Análisis de Señales – Clasificación de Imágenes

Grupo A

Andrea Bonetti

Pablo Catret Ruber

Alejandro Dionis Ros

Adrián Lizzadro Plá

Mario Soto Ramos

Sergio Sebastia Garcia

Contenido

1. Introducción

El entendimiento y procesamiento de los datos son puntos primordiales para desarrollar modelos predictivos, es por ello por lo que, para resolver el problema que expone la resolución de este trabajo se aplicarán distintas soluciones de procesamiento de datos de tal forma que el modelo sea capaz de extraer la mayor cantidad de relaciones entre los datos para predecir con el mayor porcentaje de acierto posible. Se pondrá especial atención a la investigación y desarrollo de distintas maneras de procesar los datos antes de ser consumidos por el modelo, mientras que la selección de este se considerará como un punto secundario durante la implementación del proyecto.

Los datos o señales que se necesitan procesar en este proyecto son de tipo imagen, las cuáles presentaran una variedad de desafíos específicos de su naturaleza como la gran cantidad de memoria que ocupan (dependiendo de la resolución) y la diversidad de resoluciones y canales en los que se codifican, entre otros.

1.1 Datos

Los datos que han sido necesarios recopilar para el entrenamiento y prueba del modelo son fotografías de distintos paisajes de noche, de día y nublado. Estos tres tipos son las distintas posibilidades en las que se puede etiquetar una imagen, es decir las clases.

Cada integrante ha capturado distintas imágenes con su dispositivo móvil de los paisajes y de parte del cielo para facilitar la identificación de las clases.

A la hora de juntar todas las imágenes recopiladas a partir de distintos dispositivos ha surgido la complejidad de que los datos no están representados de la misma forma (algunas tienen distintas resoluciones, otras tienen distintos formatos de imagen). Para solventar este problema la mejor solución es normalizar los datos al mismo formato y a la misma resolución, la cual se ha implementado durante el preprocesamiento y será explicada más detalladamente durante su sección específica en el documento.

Todos estos datos han sido almacenados en distintas carpetas por clase, de forma que nos facilite el etiquetado manual de las entradas.

1.2 Metodología de Trabajo

La forma de trabajar escogida para desarrollar el proyecto se basa en la distribución de responsabilidades y, como su nombre indica, se trata de dividir el proyecto en distintas áreas de trabajo donde cada una es asignada a un responsable para el diseño y desarrollo de esta.

Las distintas áreas de trabajo que hemos identificado son las siguientes:

- **Importación y Normalización de Datos:** En esta área se implementa la importación del dataset en la aplicación y se normalizan los datos para que las características extraídas durante la siguiente fase de preprocesamiento determinen las decisiones del modelo y no factores externos como la resolución de las imágenes, su estructura o su formato.

- **Preprocesamiento de las Datos:** El preprocesamiento consiste en realizar transformaciones de los datos de entrada que los preparan para ser consumidos por el modelo y facilitar el aprendizaje de este, además de conducirlo hacia el objetivo que debe lograr.
- **Selección del Modelo:** La selección del modelo también es un punto importante para obtener unos resultados adecuados. Depende del tipo de datos que estemos tratando y del aprendizaje compararemos entre unos posibles candidatos u otros. En el caso de este proyecto, como trabajamos con datos etiquetados será necesario comparar distintos algoritmos de aprendizaje supervisado.
- **Interpretación y Visualización de resultados:** La interpretación de los resultados tiene mucho peso a la hora de seleccionar el modelo definitivo que queremos utilizar para conseguir el objetivo que estemos buscando. Como es necesario comparar entre los distintos resultados que nos devuelve cada modelo será necesario incluir visualizaciones de estos para tomar la solución óptima.
- **Desarrollo de la documentación:** Cada uno de los pasos de la implementación y los procesos se detallan en un documento.

2. Importación de Imágenes

Esta fase del proyecto es el inicio de la implementación en código y consiste en importar todas las imágenes recopiladas dentro de nuestro programa para, posteriormente, ser procesadas, manipuladas y utilizadas por los modelos.

Dentro de este punto se explicará cómo se han obtenido las imágenes, cómo se han almacenado y qué pasos han sido necesarios implementar para poder hacer uso de ellas dentro del código.

2.1 Construcción y Etiquetado del Dataset

El dataset o conjunto de información que se ha construido ha sido elaborado a partir de fotografías reales de paisajes y ciudades, tomadas por los integrantes del proyecto, haciendo uso de sus dispositivos móviles.

El requisito necesario para escoger las fotos era que pudiera visualizarse, al menos, una parte del cielo. Esto se debe al objetivo principal del modelo de inteligencia artificial que es clasificar si las imágenes son de un día soleado, nublado o si, en cambio, se han tomado durante la noche.

Para almacenar y compartir este dataset entre los integrantes del grupo se ha creado una carpeta con tres subcarpetas (una por cada clase) usando Google Drive. Esta estructura también permite facilitar el etiquetado de las muestras manual desde el código ya que cada subcarpeta tiene el nombre de la clase correspondiente.

La implementación del etiquetado se ha realizado a parte del código principal, en el script “generacionMetadatos.R”. En este script se utilizan los nombres de las carpetas, junto a la dirección de la carpeta local del dataset (“./data”) para recorrer todas las imágenes y estructurar su path local junto con el nombre de la clase a la que pertenecen, todo ello almacenado dentro de un contenedor de tipo dataframe y posteriormente exportado a un CSV para poder acceder desde el código principal cuando se considere necesario.

El balanceo de estas tres clases va a ser un punto importante para que el modelo nos devuelva buenos resultados sin importar la clase que estemos intentado predecir. Para

ello, se han obtenido unos porcentajes de la proporción por clase y se han mostrado con una gráfica circular para tener un control de la distribución.

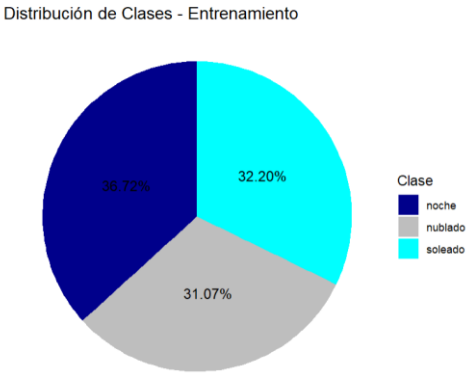

Ilustración 1: Distribución de Clases - Entrenamiento

Existen un total de 251 imágenes en el dataset, de las cuáles las etiquetadas como noche son 92, las etiquetadas como nublado son 78 y las 81 restantes son las etiquetadas como soleado. Al dividir en train y test quedan 178 para entrenamiento y 74 para probar el modelo.

2.2 Importación

Para el proceso de importación se han incluido las imágenes almacenadas en Google Drive como una carpeta local del proyecto llamada “data”. A partir de esta carpeta se leerán cada una de las imágenes y se comprobarán si tienen formato “.jpg” (es el formato estándar que hemos escogido para normalizar las imágenes) y las que tengan uno distinto serán transformada a este otro.

Una vez convertidas todas al mismo formato se utilizarán las rutas para identificar a cada imagen y se dividirá en conjunto de entrenamiento y conjunto de pruebas. El mismo procedimiento se aplicará para el contenedor de metadatos que se construirá con el script explicado en el punto 2.1. Además, se codificarán las etiquetas con números para que pueda ser interpretado por el modelo. (NOCHE -> 0, SOLEADO -> 1, NUBLADO -> 2).

Por último, se aplicará un reescalado a las imágenes para trabajar a partir de una resolución uniforme en los datos.

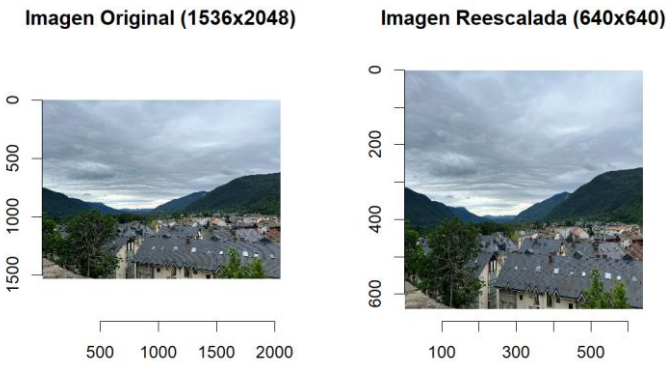

Ilustración 2: Reescalado Original a 640x640

3. Preprocesamiento de Imágenes

Para que el modelo sea capaz de aprender adecuadamente a resolver un problema específico, dependiendo de los datos con los que estemos tratando, puede ser necesario ciertos preprocesamientos sobre ellos que faciliten el entrenamiento y lo "guíen" hacia el comportamiento deseado, ya que no siempre es suficiente con alimentar al modelo con los datos crudos del dataset para obtener buenos resultados.

A los datos que se utilizan como entradas para el entrenamiento del modelo se le denominan características y pueden ser datos crudos sin preprocesamiento, combinaciones de varios datos o funciones matemáticas que tienen un significado el cual le permite al modelo tomar mejores decisiones.

Gran parte de las veces la fase de preprocesamiento no sólo consiste en extraer información significativa para la toma de decisiones, sino que también es necesario preparar o estructurar los datos ya disponibles en un formato que sea compatible y entendible para el modelo que estamos utilizando (Estructurar los datos en contenedores, dataframes, por columnas, por filas, por tipos compatibles, etc).

Un problema típico, es el de la aparición de datos total o parcialmente vacíos o nulos que pueden añadir complejidad al entrenamiento del modelo si no se procesan adecuadamente. En el caso de este proyecto, como el dataset de imágenes ha sido desarrollado desde cero, el equipo se ha asegurado de introducir valores válidos y, por lo tanto, no es necesario tener en cuenta este tipo de problemática ya que se ha tenido en consideración desde el principio de toma de datos.

A partir de las imágenes de entrada se aplicarán un conjunto de técnicas de preprocesamiento para extraer cualitativamente una serie de características que constituirán las entradas que se le suministrará al modelo, cada una de ellas será detallada en los siguientes subapartados.

3.1 Selección de Características

De los datos disponibles en las entradas del dataset es necesario seleccionar cuáles son los que pueden aportar información significativa en la toma de decisión del modelo. Para ello, lo normal es realizar un estudio o análisis exploratorio para encontrar las relaciones entre características de entrada y resultados a través de visualización de gráficas y datos, sin embargo, este caso sólo trata con imágenes, donde las características son los canales de color de cada píxel de la imagen (RGB) y donde es complejo encontrar relaciones, ya que los patrones que buscamos para identificar o etiquetar una entrada pueden ser cambiantes. Teniendo en cuenta, que después de reescalar los datos a una resolución de 640x640 y que cada píxel contiene tres canales de color, sólo una imagen está compuesta de 1.2 M de características lo que es inviable para entrenar modelos de aprendizaje máquina por la memoria que puede llegar a ocupar y, además, porque sería muy complejo encontrar relaciones entre tanta cantidad de características y el entrenamiento no daría lugar a un modelo funcional.

Por ello, será necesario aplicar una serie de funciones a los datos que nos devuelvan valores significativos para el modelo sin necesidad de incluir tanta cantidad de características.

Las características que se han seleccionado como entradas al modelo son listadas y se explicarán en detalle durante el siguiente subapartado:

- Medianas de la imagen en cada componente de color RGB (3 características)

- Skewness (Asimetría) por cada componente de color (3 características)
- Kurtosis (Distribución de intensidades) por cada componente de color (3 características)
- Contraste promedio
- Porcentaje de colores claros y oscuros (2 características)
- Porcentaje de colores grises, negro azulado y azul claro (3 características)
- Medianas de la imagen en cada componente de color HSV (3 características)
- Frecuencias altas y bajas de la imagen (Cambios en las intensidades y colores) (2 características)

3.2 Técnicas de preprocesado

Todas las técnicas que se han listado durante el apartado anterior serán explicadas a continuación:

RGB

Para comenzar, la más sencilla consiste en condensar toda la información de los canales de colores para cada píxel a partir de la media o la mediana. En este caso, se escogió la mediana ya que es menos sensible a outliers y por los datos que estamos tratando es probable que existan durante transiciones bruscas de color. Esto permite obtener tres características, una por cada canal de color.

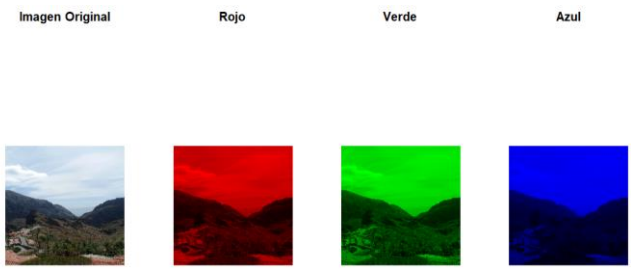

Ilustración 3: Imágen dataset RGB

La distribución de los rangos de valores por cada componente la podemos visualizar utilizando un boxplot para cada clase.

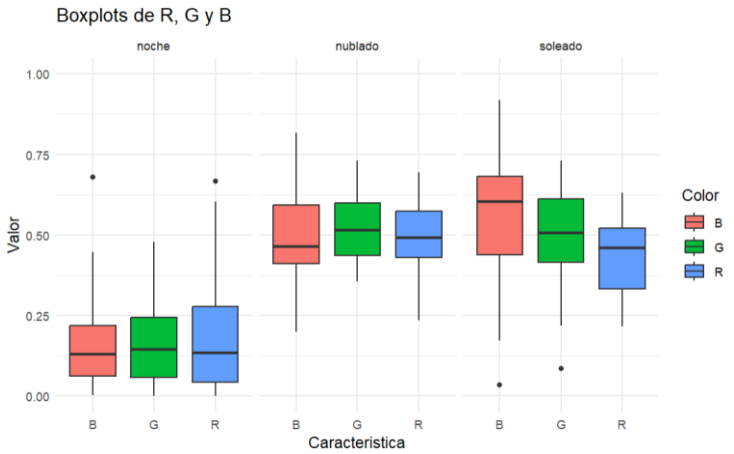

Ilustración 4: Distribución de valores en los canales de color para cada clase

En base a nuestros datos lo que podemos visualizar en este gráfico es que las imágenes de noche tienen valores por debajo de 0.3 (Los valores de cada componente están delimitados entre 0 y 1) en cada uno de los componentes de color, lo que hará que sea fácil de distinguir para el modelo de las otras dos clases, sin embargo, a pesar de que las distribuciones de las clases nublado y soleado son distintas, se solapan para una gran conjunto de valores, lo que hará más difícil de distinguir si es de una u otra.

HSV

Una transformación habitual durante el preprocesamiento de este tipo de datos es la de convertir los canales RGB a HSV ya que aún siendo sólo un cambio en el modelo de color, permite aportar información de valor al modelo porque en lugar de separar por colores separa por tonalidad, saturación y brillo las cuales son unas características que tienen un rango de valores definido para cada una de las clases que intentamos predecir (para nublado una saturación baja y brillos medios, para noche brillos bajos y tonos azulados y para soleado brillos altos y saturaciones altas), es decir que existen unas relaciones entre las características que introducimos y los resultados que nos devuelve el modelo, por lo tanto, son buenos candidatos para mejorar la toma de decisiones.

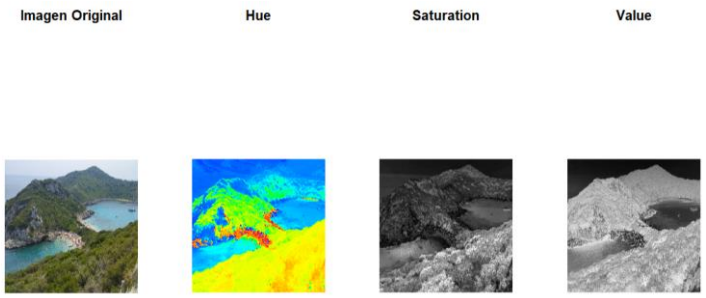

Ilustración 5: Imágen dataset HSV

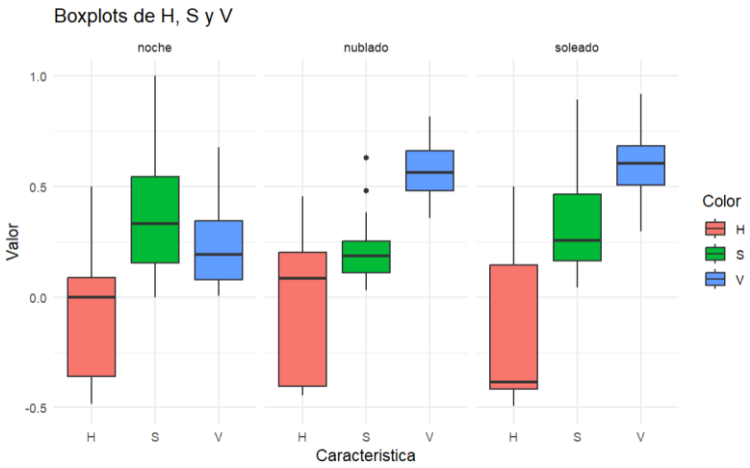

Ilustración 6: Distribución de valores por cada canal HSV para cada clase

Skewness, Kurtosis y Contraste Promedio

Además de la mediana para ambos modelos de color se pueden extraer características estadísticas avanzadas de las imágenes como el skewness, kurtosis y contraste promedio.

El skewness o asimetría mide si los datos están distribuídos simétricamente o tienen un sesgo en algún lado. La interpretación de este valor se observa de la siguiente manera: Si el valor es 0, la distribución es simétrica, si es positivo los valores tienden a acumularse en el lado izquierdo y si es negativo, en cambio, al lado derecho. Tratándose del procesamiento de imágenes tanto si el coeficiente obtenido es negativo como positivo puede significar cambios en los niveles de brillos o de color predominantes.

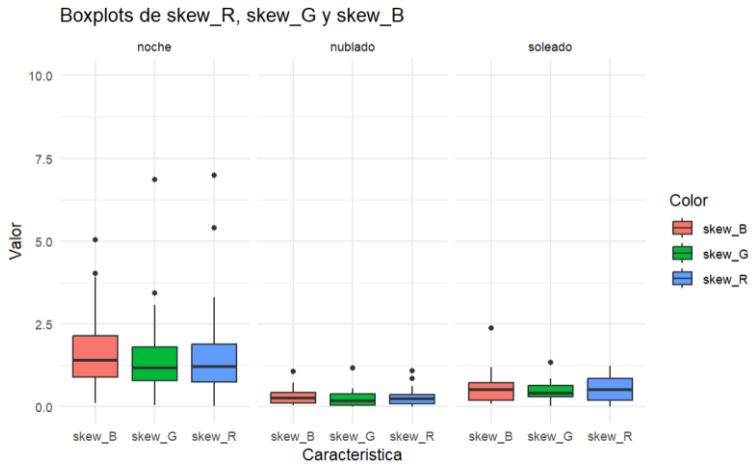

Ilustración 7: Distribución de Skewness por cada canal de color para cada clase

La Kurtosis mide la distribución de intensidades de los píxeles en imágenes digitales. Se estudia observando si es un valor grande positivo (>3) el cual indica una distribución con picos muy elevados (representa valores muy similares, por lo tanto se identifican con zonas uniformes y con pocos bordes) y colas pesadas o, en cambio, si es un valor bajo (<3) que determina lo contrario, más similar a una distribución uniforme, más plana y con colas más suaves (lo que representa cambios muy bruscos y repetitivos a lo largo de la imagen, característico de cielos nublados).

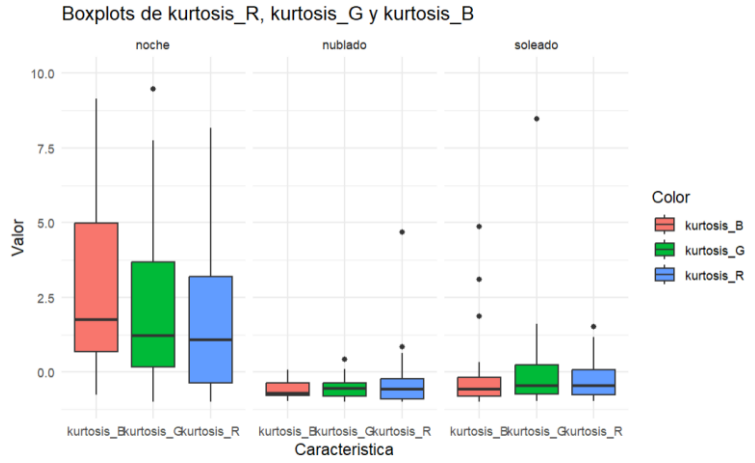

Ilustración 8: Distribución de Kurtosis por cada canal de color para cada clase

El contraste promedio mide la diferencia media de intensidad entre píxeles adyacentes en una imagen. Es útil para evaluar la "nitidez" de los patrones, donde un contraste alto indica bordes marcados y transiciones abruptas, típico en imágenes soleadas con sombras fuertes y un contraste bajo indica transiciones suaves, típico en imágenes nubladas o nocturnas. A partir del contraste promedio también se puede obtener el brillo

promedio ya que este simplemente lo calculamos a partir de la media de los píxeles por filas y por componentes de color.

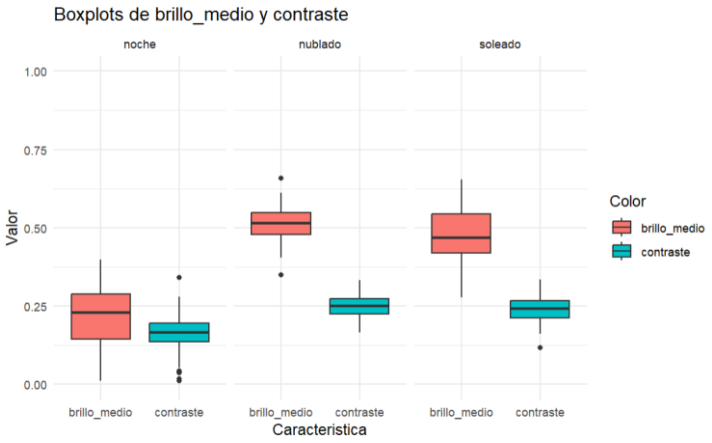

Ilustración 9: Distribución de brillo y contraste para cada clase

Tal y como se puede visualizar en la distribución de estas dos características podemos delimitar fácilmente la noche de las imágenes nubladas y soleadas y distinguir mejor, estas dos últimas con el rango de valores del brillo medio, ya que las imágenes nubladas muestran un brillo muy similar en sus muestras, mientras que las imágenes soleadas varían más en cuanto a la característica

Porcentaje de píxeles claros y porcentaje de píxeles oscuros

A partir del brillo y, sabiendo el aspecto o características típicas de las distintas clases con las que el modelo clasifica (para noche brillos bajos y para soleados y nublados brillos altos) podemos simplificarla más, delimitando cuánta parte de la imagen es oscura y cuánta es clara. Para ello consideramos que, por encima de 0.7 del valor del brillo el píxel es claro y si, en cambio, el valor está por debajo de 0.3 es oscuro. Con estos dos umbrales fijos determinamos en proporción, cuántos píxeles oscuros y claros tiene la imagen.

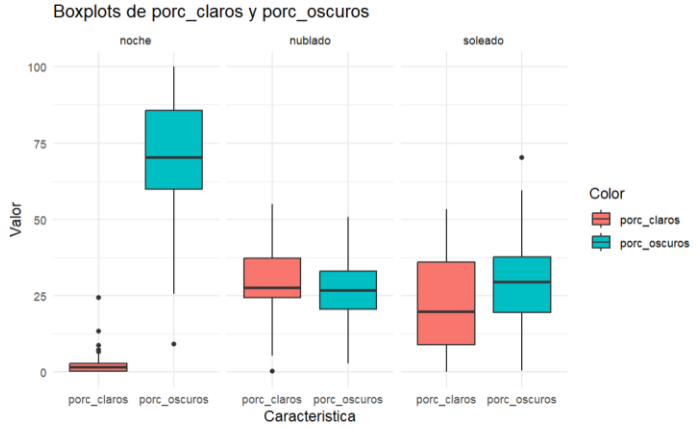

Ilustración 10: Distribución de porcentaje de píxeles claros y oscuros para cada clase

Observando la distribución de estas dos características se puede observar como claramente para las imágenes de noche predominan los píxeles oscuros aunque pueda haber partes donde exista iluminación. Las imágenes nubladas presentan tanto partes claras como partes oscuras, al igual que las soleadas, lo que tiene sentido teniendo en cuenta las imágenes que se han utilizado para la clase soleada, donde se incorporan también fotos de atardecer donde sólo una parte de las imágenes son claras, mientras el resto de la foto muestra las partes donde está oscureciendo. Otra peculiaridad que ocurre con este tipo de imágenes soleadas es que las que se han tomado durante el día no incluyen al sol ni partes con mucho brillo, es por eso por lo que parte de estas imágenes no son tomadas en cuenta por estas características, porque se sitúan por encima del umbral de oscuridad y por debajo del umbral de claridad.

Porcentaje de grises, azules oscuros y azules claros

Al haber descubierto que este tipo de características creadas a partir de umbrales funcionan bien para la toma de decisiones del modelo, se han llegado a incluir tres más, que están particularmente relacionadas con las imágenes existentes en el dataset: el porcentaje de píxeles que son grises, azules oscures y azules claros los cuales caracterizan a los días nublados, noches y soleados respectivamente.

Utilizando los componentes de color RGB se han realizado operaciones condicionales para umbralizar distintos rangos de valores que representan a cada uno de estos tres colores. Las condiciones son las siguientes:

gris <- abs(R - G) < 0.1 & abs(G - B) < 0.1 & abs(R - B) < 0.1

negro_azulado <- R < 0.2 & G < 0.2 & B > 0.4

azul_claro <- R > 0.4 & G > 0.4 & B > 0.6

Con ello, obtenemos una distribución de colores de la siguiente forma:

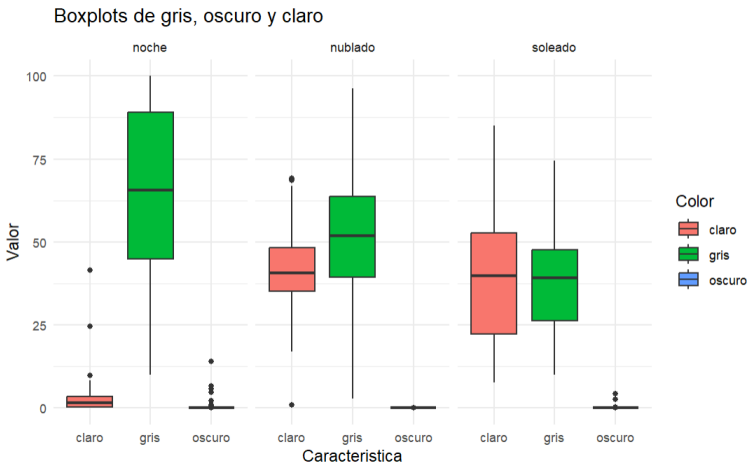

Ilustración 11: Distribución de porcentaje de píxeles claros, grises y oscuros para cada clase

Las imágenes nocturnas se componen principalmente de los grises que hemos especificado, aunque muchas también presentan partes de azul oscuro, como muestran los outliers. En el caso de los outliers del color claro se debe a las que presentan un gran porcentaje de iluminación (farolas, luces, etc). Nublado y Soleado sigue siendo muy similar comparando estas características más específicas, debido, como se ha explicado con anterioridad, a los atardeceres que también están incluidos en la categoría soleada,

sin embargo, la mediana de los valores grises es mayor en las imágenes nubladas como cabría esperar.

Transformada de Fourier y Análisis de Frecuencias

Otra forma de determinar cómo varían los colores e intensidades a lo largo de la imagen, es fijarse en las frecuencias existentes de la señal, frecuencias bajas representan cambios lentos y suaves, como áreas de color uniforme o gradientes suaves, como el brillo general o la uniformidad (útil para detectar nublado y noche). De la misma manera, frecuencias altas representan cambios rápidos, como bordes, texturas, detalles finos o ruido (útil para imágenes de día y soleado).

Aplicando la Transformada de Fourier a la imagen, analizamos cómo cambian las intensidades en términos de frecuencias y lo que obtenemos es una matriz con las frecuencias bajas en el centro y las frecuencias altas hacia las esquinas. Para poder separar las frecuencias, se utiliza una máscara circular de valor 1 en el área definida por el radio y 0 fuera de ella, para frecuencias bajas y el complemento de dicha máscara para frecuencias altas bajas. Hecha la división, lo único que falta es el cálculo de la energía, que se obtiene simplemente sumando el valor absoluto (magnitud) de las componentes de frecuencia dentro de cada región (baja y alta). Esto devuelve una medida de cuánta información está contenida en cada rango de frecuencias. Como se puede comprobar en las siguientes gráficas, las que tienen menor frecuencia son las etiquetadas como noche y nublado y las que tienen mayor valor en las altas frecuencias son las soleadas.

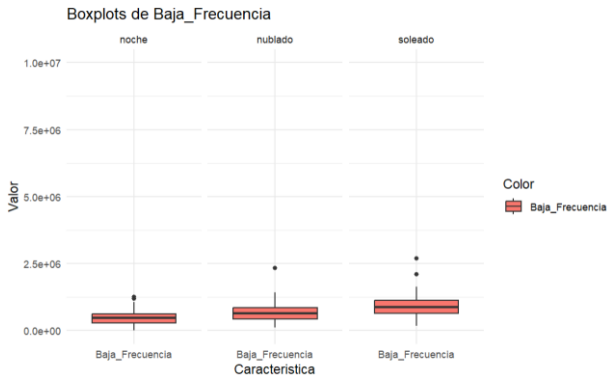

Ilustración 12: Valor de bajas frecuencias por clase

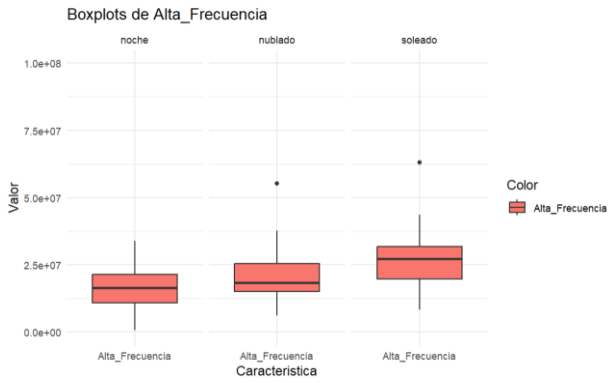

Ilustración 13: Valor de altas frecuencias por clase

4. Selección del modelo

Una vez los datos están preparados y contienen la información necesaria para que el modelo sea capaz de aprender a realizar una tarea, comienza el proceso de entrenamiento. Depende del algoritmo que se escoja este entrenamiento minimizará la función de error de una forma u otra.

Ya que durante esta asignatura no se ha trabajado ni se ha estudiado los diferentes modelos de aprendizaje máquina, cómo funcionan y para qué tipo de datos se suelen utilizar se realizará una comparativa de los resultados para distintos modelos que sean adecuados para los datos que tratamos y para el aprendizaje supervisado que buscamos.

Normalizacion y Estandarización de los datos

Previamente a entrenar los modelos se realizará un normalizado o estandarización de las características ya que, como se puede apreciar en las gráficas del apartado de preprocesamiento, cada una tiene distribuciones distintas y algunas con magnitudes muy superiores al resto, lo que daría más importancia a estas características y lo que se quiere lograr es que se le de la misma importancia a cada una.

Después de probar la normalización basada en máximos y mínimos, se observó que existían características que tomaban valores muy bajos. Todo esto a causa de los outliers, los cuáles representaban máximos o mínimos muy alejados de la distribución real de los datos.

```
(x - min(x)) / (max(x) - min(x))
```

Ilustración 14: Función de Normalización para los datos de entrada

Para resolver este problema se aplicó una estandarización de los datos que tiene en cuenta la media y la desviación estándar y distribuye mejor los coeficientes entre el rango deseado sin ser tan sensible a outliers.

```
(x - mean(x)) / sd(x)
```

Ilustración 15: Función de Estandarización para los datos de entrada

Prueba de Modelos

Para probar cada uno de los modelos compatibles con el problema a resolver se ha utilizado una metodología sencilla. Primero instanciando el modelo, entrenándolo para los datos de train con sus respectivas características y probándolo, utilizando el mismo para predecir resultados en base a los datos de test. Además se han probado distintos parámetros de configuración para cada modelo (dentro de un script aislado del código principal llamado “parámetros_modelos.R”) y se han seleccionado los que mejores resultados devolvían. Por último, se evalúa el modelo comparando las etiquetas reales con las predichas, extrayendo el porcentaje de acierto y mostrando las matrices de confusión para poder visualizar más intuitivamente los resultados.

A continuación, se mostrará una tabla con los distintos algoritmos de entrenamiento utilizados, sus porcentajes de acierto y los parámetros que han generado mejores resultados (análisis por prueba y error):

Algoritmo	Porcentaje de Acierto	Índice Kappa	Parámetros
SVR (Support Vector Classifier)	**0.8243243**	**0.7363661**	**Type = C-Classification** **Kernel = Radial** **Cost = 10** **Gamma = 0.1**
XGBoost	0.8108108	0.7153846	Objetive = multi:softmax Num_class = 3 Eta = 0.3 Max_depth = 3
KNN	0.7972973	0.6963064	K = 6
Random Forest	0.8243243	0.7354963	Ntree = 100 Mtry = 5 Importance = TRUE
Naive Bayes	0.6081081	0.4048808	Valores por defecto de la librería e1071
Regresión logística	0.6486486	0.4705559	Valores por defecto de la librería nnet

El porcentaje de acierto se calcula sumando todas las respuestas del modelo acertadas divididas entre el total de predicciones que se han hecho. Esto da lugar a un valor entre 0 y 1 en el que 0 significa que no ha acertado ninguna predicción y 1 significa que ha acertado todas.

El índice Kappa es otra medida de evaluación del modelo que se calcula de forma distinta. Este coeficiente no mide directamente si los resultados están bien o no, sino la concordancia entre los valores de los dos evaluadores, sin embargo, como estos evaluadores son los reales y los predichos su concordancia si que está relacionada con el nivel de acierto. Para calcularla se utiliza la siguiente fórmula:

$$\kappa = \frac{p_o - p_e}{1 - p_e}$$

Ilustración 16: Cálculo del índice Kappa

Donde p0 es el número de coincidencias entre el número total de predicciones y pe la probabilidad hipotética de una coincidencia aleatoria.

Cuanto mayor es el coeficiente kappa mayor es la concordancia entre las respuestas reales y las predichas y, por tanto, mayor es el número de muestras que acierta.

Como se puede observar, el algoritmo que devuelve mejores resultados para ambas métricas de evaluación es el SVR. Usando este algoritmo se van a mostrar algunas de las imágenes que no se han acertado durante la clasificación junto a su posible explicación.

Del total de las imágenes a predecir (74) se han acertado 61 y 13 han fallado al clasificar algunas de ellas son las siguientes:

Ilustración 17: Imágen Test mal clasificada 1

Como se puede observar de esta predicción tiene sentido la equivocación ya que, a pesar de ser una imagen de noche, se compone de más colores claros y brillantes (a causa de la iluminación) que de colores oscuros.

Ilustración 18: Imágen Test mal clasificada 2

En este caso es lo contrario, es una imagen de soleado pero con una gran parte oscura que presente un patrón uniforme, además no presenta brillos.

Ilustración 19: Imágen Test mal clasificada 3

Este último caso es el que menos sentido tiene, sin embargo, si revisamos las características extraídas durante la fase de preprocesamiento, las frecuencias obtenidas durante la transformada de Fourier y el coeficiente de Kurtosis, lo más probable es que presenten una Kurtosis baja y frecuencias bajas que caracterizan a las imágenes nubladas ya que gran parte de la imagen es el cielo que presenta cambios suaves y uniformes a lo largo del eje Y.

Para visualizar mejor todo el conjunto de estadísticas e identificar distintos grupos de forma intuitiva se ha hecho uso de algunos algoritmos de reducción de dimensionalidad como PCA, MDS y T-SNE

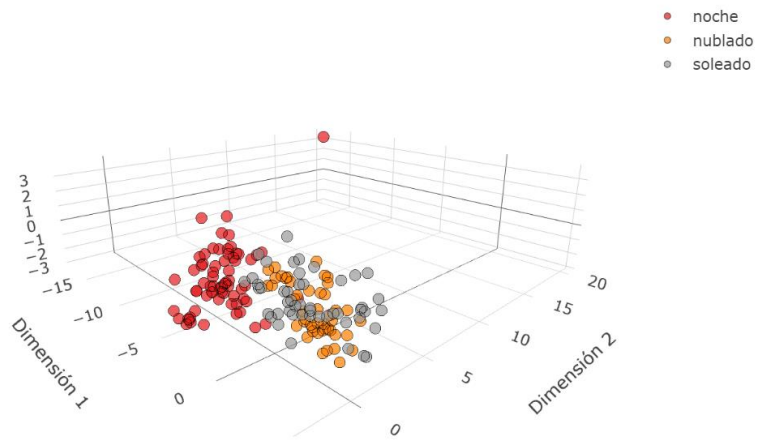

Ilustración 20: Visualización 3D de los datos aplicando PCA a las características originales

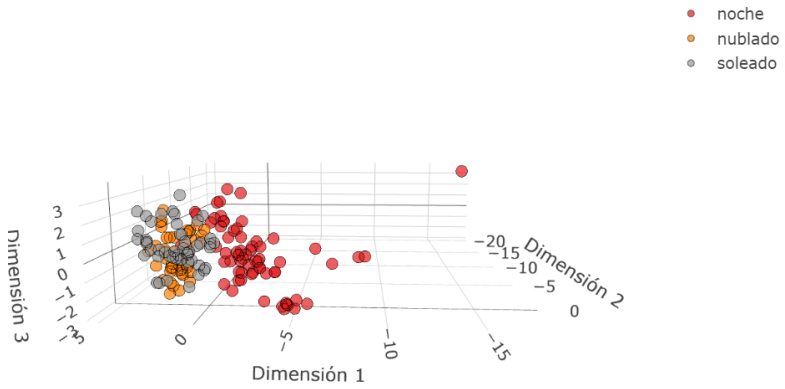

Ilustración 21: Visualización 3D de los datos aplicando MDS a las características originales

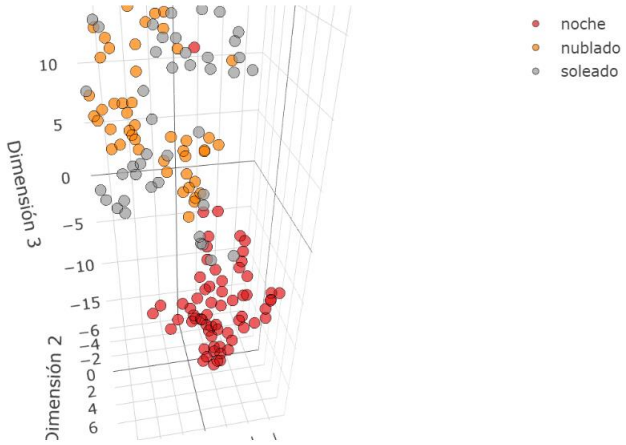

Ilustración 22: Visualización 3D de los datos aplicando t-SNE a las características originales

Para los 3 algoritmos es bastante fácil diferenciar el grupo de las imágenes de noche, sin embargo, para las imágenes nubladas y las soleadas no es tan fácil, esto se debe a que dentro de las soleadas también se incluyen una serie de imágenes que podrían clasificarse con otra etiqueta si se hubiera decidido aumentar la complejidad. Dentro de la clase soleada tenemos etiquetadas imágenes que representan también el atardecer, las cuáles tienen todavía tonos naranjas, pero también tonos oscuros y grises. Tampoco se ha diferenciado en el etiquetado, a partir de qué cantidad de nubes se considera nublado, ya que puede pasar que en algunas imágenes se pueda observar por alguna

parte el brillo del sol y por otras contener nubes. Estos hechos hacen que no se haya podido separar completamente los grupos nublado y soleado, sin embargo, a partir del conjunto de técnicas de preprocesamiento aplicadas y el modelo seleccionado se ha podido alcanzar un porcentaje de acierto bastante elevado.

Aprovechando la reducción de dimensionalidad realizada con estos últimos algoritmos se ha probado a entrenar un modelo con PCA y comprobar si se puede mejorar el porcentaje para alguno de los modelos. Los resultados, teniendo en cuenta que se han usado los mismos parámetros de configuración que en la iteración anterior sin PCA para cada modelo, han sido los siguientes:

Algoritmo	Porcentaje de Acierto	Índice Kappa
SVR (Support Vector Classifier)	0.8108108	0.7157750
XGBoost	0.7837838	0.6753496
KNN	0.8378378	0.7570451
Random Forest	0.7837838	0.6754386
Naive Bayes	0.7702703	0.6544905
Regresión logística	**0.8648649**	**0.7963116**

Tras incluir todas las características utilizadas en el entrenamiento sin PCA y transformarlas usando este algoritmo, regresión logística es el modelo que mejores resultados ha mostrado. Para ello se ha limitado la reducción de la dimensionalidad hasta 7 variables ya que analizando la proporción acumulativa de la información que se perdía (0.933) y la desviación estándar (0.77) representaba un valor adecuado para no perder calidad en los datos introducidos pudiendo, a la vez, reducir el número de características.

5. Conclusión

Dependiendo del modelo escogido se ha podido comprobar que el porcentaje de acierto aumenta o disminuye. Sin embargo, la mayoría se asemejan bastante y donde reside realmente la verdadera mejoría en las predicciones es en la selección de las características. Añadiendo, poco a poco, nuevas columnas al entrenamiento que contienen información relevante para la discriminación es cómo realmente se ha contribuido a la evolución del modelo.

También se ha podido comprobar como la construcción del dataset mientras se tiene en cuenta que información va a ser importante extraer para alimentar el algoritmo es clave para separar las distintas clases, porque cuanto más alejado esté la información contenida en una clase de las demás más asequible será para el modelo diferenciar entre esta y el resto.

Por último, recalcar también la importancia de una buena visualización de las características y de los datos porque además de ayudar a entender como predice el modelo y en base a que datos toma las decisiones, ha ayudado a encontrar varios errores en el código como la falta de la normalización de las características antes de ser consumidas por el modelo, que posteriormente se corrigió.