objetos que fue desarrollado por Sun Microsystems que se incorporó al ámbito de la informática en los años noventa. La idea de Java es que pueda realizarse programas con la posibilidad de ejecutarse en cualquier contexto, en cualquier ambiente, siendo así su portabilidad uno de sus principales logros (Oracle, 2019).

Posteriormente fue adquirido por Oracle y en la actualidad puede utilizarse de modo gratuito, pudiéndose conseguir sin problemas un paquete para desarrolladores que oriente la actividad de programar en este lenguaje. Puede ser modificado por cualquiera, circunstancia que lo convierte en lo que comúnmente se denomina “código abierto” (Alvarez, 2019).

NetBeans entorno de desarrollo integrado (IDE) es un proyecto realizado por la Empresa *Sun MicroSystems* en junio del 2000, con un entorno de desarrollo integrado gratuito y de código abierto diseñada para crear y ejecutar programas en tecnologías para escritorio, aplicaciones web y dispositivos

móviles desarrolladas en diversos lenguajes como PHP, HTML, C++, C# entre otros esencialmente en java (M. González, 2015).

Es capaz de simplificar la gestión de grandes proyectos optimizando el código en nuestros proyectos o aplicaciones con la intención de que podamos ejecutar más rápido y usando menos memoria (Juan Manuel Gimeno, 2013).

Marvin Plugins es un *Framework* de procesamiento de imágenes de código abierto capaz de integrarse a diferentes aplicaciones java para el procesamiento de imágenes utilizando diferentes complementos, desarrollados para manipular imágenes y el desarrollo de aplicaciones (Archanjo & Muñoz, 2019).

Marvin facilita el uso de muchos filtros capaces de manipular el color y la apariencia, a continuación, se muestran los filtros que se usaron para el procesamiento de imágenes:

- BlackAndWhite: Dada la imagen y un nivel entre [-100,100] de blanco y negro, puede cambiar el color de los pixeles de la imagen a blanco y negro
- BrightnessAndContrast: Dada la imagen y un nivel entre [-127,127] de brillo y contraste, puede establecer el brillo y regular el contraste de una imagen.
- Circles: En una imagen se aplica una simulación de tonos continuos de círculos negros de diferentes tamaños.
- ColorChannel: En este filtro se puede usar un canal de color diferente y aplicarlo en la imagen dada. Se usaron 3 de los colores:
    - ColorChannel_1 corresponde al filtro de color rojo.
    - ColorChannel_2 corresponde al filtro de color verde.
    - ColorChannel_3 corresponde al filtro de color azul.
- Dithering: En una imagen se aplica una simulación de tonos continuos utilizando el enfoque de tramado el cual reduce el color de la imagen.
- EdgeDectector: En una imagen se aplica el filtro de detección de bordes, extrayendo puntos de una imagen en donde el brillo de esta cambia resaltando los puntos más impórtate.
- Emboss: Dada la imagen puede aplicarse el filtro de relieve para conseguir una mayor profundidad y contraste en las imágenes.
- ErrorDifussion: En la imagen se aplica una simulación de tonos continuos utilizando el enfoque de difusión de errores convirtiendo una imagen de varios niveles en una imagen binaria (gris y blanco).
- GaussianBlur: Dada la imagen se puede aplicar un filtro de desenfoque gaussiano produciendo un efecto nebuloso en la imagen.
- grayScaleQuantization: Dada la imagen se puede aplicar el filtro escala de grises con distintos niveles dando como resultado una imagen con un gris más claro o más obscuro.

- Harris: En una imagen se aplica el filtro mostrando puntos rojos extrayendo pixeles característicos de una imagen.
- HistogramEqualization: Es un proceso que se usa para ajustar y aumentar el contraste de las imágenes.
- Maximum: El filtro máximo generalmente se aplica a una imagen para eliminar el ruido negativo aclarando la imagen.
- Median: El filtro mediano se usa normalmente para reducir el ruido de sal y pimienta en una imagen, algo así como aclarar detalles útiles en la imagen y eliminar detalles inservibles en una imagen.
- Minimum: El filtro mínimo generalmente se aplica a una imagen para eliminar el ruido positivo obscureciendo la imagen
- Mode: Se usa para eliminar el ruido de una imagen al reemplazar los píxeles con el valor de píxel más frecuente seleccionado reduciendo la imagen haciéndola un poco más pequeña.
- NoiseReduction: Se usa en una imagen de tal manera que eleve la calidad de la imagen.
- Pixelize: Dada la imagen se aplica para redibujar la imagen usando grandes bloques de color.
- Prewitt: Dada la imagen se aplica el filtro de detección de bordes dándole un nivel de intensidad mayor a los puntos o bordes de la imagen.
- Roberts: Similar al prewitt, dada la imagen se aplica el filtro de detección de bordes de la imagen.
- Rylanders: En la imagen se aplica una simulación de tonos continuos mediante el uso de puntos.
- Sepia: Dada la imagen se puede aplicar el filtro sepia dándole un tono rojo anaranjado oscuro a la imagen.
- Sobel: Similar al prewitt, dada la imagen se aplica el filtro de detección de bordes de la imagen
- Thresholding: Dada la imagen con este filtro se puede establecer el color de los pixeles a blanco y negro separando los objetos de una imagen que más nos interese.
- Invert: Dada la imagen se puede aplicar un filtro que invierte los colores de los pixeles de la imagen dándole un efecto negativo a la imagen.

Weka, es un software de código abierto que se utiliza en el aprendizaje automático y la minería de datos. Este software utiliza herramientas de visualización y algoritmos de clasificación para el análisis de datos y modelos predictivos (Sharma & Jain, 2013).

Machine learning es una rama de la Inteligencia artificial emplea diversas técnicas en las que se incluye métodos matemáticos permitiendo que los sistemas inteligentes/ordenadores) aprendan de forma continúa mejorando la experiencia de los sistemas. También se ocupa del diseño y aplicaciones de

algoritmos de aprendizaje automático para poder convertir datos en programas (F. A. González, 2015).

Los Algoritmos de clasificación operan sobre la información provista por un conjunto de muestras cuyo objetivo es identificar a que clase o categoría pertenece la nueva muestra (Gasca, Granados, Hernández, Hernández, & Soto, 2018). Existen diferentes tipos de algoritmos los cuales son:

- BayesNet: es un modelo clasificador probabilístico, son ideales para tomar un evento que ocurrió y predecir la probabilidad de que cualquiera de las posibles causas conocidas sea el factor contribuyente.
- NaiveBayes: es un modelo clasificador probabilístico, este modelo asume que las variables son independientes entre sí por lo que ciertas características de las variables no afectaran el evento a ocurrir.
- NaiveBayesUpdateable: es un modelo clasificador probabilístico, este modelo es una actualización del modelo NaiveBayes donde se puede aumentar gradualmente el número de datos para una mejor clasificación.
- IBk: El algoritmo clasifica los datos en base a instancias, almacena los datos presentados y cuando una nueva instancia es encontrada, un conjunto de instancia relacionadas es devuelta y usada para clasificar la instancia consultada.
- LWL: El algoritmo se basa en instancias para asignar pesos. Asigna pesos usando un método basado en ejemplos y construye u clasificador a partir de los ejemplos ponderados.
- AdaBoostM1: El algoritmo sólo pueden abordar problemas de clase nominal. A menudo se puede usar junto con otros tipos de algoritmos para mejorar el rendimiento de clasificación.
- AttributeSelectedClassifier: El algoritmo es una combinación de 2 pasos: reducción dimensional a través de la selección de atributos, para luego pasar a la clasificación.
- Bagging: Es un algoritmo que crea diferentes modelos usando muestras aleatorias para luego combinar los resultados y mejorar la clasificación.
- ClassificationViaRegression: Es un algoritmo que construye un modelo de regresión para conocer el efecto que una o varias variables pueden causar sobre otra, e incluso predecir en mayor o menor grado valores de una variable a partir de otra.
- FilteredClassifier: Realiza la clasificación usando datos que han pasado por un filtro aleatorio.
- LogitBoost: Es un algoritmo que realiza la clasificación usando un esquema de regresión y puede manejar problemas de múltiples clases.

- MultiClassClassifier: Este algoritmo maneja conjuntos de datos de múltiples clases. Este clasificador también es capaz de aplicar códigos de salida de corrección de errores para una mayor precisión.
- RandomCommittee: Es un algoritmo que construye un modelo clasificador aleatorio. Cada modelo se construye utilizando una semilla de números aleatorios diferente (pero basada en los mismos datos).
- RandomizableFilteredClassifier: Es una variación simple del FilteredClassifier que instancia el modelo con un filtro aleatorio.
- RandomSubSpace: Consiste en múltiples árboles construidos sistemáticamente elegidos al azar que mantiene la mayor precisión en los datos.
- JRip: Este algoritmo implementa reglas, para la reducción de errores para realizar una rápida y eficacia clasificación de los datos.
- OneR: Este algoritmo utiliza un solo modelo clasificador; lo que solo utiliza atributos de error mínimo para la predicción, lo que divide un todo (conjunto de datos) en partes pequeñas con el fin de facilitar los cálculos de los atributos numéricos.
- PART: Este algoritmo genera una lista de decisiones, reglas y utiliza la técnica de aprendizaje de reglas “divide y vencerás” con el fin de descartar o ajustar reglas individuales con el fin de mejorar la clasificación.
- J48: El algoritmo es la implementación en lenguaje JAVA del algoritmo C4.5 genera un árbol de decisión a partir de los datos mediante particiones realizadas, se considera todas las pruebas posibles que pueden dividir el conjunto de datos y selecciona la prueba que resulta en la mayor ganancia de información.
- RandomForest: Este algoritmo es una combinación de árboles predictores tal que cada árbol depende de sus valores aleatorios.
- RandomTree: Es un algoritmo que utiliza un conjunto aleatorio de datos para construir un árbol de decisión. en un RandomTree, cada nodo se divide utilizando el mejor entre los subconjuntos de predicadores elegidos al azar en ese nodo
- REPTree: es un árbol de aprendizaje de decisión rápida y construye un árbol de decisión utilizando muestras numéricas para la clasificación de datos (Díaz-Barrios et al., 2015).

Encontrar el mejor modelo predictivo procesando el *dataset* PKLot mediante técnicas de *machine learning*, para determinar automáticamente y en tiempo real las plazas de parqueo disponibles en un parqueadero institucional.

- Determinar el conjunto de técnicas en el filtrado de imágenes, previa su integración en el modelo.
- Analizar la eficiencia de los algoritmos de clasificación para este tipo de datos.

- Integrar el modelo y los filtros de preprocesamiento en el sistema del proyecto DAPLP.

2. METODOLOGÍA

2. METODOLOGÍA

2.1. MÉTODO EXPERIMENTAL

En esta metodología el investigador primero define el problema de investigación, esto ayuda a enfocarse en una investigación lo más estrecha posible para un mejor estudio (Badii Zabeh, Castillo, Rodríguez, Wong, & Villalpando Cadena, 2007).

En la metodología experimental el investigador recopila los datos (muestra) relacionados de acuerdo con el entorno (población) en el que se trabaja, estos datos son manipulados para lograr obtener la solución.

La población será el conjunto de individuos o elementos que muestran características similares en el cual se muestra interés al momento de obtener resultados (Ventura-León, 2017).

Para obtener una muestra representativa existen ciertos aspectos que el investigador debe conocer La cantidad que representa debe ser una porción significativa extraída de la población total. Una vez conseguido el tamaño de la muestra, dicha muestra se extraerá aleatoriamente de la población total, una vez que se conozca el tamaño ideal de la muestra (Luis, 2004).

En principio, escoger una muestra es esencial para obtener resultados válidos en cada experimentación a realizar. Estos resultados dependerán de las técnicas elegidas asegurando así que el experimento sea realizado correctamente y que lo resultados obtenidos sean un reflejo de lo que es el mundo real, ayudando a la sociedad a mejorar su vida de la mejor manera posible (Otzen & Manterola, 2017).

2.2. MATERIALES Y HERRAMIENTAS

- Laptop: Windows 10, Core i3 con 4 GB y 500 GB en disco
- Notepad++ v7.6.6
- IDE NetBeans v8.2 versión con JDK 1.8.0_172
- Marvin Plugins v1.5.5
- Weka v3.8: Explorer y Experimenter
- Dataset PKLot: conjunto de imágenes de estacionamientos
- Dataset CNRPark: conjunto de imágenes de estacionamientos

En la Figura 1 se exponen las etapas que se siguieron para efectuar la experimentación para el desarrollo del trabajo de titulación.

Figura 1. Etapas de la experimentación para desarrollar el trabajo de titulación

2.3. SELECCIÓN DE LAS IMÁGENES

Para efectuar este experimento se eligió un conjunto de imágenes, con el que se realizó pruebas de preprocesamiento de imágenes y creación de archivos arff.

Para comprobar y comparar resultados se seleccionó otro conjunto de imágenes con características diferentes al anterior.

2.4. TAMAÑO DE LA MUESTRA

Se debe trabajar con una muestra representativa con base al *dataset* PKLot, para determinar el tamaño de la muestra, se utilizó la fórmula que se muestra en la Ecuación 1.

$$n = \frac{N\sigma^2 Z_{\propto}^2}{e^2(N-1)+\sigma^2 Z_{\propto}^2} \quad [1]$$

Donde:

n: Tamaño de la muestra

N: Tamaño de la población

σ: Desviación estándar de la población

Z∝: Valor constante obtenido mediante niveles de confianza

e: Error muestral

2.5. PROCESAMIENTO DE LAS IMÁGENES

Se realizó un primer programa con la ayuda de NetBeans IDE (Entorno de desarrollo integrado), para obtener muestras aleatorias de los *datasets*, cambiar el tamaño de las imágenes y utilizar técnicas de filtrado de imágenes.

2.5.1. SELECCIÓN DE UNA MUESTRA ALEATORIA SOBRE EL DATASET PKLOT.

Se utilizó el primer programa para obtener una muestra aleatoria representativa, debido al amplio conjunto de imágenes que posee el *dataset* PKLot,

2.5.2. TAMAÑO EN LAS IMÁGENES

Se cambió el tamaño de las imágenes de 100x100 a 10x10 con el primer programa, para procesar de manera rápida las imágenes.

2.5.3. TÉCNICAS DE FILTRADO EN LAS IMÁGENES

Para obtener buenos resultados en la experimentación, con el primer programa se aplicó los filtros prediseñados de Marvin y RGB que se muestran en la Tabla 1.

Tabla 1. Filtros prediseñados

Filtros de Marvin		
BlackAndWhite	GaussianBlur	Pixelize
BrightnessAndContrast	grayScaleQuantization (2,4,8,16,32)	Prewitt
Circles	Harris	Roberts
ColorChannel (Rojo, Verde, Azul)	HistogramEqualization	Rylanders
Dithering	Maximum	Sepia
EdgeDectector	Median	Sobel
Emboss	Mode	Thresholding
ErrorDifussion	NoiseReduction	Invert
Filtros por RGB		
Escala de grises		

2.6. CREACIÓN DE ARCHIVOS

Se realizó un segundo programa con la ayuda de NetBeans IDE, para proporcionar de un formato a los archivos, extraer los datos esenciales de una imagen, escribir los datos en los archivos, agregar nombre a los archivos y guardar con extensión arff, con el objetivo de que la aplicación Weka pudiera procesarlos.

2.6.1. IMPLEMENTANDO FORMATO WEKA EN LOS ARCHIVOS.

Para que Weka pueda leer los archivos se debe definir relación, atributos, clase de los datos y los datos como se muestra en la Figura 2.

```
1  @relation (En base a los datos)
2
3  @attribute numeric(Pixeles de una imagen),
4             integer(Cada de caracteres),
5             date(H:m; d-M) (En base a los datos)
6
7  @attribute Clase: Estacionamiento{Empty, Ocuppied};
8             Numeros{Enteros, Reales
9             Seres Vivos{Plantas, Animales}
10            (En base a los datos)
11
12 @data
13 (Los datos)
```

Figura 2. Ejemplo archivo arff

2.6.2. EXTRACIÓN DE LOS DATOS

Se extrajo los atributos numéricos (pixeles) de cada imagen y el atributo clase (*Empty, Occupied*).

2.6.3. ESCRIBIR LOS ARCHIVOS ARFF

En esta etapa se escribieron los atributos extraídos de las imágenes en el archivo arff

2.6.4. COLOCAR Y GUARDAR LOS ARCHIVOS ARFF

Finalizada la etapa de formato, extracción y escritura de un archivo Weka se guardó el archivo con un nombre especifico y la extensión arff.

2.7. EXPERIMENTACIÓN WEKA

Para efectuar un análisis completo de los algoritmos de clasificación que se muestran en la Tabla 2, se realizó pruebas en el Weka Explorer.

Tabla 2. Algoritmos en la aplicación Weka

Algoritmos de clasificación		
BayesNet	Bagging	RandomSubSpace
NaiveBayes	ClassificationViaRegression	JRip
NaiveBayesUpdateable	FilteredClassifier	OneR
IBk	LogitBoost	PART
LWL	MultiClassClassifier	J48
AdaBoostM1	RandomCommittee	RandomForest
AttributeSelectedClassifier	RandomizableFilteredClassifier	RandomTree
REPTree		

2.7.1. ANÁLISIS Y OBSERVACIÓN DE LOS ALGORITMOS DE CLASIFICACIÓN

Existen algoritmos para diferentes tipos de datos, con la aplicación Weka Explorer se analizó y observó que algoritmos fueron compatibles y cuáles procesaron rápidamente los *datasets*.

2.7.2. PRIMER EXPERIMENTO EN WEKA

Para determinar qué *datasets* se clasificaban correctamente usando los algoritmos, se efectuó experimentos en la aplicación Weka Experimenter, la primera prueba se realizó sin aplicar ningún filtro y la segunda prueba se aplicó el filtro gris en las imágenes.

2.7.3. SEGUNDO EXPERIMENTO EN WEKA

Se compararon dos *datasets*, el primero incluía el atributo tiempo y al segundo se le eliminó el atributo tiempo, se analizó y determinó si el atributo tiempo, influye en la clasificación de los *datasets*.

2.7.4. TERCER EXPERIMENTO EN WEKA

Se aplicó diferentes filtros en las imágenes, se creó nuevos datasets, se los procesó en la aplicación Weka y se determinó qué filtros ayudaban a obtener mejores resultados, posteriormente se combinó los mejores filtros y se realizó una segunda prueba para determinar si las combinaciones de filtros mostraban mejores resultados que usándolos individualmente.

2.7.5. EXPERIMENTO FINAL EN WEKA

Se realizó pruebas con el *dataset* CNRPark para corroborar los resultados obtenidos en cada experimento realizado sobre la muestra del *dataset* PKLot, con el propósito de realizar la experimentación final sobre el *dataset* PKLot completo.

2.7.6. CREACIÓN DEL MODELO EN WEKA

En la etapa final se procedió a crear el modelo en la aplicación Weka Explorer con el dataset PKLot, en base al porcentaje de clasificación.

2.8. INTEGRACIÓN DEL MODELO Y FILTROS DE PREPROCESAMIENTO EN EL PROYECTO DAPLP

El modelo creado en Weka y el filtro elegido, fueron integrados en el proyecto DAPLP, con el fin de lograr procesar las imágenes y predecir qué plaza de parqueo está libre u ocupada, y así culminar con la entrega del trabajo

2.8.1. PROGRAMA DE FILTRADO Y CLASIFICACIÓN DE IMÁGENES

Se realizó un tercer programa con la ayuda de NetBeans IDE, capaz de realizar preprocesamiento de imágenes y predecir las plazas de parqueo, con la finalidad de evidenciar que el modelo obtenido se integró apropiadamente.

2.8.2. IMPLEMENTACIÓN FINAL EN EL PROYECTO DAPLP

La implementación se realizó integrando el modelo Weka en el aplicativo escritorio, con la finalidad de poder utilizar el modelo.

2.8.3. CAMBIOS Y REUTILIZACIÓN DEL CÓDIGO

Se realizó los respectivos cambios, lo primordial fue reutilizar el código del tercer programa e integrarlo sobre el aplicativo escritorio con la finalidad de que el aplicativo cargue y procese el modelo.

2.8.4. PRUEBAS DEL APLICATIVO

Se realizó pruebas que evidencien que el modelo se integró correctamente al aplicativo escritorio.

3. RESULTADOS Y DISCUSIÓN

3. RESULTADOS Y DISCUSIÓN

3.1. SELECCIÓN DE LAS IMÁGENES

El primer conjunto de imágenes seleccionado fue el *dataset* PKLot debido a la amplia cantidad de imágenes segmentadas que había para procesar.

El segundo conjunto de imágenes seleccionado fue el *dataset* CNPARK, debido a que se requería demostrar las pruebas hechas con el *dataset* PKLot.

3.2. MUESTRA

Se obtuvo el tamaño de la muestra sobre la Ecuación 1, con la finalidad de determinar qué cantidad de imágenes del PKLot se utilizó.

$$n = \frac{N\sigma^2 Z_{\alpha}^2}{e^2(N-1)+\sigma^2 Z_{\alpha}^2} \quad [1]$$

$$n = \frac{695850(0.5^2)1{,}96^2}{0.05^2(695850-1)+0.5^2 1{,}96^2}$$

$$n = \frac{668294{,}34}{1.714{,}6225+0{,}9604}$$

$$n = 389{,}543601 \approx 390$$

3.3. PROCESAMIENTO DE IMÁGENES

Se clasificó las imágenes en plazas de parqueo libres u ocupadas (*Empty*, *Occupied*), las capturas de las imágenes de parqueos estaban nombradas con la fecha (Año_mes_dia), la hora (H_m_s) y un número de estacionamiento, por ejemplo 2020_02_20_10_22_#08 como se muestra en la Figura 3.

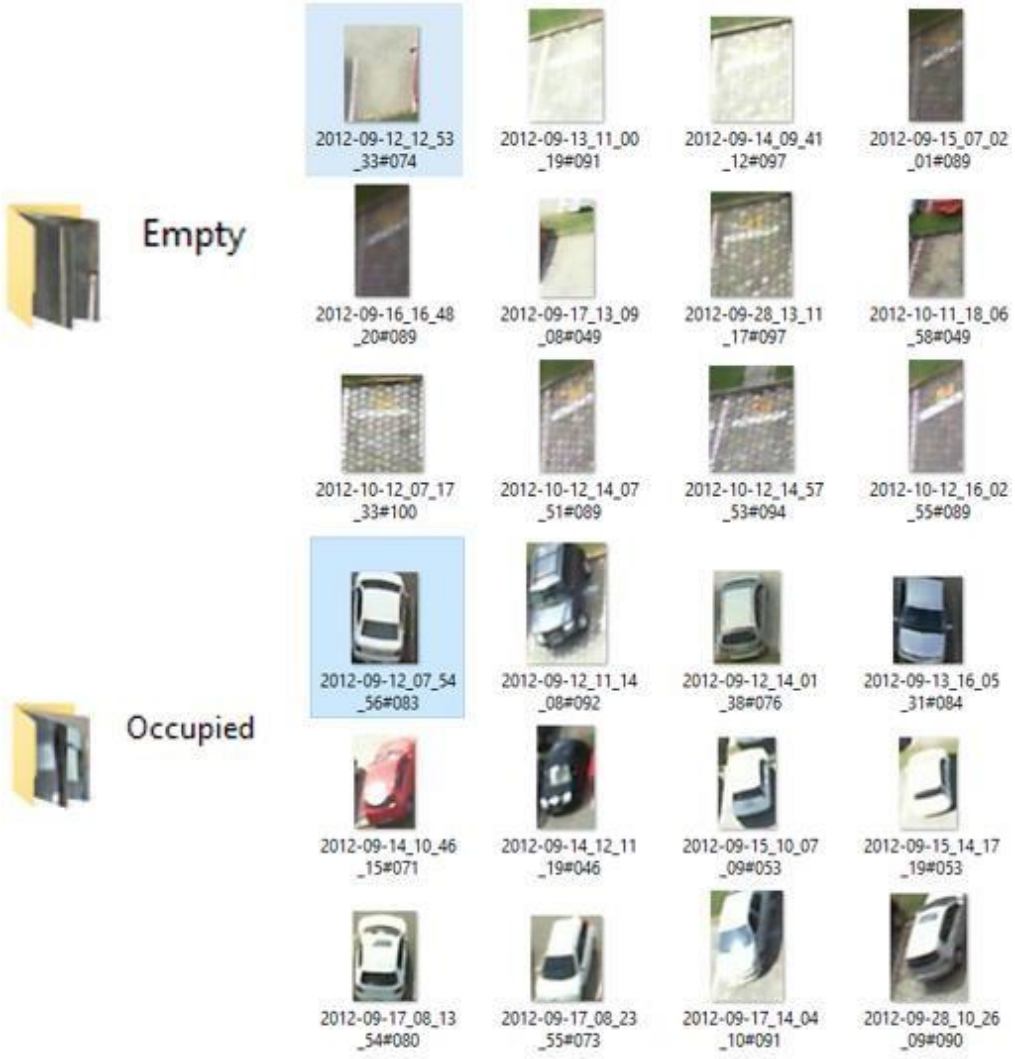

Figura 3. Clasificación de las imágenes

3.4. TAMAÑO DE IMÁGENES

Se creó diferentes conjuntos de imágenes cambiándoles de tamaño como se muestra en la Figura 4.

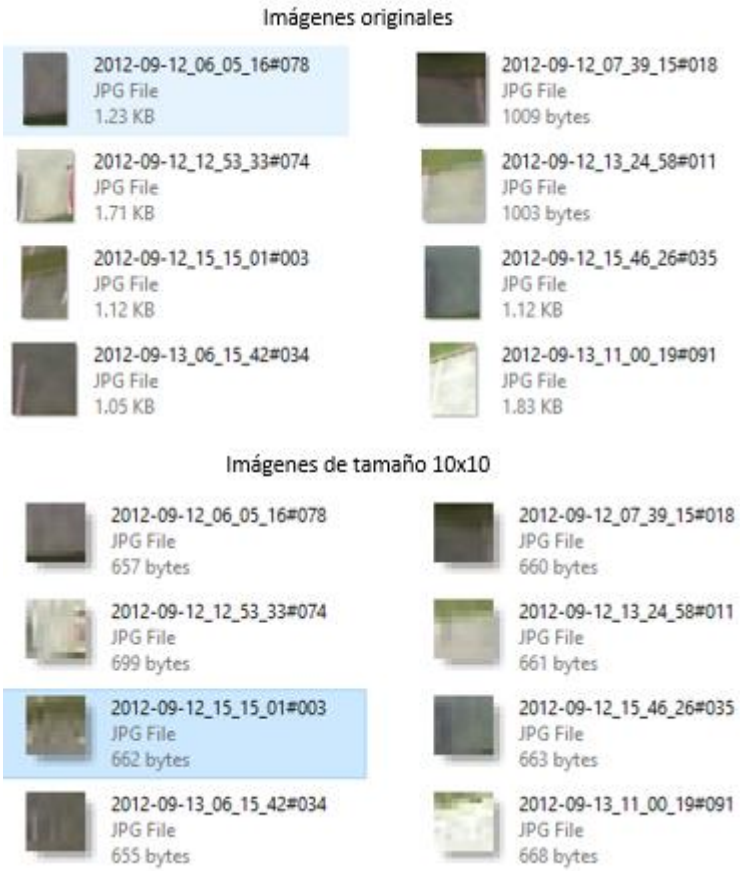

Figura 4. Cambio de tamaño a las imágenes

3.5. FILTRO EN LAS IMÁGENES

Una vez que se aplicó los filtros de Marvin y RGB se crearon nuevos conjuntos de imágenes, como se muestra en la Figura 5.

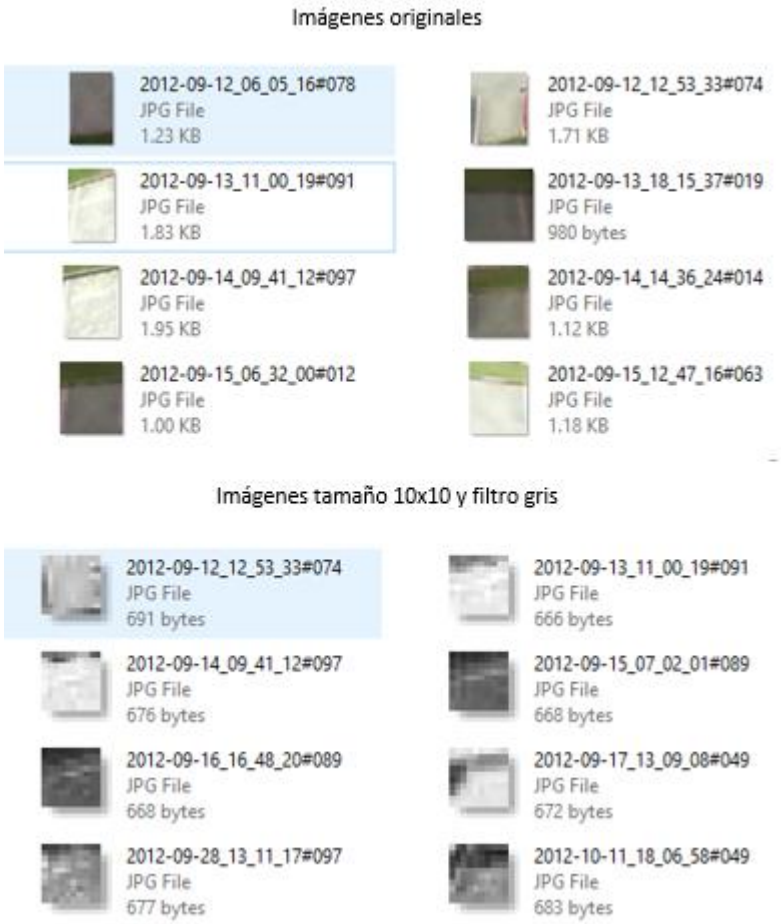

Figura 5. Aplicación de filtro gris en las imágenes

3.6. CREACIÓN DE ARCHIVOS ARFF

Se crearon los archivos con el segundo programa, se les proporcionó un formato y se cargó los datos como se muestra en la Figura 6.

```
1  @relation Dataset_PKLot_10x10_Gris
2
3  @attribute Tiempo date 'HH:mm'
4  @attribute Pixel1 numeric
5  @attribute Pixel2 numeric
6  @attribute Pixel3 numeric
7  @attribute Pixel4 numeric
8  @attribute Pixel5 numeric
9  @attribute Clase {Empty,Occupied}
10
11 @data
12
13 13:50,11,23,42,12,54,Empty
14 08:10,33,56,12,43,76,Occupied
```

Figura 6. Estructura de un archivo arff

3.7. CREACIÓN DE LOS ARCHIVOS CON LAS IMÁGENES

Se creó un archivo para cada conjunto de imágenes con tamaño diferente como se muestra en la Figura 7.

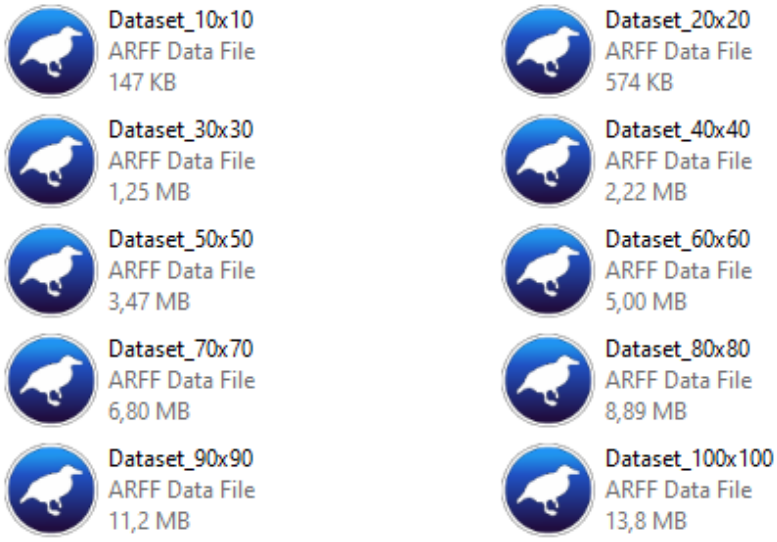

Figura 7. *Dataset* para cada conjunto de imágenes

3.8. EXPERIMENTACIÓN EN WEKA

3.7.1. ALGORITMOS DE CLASIFICACIÓN COMPATIBLES

Como resultado del análisis y observación de los algoritmos se obtuvo que 20 fueron los compatibles con los archivos arff creados, en la aplicación Weka se observó que muchos de los algoritmos estaban deshabilitados y no se pudieron utilizar debido al tipo de dato que se trató, los algoritmos compatibles se muestran en la Tabla 3.

Tabla 3. Algoritmos de clasificación Weka compatibles

Algoritmos de clasificación			
Incompatibles	**Compatibles**		
BayesNet	IBk	FilteredClassifier	OneR
	LWL	LogitBoost	PART
	AdaBoostM1	MultiClassClassifier	J48
NaiveBayes	AttributeSelectedClassifier	RandomCommittee	RandomForest
NaiveBayesUpdateable	Bagging	RandomizableFilteredClassifier	RandomTree
	ClassificationViaRegression	RandomSubSpace	REPTree
	JRip		

3.7.2. TIEMPO DE RESPUESTA

A pesar de que Weka no muestra en segundos, minutos o horas los tiempos de respuesta al utilizar cada algoritmo sobre el *dataset*, se observó qué algoritmos mostraban una repuesta de manera inmediata, catalogándolos por respuesta lenta o rápida como se muestra en la Tabla 4.

Tabla 4. Respuesta de los algoritmos de clasificación

	Dataset_10x10
Algoritmos	**Respuesta**
Logistic	Lento
IBk	Rápido
LWL	Lento
AdaBoostM1	Rápido
AttributeSelectedClassifier	Lento
Bagging	Rápido
ClassificationViaRegression	Rápido
FilteredClassifier	Rápido
LogitBoost	Rápido
MultiClassClassifier	Lento
RandomCommittee	Rápido
RandomizableFilteredClassifier	Rápido
RandomSubSpace	Rápido
JRip	Rápido
OneR	Rápido
PART	Rápido
J48	Rápido
RandomForest	Rápido
RandomTree	Rápido
REPTree	Rápido

Los resultados mostraron que 16 algoritmos fueron los más rápidos en procesar los *datasets*, esos fueron los algoritmos que se optó por utilizar.

3.7.3. PRIMER EXPERIMENTO EN WEKA

Los resultados mostraron que el *dataset* de tamaño 10x10 en comparación a los otros, no se halló diferencia significativa con respeto al porcentaje de clasificación que se muestra en la Tabla 5.

Se eligió el dataset de 10x10, debido a que fue el más conveniente, era el más pequeño y rápido en procesarse. En la aplicación Weka Experimenter se cambió las condiciones para que los resultados, muestren el mejor algoritmo, como se muestra en la Tabla 6. Los resultados mostraron que el algoritmo *RandomForest* fue el mejor con un porcentaje de 93.74%.

Como se muestra en la Figura 8 se aplicó filtro gris a todos los conjuntos de imágenes y se creó nuevos *datasets*.

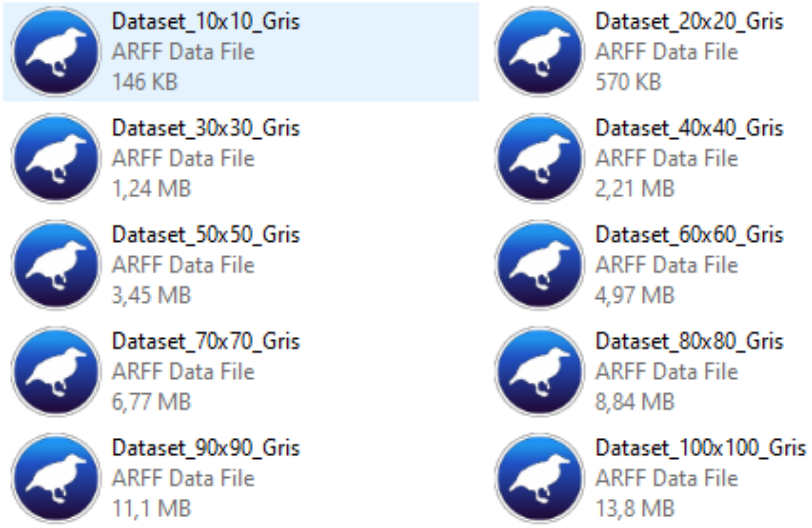

Figura 8. *Datasets* con filtro gris

Los nuevos *datasets* fueron procesados en la aplicación Weka Experimenter, los resultados fueron similares en comparación con los *datasets* originales, el algoritmo *RandomForest* siguió siendo el mejor con un 93.90% como se muestra en la Tabla 7.

El resultado obtenido fue que el archivo dataset con tamaño 10x10 y filtro gris habían mejorado los resultados de clasificación.

Tabla 5. Resultados por mejor *dataset*

	Datasets									
	10x10	20x20	30x30	40x40	50x50	60x60	70x70	80x80	90x90	100x100
Algoritmos	% correcto de clasificación									
rules.OneR	68.46	70.85	70.90	68.26	67.08	73.77	68.51	70.46	72.51	68.15
lazy.IBk	74.08	75.63	75.77	76.28	75.82	76.10	75.90	76.03	75.67	75.51
meta.AdaBoostM1	75.38	78.31	80.18	80.28	79.97	78.62	77.67	79.08	78.59	81.44
meta.RandomizableFiltered	75.92	75.59	76.56	77.05	80.97	75.51	76.41	76.03	77.26	76.64
meta.ClassificationViaReg	81.23	84.13	84.87	82.18	84.92	82.26	83.13	82.36	81.62	82.41
meta.LogitBoost	81.97	82.79	82.95	83.41	84.05	82.31	82.31	82.59	82.82	84.54
trees.REPTree	82.44	81.41	83.62	83.38	83.67	83.36	82.85	83.26	83.82	83.95
rules.JRip	82.67	82.90	82.79	81.56	85.95	83.64	81.33	82.15	81.59	80.69
trees.RandomTree	84.33	84.64	85.49	84.95	84.85	85.64	85.69	86.10	84.95	84.41
rules.PART	84.41	84.64	84.64	84.26	86.41	84.13	84.08	84.46	84.13	84.85
trees.J48	84.87	86.41	86.18	85.44	87.62	85.82	86.64	86.74	86.13	86.33
meta.FilteredClassifier	86.69	87.41	87.74	87.62	87.62	87.31	88.18	88.23	87.21	88.64
meta.Bagging	87.69	87.64	88.59	88.18	88.31	88.38	89.05	88.21	88.13	88.77
meta.RandomSubSpace	88.87	88.79	88.72	89.54	89.08	89.51	89.31	89.10	88.95	89.05
meta.RandomCommittee	91.28	93.51	93.21	93.67	93.18	93.49	93.21	93.69	92.92	93.41
trees.RandomForest	93.74	94.18	94.41	94.59	94.41	94.36	94.21	94.59	94.31	94.49

Tabla 6. Resultados por mejor Algoritmo de clasificación

	Algoritmos															
Datasets	trees.RandomForest	lazy.IBk	meta.AdaBoostM1	meta.Bagging	meta.ClassificationViaReg	meta.FilteredClassifier	meta.LogitBoost	meta.RandomCommittee	meta.RandomizableFiltered	meta.RandomSubSpace	rules.JRip	rules.OneR	rules.PART	trees.J48	trees.RandomTree	trees.REPTree
	% correcto de clasificación															
10x10	93.74	74.08	75.38	87.69	81.23	86.69	81.97	91.28	75.92	88.87	82.67	68.46	84.41	84.87	84.33	82.44
20x20	94.18	75.63	78.31	87.64	84.13	87.41	82.79	93.51	75.59	88.79	82.90	70.85	84.64	86.41	84.64	81.41
30x30	94.41	75.77	80.18	88.59	84.87	87.74	82.95	93.21	76.56	88.72	82.79	70.90	84.64	86.18	85.49	83.62
40x40	94.59	76.28	80.28	88.18	82.18	87.62	83.41	93.67	77.05	89.54	81.56	68.26	84.26	85.44	84.95	83.38
50x50	94.41	75.82	79.97	88.31	84.92	87.62	84.05	93.18	80.97	89.08	85.95	67.08	86.21	87.62	84.85	83.67
60x60	94.36	76.10	78.62	88.38	82.26	87.31	82.31	93.49	75.51	89.51	83.64	73.77	84.13	85.82	85.64	83.36
70x70	94.21	75.90	77.67	89.05	83.13	88.18	82.31	93.21	76.41	89.31	81.33	68.51	84.08	86.64	85.69	82.85
80x80	94.59	76.03	79.08	88.21	82.36	88.23	82.59	93.69	76.03	89.10	82.15	70.46	84.46	86.74	86.10	83.26
90x90	94.31	75.67	78.59	88.13	81.62	87.21	82.82	92.92	77.26	88.95	81.59	72.51	84.13	86.13	84.95	83.82
100x100	94.49	75.51	81.44	88.77	82.41	88.64	84.54	93.41	76.64	89.05	80.69	68.15	84.85	86.33	84.41	83.95
Nota: Weka nos permite poner en primer lugar el mejor resultado y compararlo con el resto.																

Tabla 7. Resultados por mejor Algoritmo de clasificación con filtros gris

	Algoritmos															
Datasets	trees.RandomForest	lazy.IBk	meta.AdaBoostM1	meta.Bagging	meta.ClassificationViaReg	meta.FilteredClassifier	meta.LogitBoost	meta.RandomCommittee	meta.RandomizableFiltered	meta.RandomSubSpace	rules.JRip	rules.OneR	rules.PART	trees.J48	trees.RandomTree	trees.REPTree
	% correcto de clasificación															
10x10	93.90	75.10	77.77	88.18	85.13	87.38	80.87	92.72	76.44	88.41	82.85	65.69	84.77	84.87	83.41	83.74
20x20	93.95	75.03	77.49	88.13	82.49	87.64	81.00	92.95	76.77	87.92	81.62	67.46	83.97	85.03	83.05	82.72
30x30	94.10	75.31	76.64	87.41	82.77	87.79	83.28	92.38	76.18	87.69	82.26	65.90	82.74	85.10	84.62	82.95
40x40	94.51	74.95	78.51	87.44	80.31	87.49	82.38	93.23	75.79	88.23	82.74	65.21	82.90	85.33	84.56	83.15
50x50	94.46	75.33	81.10	87.95	81.23	87.74	83.79	92.64	74.82	88.72	81.82	71.23	82.90	85.54	84.31	83.69
60x60	94.10	75.67	77.51	87.62	81.77	87.38	83.03	93.21	73.82	88.87	82.21	65.36	83.15	85.90	84.46	82.51
70x70	94.15	75.46	77.95	87.05	82.13	87.67	82.46	93.13	74.08	87.82	81.67	68.95	82.85	84.62	84.79	82.46
80x80	94.21	75.72	79.18	87.23	81.77	87.90	83.90	92.67	75.64	88.10	80.51	67.03	83.33	85.03	85.82	82.15
90x90	94.15	75.77	78.41	87.56	81.90	87.56	83.51	93.23	75.54	87.38	81.85	68.54	83.51	84.62	86.46	82.44
100x100	94.05	75.67	78.74	87.54	82.62	87.28	83.08	93.26	75.08	88.18	81.10	65.54	83.59	84.56	84.90	83.31
Nota: Weka nos permite poner en primer lugar el mejor resultado y compararlo con el resto.																

3.7.4. SEGUNDO EXPERIMENTO EN WEKA

Una vez que se obtuvo el dataset_10x10 como mejor *dataset,* a partir de este se crearon 2 nuevos *datasets*, el primero incluía el atributo tiempo y al segundo se le quito este atributo, con estas condiciones se los procesó en la aplicación Weka Experimenter.

Los resultados mostraron que el *dataset* que no incluía el atributo tiempo no obtuvo mejor resultado con respecto al *dataset* que incluía el atributo tiempo como se muestra en la Tabla 8. Con estos resultados se tomó la decisión de conservar el atributo tiempo, debido a que es un dato que no influyó mucho al momento de clasificar los *datasets*.

Tabla 8. Resultados de los *datasets* con y sin tiempo

	Algoritmo
Datasets	meta.RandomizableFiltered
	% correcto de clasificación
10x10	93.74
10x10_sin Tiempo	93.77

3.7.5. TERCER EXPERIMENTO EN WEKA

Se crearon nuevos *datasets* aplicando el tamaño 10x10 y los diferentes tipos de filtros de Marvin como se muestra en la Figura 9, con la finalidad de conseguir mejores resultados

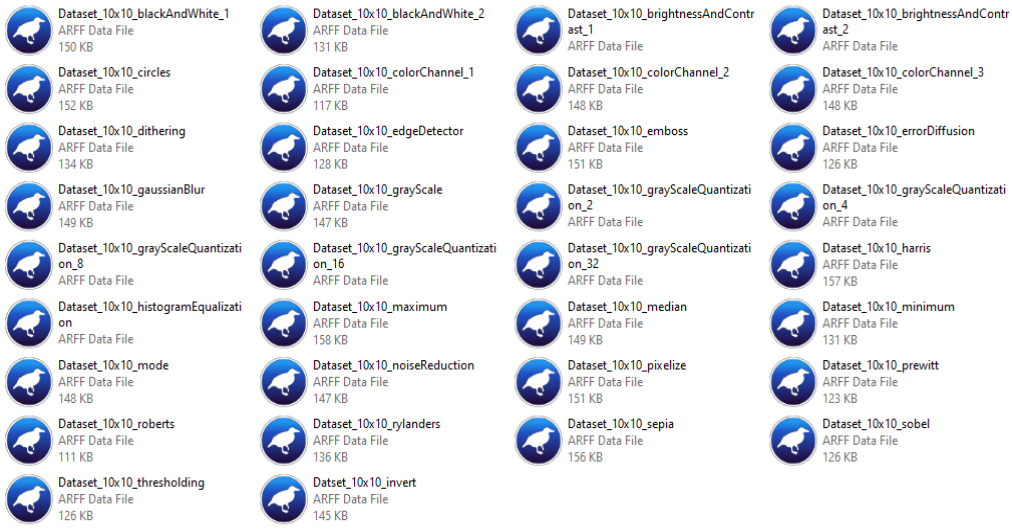

Figura 9. *Datasets* con tamaño 10x10 y con filtros Marvin

Estos archivos fueron procesados en la aplicación Weka Experimenter, los cuales mostraron resultados muy prometedores como se muestra en la Tabla 9, el mejor de todos los filtros aplicados fue el filtro colorChannel_1 con un 94.82% de clasificación.

Luego se eligieron los otros 2 mejores filtros, el filtro grayScaleQuantization_8 con un 94.72% y el filtro sepia con un 94.28%, los cuales se aplicó simultáneamente sobre el conjunto de imágenes y se creó nuevos *datasets* como se muestra en la Figura 10.

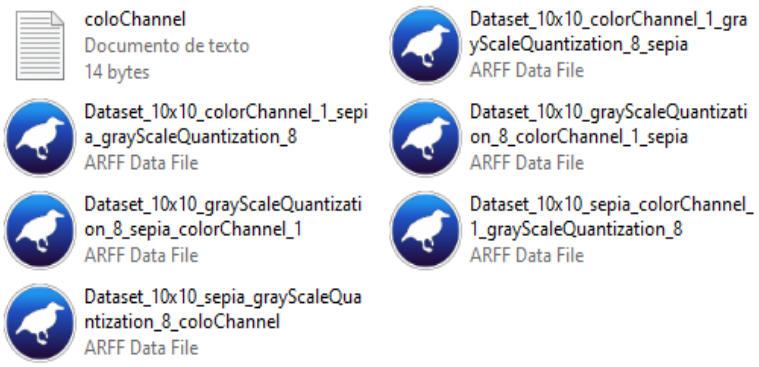

Figura 10. *Dataset* con filtros combinados

Se hizo una segunda prueba utilizando los *datasets* con filtros combinados, los cuales presentaron un mejor porcentaje de clasificación en comparación a cada filtro individual utilizado, la mejor de todas las combinaciones fue el filtro colorChannel_1_grayScaleQuantization_8_sepia con un 96.46% de clasificación como se muestra en la Tabla 10.

Tabla 9. Resultados por mejor filtro

	Filtros															
Algoritmo	colorChannel_1	blackAndWhite_1	blackAndWhite_2	brightnessAndContrast_1	brightnessAndContrast_2	circles	colorChannel_2	colorChannel_3	Dithering	edgeDetector	emboss	errorDiffusion	gaussianBlur	grayScale	grayScaleQuantization_16	grayScaleQuantization_2	grayScaleQuantization_32
	% correcto de clasificación																
trees.RandomForest	94.82	80.44	87.49	93.15	86.97	73.28	92.21	93.59	88.54	91.77	92.95	72.62	91.28	94.13	94.05	84.79	94.21
Nota: Weka nos permite poner en primer lugar el mejor resultado y compararlo con el resto.																	

Continúa…

Continuación…

	Filtros																
Algoritmo	grayScaleQuantization_4	grayScaleQuantization_8	harris	histogramEqualization	maximum	median	minimum	mode	noiseReduction	Pixelize	prewitt	roberts	rylanders	sepia	sobel	thresholding	invert
	% correcto de clasificación																
trees.RandomForest	93.33	94.72	90.87	83.79	76.49	85.28	89.85	85.33	92.67	59.82	92.79	90.97	81.31	94.28	92.44	83.62	93.69
Nota: Weka nos permite poner en primer lugar el mejor resultado y compararlo con el resto.																	

Tabla 10. Resultados de las combinaciones de filtros

	Datasets					
Algoritmo	colorChannel_1_grayScaleQuantization_8_sepia	colorChannel_1_sepia_grayScaleQuantization_8	grayScaleQuantization_8_colorChannel_1_sepia	grayScaleQuantization_8_sepia_colorChannel_1	sepia_colorChannel_1_grayScaleQuantization_8	sepia_grayScaleQuantization_8_coloChannel
	% correcto de clasificación					
trees.RandomForest	96.46	92.79	94.03	94.64	93.44	95.90
Nota: Weka nos permite poner en primer lugar el mejor resultado y compararlo con el resto.						

3.7.6. EXPERIMENTO FINAL EN WEKA

Se crearon 2 nuevos *datasets* con el conjunto de imágenes CNRPark, el primer *dataset* se realizó aplicando filtro gris a las imágenes y el segundo se realizó aplicando la mejor combinación de filtros a las imágenes, posteriormente se creó un nuevo *dataset* con el conjunto de imágenes PKLot completo, se les aplicó el filtro gris creando 3 archivos nuevos como se muestra en la Figura 11.

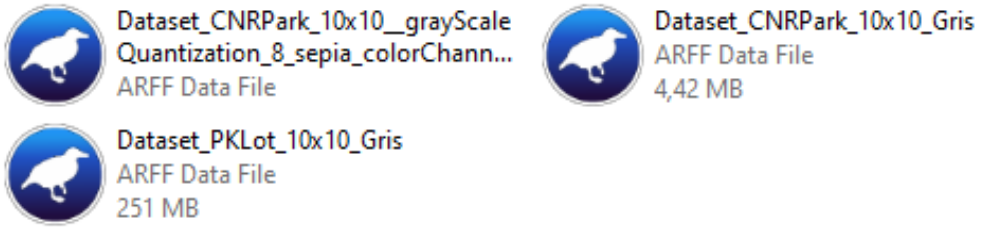

Figura 11. *Datasets* con tamaño 10x10 y los diferentes filtros combinados

Por último, se efectuó la experimentación en la aplicación Weka Experimenter, los resultados obtenidos se muestran en la Tabla 11.

Tabla 11. Resultados finales de los *datasets*

	Algoritmo
Dataset	trees.RandomForest
	% correcto de clasificación
CNRPark_10x10_grayScaleQuantization_8_sepia_colorChannel_1	96.81
CNRPark_10x10_Gris	99.28
PKLot_10x10_Gris	99.41

Se demostró que el combinar filtros no se encontró una mejora significativa que, al emplear solo el filtro gris, como se muestra en la Tabla 10. Con estos resultados se tomó la decisión de seleccionar el *dataset* con mejor porcentaje de clasificación para la creación del modelo definitivo.

3.7.7. CREACIÓN DEL MODELO EN WEKA

El *dataset* PKLot con tamaño 10x10 y filtro gris con un 99.41% de clasificación fue el mejor resultado obtenido en la última experimentación, por ende, se optó por utilizar el archivo y se creó el modelo Weka en la aplicación Weka Explorer que se muestra en la Figura 12.

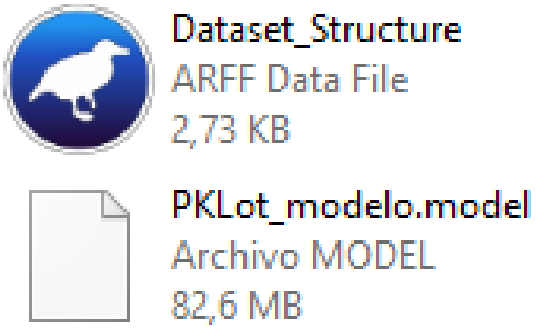

Figura 12. Modelo y estructura del *dataset*

3.9. CLASIFICACIÓN, PREDICIÓN E INTEGRACIÓN DEL MODELO.

3.8.1. TERCER PROGRAMA O PROGRAMA PRUEBA

El modelo fue capaz de integrarse al programa prueba mostrando en su totalidad la predicción de las plazas de parqueo libres u ocupadas como se muestra en la Figura 13.

```
Programa Predicción Estacionamiento
Estacionamiento: #002
Estado: Empty
Estacionamiento: #009
Estado: Empty
Estacionamiento: #001
Estado: Empty
Estacionamiento: #001
Estado: Occupied
Estacionamiento: #005
Estado: Occupied
Estacionamiento: #010
Estado: Empty
Estacionamiento: #019
Estado: Empty
Estacionamiento: #002
Estado: Empty
Estacionamiento: #022
Estado: Occupied
Estacionamiento: #031
Estado: Occupied
Estacionamiento: #033
Estado: Occupied
Estacionamiento: #032
Estado: Occupied
```

Figura 13. Predicción de las plazas de parqueo

3.8.2. INTEGRACIÓN DEL MODELO EN EL PROYECTO DAPLP

El proyecto DPALP ya constaba de un aplicativo escritorio con una interfaz que cargaba y leía los modelos Weka como se muestra en la Figura 14, como resultado hubo errores de compilación que se debían cambiar.

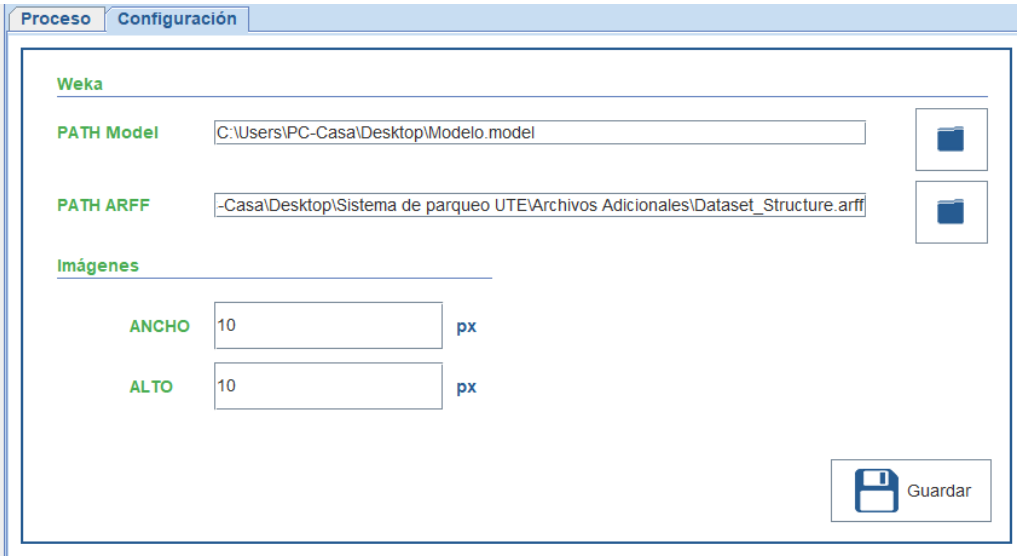

Figura 14. Aplicativo Escritorio cargar y lectura del modelo y archivo Weka

3.8.3. CAMBIOS EN EL APLICATIVO ESCRITORIO

Se utilizó parte del código del programa prueba que se muestra en la Figura 15, debido a que el código implementado anteriormente servía solo para un tipo de dato y algoritmo de clasificación diferente.

```
//Modificador Por Daniel Gómez
public String predecir(String name, int id, String id_parking, String URL) {

    try {
        //Carga el modelo
        Classifier cls = (RandomForest) weka.core.SerializationHelper.read(MODEL);
        //Carga la estructura de los datos que hallará en el archivo
        // "Dataset_Structure.arff"
        Instances dataSet = new ConverterUtils.DataSource(STRUCTURE).getDataSet();
        dataSet.setClassIndex(dataSet.numAttributes()-1);
        //Asigna el índice de la clase al datasetset (es el último atributo)
        Instance inst = new DenseInstance(dataSet.numAttributes());//Weka 3.8
        inst.setDataset(dataSet);
        //Carga los datos de la instancia a ser clasificada
        //la forma de cargar los datos de la instancia dependerá de su problema
        //String[] s = data.split(",");
        for (int i = 0; i < pixel.length; i++) {
            //double d = Double.parseDouble(s[i]);
            inst.setValue(i, pixel[i]);
        }

        //clasifica la instancia según el modelo
        double pred = cls.classifyInstance(inst);
          System.out.println(pred);
        //presenta la clasificación
```

Figura 15. Código implementado en el aplicativo Weka

3.8.4. PRUEBAS DEL APLICATIVO

Una vez que se realizó los cambios, se corrió el proyecto mostrando como resultados que el modelo se integró satisfactoriamente prediciendo las plazas de parqueo que llegaban en ese momento como se observa en la Figura 16 y Figura 17.

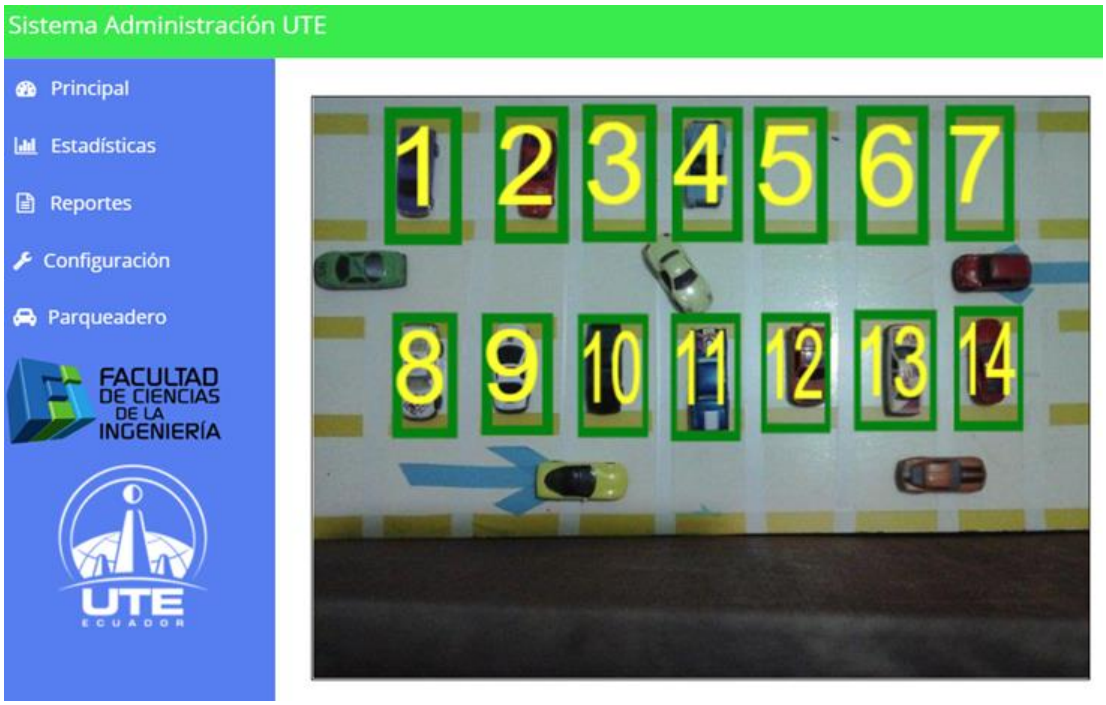

Figura 16. Plazas de parqueo libres, ocupados, marcadas y enumeradas

```
Predicción: 2020-02-10_16_36_06 60 1 Ocupado
Predicción: 2020-02-10_16_36_06 60 2 Ocupado
Predicción: 2020-02-10_16_36_06 60 3 Libre
Predicción: 2020-02-10_16_36_06 60 4 Libre
Predicción: 2020-02-10_16_36_06 60 5 Libre
Predicción: 2020-02-10_16_36_06 60 6 Libre
Predicción: 2020-02-10_16_36_06 60 7 Libre
Predicción: 2020-02-10_16_36_06 60 8 Libre
Predicción: 2020-02-10_16_36_06 60 9 Ocupado
Predicción: 2020-02-10_16_36_06 60 10 Ocupado
Predicción: 2020-02-10_16_36_06 60 11 Ocupado
Predicción: 2020-02-10_16_36_06 60 12 Ocupado
Predicción: 2020-02-10_16_36_06 60 13 Ocupado
Predicción: 2020-02-10_16_36_06 60 14 Ocupado
```

Figura 17. Predicción de la plazas libres y ocupadas del parqueo

4. CONCLUSIONES Y RECOMENDACIONES

4. CONCLUSIONES Y RECOMENDACIONES

4.1. CONCLUSIONES

Luego de realizar las experimentaciones en Weka y utilizar las técnicas de preprocesamiento de imágenes se concluye que:

- El conjunto de técnicas en el filtrado de imágenes es muy útil cuando se hace este tipo de experimentos, se pudo conseguir muy buenos resultados.
- La mayoría de algoritmos de clasificación implementados fueron muy eficientes al procesar este tipo datos, clasificándolos en su mayoría en una escala alta, obteniendo así resultados favorables.
- Se logró integrar de manera eficiente nuestro modelo al proyecto DAPLP, utilizando las técnicas de *machine learning* y los filtros de preprocesamiento estudiados.
- Se obtuvo el mejor modelo predictivo con un 99.41% de clasificación, capaz de procesar cualquier cantidad de datos, determinando automáticamente en tiempo real las plazas de parqueo disponibles en una institución.

4.2. RECOMENDACIONES

- Se recomienda utilizar otros métodos de investigación que nos muestren otros tipos de situaciones a la que está ligado el problema.
- Se recomienda implementar otros tipos de técnicas de preprocesamiento que den una perspectiva diferente.
- Se recomienda investigar las técnicas que *machine learning* ofrece ya que existen soluciones diferentes para este tipo de problema.
- Se recomienda utilizar diferentes tipos de datos y probar con los algoritmos no probados en esta práctica pueden que ofrezcan resoluciones distintas.

5. BIBLIOGRAFÍA

5. BIBLIOGRAFÍA

Alvear-Puertas, V., Rosero-Montalvo, P., Peluffo-Ordóñez, D., & Pijal-Rojas, J. (2017). Internet de las Cosas y Visión Artificial, Funcionamiento y Aplicaciones: Revisión de Literatura. *Enfoque UTE*, *8*(1), 244. https://doi.org/10.29019/enfoqueute.v8n1.121

Amato, G., Carrara, F., Falchi, F., Gennaro, C., & Vairo, C. (2016). Car parking occupancy detection using smart camera networks and Deep Learning. *Proceedings - IEEE Symposium on Computers and Communications*, *2016-Augus*(Dl), 1212–1217. https://doi.org/10.1109/ISCC.2016.7543901

Ávalos, H., Gómez, E., Ordónez-Camacho, D., & Taipe, O. (2018). Implementation of a system for the administration, configuration and monitoring of parking areas. *Proceedings - 3rd International Conference on Information Systems and Computer Science, INCISCOS 2018*, 356–360. https://doi.org/10.1109/INCISCOS.2018.00058

Badii, C., Nesi, P., & Paoli, I. (2018). Predicting Available Parking Slots on Critical and Regular Services by Exploiting a Range of Open Data. *IEEE Access*, *6*, 44059–44071. https://doi.org/10.1109/ACCESS.2018.2864157

Badii Zabeh, M., Castillo, J., Rodríguez, M., Wong, A., & Villalpando Cadena, P. (2007). Diseños experimentales e investigación científica. *Innovaciones de Negocios*, *4*(8), 283–330.

Cheng, L., & Qiao, T. (2016). Localization in the parking lot by parked-vehicle assistance. *IEEE Transactions on Intelligent Transportation Systems*, *17*(12), 3629–3634. https://doi.org/10.1109/TITS.2016.2547987

De Almeida, P. R. L., Oliveira, L. S., Britto, A. S., Silva, E. J., & Koerich, A. L. (2015). PKLot-A robust dataset for parking lot classification. *Expert Systems with Applications*, *42*(11), 4937–4949. https://doi.org/10.1016/j.eswa.2015.02.009

Díaz-Barrios, H., Alemán-Rivas, Y., Cabrera-Hernández, L., Morales-Hernández, A., Del Carmen Chávez-Cárdenas, M., & Casas-Cardoso, G. M. (2015). Algoritmos de aprendizaje automático para clasificación de Splice Sites en secuencias genómicas. *Revista Cubana de Ciencias Informáticas*, *9*(4), 155–170.

Gasca, P. J. C., Granados, J. A. R., Hernández, M. Á. V., Hernández, I. de J. R., & Soto, R. R. (2018). Métodos de clasificación: Análisis de fertilidad. *Pistas Educativas*, *35*(111), 43–57. Retrieved from http://itcelaya.edu.mx/ojs/index.php/pistas/article/view/356

González, F. A. (2015). Modelos de aprendizaje computacional en reumatología. *Revista Colombiana de Reumatologia*, *22*(2), 77–78. https://doi.org/10.1016/j.rcreu.2015.06.001

González, M. (2015). Herramienta de Desarrollo Netbeans. *Universidad Del Norte*, 5. Retrieved from https://www.consultorjava.com/wp/wp-content/uploads/2015/09/herramienta_desarrollo_netbeans.pdf%0Ahttp://www.consultorjava.com/wp/wp-content/uploads/2015/09/herramienta_desarrollo_netbeans.pdf

Juan Manuel Gimeno, J. L. G. (2013). *Introducción a Netbeans*. 15–55. https://doi.org/10.1111/j.1399-5618.2008.00597.x

Lin, T., Rivano, H., & Le Mouel, F. (2017). A Survey of Smart Parking Solutions. *IEEE Transactions on Intelligent Transportation Systems*, *18*(12), 3229–3253. https://doi.org/10.1109/TITS.2017.2685143

Luis, P. (2004). Población “Muestra y Muestreo.” *Punto Cero*, *09*(08), 69–74.

Mancilla, P. C. S., Ibañez, J. A. G., & Castillo, J. J. C. (2014). Avances en Tecnologías de Información Research in Computing Science. *Instituto Politécnico Nacional, Centro de Investigación En Computación México 2014*.

Masmoudi, I., Wali, A., Jamoussi, A., & Alimi, A. M. (2014). Vision based system for vacant parking lot detection: VPLD. *VISAPP 2014 - Proceedings of the 9th International Conference on Computer Vision Theory and Applications*, *2*, 526–533. https://doi.org/10.5220/0004730605260533

Nieto, R. M. (2018). *Detección de vehículos en entornos Multi-cámara utilizando información Contextual*.

Ordóñez, D., Gómez, E., & Ávalos, H. (2018). Propuesta arquitectural para un sistema de detección automática de parqueaderos libres. *2018 International Conference on Information Systems and Computer Science (INCISCOS)*, 5.

Otzen, T., & Manterola, C. (2017). Técnicas de Muestreo sobre una Población a Estudio. *International Journal of Morphology*, *35*(1), 227–232. https://doi.org/10.4067/S0717-95022017000100037

Sharma, T. C., & Jain, M. (2013). WEKA Approach for Comparative Study of Classification Algorithm. *(IJARCCE) International Journal of Advanced Research in Computer and Communication Engineering*, *2*(4), 1925–1931.

Taipe, O. W. (2018). *Sistema Web para la Administración, Configuración y Monitoreo de parqueos libres dentro de la Universidad UTE utilizando Software Libre*.

Ventura-León, J. L. (2017). ¿Población o muestra?: Una diferencia necesaria. *Nursing Research*, *39*(5), 309. https://doi.org/10.1097/00006199-199009000-00016

Enlace URKUND:
https://secure.urkund.com/view/61981305-224275-818706