UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Informática

Estimación del tamaño de las "bounding box" de palabras en texto manuscrito a partir de su transcripción

Trabajo Fin de Grado

Grado en Ingeniería Informática

AUTOR/A: Maroto Llamas, Javier

Tutor/a: Martínez Hinarejos, Carlos David

CURSO ACADÉMICO: 2021/2022

Resumen

Debido al incremento en los documentos manuscritos digitalizados, estimar el tamaño de las palabras escritas por un autor resulta relevante en el campo de la visión por computador, pues permite estimar el tamaño de las ventanas de búsqueda en las imágenes. Siendo un modelo de *Query-by-String*, permite la búsqueda de cualquier palabra aunque no esté en los datos de entrenamiento. Esto tiene aplicaciones tanto en el procesado de documentos como en la detección de palabras clave o el reconocimiento de entidades nombradas.

En este trabajo, se estudia este problema y se realiza una aproximación aplicando distintas técnicas, métodos y clasificadores de aprendizaje automático. Se expone así el conjunto de datos obtenido, la metodología usada para obtener los resultados y los resultados finales, que dentro de las limitaciones son bastante favorables.

Palabras clave: Procesado de texto manuscrito, procesado de documentos, aprendizaje automático, Modelos estadísticos de predicción, Procesado de imágenes, Detección de palabras (*word spotting*) en texto manuscrito, visión por computador, Bayes, *Random forest*

Abstract

Due to the increase in digitized handwritten documents, estimating the size of the words written by an author is relevant in the field of computer vision, as it allows estimating the size of search windows in images. Being a Query-By-String model, it allows the search of any word even if it was not included in the training dataset. This has key applications in document processing tasks such as keyword spotting or named-entity recognition.

In this work, this problem is studied and an approximation is made by applying different techniques, methods and classifiers of machine learning. The dataset obtained, the methodology used to obtain the results and the final results, which within the limitations are quite favorable, are thus presented.

Key words: Handwritten text processing, Document processing, Statistical prediction models, machine learning, Image processing, Keyword Spotting on handwritten text, computer vision, Naive Bayes, Random forest

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

A Carlos, por ayudarme dirigiendo este trabajo y solucionando todas mis dudas con una abierta y amable comunicación.

A mis amigos y compañeros, pues su apoyo en lo emocional y en lo académico ha sido imprescindible durante todo el grado.

A mi familia, especialmente a mis padres, pues sin su apoyo nada de esto habría sido posible. **Os quiero** dedicar este párrafo, pues creo que me ha representado últimamente:

«Estoy haciendo progreso, pero ya he perdido la esperanza de predecir cuándo llegaré al final, cuándo estará listo. Cada vez que lo hago, que intento predecir, no logro cumplir con esa fecha y todos se molestan conmigo. No tiene sentido. Estará listo cuando esté listo» - *George R.R. Martin*

CAPÍTULO 1
Introducción

1.1 Motivación

Los recientes avances en el campo de la inteligencia artificial y la digitalización han permitido el acceso a miles de documentos manuscritos escaneados y colecciones. Sin embargo la variabilidad y ambigüedad de los trazos, que varían de persona a persona y en el tiempo, junto a lo costoso que es el proceso de obtener un conjunto de datos etiquetado, hace complicada la tarea de automatizar las transcripciones en los mismos.

Una tarea habitual sobre documentos manuscritos escaneados es la búsqueda de una palabra en los mismos. Para esta tarea se puede optar como aproximación la realización de una transcripción, con posibles alternativas, y la posterior búsqueda de la palabra en esa transcripción, pero esto tiene el problema de que es necesario realizar una transcripción automática y no siempre se tiene disponible ese tipo de herramienta. La alternativa consiste en una búsqueda directa por imagen, pero en este caso es necesario saber el tamaño de la subimagen a buscar dentro de la imagen del documento, a fin de determinar la ventana de búsqueda.

El cuadro delimitador de las palabras, al cuál nos referimos en este trabajo como *bounding box*, es la caja que incluye toda la palabra con todas sus letras y a tamaño completo, sin dejar trazos que cubrir.

Conocer el tamaño de la *bounding box* permite delimitar esa ventana de búsqueda y por tanto habilita la búsqueda mediante comparativa gráfica o por la transcripción de los contenidos que ocupa.

Una de las ventajas de poder estimar una ventana de búsqueda frente a los métodos de indexado basados en palabras es el hecho de poder predecir el tamaño de palabras que no están presentes en el vocabulario estudiado, pues para estimar el tamaño de una palabra nos basamos en su composición a nivel de carácter.

Las aplicaciones relacionadas son claves en el procesado de documentos: detección de palabras clave, reconocimiento de entidades nombradas, estadísticas de vocabulario, etc.

1.2 Objetivos

El objetivo principal del trabajo es que, a partir de un conjunto de imágenes de las cuales se tiene su transcripción, se estime un modelo que, dada una transcripción de una palabra, determine el tamaño óptimo de su "bounding box", a fin de establecer la ventana de búsqueda en la imagen. Hemos separado los objetivos de esta forma:

- Crear un corpus y discretizar apropiadamente los datos obtenidos.
- Entrenar un modelo que dada una palabra estime su altura y anchura, ya sea por combinación de dos modelos que estimen de forma independiente altura y anchura o uno que lo estime de manera conjunta.
- Probar la calidad de los modelos y las diferentes mejoras propuestas.

1.3 Metodología

Con el fin de alcanzar los objetivos de este trabajo, se plantearon inicialmente los siguientes pasos:

- Obtener las muestras de las *bounding boxes*.

  Una vez realizada esta tarea, debido al limitado tiempo y esfuerzo del que se dispone, es posible que obtengamos un pequeño número de muestras pero suficiente para nuestros objetivos.
- Tratar la estimación como un problema de clasificación en grupos de tamaños, por ejemplo, entre 50 y 60 píxeles se asociaría a una clase, entre 60 y 70 se asociaría a otra. Por tanto, discretizar las muestras de acuerdo a ello.
- Construir un modelo para estimar la altura y otro para la anchura en base a la idea de usar los caracteres que componen su transcripción como características relevantes.
- Combinar ambos modelos para estimar la anchura y altura de manera conjunta.
- Analizar los resultados obtenidos y proponer mejoras.

1.4 Estructura de la memoria

Este documento se divide en un total de 7 capítulos:

- En el capítulo 2 se describen trabajos ya realizados en campos cercanos a este trabajo, tanto los problemas como los procedimientos y métricas experimentados en éstos.
- En el capítulo 3 se detallan conceptos teóricos clave que se han usado durante la metodología: los modelos de aprendizaje automático usados, sus métricas más comunes y la representación de las muestras.
- En el capítulo 4 se exponen los problemas a los que nos enfrentamos en este trabajo y sus soluciones, justificando la elección de los distintos clasificadores, las métricas y las técnicas.

- En el capítulo 5 se muestran las tecnologías utilizadas y el análisis del conjunto de datos.
- En el capítulo 6 se presentan y comparan los resultados de las distintas pruebas efectuadas en los clasificadores ya entrenados.
- Finalmente, en el capítulo 7 se concluye el trabajo, exponiendo los objetivos cumplidos y relacionando lo trabajado con los conocimientos adquiridos durante el grado.

Adicionalmente, en el apéndice A se encuentra un análisis de las características resultantes del discretizado efectuado en el capítulo 4.

CAPÍTULO 2
Estado del arte

2.1 Transcripción y reconocimiento de texto manuscrito

Una de las aplicaciones que tiene estimar las *bounding boxes* de las palabras es poder realizar la búsqueda en un texto manuscrito a partir de dividir la imagen en ventanas del tamaño de la palabra buscada.

Para esta finalidad se puede optar por realizar la transcripción directa de los documentos mediante modelos de inteligencia artificial como el de la plataforma Transkribus.

No obstante, es una tarea compleja, como podemos comprobar mediante la demostración disponible en su página oficial.[1] Dependiendo del idioma, de la complejidad de los trazos y la legibilidad del texto manuscrito, puede resultar no ser muy precisa; por tanto, habría que entrenar modelos específicos. En la Figura 2.1 se ve claramente esto.

Figura 2.1: Ejemplo de una transcripción de alemán realizada mediante la plataforma Transkribus usando el modelo German Kurrent M2.

[1] https://readcoop.eu/transkribus/

Muchos de los sistemas o modelos para la transcripción funcionan a nivel de línea, transformando la línea de la imagen en vectores de características e introduciéndolos en un modelo óptico para reconocer el texto manuscrito [38]. Estos pueden ser modelos para la transcripción de texto manuscrito, siendo los más tradicionales los basados en el indexado y el uso de modelos ocultos de Markov (HMM) [28] junto al algoritmo de Viterbi.

Más recientemente se ha estudiado el uso redes neuronales recurrentes convolucionales (CRNNs), con buenos resultados como en [1], donde se utilizan para reconocer prescripciones médicas manuscritas, o en [7], donde se utiliza un entrenamiento progresivo usando pocos datos etiquetados para entrenar la red.

En las CRNNs se destaca usualmente el uso de unidades LSTMs *Long short-term memory* o BLSTMs *Bidirectional Long short-term memory* [33] para realizar el trabajo de decodificación y recordar el contexto, con el fin de solucionar el fallo de las RNNs para recordar el contexto.

En cuanto al uso de CRNNs en documentos históricos manuscritos, en [23] se estudia su uso para la transcripción de texto histórico manuscrito en griego antiguo, con resultados positivos, como también se ha visto en otros estudios, como por ejemplo en [7].

Se ha estudiado también el uso de detección a nivel de documento y a nivel de párrafo [3], con resultados interesantes a pesar de que los resultados de los sistemas a nivel de línea siguen siendo superiores.

Todos estos sistemas son complejos, costosos y requieren del ajuste de muchos hiperparámetros. Además, como se menciona en [31], todavía hay problemas como los documentos deteriorados, las diferentes condiciones de escaneo, los signos de puntuación en los finales de línea, etc. Es por esto que puede ser más eficiente el hecho de realizar una búsqueda de palabras sobre texto manuscrito sin hacer la transcripción.

2.2 Detección o búsqueda de palabras en texto manuscrito

La detección de palabras o *Word Spotting* es una tarea del ámbito de la recuperación de la información y es más cercano a la detección de objetos visuales. El concepto deja de ser el reconocimiento de la palabra y pasa a ser un problema de *pattern matching*, con dos posibles fines: obtener la probabilidad de que una palabra se encuentre en un documento o delimitar todas las apariciones de cierta palabra dada una imagen.

En [11] se clasifican muchos de los modelos según la entrada que le demos al sistema en:

- *Query-by-String* [13]: La entrada es una palabra escrita. Los modelos de los caracteres se aprenden de manera *offline* por separado y se combinan para obtener las probabilidades de las palabras. La Figura 2.2 muestra un sistema de este tipo.
- *Query-by-Example* [29]: La entrada es una imagen de la palabra a buscar y se obtiene en qué zonas de la imagen objetivo aparece la palabra indicada.

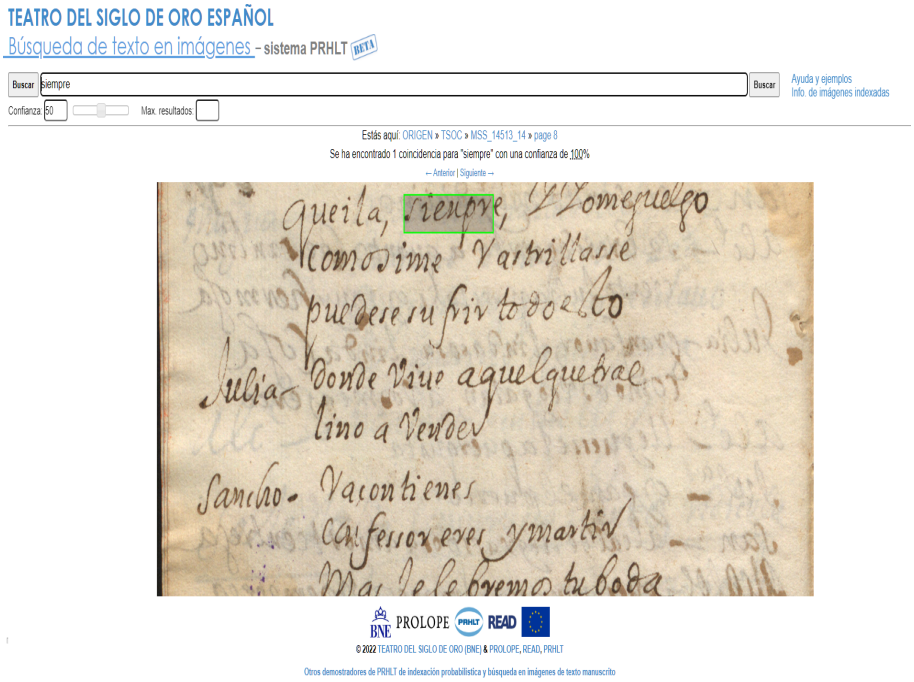

Figura 2.2: Demo del *Pattern Recognition and Human Language Technology* Research center (PRHLT) que usa CRNNs sobre textos del teatro del siglo de oro. En verde, la *bounding box* correspondiente para la palabra "siempre".

Además, en [11] se evalúa positivamente añadir información semántica (contextual) en el proceso de *Word Spotting*.

A modo de ejemplo, el modelo propuesto en [2] acepta como entrada tanto texto como imágenes, llevándolo a un espacio de representación más discriminatorio que permite que el sistema sea además multiautor.

Muchas de las aproximaciones al problema se basan en el uso de la técnica de *Dynamic Time Warping* [14] para la comparación de las imágenes [4] y HMMs [12].

En concreto, en [4] se presenta un método para indexar de forma semiautomática colecciones de documentos, basado en la extracción de características mediante histogramas de gradientes orientados (HOG) y el uso de una ventana deslizante o *sliding window* que recorre las imágenes, para luego poder aplicar la técnica de *Dynamic Time Warping* sobre las imágenes. En [4] se aplica a las *sliding windows* un tamaño estimado experimentalmente en [36]. Este trabajo podría ser de utilidad para darle más exactitud o flexibilidad al tamaño de estas ventanas.

También es posible clasificar muchas de las soluciones propuestas en dos grupos:

- Basados en la segmentación de las imágenes, *Segmentation-based*:
  - Con segmentación a nivel de línea, *Line-segmentation* [17].
  - Con segmentación a nivel de palabra, *Word-segmentation* [21].
- Libres de segmentación, *Segmentation-free* [12, 29, 14].

La segmentación de las imágenes no es un problema trivial, sino que es responsable de buena parte de las complicaciones que tiene el reconocimiento y la búsqueda de texto manuscrito [20]. De hecho, muchos trabajos hacen grandes asunciones en esta materia, como por ejemplo que se pueden binarizar las imágenes y obtenerlas en blanco sobre fondo negro, como ocurre en [21].

Es por esta razón que surgen trabajos en los que se busca reducir o eliminar el preproceso necesario de segmentación de las palabras en los documentos, dado que fallos en la segmentación suelen inducir fallos en el reconocimiento [12]. Véase, por ejemplo, [29], en el cual se propone el uso de una representación *Bag-of-Features*, extrayendo características de las imágenes y realizando agrupaciones, junto a un HMM (ya que son muy efectivos codificando información secuencial), que se estima solamente con el procesado de la *query*, para luego realizar la comparación sobre el documento.

Uno de los grandes problemas que suelen experimentar o intentan solucionar todos estos trabajos son, como se explica en [2], la imposibilidad de la búsqueda de palabras que estén fuera del vocabulario de entrenamiento y el tiempo que se requiere para realizar estas tareas de búsqueda. Además, buena parte de las soluciones propuestas son del tipo *Query-by-Example* y por tanto están limitadas a palabras de las que dispongamos de ejemplos. Este trabajo busca estudiar una solución a estos problemas.

2.3 Métricas para la evaluación de detectores de objetos

Con el propósito de valorar la optimalidad de las soluciones y poder realizar comparaciones en el área de la visión por computador y más concretamente en la segmentación semántica (en la detección de *bounding boxes*), se han utilizado distintas métricas en diversos trabajos. Algunas de las más comunes son:

- *Pixel accuracy*: No es considerada la mejor métrica y es reemplazada por muchos autores por el índice de Jaccard, debido a los problemas que genera el desequilibrio de las clases.
- Índice de Jaccard o *Intersection over Union* (IoU) [9, 16]: La más recurrente, es la región solapada entre el total de las regiones.
- *Dice coefficient* o *F1 score* [34]: Similar al índice de Jaccard. No obstante, penaliza menos los errores de clasificación, es más similar a la "media" mientras que en IoU sería "el peor caso".
- *Local Consistency Error (LCE)* y *Global Consistency Error (GCE* [19, 35, 24]: Métricas que permiten diferenciar si una región es subconjunto de otra.

En el campo del reconocimiento de objetos la usual elección, por ejemplo para los *datasets PASCAL Visual Object Classes (VOC)* [10] y *MS Common Objects in Context* [18] (COCO) suele ser una combinación junto a otra métrica, *Mean Average Precision* (mAP), que a su vez, hace uso de la métrica *Average Precision* (AP) y ésta de las de precisión y exhaustividad (*precision and recall*), estimando éstas últimas mediante *Intersection over Union*.

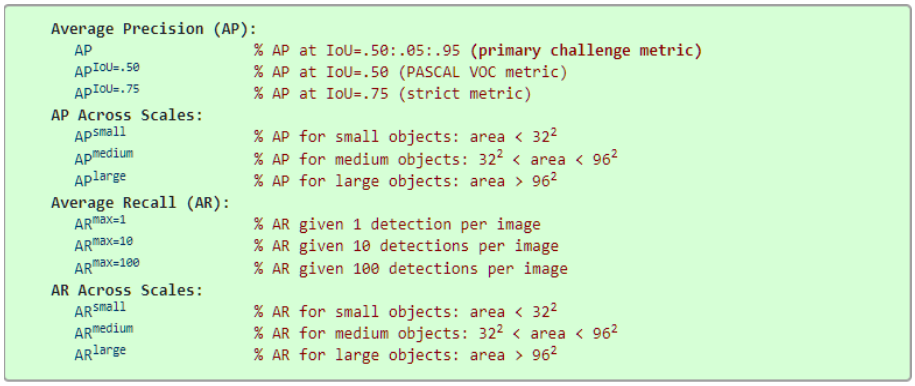

Figura 2.3: Métricas usadas en el dataset COCO. [2]

Para el caso de COCO, por ejemplo, la mAP (que en la Figura 2.3 es llamada AP) se calcula colocando umbrales a la IoU, desde 0.50 hasta 0.95 en rangos de 0.05. Es decir, si el valor de la IoU es mayor que el umbral entonces es un acierto. También definen el *Average Recall* como el máximo *recall* dado un número fijo de detecciones por imagen, haciendo la media entre todas las categorías e IoU.

Hemos visto cómo la selección de las métricas apropiadas no es sólo crucial sino que es un campo muy variado, habiendo muchas distintas para cada tipo de problema. La elección de las métricas para este trabajo se discute detalladamente en el capítulo 4: análisis del problema.

[2]Imagen extraída de: https://cocodataset.org/#detection-eval

CAPÍTULO 3
Conceptos teóricos

En este capítulo se introducen los conceptos teóricos de las tecnologías o procedimientos utilizados en este trabajo, de forma muy centrada de cara a los siguientes capítulos.

3.1 Aproximación *Bag of words*

El texto no puede ser tratado de forma directa por los algoritmos de aprendizaje automático, sino que requiere de una primera extracción de características y transformación en vectores.

El modelo de representación de la bolsa de palabras o *Bag of words* es muy útil para la representación de texto y fácil de implementar. Es una bolsa de palabras porque se pierde la información sobre el orden en el que aparecen y sólo se tiene en cuenta la presencia o ausencia de cada palabra [26]. La Figura 3.1 ilustra un ejemplo en el que se puede ver cómo se pierde el orden de la oración.

	about	bird	heard	is	the	word	you
About the bird, the bird, bird bird bird	1	5	0	0	2	0	0
You heard about the bird	1	1	1	0	1	0	1
The bird is the word	0	1	0	1	2	1	0

Figura 3.1: Ejemplo de una representación *Bag of Words*. [1]

Para el problema de la extracción de características en las palabras se ha propuesto un modelo de representación *Bag of letters* que funciona de manera similar: el vector de características viene dado por las apariciones de las letras en la palabra. Así el vocabulario viene dado por las letras presentes en el conjunto de datos.

[1] Imagen extraída de: https://openclassrooms.com/en/courses/6532301-introduction-to-natural-language-processin
6980811-apply-a-simple-bag-of-words-approach

La principal desventaja es la pérdida del orden, el cual podría ser importante. Por ejemplo, en el caso de las palabras, no se considera que el autor escriba la letra "p" de manera diferente al inicio de palabra. No obstante, no es plausible tenerlo en cuenta en el contexto de este trabajo puesto que la cantidad de datos es lo bastante limitada como para que obtengamos cantidades poco representativas de ciertas regiones de decisión que pueden engañar al clasificador. También está la desventaja del tamaño de vocabulario, que podría darnos matrices muy grandes, pero éste está limitado en nuestro caso particular, a como máximo, las letras de la "a" a la "z" tanto mayúsculas como minúsculas y números del 0 al 9.

3.2 Clasificadores ingenuos de Bayes

Los clasificadores ingenuos de Bayes o *naive Bayes* son de los clasificadores probabilísticos más fáciles de entender pero que a su vez obtienen resultados sorprendentemente buenos en un gran abanico de problemas [40], entre los más comunes, aquellos relacionados con la clasificación de texto [39].

Están basados en el teorema de Bayes, una proposición planteada por Thomas Bayes [37]. En ésta se postula que, dado un conjunto de sucesos mutuamente excluyentes y exhaustivos, $X = \{X_1, X_2, ..., X_n\}$, y con probabilidades de cada uno distintas de cero, $\forall x \in X,\ x \neq 0$, entonces la probabilidad *a posteriori* viene dada por esta expresión:

$$P(X_i|Y) = \frac{P(Y|X_i)P(X_i)}{P(Y)}$$

Siendo:

- $P(Y|X_i)$: la probabilidad del suceso Y en la hipótesis X_i.
- $P(X_i)$ y $P(Y)$: las probabilidades a priori.

El algoritmo de clasificación es de la rama del aprendizaje supervisado y asume una independencia condicional fuerte entre las características de los datos, es decir, no hay dependencias entre las características dada una clase. De esta forma, evitamos que los parámetros a estimar escalen exponencialmente.

Sea un vector $\vec{x}$ que contenga C características e Y, una variable aleatoria que puede tomar su valor entre K distintas clases:

$$\vec{x} = (X_1, X_2, X_3, ..., X_C)^T$$

$$Y = \{y_1, y_2, y_3, ..., y_K\}$$

Usualmente, en este tipo de clasificadores se busca maximizar la probabilidad a posteriori (*MAP*).

Entonces, lo que queremos calcular es:

$$p(Y|\vec{x})$$

Aplicando la definición de la probabilidad condicional:

$$p(Y|\vec{x}) = \frac{p(\vec{x}, Y)}{p(\vec{x})}$$

En la práctica, no nos interesa el denominador ya que no depende de Y y es constante. Además, gracias a la regla de la cadena podemos reescribir la expresión:

$$p(X_1, X_2, ..., X_C, Y) = p(X_1|X_2, ..., X_C, Y)p(X_2|X_3, ..., X_C, Y)p(X_C|Y)p(Y)$$

Esto induce un problema que ya habíamos comentado antes: el de los parámetros a estimar. Asumiendo variables binarias, el número a estimar sería $2^C K - 1$ parámetros.

Aquí es cuando se realizan las asunciones que simplifican el modelo, la parte *ingenua*. Se asume que cada X_i es condicionalmente independiente, es decir, independiente del resto cuando están condicionadas a una clase Y, aunque esta asunción claramente no se cumple en la práctica.

Por tanto, esta asunción hará que la probabilidad final posterior sea más cercana a 0 o 1 de lo que deberían, es decir, el modelo está demasiado seguro de sus predicciones. De todas formas, las predicciones siguen siendo muchas veces lo bastante precisas [30].

$$P(X_1|X_2, ..., X_C, Y) = p(X_1|Y)$$

$$P(\vec{x}, Y) = p(Y)\prod_{i=1}^{C} p(X_i|Y)$$

Y por tanto:

$$P(Y|\vec{x}) \propto p(Y)\prod_{i=1}^{C} p(X_i|Y)$$

Lo que reduce drásticamente el número de parámetros a estimar a $K - 1 + KC$ parámetros.

No obstante, todavía hay un detalle numérico a tener en cuenta en el caso práctico del producto, ya que vamos a estar multiplicando valores pequeños cercanos a 0. Por tanto, el uso del logaritmo es útil, ya que además es una función monótona que nos permite conservar el orden.

$$\log p(Y) + \sum_{i=1}^{C} \log p(X_i|Y)$$

3.2.1. Estimación de parámetros

En cuanto a la estimación de las probabilidades a priori se pueden suponer equiprobables o hallar la distribución de las clases en el conjunto de datos; ésta última es la utilizada en este trabajo.

Para la estimación de los parámetros de las características, $p(\vec{x}|Y)$, lo usual es suponer una distribución probabilística de las características basada en el problema a resolver y, por tanto, los clasificadores se dividirán respecto a la distribución escogida.

Clasificador Bernoulli

En el modelo que se basa en la distribución Bernoulli, el vector de características es un vector booleano que indica la presencia o ausencia de éstas. Entonces, la probabilidad dada una clase es:

$$p(\vec{x}|Y) = \prod_{i=1}^{n} p_Y^{X_i}(1 - p_Y)^{(1-X_i)}$$

Siendo $p_Y^{X_i}$ la probabilidad de que la clase "Y" contenga la característica "i".

En la Figura 3.2 se expone el algoritmo utilizado para el entrenamiento y prueba.

```
1: TRAINBERNOULLINB(C, D)
2: V ← EXTRACTVOCABULARY(D)
3: V ← COUNTDOCS(D)
4: for all c ∈ C do
5:    N_c ← COUNTDOCSINCLASS(D, c)
6:    prior[c] ← N_c/N
7:    for all t ∈ V do
8:       condprob[t][c] ← (N_c + 1)/(N + 2)
9:    end for
10: end for
11: return V, prior, condprob
12: APPLYBERNOULLINB(C, V, prior, condprob, d)
13: V_d ← EXTRACTTERMSFROMDOC(V, d)
14: for all c ∈ C do
15:    score[c] ← log prior(c)
16:    for all t ∈ V do
17:       if t ∈ V_d then
18:          score[c]+ = log condprob[t][c]
19:       else
20:          score[c]+ = log(1 − condprob[t][c])
21:       end if
22:    end for
23: end for
24: return argmax_{c∈C} score[c]
```

Figura 3.2: Algoritmo *naive bayes* para el modelo Bernoulli, tanto entrenamiento como prueba. El suavizado en la línea 8 es el suavizado de Laplace. [2]

[2] Algoritmo extraído de: https://nlp.stanford.edu/IR-book/html/htmledition/the-bernoulli-model-1.html. Versión html del libro [22]

Clasificador Multinomial

En este caso, el vector de características es un vector que representa las frecuencias, de tal manera que es un histograma de éstas. En este caso, la probabilidad de observar un histograma $\vec{x}$ es:

$$p(\vec{x}|Y) = \frac{(\sum_{i=1}^{n} X_i)!}{\prod_{i=1}^{n} X_i!} \prod_{i=1}^{n} p_Y^{X_i}$$

Características no presentes

Teniendo en cuenta nuestro caso de estudio, sabemos que es probable que hayan algunas letras que son menos comunes y es posible que no aparezcan o aparezcan muy pocas veces entre los datos escogidos.

Si para una clase y característica no hay ocurrencias en el conjunto de entrenamiento entonces la probablidad estimada será 0. Para evitar esto se suele regularizar el clasificador usando el suavizado de Laplace (cuando se añade un recuento de más) o el suavizado de Lidstone (en el caso general).

Esto, aun así, puede inducir a errores en la estimación, sobre todo cuando se usan clasificadores ingenuos de Bayes. No obstante, tenemos la certeza de que ciertas letras son similares entre sí; por ejemplo, en cuanto a altura, la aproximación de una ñ a una n no es para nada absurdo. De esta forma además podemos reducir la maldición de la dimensionalidad, por la cual la cantidad de datos para estimar nuestros parámetros crece enormemente con el número de características.

3.3 Árboles aleatorios, *Random forests*

Random forest es una técnica de aprendizaje supervisado por conjuntos basada en árboles de decisión [6]. Éste construye y combina la salida de los distintos árboles. Si lo usamos para la clasificación, la salida es la clase seleccionada por la mayoría; sin embargo, para regresión se usa la media de las salidas de los árboles individuales.

3.3.1. Árboles de decisión

Los árboles de decisión son modelos muy buenos para predecir relaciones líneales y no líneales entre las características y el resultado. Se puede pensar en ellos como en un conjunto de *if-else*, de forma que en cada nodo se toma una decisión, buscando la mejor forma de seccionar los datos.

A pesar de ser prácticos, los árboles de decisión tienden al *overfit*, es decir, al sobreentrenamiento para encontrar los resultados deseados. Ésta es una de las desventajas que intenta corregir el *random forest*.

3.3.2. Aprendizaje por conjuntos (*ensemble learning*)

El aprendizaje por conjuntos se basa en la idea de que un grupo de clasificadores, en este caso árboles de decisión, pueden generar otro clasificador. Entre los más conocidos están el *bootstrap aggregating (bagging)* [5] y el *boosting* [32]. El *bagging*, usado en los *random*

forests, consiste en seleccionar grupos de N muestras aleatorias con reemplazamiento (es decir, la misma muestra puede aparecer más de una vez en el entrenamiento), y ajustar cada uno de los árboles de decisión en base a un grupo de estas muestras. La Figura 3.3 ilustra su funcionamiento.

Esto incrementa la varianza sin incrementar el sesgo, pues aunque un árbol de decisión pueda ser sensible al ruido en el entrenamiento, la media de estos árboles no lo es siempre y cuando no estén correlacionados. Mediante el *bagging* evitamos esta correlación.

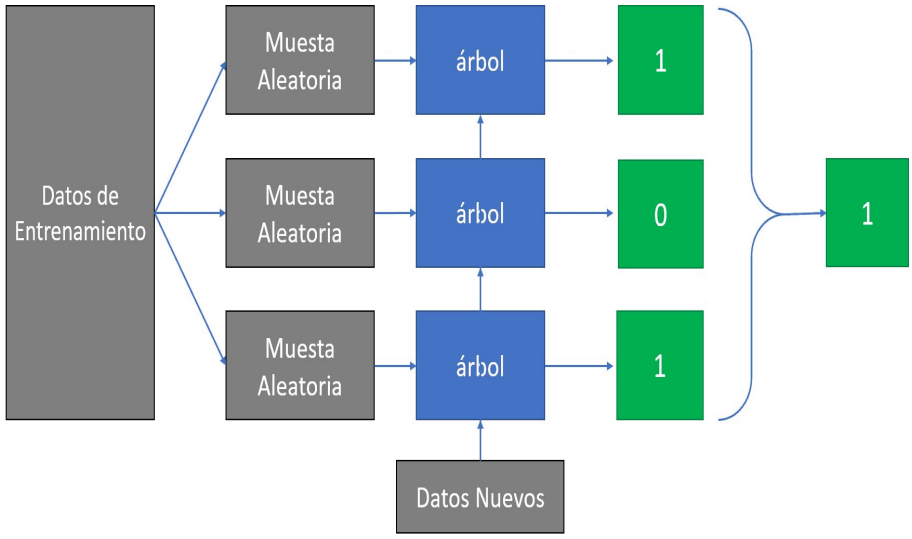

Figura 3.3: Diagrama informativo sobre cómo funciona en terminos generales el algoritmo *random forest*. [3]

Además de esto, el algoritmo de aprendizaje *random forest* también hace una separación en subconjuntos de las características, porque si algunas de las características fueran muy determinantes en los árboles de decisión, entonces podrían correlacionarse porque serían seleccionadas por muchos árboles [15].

3.4 Validación cruzada

La validación cruzada es una técnica para evaluar resultados y garantizar que son independientes de la partición de datos de entrenamiento y prueba, con el fin de evaluar la estabilidad y cómo se comportarán los modelos frente a datos nuevos y reales.

Hay varias formas de aplicar esta técnica, siendo una de las más comunes la validación cruzada de K iteraciones. Ésta consiste en la separación de los datos en K grupos, realizando el entrenamiento del modelo con K-1 grupos y un conjunto como datos de prueba, junto a la posterior media aritmética ponderada de los resultados.

[3] Imagen extraída de: https://www.iartificial.net/random-forest-bosque-aleatorio/

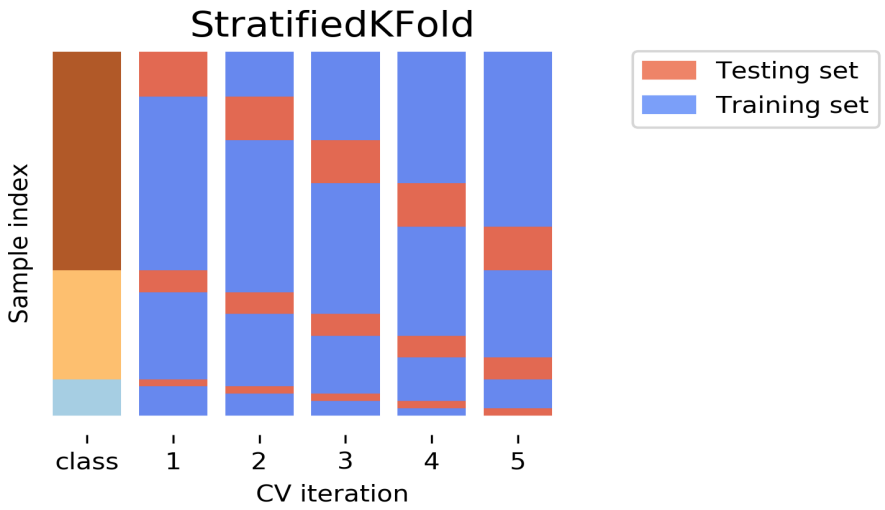

Figura 3.4: Diagrama que muestra la separación de los grupos en la variante *stratified* de la validación cruzada. [4]

En el caso concreto de este trabajo, se utilizará una variante *stratified*. Esto se debe a que cuando se dispone de un conjunto de datos desequilibrado entre las clases, realizar la separación de forma aleatoria o de forma secuencial no es lo óptimo, pues no vamos a mantener la proporción entre las clases y no vamos a poder evaluar correctamente los modelos. La diferencia radica entonces en mantener la proporción de cada clase durante la separación en K grupos (con como mucho 1 muestra de diferencia). En la Figura 3.4 se observa la separación de los datos de prueba y entrenamiento de forma proporcional a las clases.

El principal problema que suele presentar es la lentitud computacional, pues hay que repetir el proceso de entrenamiento y prueba durante K iteraciones. No obstante, esto no será una complicación en la práctica puesto que los modelos a implementar no son tan costosos computacionalmente ni disponemos de tantos datos como para que resulte un reto computacional.

3.5 *Statistical data binning*

Con el fin de agrupar los datos continuos o semicontinuos utilizamos una técnica de preprocesado de datos llamada *statistical data binning*. Consiste en agrupar los valores tal que obtengamos ciertos intervalos, por ejemplo, agrupar los valores entre 100 y 150 bajo el mismo intervalo, los de 150 a 200 bajo otro distinto, etc. Se puede apreciar de forma clara en el ejemplo de la Figura 3.5.

[4] Imagen extraída de: https://amueller.github.io/aml/04-model-evaluation/1-data-splitting-strategies.html

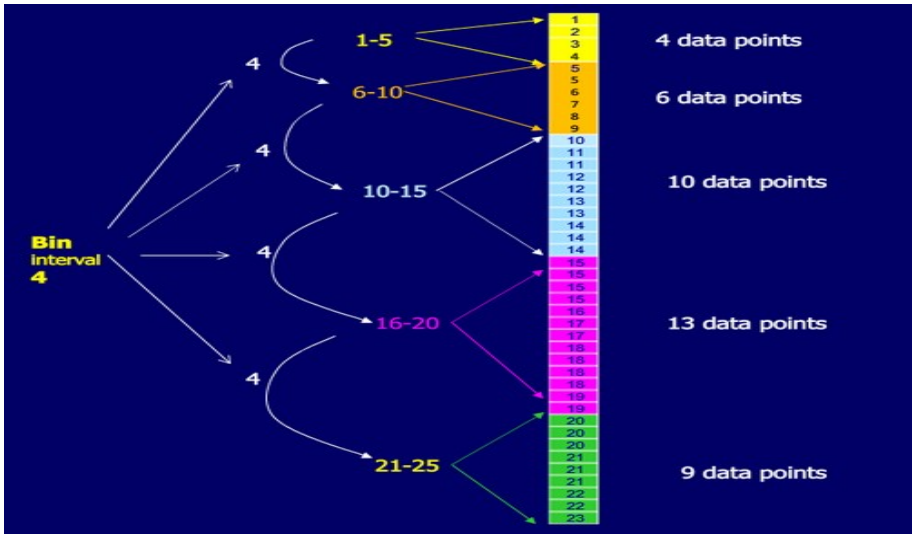

Figura 3.5: Diagrama de un ejemplo de *data binning* en intervalos de 4 valores. [5]

De hecho, esto sirve no sólo para reducir los problemas a un problemas de clasificación en clases (de valores discretos), reduciendo el ruido en el proceso, sino que también es útil para usarse en los histogramas, de tal forma que podamos discernir distribuciones de frecuencias.

3.6 *F-measure*, precisión y *recall*

Para la definición de las siguientes métricas de evaluación hay que explicar los siguientes términos:

- *True Positive* (TP): Muestras predichas positivas que se clasificaron correctamente.
- *True Negative* (TN): Muestras predichas negativas que se clasificaron correctamente.
- *False Positive* (FP): Se predijo positivo y se clasificaron incorrectamente.
- *False Negative* (FN): Se predijo negativo y se clasificaron incorrectamente.

Esto es aplicable también a problemas de clasificación en varias clases, reduciéndolo a clasificaciones binarias. Así, surgen las estrategias *One vs Rest* y *One vs One*:

- *One vs Rest*: Para cada clase, consideramos esa clase como positiva y el resto como negativas.
- *One vs One*: Para cada par de clases, consideramos una como positiva y la otra como negativa. Se repite tantas veces como combinaciones de clases haya.

Usualmente se realiza la media ponderada entre los resultados obtenidos con las estrategias mencionadas anteriormente. También es posible, no obstante, simplemente mostrar los valores por clase.

[5] Imagen extraída de: https://datacadamia.com/data_mining/discretization

Por otro lado, definiremos la precisión como:

$$precision = \frac{TP}{TP + FP}$$

Es decir, del total de muestras clasificadas como positivas, cuántas se clasificaron correctamente.

También definiremos el *recall* como:

$$recall = \frac{TP}{TP + FN}$$

Lo cual se traduce en: del total de muestras positivas, cuántas se clasificaron correctamente. La precisión nos da una medida de cómo de válidos son nuestros resultados, mientras que el *recall* nos da una medida de cómo de exhaustivos son.

Con estos conceptos podemos definir la *F-measure* como la media armónica de la precisión y el *recall*:

$$F = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

CAPÍTULO 4

Análisis del problema y propuesta de solución

4.1 Agrupación de características

En nuestro problema, tenemos un conjunto de letras que son similares en cuanto a la altura (que suele ser lo común en los textos manuscritos y variará de autor a autor). Por tanto, podemos reducir las características, simplificando el trabajo a los clasificadores ingenuos de Bayes. En tipografía, se define de la forma que muestra la Figura 4.1.

Figura 4.1: Infografía que explica las diferentes alturas que componen una palabra. [1]

De acuerdo con esto, tendríamos:

- Letras con la altura de la x: “a”, “c”, “n”...
- Letras con asta ascendente: “b”, “d”, “t”...
- Letras con asta descendente: “q”, “p”, “g”...
- Letras con asta descendente y ascendente: “f”.
- Letras mayúsculas y números.

En el caso de la estimación de las anchuras, discernir grupos similares requeriría un estudio más exhaustivo y obviamente variaría de autor a autor; no obstante, las letras más anchas suelen ser la letra “m”, la “w” y las mayúsculas, habiendo además, una correlación con el número de caracteres de la palabra.

[1] Imagen extraída de: https://es.wikipedia.org/wiki/Altura_de_la_x

4.2 Comparación de dos *bounding boxes*, elección de las métricas

Usualmente, para la comparación de *bounding boxes* se utiliza el coeficiente de Jaccard para hacer la comparación del solapamiento entre áreas [27].

El coeficiente de Jaccard mide el grado de similaridad entre conjuntos. Éste también se conoce como *Intersection over Union (IoU)* por su característica definición:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

Ésta definición se expresa de manera visual en la Figura 4.2

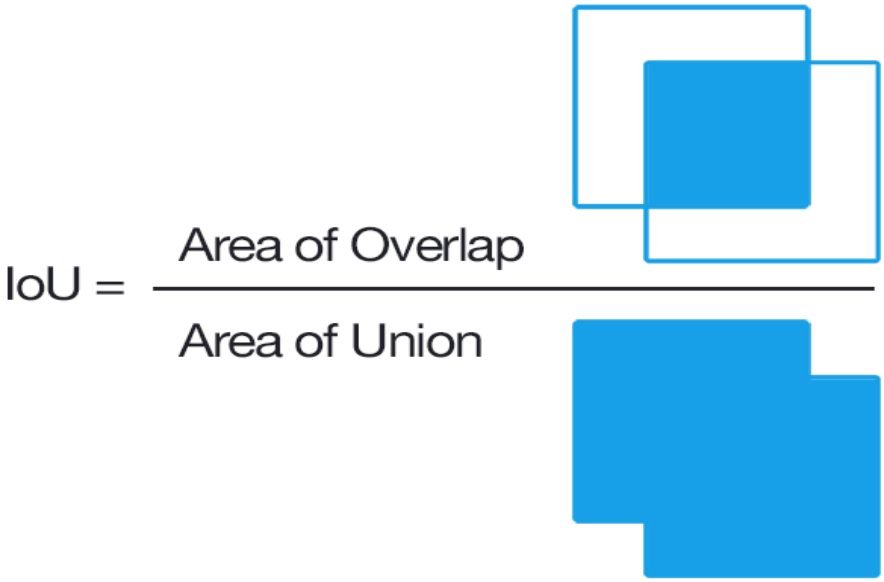

Figura 4.2: Cálculo de la IoU expresado de forma visual. Arriba se muestra la intersección de las regiones de las áreas y abajo la unión de ellas. [2]

La distancia de Jaccard, sin embargo, mide el grado de disimilud; por tanto, es básicamente el concepto complementario al anterior.

$$D_J(A,B) = 1 - J(A,B)$$

[2]Imagen extraída de: https://en.wikipedia.org/wiki/Jaccard_index#

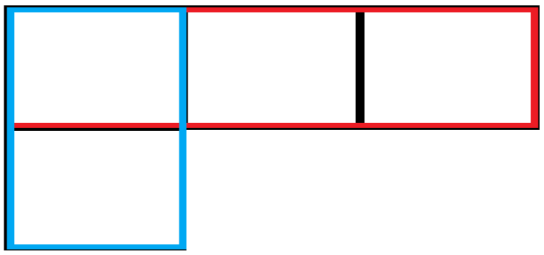

Figura 4.3: Ejemplo ilustrativo sobre cómo dos *bounding boxes* se comparan entre sí. En rojo, una de ellas, en azul, la otra.

El ejemplo de la Figura 4.3 ilustra cómo se aplicará en nuestro caso particular. Normalmente, la comparación de las *bounding boxes* se realiza tomando en cuenta la posición concreta de cada una aparte de su tamaño. Sin embargo, en nuestro caso no es necesario y bastará con el cálculo del área de cada una.

Sin embargo, la IoU no va a ser adecuada como métrica, pues una diferencia en el área predicha de las mismas unidades no va a producir un mismo resultado. Pongamos un ejemplo:

- Tenemos un área original de 20 píxeles.
- Predecimos un área de 30 píxeles, englobando por completo el área original.
- La intersección resultante es el total del área original, 20 píxeles.
- La unión es la suma de ambas, menos la región solapada: 50-20 = 30. La IoU resultante es: $\frac{2}{3}$
- Si predecimos, sin embargo, un área más pequeña que la original, digamos de 10 píxeles, la intersección resultante sería 10 píxeles.
- La unión sería 10+20-10 = 20. La IoU resultante sería $\frac{1}{2}$.

Por tanto, bajo la misma diferencia total se obtienen distintos valores, así que hay que usar otra métrica más apropiada. Se consideraron:

- Modificación al cálculo de la IoU
- *Pixel accuracy*: el porcentaje de píxeles bien clasificados. Tiene el mismo problema que la IoU, por tanto, no es adecuada.
- *Global Consistency Error* (GCE), *Local Consistency Error* (LCE)

Para ambas, GCE y LCE, se define para el píxel p_i, el *Local Refinement Error* (LRE) como:

$$E(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \backslash R(S_2, p_i)|}{|R(S_1, p_i)|}$$

Donde S_i denota la máscara (la predicha y la *ground-truth*) y $R(S_i, p_i)$ denota la región de S_i que contiene el píxel i. $|\cdot|$ Denota la cardinalidad y $\setminus$ la diferencia entre conjuntos. No es simétrica y, por tanto, hay que calcularla en ambos sentidos.

Como nuestro problema es más simple que el descrito en [24], podemos simplificarlo. Así, vamos a obtener una métrica distinta inspirada en la anterior:

$$E(B_1, B_2) = \frac{|R(B_1) \setminus R(B_2)|}{|R(B_1)|}$$

$$E(B_1, B_2) = Ratio(B_1, B_2) = \frac{|A(B_1) - A(B_2)|}{A(B_1)}$$

Siendo $A(B_1)$ el área de la *bounding box* original y $A(B_2)$ el área de la predicha. El resultado nos soluciona el problema, pero no es una métrica, pues es asimétrica. De nuevo, en la Figura 4.4 se muestra el cálculo de forma visual.

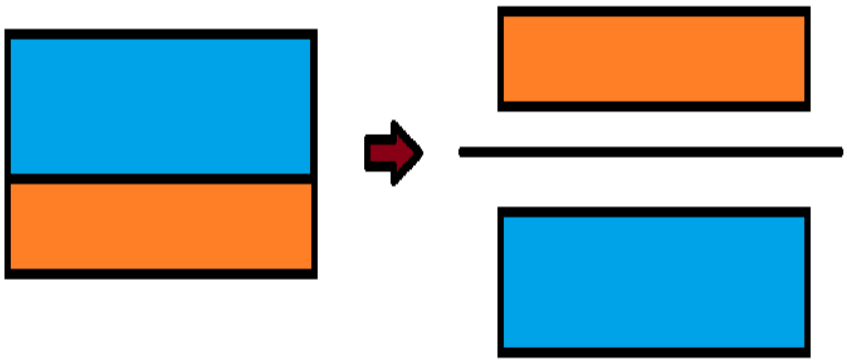

Figura 4.4: Cálculo del ratio expresado de forma visual. En el numerador se muestra la diferencia entre las áreas y en el denominador el área original. [3]

Esta aproximación es interesante pues nos permite también no sólo comparar resultados, sino obtener la relación, ya que no está definida entre 0 y 1 sino entre 0 y $+\infty$, siendo 0 la mejor aproximación y $+\infty$ la peor.

Como métrica real, que cumple la definición, usaremos la diferencia en valor absoluto de las áreas.

Usando ésta y otras métricas que resultan interesantes podemos evaluar los modelos:

- La diferencia en área, tanto la máxima como la mínima, entre todas las muestras.
- La mínima posible diferencia en área máxima entre todas las muestras (si se clasificaran correctamente todas las muestras).

[3] Imagen extraída de: https://en.wikipedia.org/wiki/Jaccard_index#

- La media de la diferencia entre áreas y la mejor media posible.
- El mejor ratio conseguido y el mejor ratio posible.
- El ratio medio y el mejor ratio medio posible.
- *Precision*, *recall* y *F-score*.

Nótese que, al estar clasificando datos continuos o casi continuos en clases, lo más probable es que no haya una solución perfecta; por tanto, para obtener y poder comparar la mejor solución respecto a la separación de los datos en clases, es crucial el cálculo de la mejor situación (correcta clasificación de todas las muestras).

4.3 Selección de los clasificadores

4.3.1. Clasificador de altura

Para la clasificación de las alturas se ha de considerar lo escrito en el apartado de agrupación de características. La altura de una *bounding box* de una palabra vendrá caracterizada por la altura de la letra más alta. Por tanto, palabras que contengan letras ascendentes o descendentes serán más altas por razones lógicas.

Teniendo en cuenta que estamos buscando la presencia o ausencia de ciertas letras, proponemos un clasificador ingenuo de Bayes basado en una distribución Bernoulli.

4.3.2. Clasificador de anchura

La anchura de la *bounding box* de una palabra, sin embargo, tendrá como características, además de las letras que la componen, el número de veces que aparece cada una. Es bajo esta suposición que proponemos un clasificador ingenuo de Bayes basado en una distribución Multinomial como solución.

4.3.3. *Random forest*

Ya hemos explicado las bondades de los árboles aleatorios y cómo se ajustan a casi cualquier tipo de datos. Son modelos muy potentes y flexibles que permiten tanto la regresión como la clasificación de datos con características categóricas.

Sin embargo, tienden a no generalizar bien nuevos datos o variables que no hayan aparecido en el conjunto de entrenamiento y su interpretación es mucho más complicada de efectuar hasta el punto que pueden llegar a parecer cajas negras.

Podemos tratar de eludir el problema de la generalización reduciendo las características a un número mucho más pequeño que el conjunto de entrenamiento y con el cual se hagan diversas combinaciones; como hemos visto, podríamos hacer esa agrupación para la altura.

En el caso de la anchura podemos analizar qué letras no aparecen y hacer algo similar; por ejemplo, si una “k” se parece en anchura a una “d” podemos agruparlas bajo la misma característica. Hay que tener en cuenta que la anchura tiene una característica más que es bastante relevante y es la longitud de la palabra.

Otra ventaja más de usar *random forest* es, además de lo anterior, el hecho de poder construir un regresor y a la vez un clasificador, pudiendo comparar ambas alternativas en términos de qué área es más parecida a la real, por ejemplo.

4.3.4. El problema del desequilibro entre clases

Gran parte de los conjuntos de datos de clasificación reales tienen desequilibro entre clases. Esto induce problemas, pues muchos de los algoritmos de clasificación suelen tener sesgo hacia las clases mayoritarias, como es el caso de *naive bayes*.

A pesar de esto hay varias maneras posibles de atacar el problema:

- Tratando los datos: como *undersampling* (casi nunca recomendado pues perdemos datos), *oversampling* (que puede incrementar el riesgo de sobreentrenamiento) o SMOTE [8] (una de las técnicas de creación de datos sintéticos, que suele funcionar mal si la generación de nuevas muestras no es precisa).

- Usando otro tipo de modelos, como *random forest*.

La familia de modelos *random forest* tiende a funcionar igual de bien para conjuntos de datos que tengan desequilibrio entre las clases. Esto es posible gracias a que se puede añadir los pesos de las clases en el algoritmo de tal forma que penalice los errores en la clase minoritaria. Además, el uso combinado de la selección de muestras con el aprendizaje por conjuntos resulta en la construcción de árboles en un conjunto de datos más equilibrado.

No solo hay problemas en la clasificación (y en la validación cruzada), sino también en la medición, pues las medidas usuales de error no suelen ser óptimas. Es por ello que usualmente se recurre al *F1-Score*, que funciona de manera fantástica bajo conjuntos de datos desequilibrados.

4.4 *Statistical data binning*

4.4.1. *Trapping rain water*

Cuando queremos discretizar valores continuos en clases hay que valorar distintas opciones de segmentación. Para realizar esta tarea podemos confiar en nuestro sentido común y habilidad para separar los datos y luego comprobar continuamente si esa separación es más o menos correcta mediante histogramas y visualización de los datos.

En esta sección se presenta una forma algo más determinista de realizarlo. Está basada en el popular problema de programación *trapping rain water*.

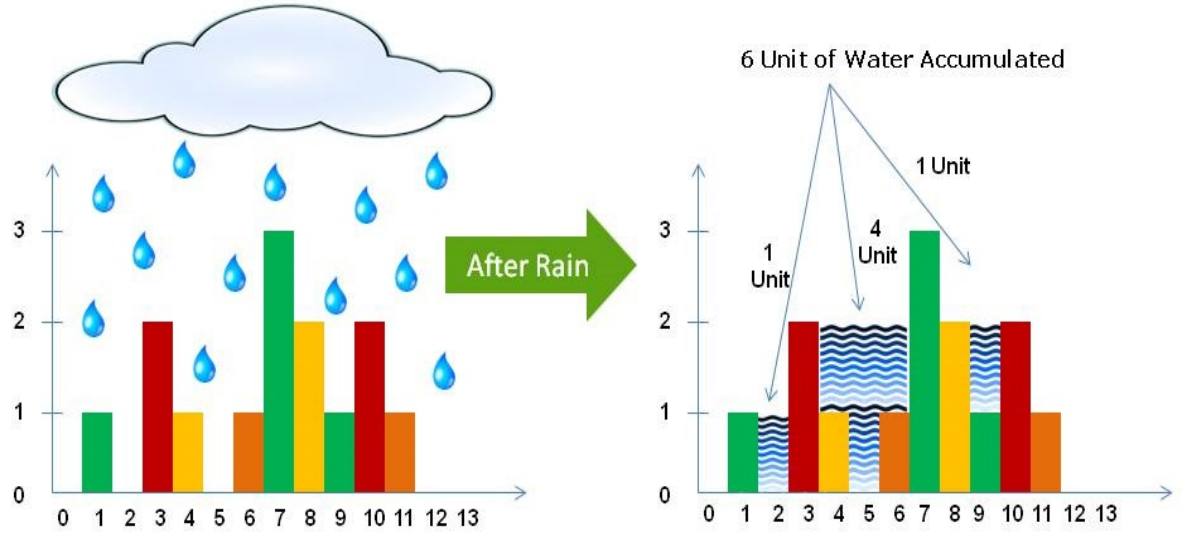

Figura 4.5: Infografía que muestra el concepto del problema *trapping rain water*.[4]

Se plantea el concepto del problema como qué ocurriría si lloviera sobre nuestro histograma. ¿Dónde se acumularía más agua? Resulta que en las zonas en las que más agua se acumula suelen ser mínimos locales aun teniendo en cuenta el ruido. La Figura 4.5 lo muestra de forma simple.

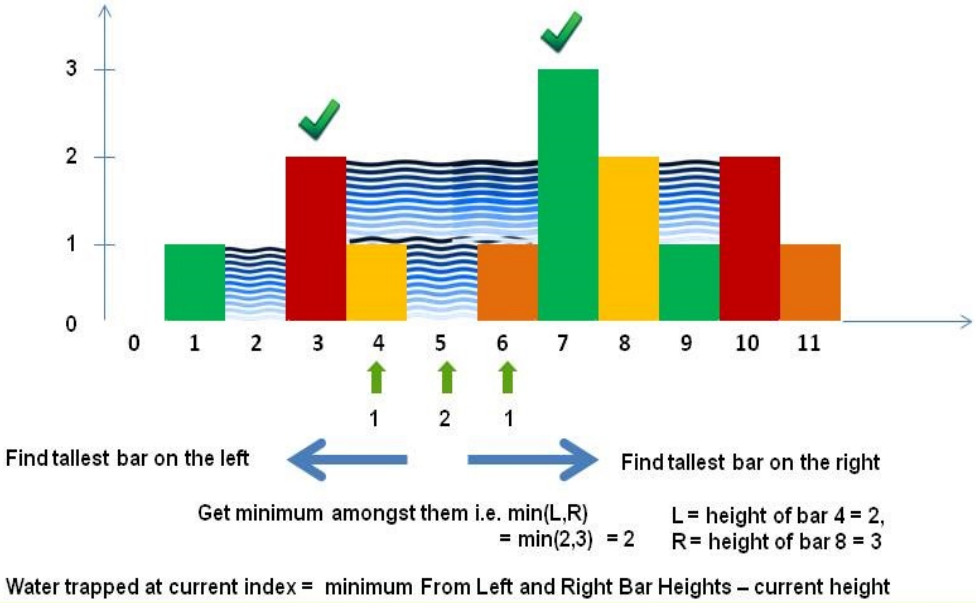

Figura 4.6: Infografía que muestra una parte de la resolución del problema *trapping rain water*.[5]

La resolución de este problema está públicamente disponible en varias páginas web, siendo además no muy complejo de resolver, con la mejor solución teniendo un coste

[4]Imagen extraída de: https://codepumpkin.com/trapping-rain-water-algorithm-problem/
[5]Imagen extraída de: https://codepumpkin.com/trapping-rain-water-algorithm-problem/

computacional en tiempo y espacio líneal (O(n)) con el número de barras. En la Figura 4.6 se puede ver una parte de la resolución.

De todas formas, nosotros lo vamos a utilizar (con a una pequeña modificación con el fin de guardar los índices) para analizar en qué zonas se podrían presentar mínimos locales y así valorar estas zonas como posibles puntos de segmentación. Es decir, si tenemos un mínimo en el píxel 30 y el siguiente más grande está en el píxel 60, podríamos valorar una segmentación desde el píxel 30 al 60.

CAPÍTULO 5
Marco experimental

5.1 Tecnologías utilizadas

5.1.1. Sistema operativo, entorno de desarrollo y lenguaje de programación

Como sistema operativo principal se ha utilizado *Windows 10 Pro, versión 21H2*, junto al entorno de desarrollo *Visual Studio Code*. Como lenguaje principal el código ha sido desarrollado en Python, versión 3.10, debido a la flexibilidad, sencillez y legibilidad del código, además de la disponibilidad de librerías esenciales.

5.1.2. Librerías utilizadas

Numpy

Numpy[1] es una librería muy popular para Python. Es útil para la implementación de matrices y operaciones matriciales en Python de manera eficiente, gracias al objeto *ndarray*. Muchas librerías tienen dependencia de ésta.

Scikit-learn (Sklearn)

Scikit-learn[2] [25] es una librería de software libre para Python que hace uso de Numpy y matplotlib. Proporciona herramientas eficientes para el aprendizaje automático, incluyendo algoritmos para la clasificación, regresión y análisis de grupos. Con el uso de esta librería podemos centrarnos menos en la implementación de los algoritmos y más en cómo hacer uso de ellos sobre nuestros datos.

Pickle

Con el uso de Pickle[3] podemos serializar objetos en Python. Esto es útil pues nos permite "almacenar" los modelos entrenados y otros objetos importantes para poder utilizarlos en otros contextos de manera simple.

[1] Página web de Numpy: https://numpy.org/doc/stable/index.html
[2] Página web de Scikit-learn: https://scikit-learn.org/stable/index.html
[3] Documentación de Pickle: https://docs.python.org/3/library/pickle.html

Pandas

Pandas [4] es una de las librerías por excelencia para la manipulación y análisis de datos en Python como una extensión de Numpy. Proporciona estructuras de datos flexibles y potentes para la manipulación de tablas y series temporales.

Plotly

Plotly[5] es una librería que nos permite visualizar datos, gráficas e histogramas de manera sencilla y atractiva visualmente.

5.2 Análisis exploratorio de los datos

5.2.1. Extracción de los datos

Marco legal y fuente de extracción

Se han obtenido un total de 1505 muestras, que se dividirán en 5 particiones durante la validación cruzada, con 301 muestras para test y 1204 para entrenamiento en cada iteración.

Las muestras fueron extraídas midiendo el tamaño en píxeles de las palabras, tanto en altura como en anchura y sin dejar trazos por cubrir, del texto "Noticia histórica de las fiestas con que Valencia celebró el siglo sexto de la venida a esta capital de la milagrosa imagen del Salvador [Manuscrito], escrito por Vicente Boix", desde la página 5 hasta la 10. El texto es de domino público y se puede encontrar en la página de la biblioteca valenciana[6]. En la Figura 5.1 se muestra un extracto de dicha obra y en la Figura 5.2 una *bounding box* de la palabra "sobre" extraída del mismo texto.

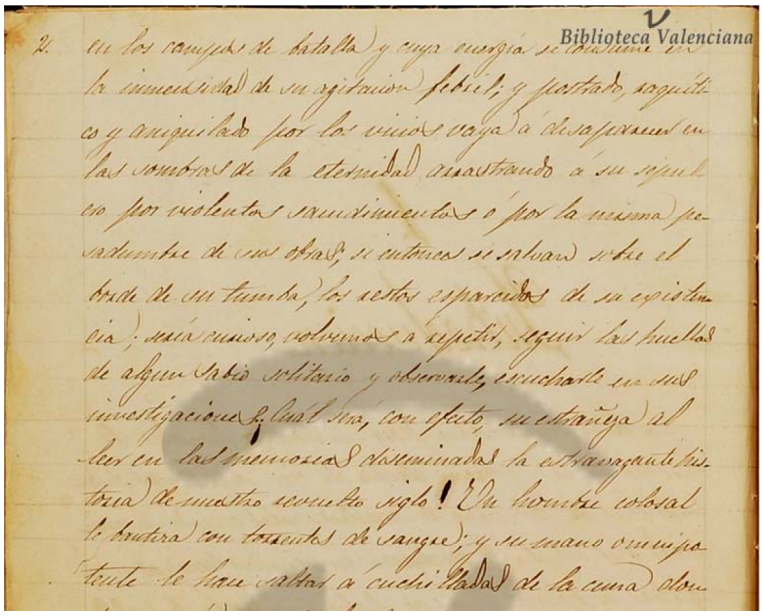

Figura 5.1: Extracto de media página del texto original. Página 5.

[4]Página web de Pandas: https://pandas.pydata.org/
[5]Página web de Plotly: https://plotly.com/python/
[6]Página web de la biblioteca valenciana: https://bivaldi.gva.es/

Figura 5.2: *Bounding box* de la palabra "sobre", de altura 36 píxeles y de anchura 89.

5.2.2. Comprobaciones iniciales

La primeras comprobaciones a realizar sobre el conjunto de datos es la ausencia de letras y la frecuencia del resto de letras que sí aparecen.

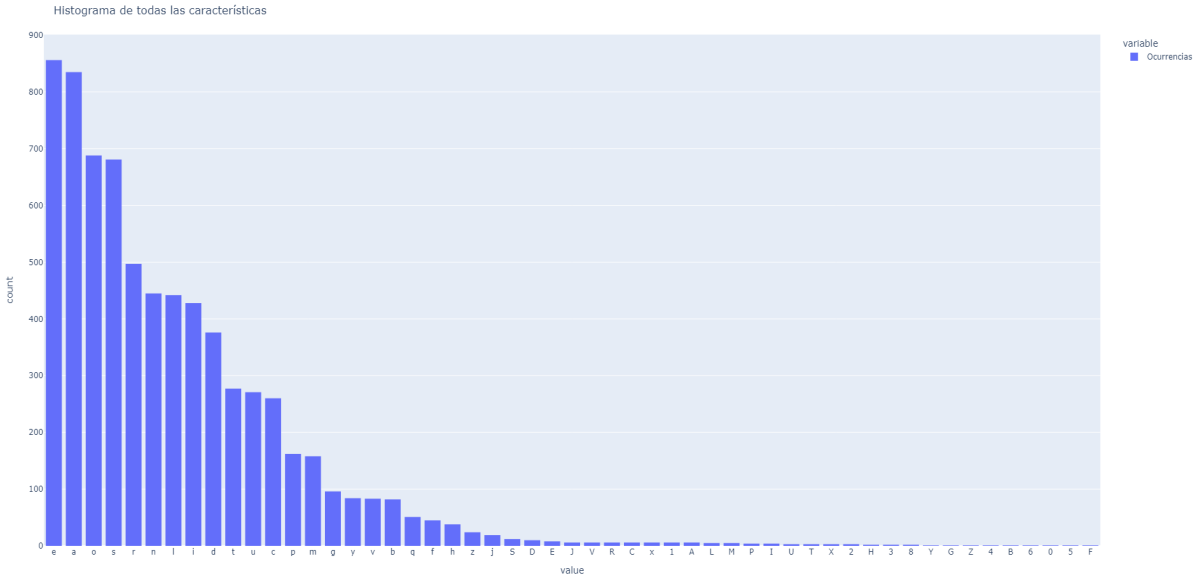

Figura 5.3: Histograma que muestra la distribución total de las características para todos los datos.

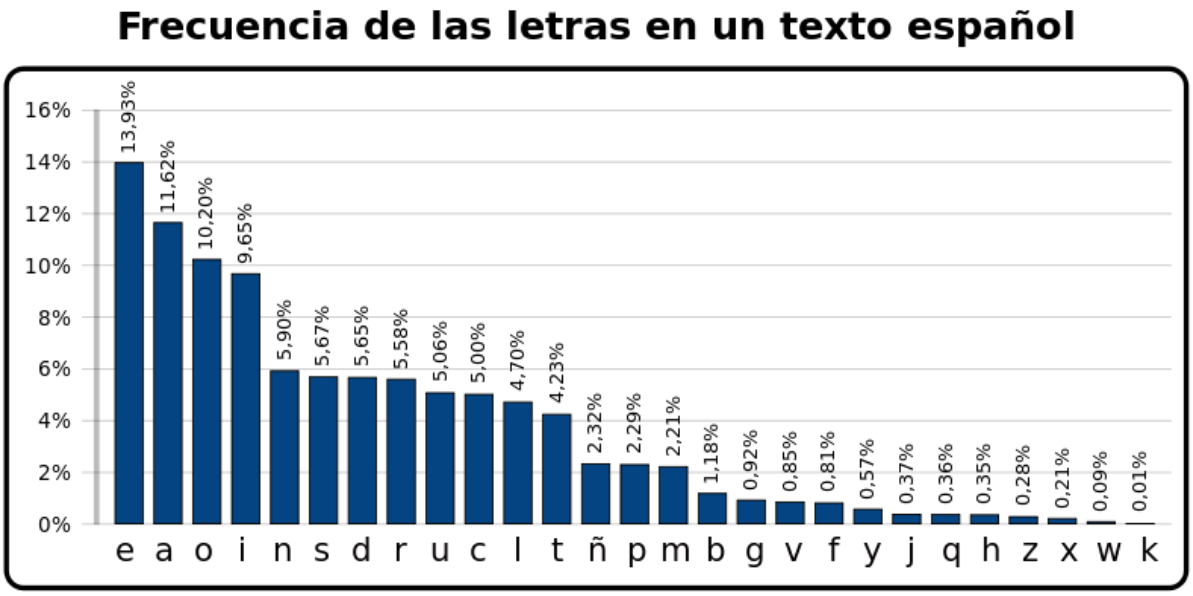

Figura 5.4: Frecuencia de las letras de un texto en castellano.[7]

[7]Imagen extraída de: https://commons.wikimedia.org/wiki/File:Frecuencias_de_las_letras_en_un_texto_espa%C3%B1ol.svg

Como podemos comprobar en el histograma presentado en la Figura 5.3, las letras más comunes del alfabeto español (véase la Figura 5.4) son las más comunes también en nuestros datos. También podemos observar cómo sigue la distribución de forma casi perfecta, lo cual es positivo para la posterior predicción.

Las letras y números que no aparecen y por tanto habrá que tener en cuenta son: “k”,“w”,“K”,“W”,“N”,“O”,“Q”,“7”,“9”. No obstante, como el número de letras mayúsculas y números es muy reducido tendremos que agruparlos en la misma categoría casi siempre, una para las mayúsculas y otra para los números. Podemos agrupar la letra “k” con la “d”, pues suelen tener similar anchura y altura, y la letra “w” con la “m” por la misma razón.

5.2.3. Distribución de los datos

Altura

Podemos vislumbrar en las Figuras 5.5 y 5.6 la diferencia en la cantidad de muestras entre las posibles clases, pues en el primer histograma ya vemos que la mayor parte de los datos se agrupan en torno a los valores más bajos (menores de 50). El valor mínimo está situado en 15 píxeles y el valor máximo en 109 píxeles.

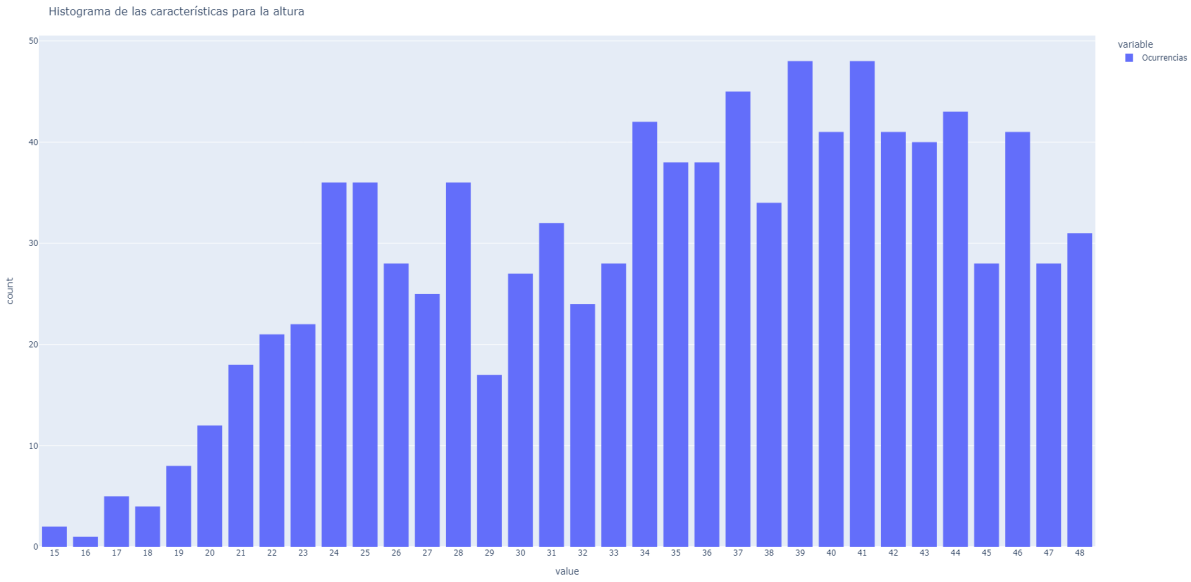

Figura 5.5: Histograma que muestra la distribución de las alturas de las palabras en el conjunto de datos. (I)

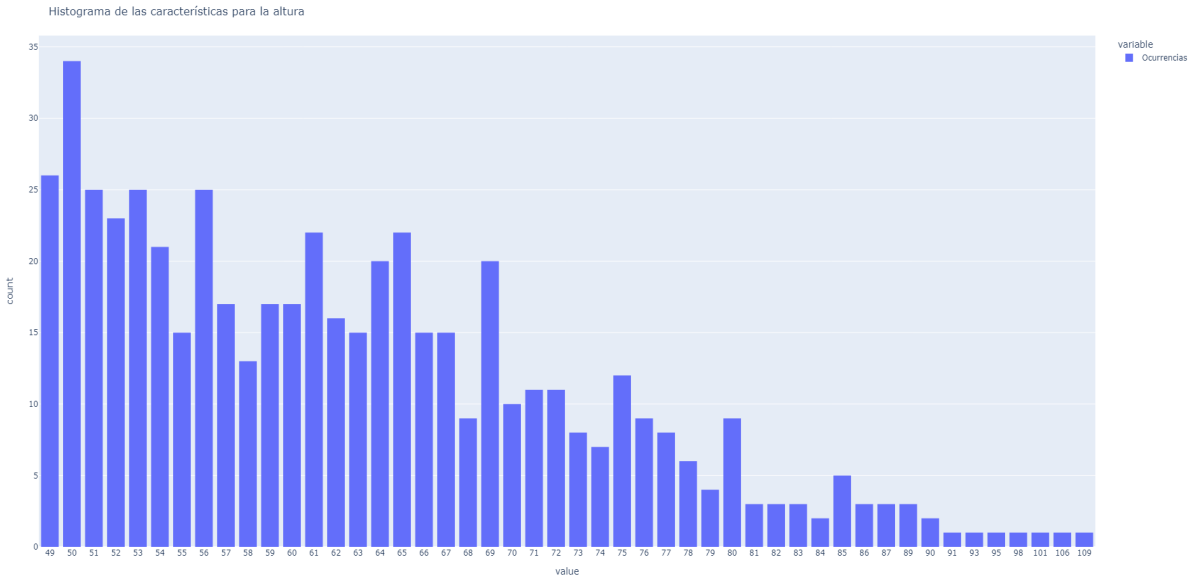

Figura 5.6: Histograma que muestra la distribución de las dimensiones de las alturas en el conjunto de datos. (II)

Anchura

Los histogramas presentados en las Figuras 5.7 a 5.11 van a mostrar la distribución de las dimensiones de las anchuras. Nótese que la escala en el eje Y varía en función del número de ocurrencias.

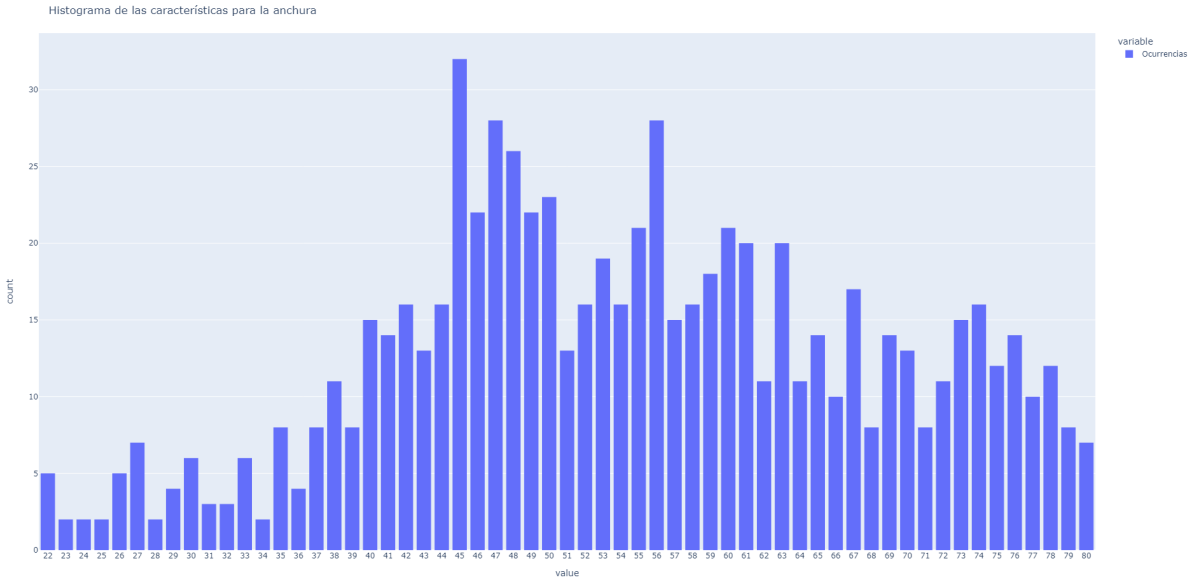

Figura 5.7: Histograma que muestra la distribución de las anchuras de las palabras en el conjunto de datos. (I)

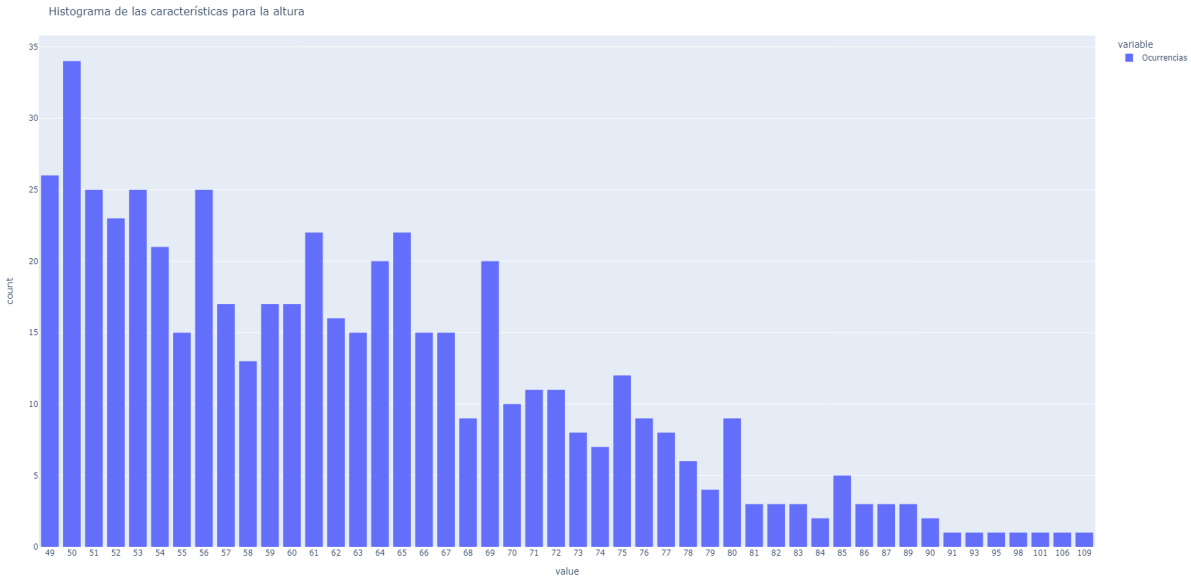

Figura 5.8: Histograma que muestra la distribución de las anchuras de las palabras en el conjunto de datos. (II)

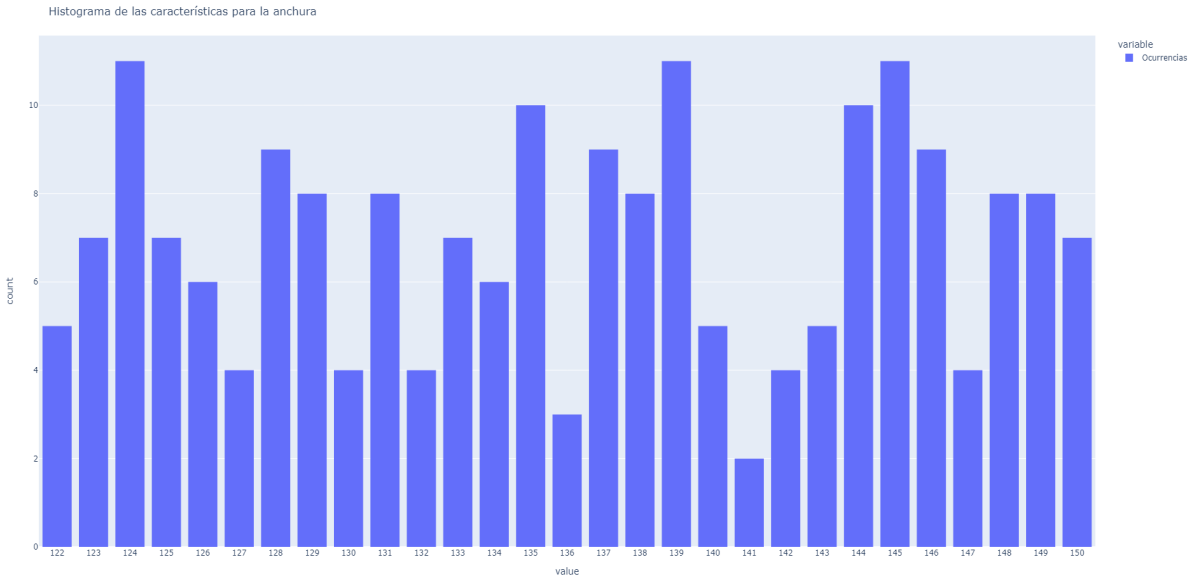

Figura 5.9: Histograma que muestra la distribución de las anchuras de las palabras en el conjunto de datos. (III)

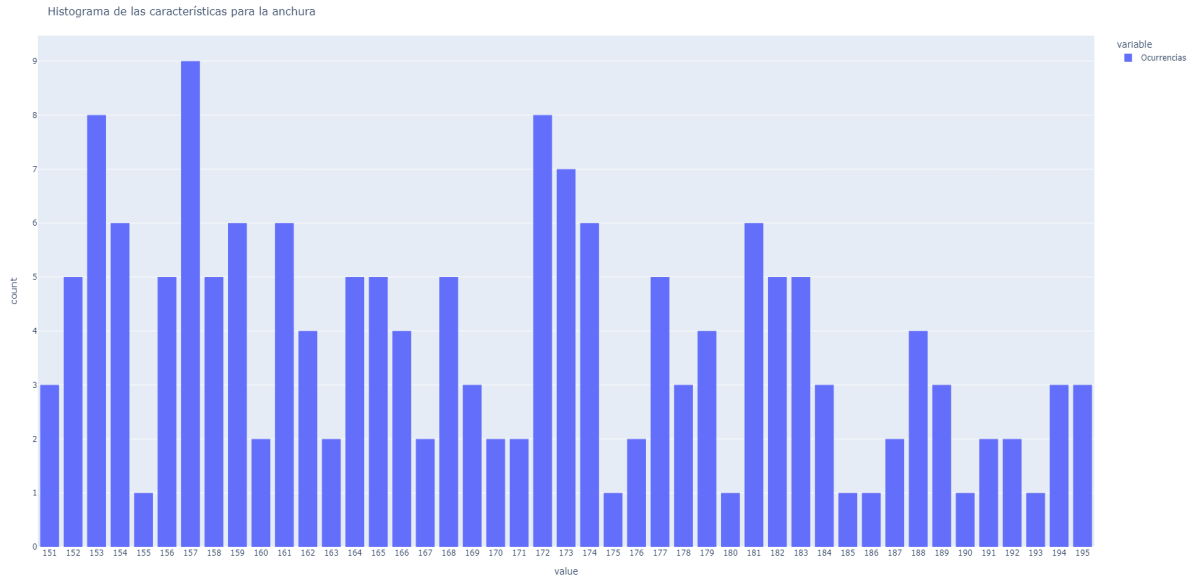

Figura 5.10: Histograma que muestra la distribución de las anchuras de las palabras en el conjunto de datos. (IV)

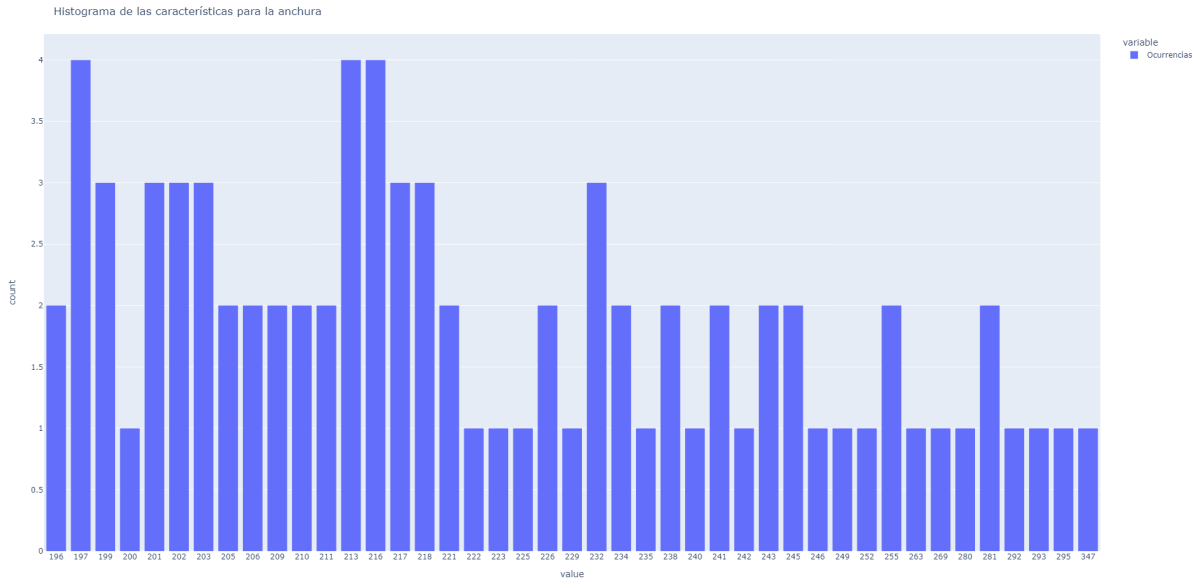

Figura 5.11: Histograma que muestra la distribución de las anchuras de las palabras en el conjunto de datos. (V)

Esta distribución ya es más difícil de interpretar, pero podemos observar algo similar, para los primeros intervalos (menor de 80 píxeles) hay muchos más valores que para los últimos (mayor de 150 píxeles). El valor mínimo es 22 píxeles y el valor máximo es 347 píxeles, un rango de valores mucho mayor para la misma cantidad de muestras.

5.2.4. Discretizados

Trapping rain water

Con el fin de detectar los posibles rangos de discretización, se ha aplicado esta técnica descrita en el apartado 4.4.1.

En la altura, los resultados de aplicarlo han sido:

- Píxel: 29, 32, 38, 45, 55, 58, 68.

En la anchura:

- Píxel: 34, 51, 57, 64, 84, 88, 99, 107, 115, 127, 141, 155, 170, 185.

Se descartan valores mayores pues hay muy pocas muestras en comparación y la agrupación no sería relevante. Se van a considerar combinaciones (que mantegan una cantidad mínima de datos, >5 %) de estos valores para el discretizado de las clases mostrado en la Tabla 5.1 y las Tablas 5.2 a 5.4.

Asimismo vamos a considerar los distintos posibles discretizados como distintos conjuntos, en los que habrán distintos grupos de clases. Cada clase se va a definir como el punto medio de su intervalo del rango de valores asignado. Poniendo el ejemplo de la Tabla 5.1, la clase 42 viene dada por el punto medio del intervalo]29, 55], es decir, usando como punto inicial el píxel 29 y como punto final el 55, ambos obtenidos de los resultados de la técnica descrita en el apartado 4.4.1.

Esto se realiza con el fin de observar cuál de los posibles discretizados va a traducirse en mejores resultados, pues dependiendo del discretizado podemos obtener mejores o peores regiones de decisión.

Altura

Conjunto 1				Conjunto 2			
Clase 22 [15, 29]	Clase 42]29, 55]	Clase 62]55, 68]	Clase 90]68, 109]	Clase 23 [15, 32]	Clase 38]32, 45]	Clase 56]45, 68]	Clase 90]68, 109]

Tabla 5.1: Tabla que muestra los distintos intervalos en píxeles para cada clase en los dos conjuntos considerados para la altura.

Para el caso de la altura, se considerarán distintas extracciones de características.

- Primera extracción, 6 características:
  - Palabras con letras ascendentes.
  - Palabras con letras descendentes.
  - Palabras con letras ascendentes y descendentes.
  - Palabras con letras mayúsculas y números.

  - Palabras con letras mayúsculas (o números) y descendentes.
  - Palabras de la altura de la x (intermedio).

- Segunda extracción, todas las letras como características:

  - Mayúsculas y números.
  - Cada letra como característica (solamente su presencia o ausencia).

- Tercera extracción:

  - Palabras con sólo letras ascendentes (incluyendo mayúsculas y números).
  - Palabras con sólo letras de la altura de la x (intermedio).
  - Palabras con sólo letras descendentes.
  - Cada letra descendente y ascendente como característica si no se cumple lo anterior, pues son las que definen la altura (solamente su presencia o ausencia).

Para un análisis más exhaustivo de las características por clase se puede ver el apéndice A. Con estas tres distintas extracciones tenemos suficiente para probar los modelos. Además, separar en más clases puede hacer que obtengamos clases aún más desequilibradas y no tiene por qué mejorar los resultados drásticamente, dado que esta separación ya cubre combinaciones de características distintivas para las clases.

Las distintas extracciones posibles combinadas con los conjuntos de la Tabla 5.1 resultan en 6 posibles combinaciones a probar.

Las Figuras 5.12 y 5.13 muestran las distribuciones de las clases para cada uno de los conjuntos:

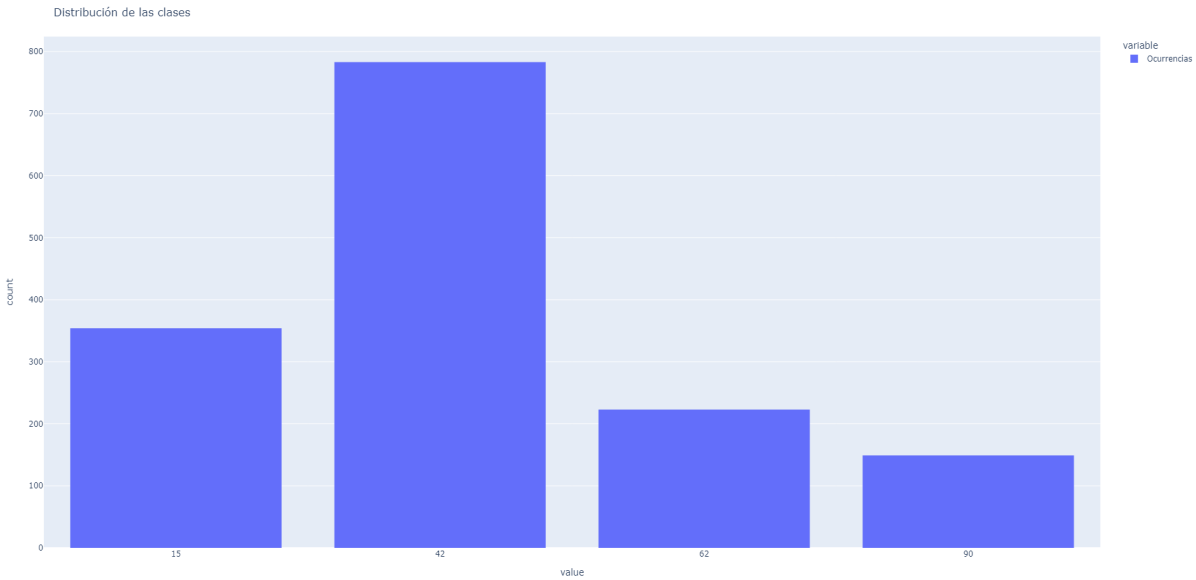

Figura 5.12: Histograma que muestra la distribución de los datos en el primer conjunto de clases.

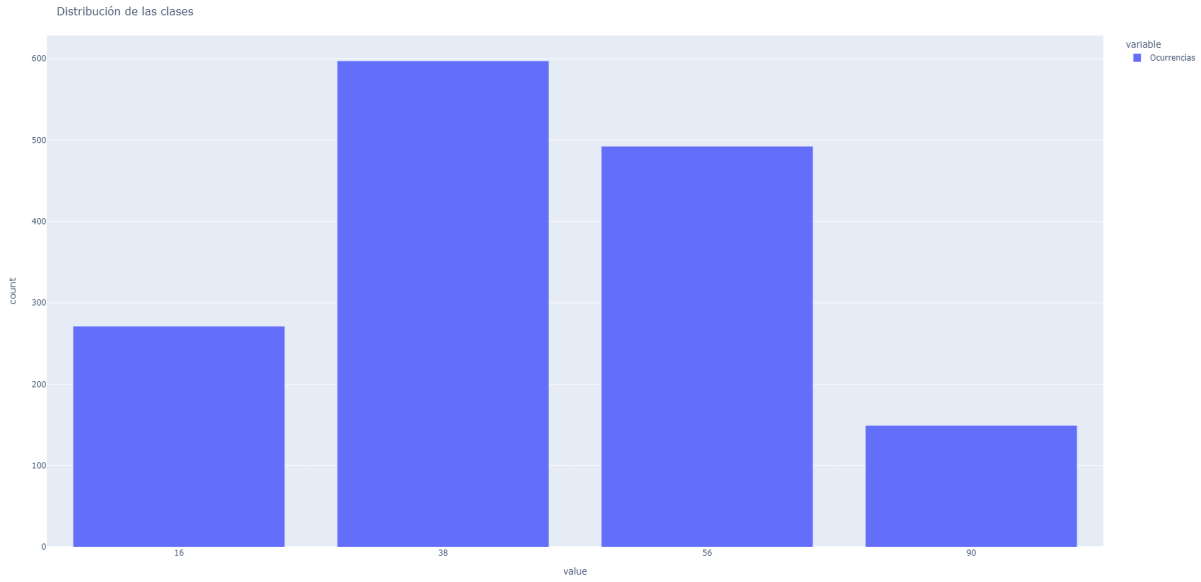

Figura 5.13: Histograma que muestra la distribución de los datos en el segundo conjunto de clases.

Anchura

En cuanto a la estimación de la anchura, las combinaciones posibles (descritas en las Tablas 5.2 a 5.4 son mucho mayores; sin embargo la extracción de características se limitará a una sola forma.

Conjunto 1					
Clase 36	Clase 67	Clase 95	Clase 117	Clase 156	Clase 220
[22,51]	]51, 84]	]84, 107]	]107,127]	]127,185]	]185,347]

Tabla 5.2: Tabla que muestra los distintos intervalos para cada clase en los tres conjuntos considerados. (I)

Conjunto 2					
Clase 36	Clase 69	Clase 101	Clase 128	Clase 163	Clase 220
[22,51]	]51, 88]	]88,115]	]115,141]	]141,185]	]185,347]

Tabla 5.3: Tabla que muestra los distintos intervalos para cada clase en los tres conjuntos considerados. (II)

Conjunto 3					
Clase 36	Clase 67	Clase 81	Clase 120	Clase 163	Clase 220
[22,51]	]51, 84]	]64,99]	]99,141]	]141,185]	]185,347]

Tabla 5.4: Tabla que muestra los distintos intervalos para cada clase en los tres conjuntos considerados. (III)

Nótese que la primera clase no es el punto medio entre los intervalos sino entre 22 (mínimo en anchura) y el máximo del primer intervalo.

En cuanto a la extracción de características:

- Letras mayúsculas y números agrupados.
- Letra “k” aproximada a la “d” y letra “w” a la “m”.
- Resto de letras (contando todas las apariciones).
- Añadimos una característica nueva relevante, la longitud de la palabra.

De nuevo, en el apéndice A se puede encontrar un análisis más detallado sobre las características.

En las Figuras 5.14 a 5.16 se muestran los distintos histogramas de las distribuciones para los tres conjuntos.

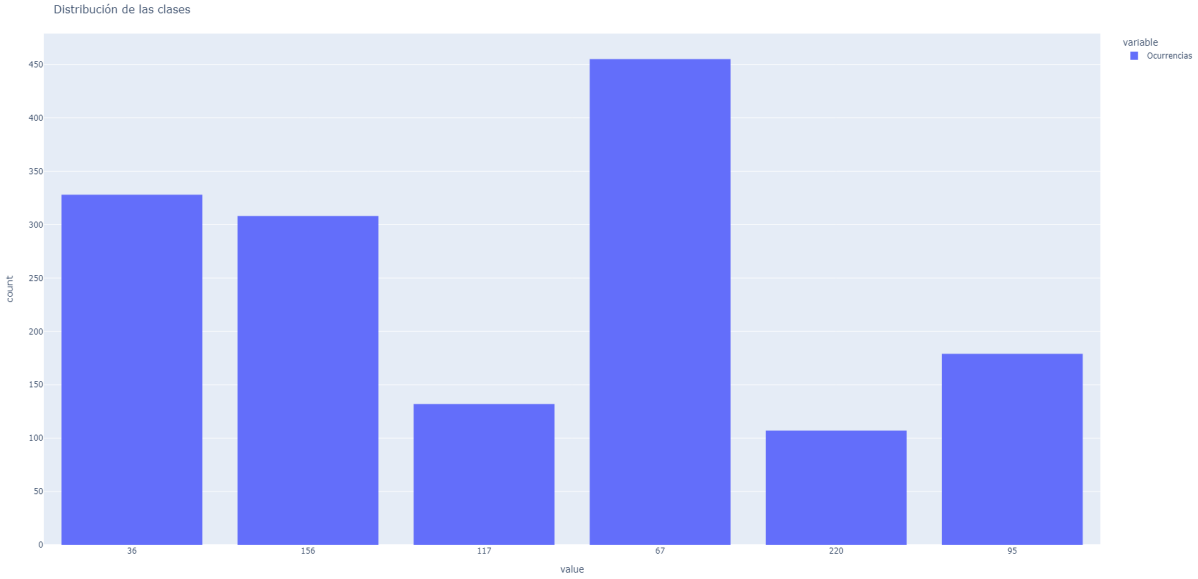

Figura 5.14: Distribución de las clases para el primer conjunto en anchura.

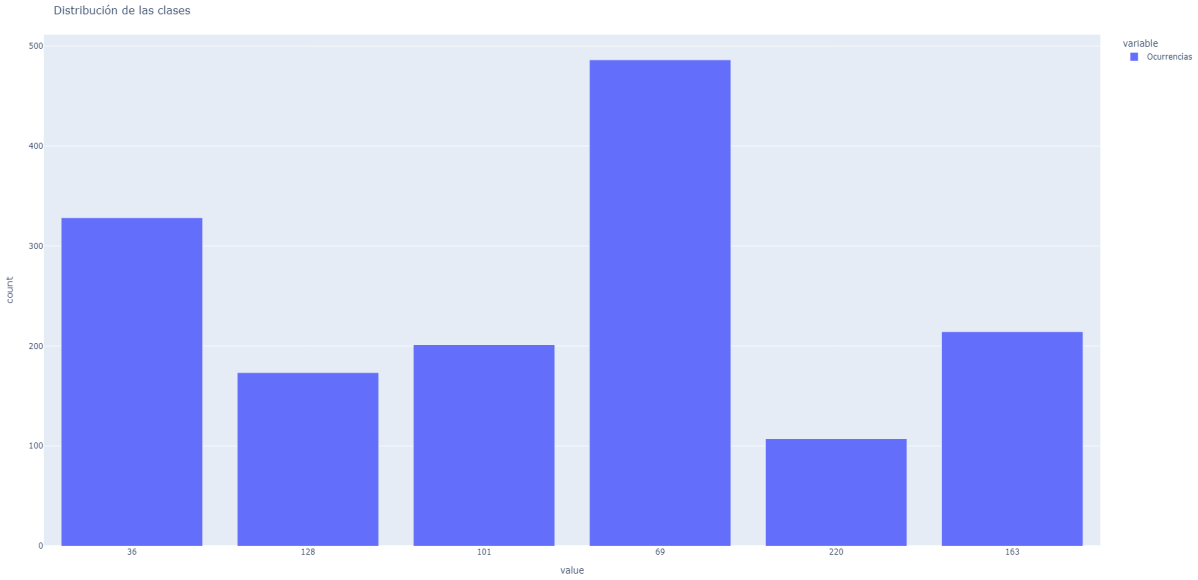

Figura 5.15: Distribución de las clases para el segundo conjunto en anchura.

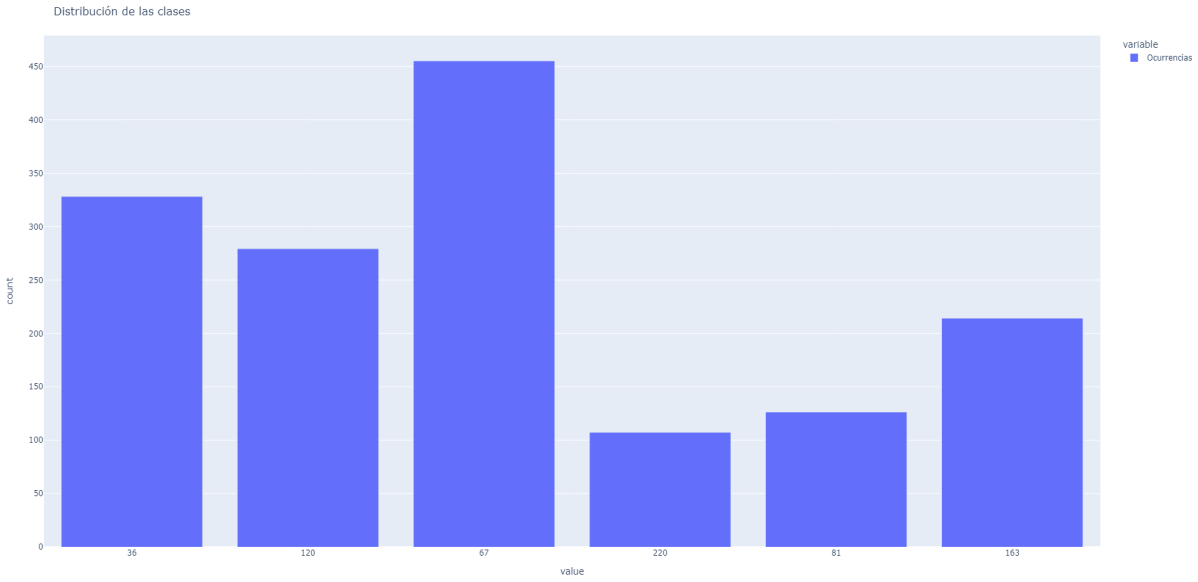

Figura 5.16: Distribución de las clases para el tercer conjunto en anchura.

CAPÍTULO 6
Resultados y discusión

Definiremos los siguientes acrónimos que serán usados a lo largo de los resultados mostrados en este capítulo:

- ErrorM: error medio de clasificación, total de muestras mal clasificadas entre el total de muestras.
- ÁreaM: área media de diferencia entre el valor real medio para la palabra estimada y el valor predicho (es decir, se predice 420px de área para la palabra "de" y la media aritmética en el conjunto de datos para esa palabra es 400px).
- Mejor ÁreaM: área media de diferencia entre el valor real medio y el predicho si se clasificaran correctamente todas las muestras. Es una cota inferior de ÁreaM.
- MinÁrea y MaxÁrea: indicadores de los mejores y peores casos respectivamente.
- RatioM: valor medio del ratio entre el valor real medio y el valor predicho.
- Mejor RatioM: valor medio del ratio entre el valor real medio y el predicho si se clasificaran todas las muestras correctamente.
- MinRatio y MaxRatio: indicador del mejor y peor caso, cuánta diferencia hay respectivamente con la *bounding box* real.

Nótese que "Mejor RatioM" puede ser un valor mayor que "RatioM" (y por tanto, irónicamente, peor), debido a que el primero se basa en si hubiera un error 0 de clasificación y el segundo se basa en la comparación con la media de los valores para el conjunto de datos original.

Las variables basadas en las etiquetas de las muestras y los aciertos nos permiten comprobar el rendimiento del clasificador, mientras que las basadas en los valores medios del conjunto de datos nos permite comprobar cuán lejos estamos de predecir un valor real.

Para el cálculo del *F1-Score* (*F-measure*) se usa una estrategia *One vs Rest*.

Los hiper-parámetros utilizados han sido:

- Para el clasificador Bernoulli y multinomial se ha usado `alpha=2.0` (parámetro de suavizado), `fit_prior=True`, `class_prior=None`.[1]

[1] Más información sobre los hiper-parámetros: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html

- Para los clasificadores *random forest*:`max_features=None`, `n_estimators=100`, `max_depth=None`, `random_state=0`.[2]
- Para los regresores *random forest*: `random_state=0`, `max_depth=35` para la altura y `max_depth=70` para la altura.[3]

6.1 Clasificación en altura

Para obtener los resultados de áreas y ratios en la clasificación en altura estimaremos la anchura para ambos conjuntos de la misma forma, con un clasificador *random forest*, así los resultados serán únicamente dependientes de la altura escogida.

6.1.1. Clasificador *naive Bayes* Bernoulli.

Primera extracción

	ErrorM	ÁreaM	Mejor ÁreaM	MaxArea	MinArea	RatioM	Mejor RatioM	MaxRatio	MinRatio
Conjunto 1	22.7 %	1058 px	850 px	7465 px	4 px	0.202	0.198	1.440	0.001
Conjunto 2	24.9 %	1070 px	823 px	9825 px	0 px	0.217	0.203	1.650	0.000

Tabla 6.1: Resultados del clasificador *naive bayes* Bernoulli para la altura, usando la extracción 1.

Extracción 1. *F1 Score*.				
Conjunto 1				
Clase 23	Clase 42	Clase 62	Clase 90	Media ponderada
0.895	0.864	0.403	0.601	0.777
Conjunto 2				
Clase 22	Clase 38	Clase 56	Clase 90	Media ponderada
0.873	0.792	0.530	0.619	0.755

Tabla 6.2: Resultados del clasificador *naive bayes* Bernoulli para la altura. Valores del *f1 score* por clase y la media ponderada (con el número de muestras por clase), usando la extracción 1.

Extracción 1. Ratio.				
Conjunto 1				
	Clase 23	Clase 42	Clase 62	Clase 90
Correcta	0.200	0.163	0.181	0.205
Incorrecta	0.239	0.326	0.373	0.193
Conjunto 2				
	Clase 22	Clase 38	Clase 56	Clase 90
Correcta	0.208	0.158	0.145	0.210
Incorrecta	0.372	0.299	0.319	0.182

Tabla 6.3: Resultados del clasificador *naive bayes* para la altura. Ratio medio por clase, dependiendo de si la predicción fue correcta o incorrecta, usando la extracción 1.

[2]Más información sobre los hiper-parámetros: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=random+forest#sklearn.ensemble.RandomForestClassifier

[3]Más información sobre los hiper-parámetros: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html?highlight=random+forest#sklearn.ensemble.RandomForestRegressor

De los resultados presentados en las Tablas 6.1 a 6.3 podemos deducir que la clase 62 no se está clasificando correctamente, dado su bajo valor de *F1-Score*, al igual que la clase 56.

Además, el error de clasificación en ambas clases significa un alto valor de ratio, es decir, la predicción está lejos del valor medio real.

Segunda extracción

	ErrorM	ÁreaM	Mejor ÁreaM	MaxArea	MinArea	RatioM	Mejor RatioM	MaxRatio	MinRatio
Conjunto 1	30.2 %	1130 px	850 px	9433 px	4 px	0.249	0.198	2.81	0.001
Conjunto 2	31.8 %	1024 px	823 px	9008 px	0 px	0.261	0.203	4.09	0.000

Tabla 6.4: Resultados del clasificador *naive bayes* Bernoulli para la altura, usando la extracción 2.

Extracción 2. *F1 Score*.				
Conjunto 1				
Clase 23	Clase 42	Clase 62	Clase 90	Media ponderada
0.699	0.786	0.404	0.487	0.680
Conjunto 2				
Clase 22	Clase 38	Clase 56	Clase 90	Media ponderada
0.690	0.762	0.608	0.530	0.700

Tabla 6.5: Resultados del clasificador *naive bayes* Bernoulli para la altura. Valores del *f1 score* por clase y la media ponderada (con el número de muestras por clase), usando la extracción 2.

Extracción 2. Ratio.				
Conjunto 1				
	Clase 23	Clase 42	Clase 62	Clase 90
Correcta	0.178	0.169	0.141	0.215
Incorrecta	0.718	0.370	0.308	0.306
Conjunto 2				
	Clase 22	Clase 38	Clase 56	Clase 90
Correcta	0.201	0.159	0.145	0.228
Incorrecta	1.069	0.358	0.270	0.243

Tabla 6.6: Resultados del clasificador *naive bayes* para la altura. Ratio medio por clase, dependiendo de si la predicción fue correcta o incorrecta, usando la extracción 2.

La combinación de la segunda extracción de características y el modelo Bernoulli mostrada en las Tablas 6.4 a 6.6 no aparenta dar buenos resultados, manteniendo un error alto en la clasificación, junto a un ratio medio y área muy altos. Sumado a eso, las clases 62 y 90 no parecen clasificarse correctamente de acuerdo al *F1-Score*.

El ratio más alto cuando la muestra es incorrecta se encuentra en la clase 22, siendo de 1.069, lo que implica que es al menos el doble de grande que la *bounding box* media. Por tanto, se tiende a errores grandes y la fácil confusión del clasificador.

Tercera extracción

	ErrorM	ÁreaM	Mejor ÁreaM	MaxArea	MinArea	RatioM	Mejor RatioM	MaxRatio	MinRatio
Conjunto 1	20.3 %	864 px	850 px	6828 px	4 px	0.179	0.198	1.102	0.001
Conjunto 2	27.0 %	919 px	823 px	6828 px	0 px	0.201	0.203	1.644	0.000

Tabla 6.7: Resultados del clasificador *naive bayes* Bernoulli para la altura, usando la extracción 3.

Extracción 3. F1 Score.				
Conjunto 1				
Clase 23	Clase 42	Clase 62	Clase 90	Media ponderada
0.895	0.865	0.580	0.608	0.803
Conjunto 2				
Clase 22	Clase 38	Clase 56	Clase 90	Media ponderada
0.873	0.782	0.568	0.622	0.756

Tabla 6.8: Resultados del clasificador *naive bayes* Bernoulli para la altura. Valores del *f1 score* por clase y la media ponderada (con el número de muestras por clase), usando la extracción 3.

Extracción 3. Ratio.				
Conjunto 1				
	Clase 23	Clase 42	Clase 62	Clase 90
Correcta	0.200	0.163	0.147	0.200
Incorrecta	0.239	0.241	0.330	0.170
Conjunto 2				
	Clase 22	Clase 38	Clase 56	Clase 90
Correcta	0.208	0.157	0.143	0.204
Incorrecta	0.305	0.259	0.265	0.211

Tabla 6.9: Resultados del clasificador *naive bayes* para la altura. Ratio medio por clase, dependiendo de si la predicción fue correcta o incorrecta, usando la extracción 3.

En los resultados de las Tablas 6.7 a 6.9 ya podemos discernir que el error de clasificación, aunque importante, no es la única métrica a considerar, de hecho puede llegar a ser irrelevante comparada con el ratio y el área media. Por ejemplo, que el área para una palabra sea 30px, y se clasifique en 22px; sin embargo, es posible que el área media de la palabra en el conjunto de datos original sea 40px; por tanto, la clasificación en la clase 42 no solo es razonable sino que es óptima.

Por otro lado, los resultados de esta clasificación, sobre todo para el conjunto 1, son bastante buenos, con un ratio de 0.179 y valores de *F1-Score* que sobrepasan el mínimo para todas las clases.

6.1.2. *Random forest*

Primera extracción de características

	ErrorM	ÁreaM	Mejor ÁreaM	MaxArea	MinArea	RatioM	Mejor RatioM	MaxRatio	MinRatio
Conjunto 1	20.3 %	1042 px	850 px	8116 px	4 px	0.197	0.198	1.44	0.001
Conjunto 2	27.2 %	925 px	823 px	10376 px	0 px	0.199	0.203	1.644	0.000

Tabla 6.10: Resultados del clasificador *random forest* para la altura, usando la extracción 1.

Extracción 1. *F1 Score*.				
Conjunto 1				
Clase 23	Clase 42	Clase 62	Clase 90	Media ponderada
0.895	0.863	0.214	0.395	0.712
Conjunto 2				
Clase 22	Clase 38	Clase 56	Clase 90	Media ponderada
0.873	0.794	0.630	0.270	0.736

Tabla 6.11: Resultados del clasificador *random forest* para la altura. Valores del *f1 score* por clase y la media ponderada (con el número de muestras por clase), usando la extracción 1.

Extracción 1. Ratio.				
Conjunto 1				
	Clase 23	Clase 42	Clase 62	Clase 90
Correcta	0.201	0.170	0.103	0.223
Incorrecta	0.239	0.404	0.274	0.218
Conjunto 2				
	Clase 22	Clase 38	Clase 56	Clase 90
Correcta	0.209	0.159	0.141	0.117
Incorrecta	0.372	0.292	0.226	0.238

Tabla 6.12: Resultados del clasificador *random forest* para la altura. Ratio medio por clase, dependiendo de si la predicción fue correcta o incorrecta, usando la extracción 1.

Como se ve en los resultados de las Tablas 6.10 a 6.12, el modelo no es capaz de discernir las diferencias entre las clases con esta extracción de características y por tanto se descarta su uso.

Segunda extracción de características

	ErrorM	ÁreaM	Mejor ÁreaM	MaxArea	MinArea	RatioM	Mejor RatioM	MaxRatio	MinRatio
Conjunto 1	17.2 %	967 px	850 px	8433 px	4 px	0.192	0.198	1.421	0.001
Conjunto 2	22.3 %	888 px	823 px	6249 px	0 px	0.197	0.203	1.650	0.000

Tabla 6.13: Resultados del clasificador *random forest* para la altura, usando la extracción 2.

Extracción 2. *F1 Score*.				
Conjunto 1				
Clase 23	Clase 42	Clase 62	Clase 90	Media ponderada
0.912	0.883	0.624	0.610	0.824
Conjunto 2				
Clase 22	Clase 38	Clase 56	Clase 90	Media ponderada
0.872	0.806	0.724	0.626	0.792

Tabla 6.14: Resultados del clasificador *random forest* para la altura. Valores del *f1 score* por clase y la media ponderada (con el número de muestras por clase), usando la extracción 2.

Extracción 2. Ratio.				
Conjunto 1				
	Clase 23	Clase 42	Clase 62	Clase 90
Correcta	0.203	0.166	0.165	0.220
Incorrecta	0.198	0.339	0.318	0.296
Conjunto 2				
	Clase 22	Clase 38	Clase 56	Clase 90
Correcta	0.212	0.159	0.148	0.222
Incorrecta	0.439	0.284	0.235	0.250

Tabla 6.15: Resultados del clasificador *random forest* para la altura. Ratio medio por clase, dependiendo de si la predicción fue correcta o incorrecta, usando la extracción 2.

Al incrementar el número de características obtenemos uno de los mejores resultados, como se muestra en las Tablas 6.13 a 6.15. Con el mejor error de clasificación y un bajo ratio, muestra que *random forest* funciona muy bien para nuestro problema.

Tercera extracción de características

	ErrorM	ÁreaM	Mejor ÁreaM	MaxArea	MinArea	RatioM	Mejor RatioM	MaxRatio	MinRatio
Conjunto 1	20.3 %	936 px	850 px	8828 px	4 px	0.187	0.198	1.440	0.001
Conjunto 2	25.9 %	875 px	823 px	8828 px	0 px	0.195	0.203	1.644	0.000

Tabla 6.16: Resultados del clasificador *random forest* para la altura, usando la extracción 3.

Extracción 3. *F1 Score*.				
Conjunto 1				
Clase 23	Clase 42	Clase 62	Clase 90	Media ponderada
0.895	0.864	0.427	0.625	0.783
Conjunto 2				
Clase 22	Clase 38	Clase 56	Clase 90	Media ponderada
0.874	0.783	0.607	0.629	0.763

Tabla 6.17: Resultados del clasificador *random forest* para la altura. Valores del *f1 score* por clase y la media ponderada (con el número de muestras por clase), usando la extracción 3.

Extracción 3. Ratio.				
Conjunto 1				
	Clase 23	Clase 42	Clase 62	Clase 90
Correcta	0.201	0.170	0.119	0.202
Incorrecta	0.239	0.227	0.327	0.214
Conjunto 2				
	Clase 22	Clase 38	Clase 56	Clase 90
Correcta	0.209	0.158	0.141	0.202
Incorrecta	0.305	0.260	0.243	0.219

Tabla 6.18: Resultados del clasificador *random forest* para la altura. Ratio medio por clase, dependiendo de si la predicción fue correcta o incorrecta, usando la extracción 3.

Como muestran las Figuras 6.16 a 6.18 son resultados ligeramente peores que los de la segunda extracción, pues es posible que estemos perdiendo características relevantes para la clasificación.

6.2 Clasificación en anchura

De la misma forma que en la clasificación en altura, estimaremos la altura para ambos conjuntos con el mismo clasificador y conjunto (clasificador *naive bayes* Bernoulli con la extracción 3 sobre el conjunto 1), dejando la variación en los resultados dependiente de la anchura escogida.

6.2.1. Clasificador *naive Bayes* multinomial.

	ErrorM	ÁreaM	Mejor ÁreaM	MaxArea	MinArea	RatioM	Mejor RatioM	MaxRatio	MinRatio
Conjunto 1	20.5 %	881 px	850 px	6942 px	4 px	0.181	0.198	1.181	0.001
Conjunto 2	22.3 %	921 px	847 px	6702 px	0 px	0.200	0.211	1.434	0.000
Conjunto 3	21.4 %	906 px	839 px	6942 px	6 px	0.193	0.192	1.433	0.001

Tabla 6.19: Resultados del clasificador *naive bayes* multinomial para la anchura.

F1 Score.						
Conjunto 1						
Clase 36	Clase 67	Clase 95	Clase 117	Clase 156	Clase 220	Media ponderada
0.681	0.861	0.789	0.778	0.748	0.657	0.772
Conjunto 2						
Clase 36	Clase 69	Clase 101	Clase 128	Clase 163	Clase 220	Media ponderada
0.688	0.735	0.811	0.830	0.799	0.778	0.761
Conjunto 3						
Clase 36	Clase 67	Clase 81	Clase 120	Clase 163	Clase 220	Media ponderada
0.792	0.785	0.815	0.830	0.773	0.570	0.780

Tabla 6.20: Resultados del clasificador *naive bayes* multinomial para la anchura. Valores del *f1 score* por clase y la media ponderada (con el número de muestras por clase).

Ratio.						
Conjunto 1						
	Clase 36	Clase 67	Clase 95	Clase 117	Clase 156	Clase 220
Correcta	0.244	0.167	0.180	0.117	0.157	0.126
Incorrecta	0.202	0.255	0.214	0.252	0.269	0.224
Conjunto 2						
	Clase 36	Clase 69	Clase 101	Clase 128	Clase 163	Clase 220
Correcta	0.254	0.180	0.147	0.131	0.170	0.155
Incorrecta	0.215	0.255	0.262	0.250	0.198	0.174
Conjunto 3						
	Clase 36	Clase 67	Clase 81	Clase 120	Clase 163	Clase 220
Correcta	0.235	0.162	0.121	0.165	0.151	0.150
Incorrecta	0.203	0.265	0.234	0.292	0.237	0.169

Tabla 6.21: Resultados del clasificador *naive bayes* para la anchura. Valores del ratio por clase dependiendo de si la predicción fue correcta o incorrecta.

Como se puede ver en las Tablas 6.22 a 6.24 los resultados no son malos, a pesar del error en clasificación; de hecho para todos los conjuntos los valores de *F1-Score* son suficientes, el ratio es bajo, y el área media está por debajo de 1000 px.

6.2.2. *Random forest*

	ErrorM	ÁreaM	Mejor ÁreaM	MaxArea	MinArea	RatioM	Mejor RatioM	MaxRatio	MinRatio
Conjunto 1	19.6 %	885 px	850 px	6892 px	4 px	0.182	0.198	1.181	0.001
Conjunto 2	21.3 %	933 px	847 px	7286 px	0 px	0.199	0.211	1.326	0.000
Conjunto 3	21.6 %	906 px	839 px	6942 px	6 px	0.191	0.192	1.433	0.001

Tabla 6.22: Resultados del clasificador *random forest* para la anchura.

F1 Score.						
Conjunto 1						
Clase 36	Clase 67	Clase 95	Clase 117	Clase 156	Clase 220	Media ponderada
0.732	0.901	0.806	0.768	0.790	0.738	0.808
Conjunto 2						
Clase 36	Clase 69	Clase 101	Clase 128	Clase 163	Clase 220	Media ponderada
0.712	0.737	0.811	0.822	0.807	0.796	0.768
Conjunto 3						
Clase 36	Clase 67	Clase 81	Clase 120	Clase 163	Clase 220	Media ponderada
0.782	0.776	0.799	0.836	0.776	0.577	0.774

Tabla 6.23: Resultados del clasificador *random forest* para la anchura. Valores del *f1 score* por clase y la media ponderada (con el número de muestras por clase).

Ratio.						
Conjunto 1						
	Clase 36	Clase 67	Clase 95	Clase 117	Clase 156	Clase 220
Correcta	0.222	0.166	0.144	0.140	0.166	0.144
Incorrecta	0.220	0.183	0.246	0.237	0.186	0.225
Conjunto 2						
	Clase 36	Clase 69	Clase 101	Clase 128	Clase 163	Clase 220
Correcta	0.254	0.181	0.142	0.142	0.167	0.157
Incorrecta	0.214	0.251	0.232	0.216	0.197	0.157
Conjunto 3						
	Clase 36	Clase 67	Clase 81	Clase 120	Clase 163	Clase 220
Correcta	0.235	0.165	0.115	0.163	0.148	0.148
Incorrecta	0.203	0.251	0.225	0.257	0.267	0.147

Tabla 6.24: Resultados del clasificador *random forest* para la anchura. Valores del ratio por clase dependiendo de si la predicción fue correcta o incorrecta.

Aunque el clasificador multinomial no daba malos resultados, *random forest* le sobrepasa en todos los aspectos. Es muy destacable la discretización del conjunto 1, que anteriormente ya denotaba una mejor separación entre las clases.

6.3 Clasificación conjunta

De los anteriores resultados se ha realizado la siguiente selección como la mejor posible dados los datos disponibles:

- Altura: Tercera extracción de características junto al clasificador *naive bayes* Bernoulli, usando el primer conjunto como discretizado. El clasificador *random forest* junto a la segunda extracción podía ser una elección casi igual de válida.
- Anchura: Clasificador *random forest*, usando el primer conjunto como discretizado.

Así, los mejores datos vienen dados por la Tablas 6.22, 6.23 y 6.24.

6.3.1. Regresión usando *random forest*

Si se usa el modelo *random forest* como regresor en vez de clasificador es posible prescindir del discretizado de los datos. Aplicando un regresor para la altura y otro para la anchura, se han obtenido los resultados de la Tabla 6.25.

Regresor random forest							
ÁreaM	Mejor ÁreaM	MaxArea	MinArea	RatioM	Mejor RatioM	MaxRatio	MinRatio
647 px	276 px	4348 px	0 px	0.12	0.09	1.588	0.0

Tabla 6.25: Resultados del regresor *random forest*.

Estos son, por bastante margen, los mejores resultados obtenidos. A pesar de haber tratado el problema anteriormente como un problema de clasificación, la regresión usando *random forest* parece ser la mejor alternativa.

Nótese que “Mejor RatioM” en este caso sí que estima el mejor ratio posible, pues está comparando cada muestra con su media. Por tanto, un ratioM de 0.12 frente a 0.09 indica una diferencia de 0.03, apenas un 3 % de diferencia en las áreas. Esto ya era discernible pues hemos pasado de obtener un área media en rangos cercanos a 1000px a una de 647px.

El peor fallo en cuanto a área obtenido es de la muestra “ensangrentadas”, que se predijo tenía 67 px de altura y 251 px de anchura y sin embargo, eran 83 px de altura y 255 de anchura.

El peor fallo en cuanto a ratio obtenido es de la muestra “o”, que se predijo tenía 25px de altura y 41 de anchura; sin embargo, eran 18 de altura y 22 de anchura.

CAPÍTULO 7
Conclusiones

La estimación del tamaño de las palabras de un texto manuscrito es una tarea compleja, que varía de autor a autor, de los trazos efectuados, del momento de escritura y de la precisión en la extracción de las muestras junto al número y exhaustividad de éstas.

En este trabajo se ha tratado este problema de dos formas, como un problema de clasificación y como un problema de regresión. Para ambos casos, se han extraído las muestras y creado distintos discretizados posibles (mediante la técnica descrita en el apartado 4.4.1) de los datos para su posible estimación, de acuerdo al primer objetivo planteado.

Se han planteado varios tipos distintos de clasificadores, *naive bayes* Bernoulli y multinomial y *random forest* de acuerdo a las características observadas en los datos. Al mismo tiempo se han planteado y probado distintas extracciones de características posibles.

A su vez, se han entrenado y probado mediante validación cruzada *stratified* y probado su calidad mediante distintas métricas, como el ratio o el *f1-score*, cumpliendo así los objetivos y la metodología planteada inicialmente.

Los resultados son favorables considerando las limitaciones, siendo la combinación de dos regresores *random forest*, uno para la anchura y otro para la altura, la mejor aproximación, con un error de ratio del 0.12, sólo un 0.03 por encima del mejor posible.

Este trabajo ha supuesto un reto, pues ha tratado de poner en práctica muchos de los conocimientos adquiridos durante el grado, además de la inclusión de conceptos nuevos y el uso de las librerías necesarias para resolver un problema en el que partíamos desde cero.

7.1 Trabajo futuro

Como posibles trabajos futuros se plantea:

- La automatización de la obtención de las muestras y su etiquetado, con el fin de ahorrar tiempo y obtener un mayor número de muestras y características.
- La aplicación de esta estimación a un sistema real de búsqueda de palabras.
- Probar otros modelos más complejos si es posible gracias a la automatización de la obtención de las muestras y diseñar una métrica que funcione bajo un umbral de altura y anchura, para garantizar que las estimaciones se parecen lo suficiente.

7.2 Relación con los estudios cursados

Para realizar este trabajo ha sido necesario poner en práctica muchos conocimientos adquiridos durante el grado. Durante este trabajo se han puesto en práctica y pulido las habilidades de programación eficiente. Expuesto de una forma más concreta:

Conocimientos generales y avanzados de programación:

- De las asignaturas introductorias a los lenguajes de programación, introducción a la informática y la programación (IIP) y programación (PRG), se han requerido todo tipo de conceptos básicos, pues no se podría haber programado eficientemente sin éstas.
- De asignaturas un poco más avanzadas, como estructuras de datos y algoritmos (EDA), se han requerido conceptos clave de estructuras de datos, como los diccionarios, los *arrays*, etc. Los conceptos estudiados en lenguajes y paradigmas de la programación (LTP) han sido clave para la comprensión del uso de las librerías, el lenguaje de programación y el proceso de depuración. Destacar también la optativa competición de programación (CACM), pues de resolver problemas similares a los de esta asignatura surge la idea del *trapping rain water*.

También hubiera sido imposible comprender muchos de los conceptos sin una buena base en estadística (EST), por poner un ejemplo, la familia de clasificadores *naive bayes* se basa en distribuciones de probabilidad.

Conocimientos de la rama de computación:

- Las asignaturas de aprendizaje automático (APR) y sistemas inteligentes (SIN) proporcionaron las bases del razonamiento probabilístico.
- Percepción (PER) es la asignatura más íntimamente relacionada con este trabajo. Gran parte de los conocimientos parten de ésta, aunque algunos conceptos fueran vistos también en otras asignaturas como APR, SIN y sistemas de almacenamiento y recuperación de la información (SAR). Tanto la extracción de características, el modelo *bag of words*, los espacios métricos, suavizados, los clasificadores basados en distribuciones de probabilidad como los conceptos del aprendizaje por conjuntos (combinación de clasificadores, *bagging y boosting*) han sido puestos en práctica en este trabajo.

No obstante, no se vieron en concreto muchas técnicas para lidiar con desequilibrios de clases, así como los clasificadores *random forest*. Por tanto, es algo que se tuvo que aprender durante este trabajo, así como no se puso en práctica el uso de la validación cruzada *stratified* y las métricas pertinentes.

El desarrollo de las siguientes competencias transversales ha sido crucial durante el transcurso del trabajo:

- CT-01. Comprensión e integración. Se ha demostrado la comprensión y la integración del conocimiento tanto de la especialización propia como en otros contextos más amplios.

- CT-02. Aplicación y pensamiento práctico. Se han aplicado los conocimientos teóricos y establecido el proceso a seguir para alcanzar los objetivos, llevar a cabo los experimentos y analizar e interpretar los datos para extraer conclusiones.
- CT-03. Análisis y resolución de problemas. Se han analizado y resuelto los problemas de forma efectiva, identificando y definiendo los elementos significativos que los constituyen.

Bibliografía

[1] Roger Achkar, Khodor Ghayad, Rayan Haidar, Sawsan Saleh y Rana Al Hajj. «Medical Handwritten Prescription Recognition Using CRNN». En: *2019 International Conference on Computer, Information and Telecommunication Systems (CITS)*. 2019, págs. 1-5. DOI: 10.1109/CITS.2019.8862004.

[2] Jon Almazán, Albert Gordo, Alicia Fornés y Ernest Valveny. «Handwritten Word Spotting with Corrected Attributes». En: *2013 IEEE International Conference on Computer Vision*. Dic. de 2013, págs. 1017-1024. DOI: 10.1109/ICCV.2013.130.

[3] Théodore Bluche. «Joint Line Segmentation and Transcription for End-to-End Handwritten Paragraph Recognition». En: *Proceedings of the 30th International Conference on Neural Information Processing Systems*. NIPS'16. Barcelona, Spain: Curran Associates Inc., 2016, págs. 838-846. ISBN: 9781510838819.

[4] Federico Bolelli, Guido Borghi y Costantino Grana. «Historical Handwritten Text Images Word Spotting Through Sliding Window HOG Features». En: *Image Analysis and Processing - ICIAP 2017*. Ed. por Sebastiano Battiato, Giovanni Gallo, Raimondo Schettini y Filippo Stanco. Cham: Springer International Publishing, 2017, págs. 729-738. ISBN: 978-3-319-68560-1.

[5] Leo Breiman. «Bagging predictors». En: *Machine learning* 24.2 (1996), págs. 123-140.

[6] Leo Breiman. «Random forests». En: *Machine learning* 45.1 (2001), págs. 5-32.

[7] Edgard Chammas, Chafic Mokbel y Laurence Likforman-Sulem. «Handwriting Recognition of Historical Documents with Few Labeled Data». En: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. 2018, págs. 43-48. DOI: 10.1109/DAS.2018.15.

[8] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall y W Philip Kegelmeyer. «SMOTE: synthetic minority over-sampling technique». En: *Journal of artificial intelligence research* 16 (2002), págs. 321-357.

[9] Mark Everingham, SM Eslami, Luc Van Gool, Christopher KI Williams, John Winn y Andrew Zisserman. «The pascal visual object classes challenge: A retrospective». En: *International journal of computer vision* 111.1 (2015), págs. 98-136.

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn y Andrew Zisserman. «The pascal visual object classes (voc) challenge». En: *International journal of computer vision* 88.2 (2010), págs. 303-338.

[11] David Fernández Mota. «Contextual word spotting in historical handwritten documents». Tesis doct. Universitat Autònoma de Barcelona, 2015.

[12] Andreas Fischer, Andreas Keller, Volkmar Frinken y Horst Bunke. «Lexicon-free handwritten word spotting using character HMMs». En: *Pattern Recognition Letters* 33.7 (2012). Special Issue on Awards from ICPR 2010, págs. 934-942. ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2011.09.009.

[13] Volkmar Frinken, Andreas Fischer, R. Manmatha y Horst Bunke. «A Novel Word Spotting Method Based on Recurrent Neural Networks». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.2 (2012), págs. 211-224. DOI: 10.1109/TPAMI.2011.113.

[14] Basilis Gatos y Ioannis Pratikakis. «Segmentation-free word spotting in historical printed documents». En: *2009 10th international conference on document analysis and recognition*. IEEE. 2009, págs. 271-275.

[15] Tin Kam Ho. «A data complexity analysis of comparative advantages of decision forest constructors». En: *Pattern Analysis & Applications* 5.2 (2002), págs. 102-112.

[16] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother y Piotr Dollar. «Panoptic Segmentation». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2019, págs. 9404-9413.

[17] Aleksander Kolcz, Joshua Alspector, Mareijke Augusteijn, Robert Carlson y G Viorel Popescu. «A line-oriented approach to word spotting in handwritten documents». En: *Pattern Analysis & Applications* 3.2 (2000), págs. 153-168.

[18] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár y C Lawrence Zitnick. «Microsoft coco: Common objects in context». En: *European conference on computer vision*. Springer. 2014, págs. 740-755.

[19] Rocio Lizarraga-Morales, Raul Sanchez-Yanez, Victor Ayala y Alberto Patlan-Rosales. «Improving a rough set theory-based segmentation approach using adaptable threshold selection and perceptual color spaces». En: *Journal of Electronic Imaging* 23 (feb. de 2014), pág. 013024. DOI: 10.1117/1.JEI.23.1.013024.

[20] G. Louloudis, B. Gatos, I. Pratikakis y C. Halatsis. «Text line and word segmentation of handwritten documents». En: *Pattern Recognition* 42.12 (2009). New Frontiers in Handwriting Recognition, págs. 3169-3183. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2008.12.016.

[21] Raghavan Manmatha y WB Croft. «Word spotting: Indexing handwritten archives». En: *Intelligent Multimedia Information Retrieval Collection* (1997), págs. 43-64.

[22] Christopher D Manning. *Introduction to information retrieval*. Syngress Publishing, 2008.

[23] K. Markou, L. Tsochatzidis, K. Zagoris, A. Papazoglou, X. Karagiannis, S. Symeonidis e I. Pratikakis. «A Convolutional Recurrent Neural Network for the Handwritten Text Recognition of Historical Greek Manuscripts». En: *Pattern Recognition. ICPR International Workshops and Challenges*. Ed. por Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante y Roberto Vezzani. Cham: Springer International Publishing, 2021, págs. 249-262. ISBN: 978-3-030-68787-8.

[24] D. Martin, C. Fowlkes, D. Tal y J. Malik. «A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics». En: *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*. Vol. 2. 2001, 416-423 vol.2. DOI: 10.1109/ICCV.2001.937655.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot y E. Duchesnay. «Scikit-learn: Machine Learning in Python». En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830.

[26] Wisam A. Qader, Musa M. Ameen y Bilal I. Ahmed. «An Overview of Bag of Words;Importance, Implementation, Applications, and Challenges». En: *2019 International Engineering Conference (IEC)*. 2019, págs. 200-204. DOI: 10.1109/IEC47844.2019.8950616.

[27] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid y Silvio Savarese. «Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2019, págs. 658, 666.

[28] Verónica Romero-Gómez, Alejandro H. Toselli, Vicente Bosch, Joan Andreu Sánchez y Enrique Vidal. «Automatic Alignment of Handwritten Images and Transcripts for Training Handwritten Text Recognition Systems». En: *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. 2018, págs. 328-333. DOI: 10.1109/DAS.2018.41.

[29] Leonard Rothacker, Marçal Rusiñol y Gernot A. Fink. «Bag-of-Features HMMs for Segmentation-Free Word Spotting in Handwritten Documents». En: *2013 12th International Conference on Document Analysis and Recognition*. 2013, págs. 1305-1309. DOI: 10.1109/ICDAR.2013.264.

[30] Stuart J Russell. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2010.

[31] Joan Andreu Sánchez, Verónica Romero, Alejandro H. Toselli, Mauricio Villegas y Enrique Vidal. «A set of benchmarks for Handwritten Text Recognition on historical documents». En: *Pattern Recognition* 94 (2019), págs. 122-134. ISSN: 0031-3203. DOI: https://doi.org/10.1016/j.patcog.2019.05.025.

[32] Robert E Schapire. «The strength of weak learnability». En: *Machine learning* 5.2 (1990), págs. 197-227.

[33] M. Schuster y K.K. Paliwal. «Bidirectional recurrent neural networks». En: *IEEE Transactions on Signal Processing* 45.11 (1997), págs. 2673-2681. DOI: 10.1109/78.650093.

[34] Reuben R. Shamir, Yuval Duchin, Jinyoung Kim, Guillermo Sapiro y Noam Harel. «Continuous Dice Coefficient: a Method for Evaluating Probabilistic Segmentations». En: *CoRR* abs/1906.11031 (2019). arXiv: 1906.11031. URL: http://arxiv.org/abs/1906.11031.

[35] Manisha Sharma y Vandana Chouhan. «Objective evaluation parameters of image segmentation algorithms». En: *International Journal of Engineering and Advanced Technology (IJEAT)* 2.2 (2012), págs. 2249-8958.

[36] Kengo Terasawa y Yuzuru Tanaka. «Slit Style HOG Feature for Document Image Word Spotting». En: *2009 10th International Conference on Document Analysis and Recognition*. 2009, págs. 116-120. DOI: 10.1109/ICDAR.2009.118.

[37] Bayes Thomas. En: *LII. An essay towards solving a problem in the doctrine of chances. By the late Rev. Mr. Bayes, F. R. S. communicated by Mr. Price, in a letter to John Canton, A. M. F. R. S.* 1763. DOI: 10.1098/rstl.1763.0053.

[38] Paul Voigtlaender, Patrick Doetsch y Hermann Ney. «Handwriting Recognition with Large Multidimensional Long Short-Term Memory Recurrent Neural Networks». En: *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. 2016, págs. 228-233. DOI: 10.1109/ICFHR.2016.0052.

[39] Haiyi Zhang y Di Li. «Naïve Bayes Text Classifier». En: *2007 IEEE International Conference on Granular Computing (GRC 2007)*. 2007, págs. 708-708. DOI: 10.1109/GrC.2007.40.

[40] Harry Zhang. «The Optimality of Naive Bayes». En: *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004)*. May 17-19, 2004 (Miami Beach, Florida, USA). Ed. por Valerie Barr y Zdravko Markov. AAAI Press, 2004, págs. 562-567.

APÉNDICE A
Histogramas de características

A.1 Altura

A.1.1. Primer conjunto

Las Figuras A.1 a A.4 son los correspondientes histogramas de características para cada una de las clases del primer conjunto, usando la primera extracción de características. Nótese que todos los histogramas de este apéndice tienen distintas escalas en el eje Y.

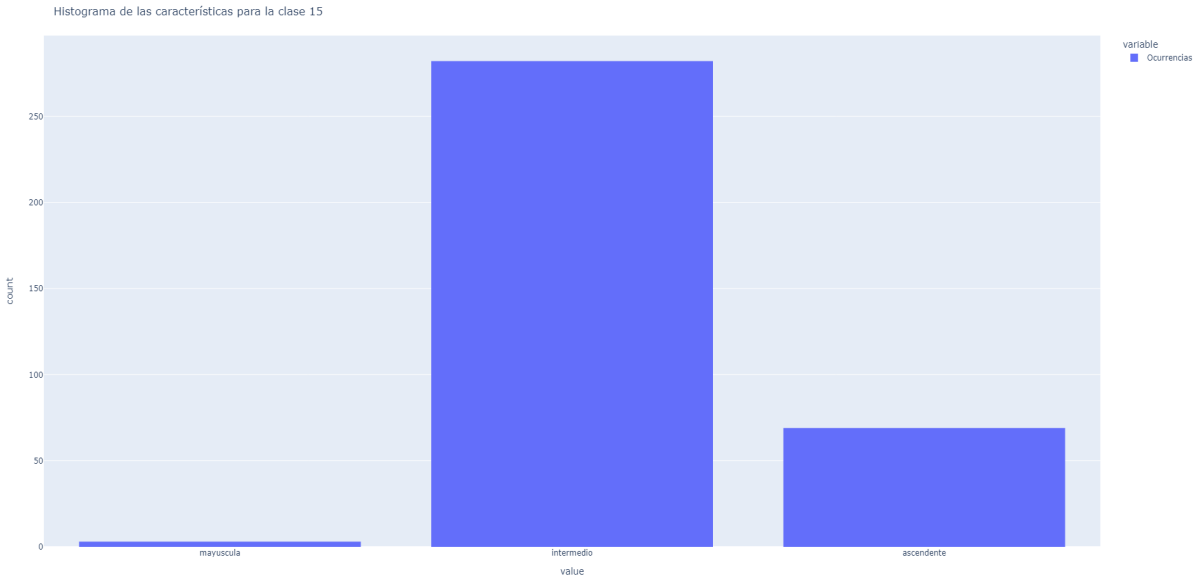

Figura A.1: Histograma que muestra la distribución de las características usando la primera extracción en el primer conjunto de clases. Clase 15

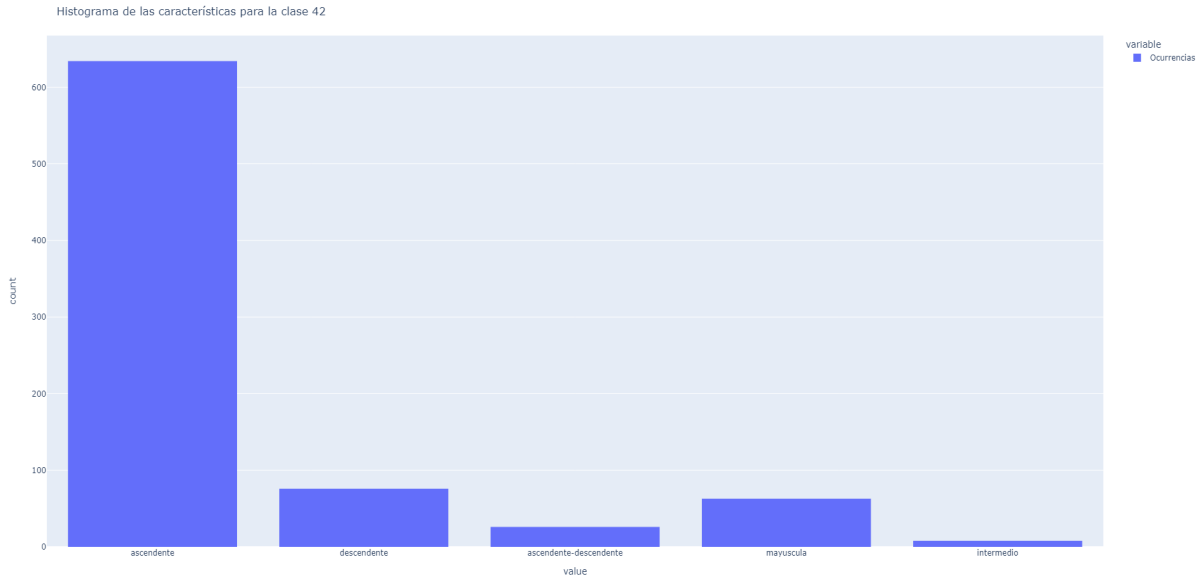

Figura A.2: Histograma que muestra la distribución de las características usando la primera extracción en el primer conjunto de clases. Clase 42

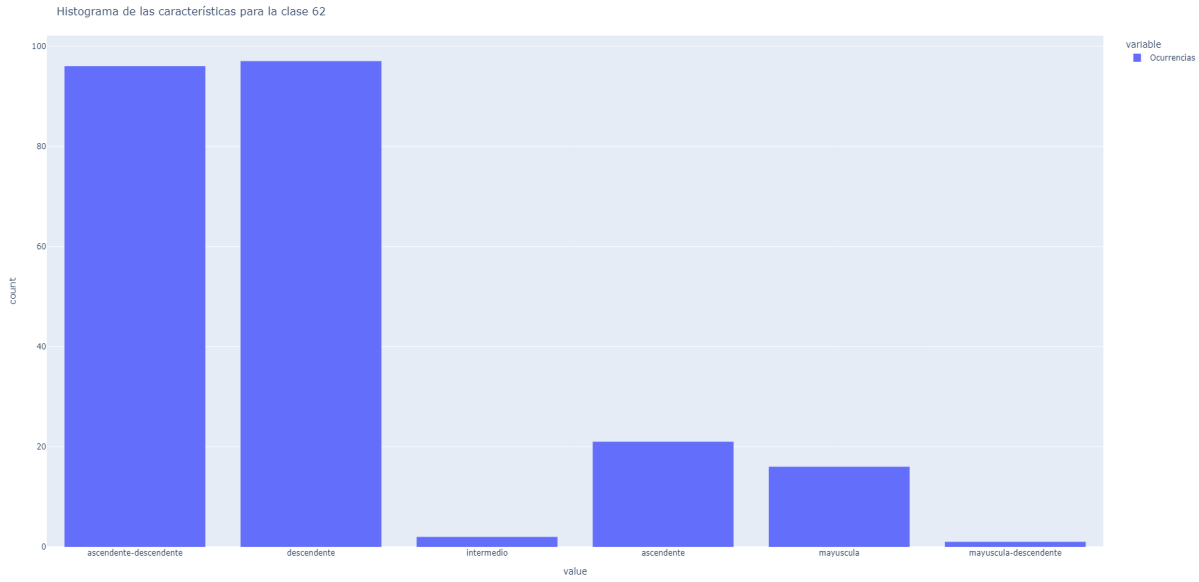

Figura A.3: Histograma que muestra la distribución de las características usando la primera extracción en el primer conjunto de clases. Clase 62

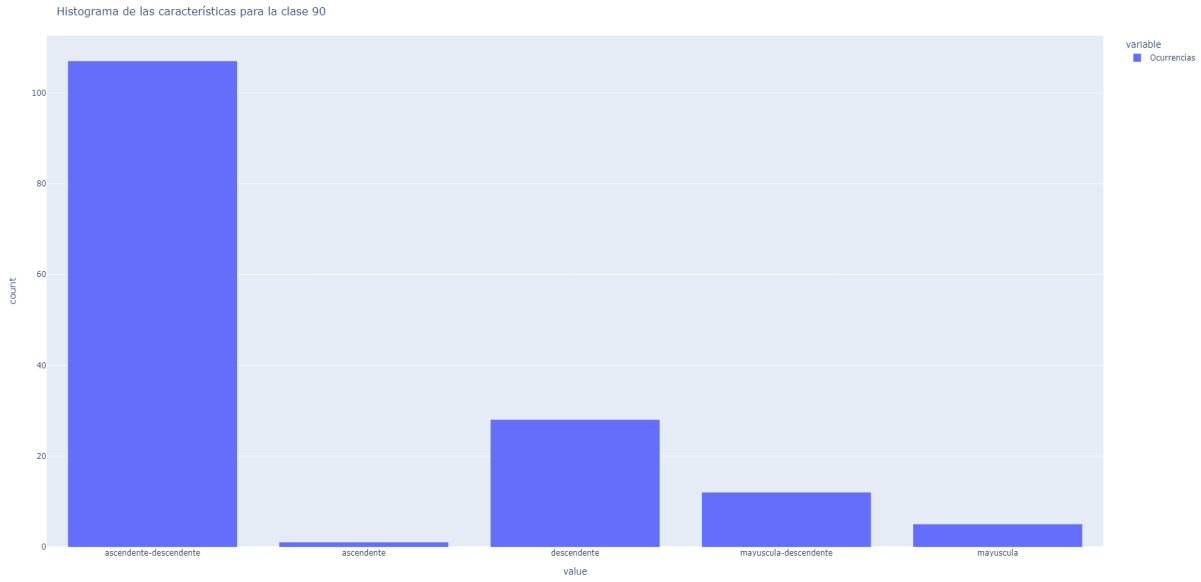

Figura A.4: Histograma que muestra la distribución de las características usando la primera extracción en el primer conjunto de clases. Clase 90

Se muestran los resultados de usar la segunda extracción en las Figuras A.5 a A.8.

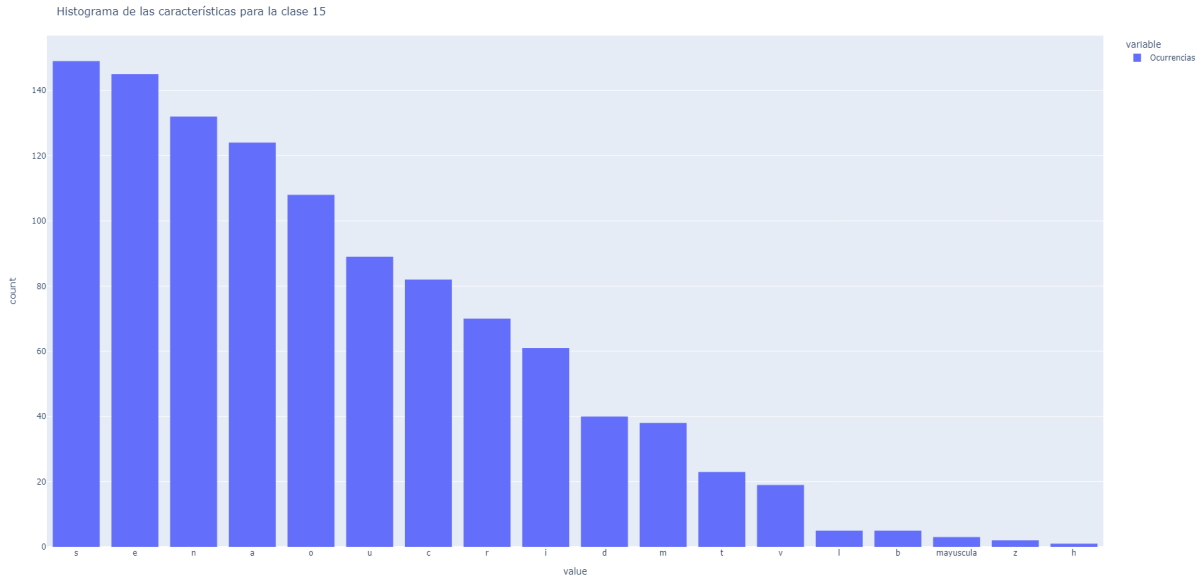

Figura A.5: Histograma que muestra la distribución de las características usando la segunda extracción en el primer conjunto de clases. Clase 15

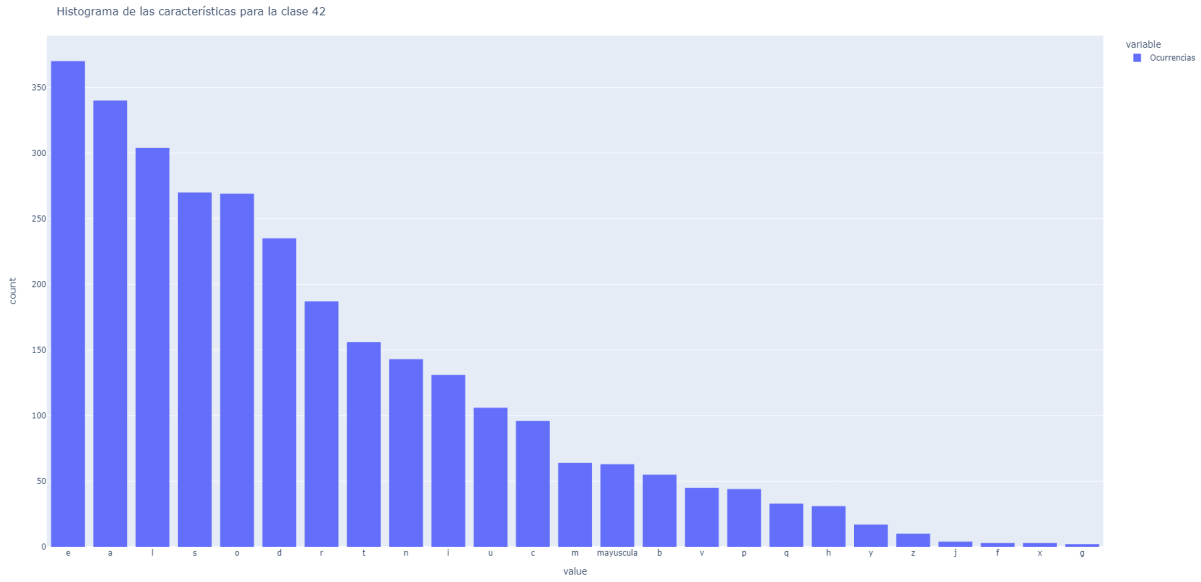

Figura A.6: Histograma que muestra la distribución de las características usando la segunda extracción en el primer conjunto de clases. Clase 42

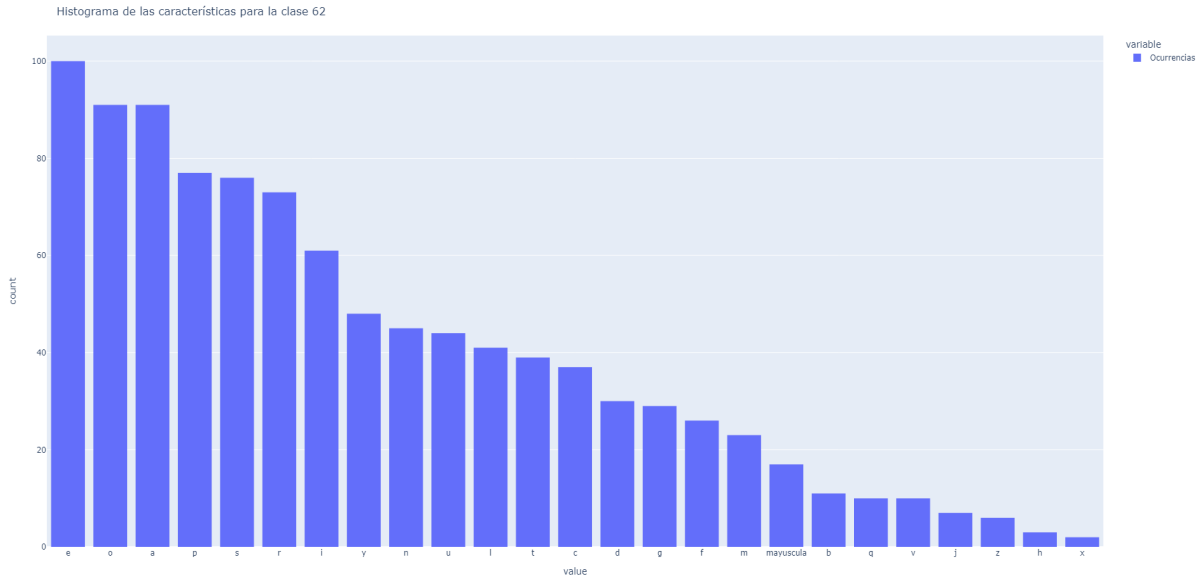

Figura A.7: Histograma que muestra la distribución de las características usando la segunda extracción en el primer conjunto de clases. Clase 62

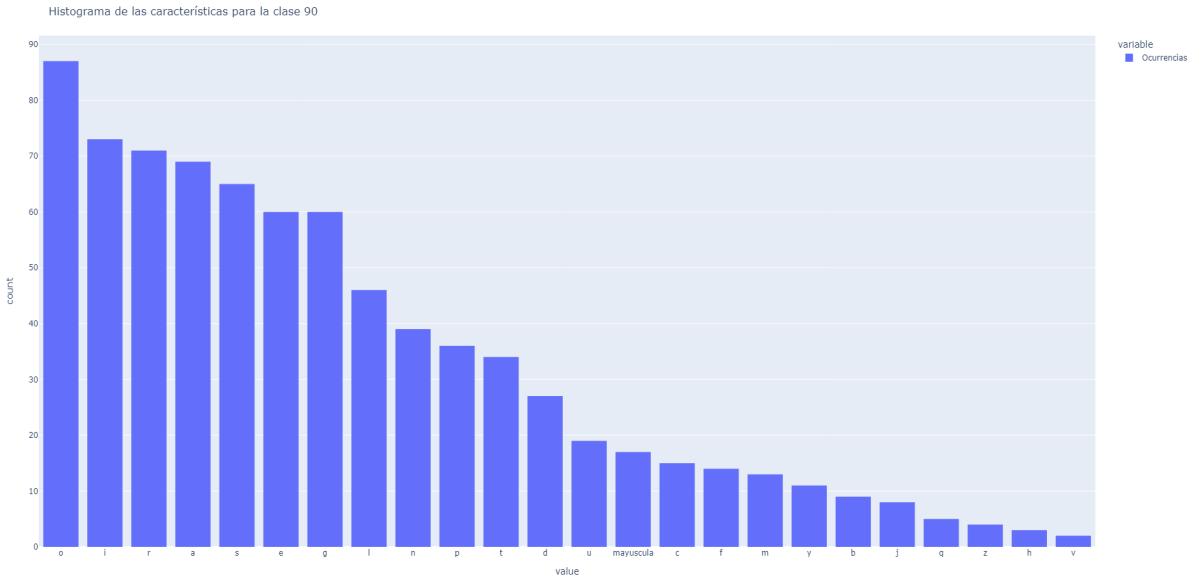

Figura A.8: Histograma que muestra la distribución de las características usando la segunda extracción en el primer conjunto de clases. Clase 90

Se pueden ver los resultados de la tercera extracción en las Figuras A.9 a A.12.

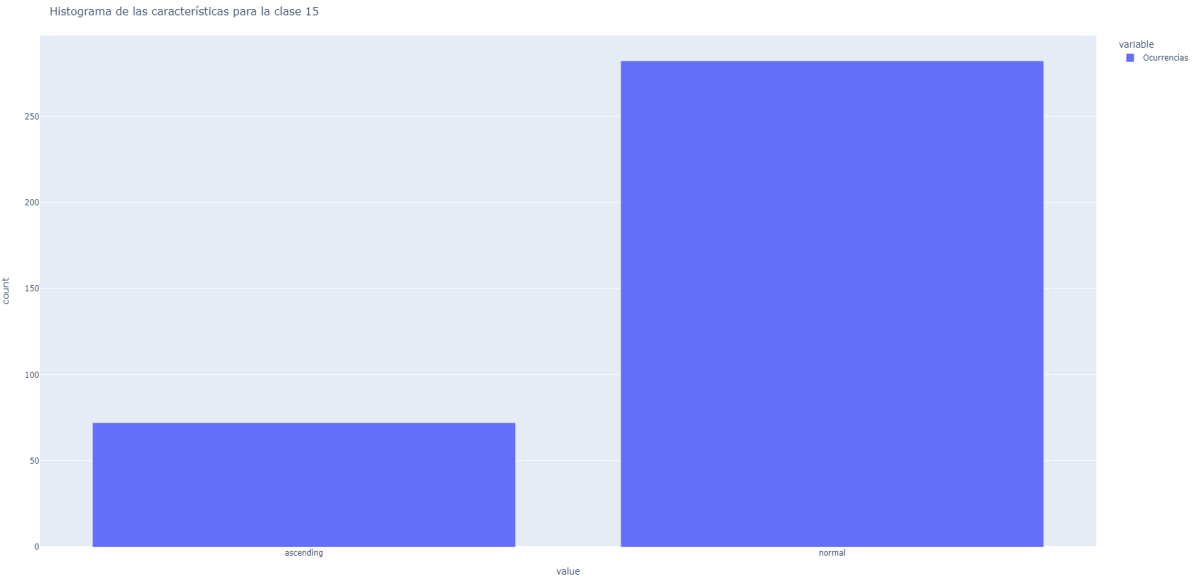

Figura A.9: Histograma que muestra la distribución de las características usando la tercera extracción en el primer conjunto de clases. Clase 15

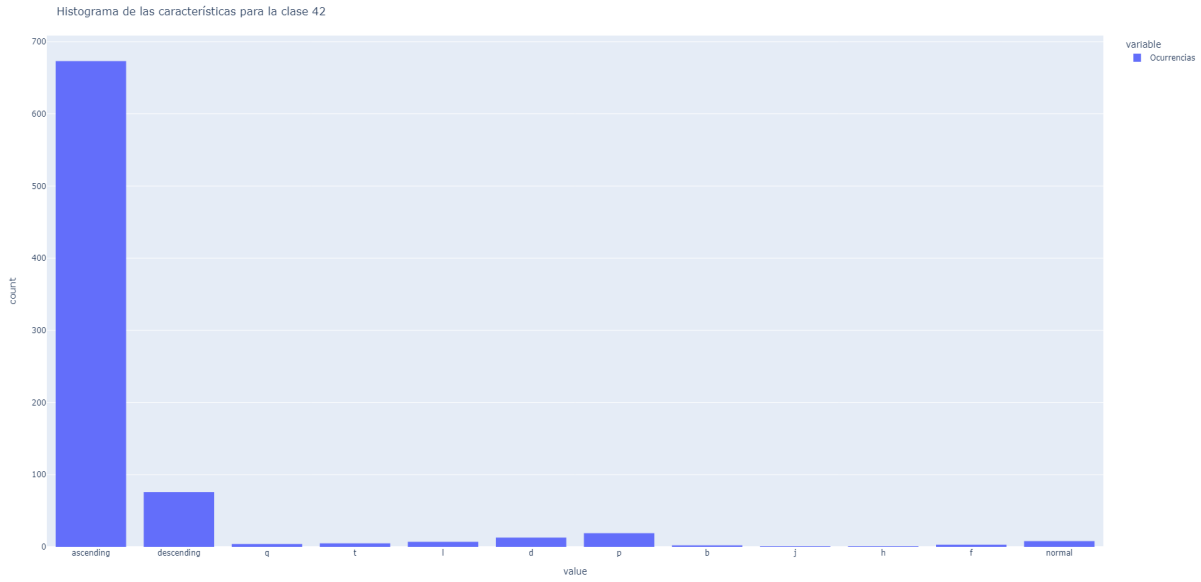

Figura A.10: Histograma que muestra la distribución de las características usando la tercera extracción en el primer conjunto de clases. Clase 42

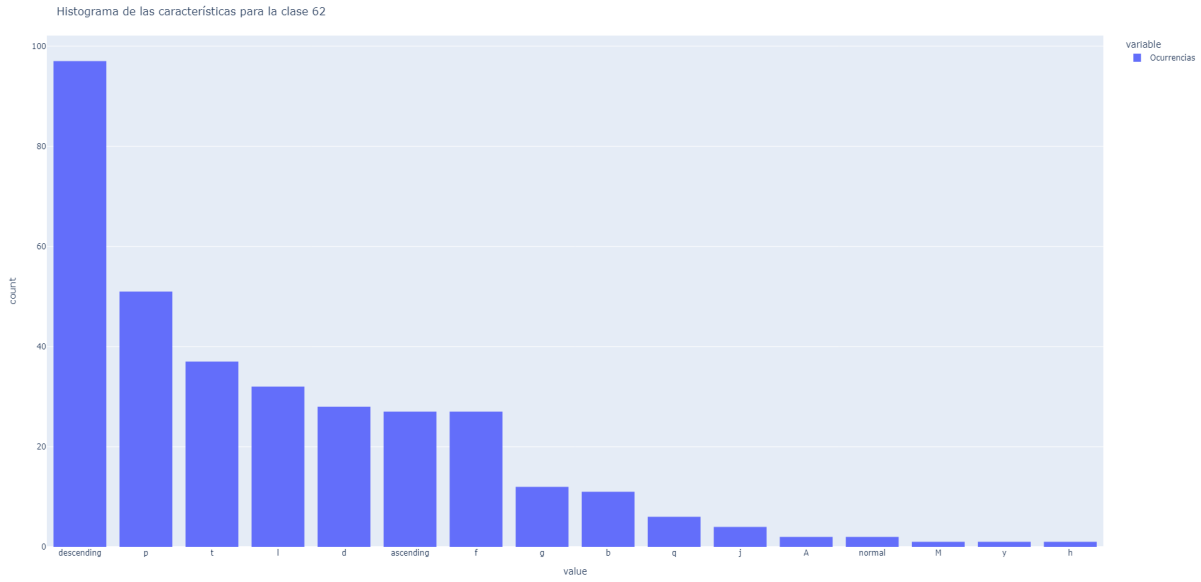

Figura A.11: Histograma que muestra la distribución de las características usando la tercera extracción en el primer conjunto de clases. Clase 62

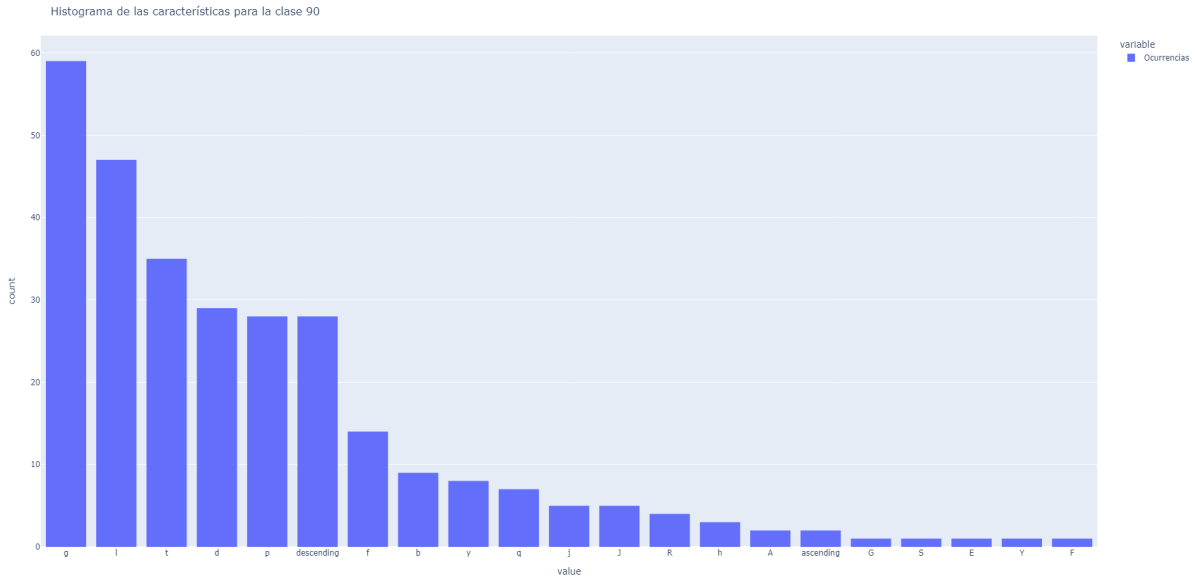

Figura A.12: Histograma que muestra la distribución de las características usando la tercera extracción en el primer conjunto de clases. Clase 90

A.1.2. Segundo conjunto

En las Figuras A.13 a A.16 son los correspondientes histogramas de características para cada una de las clases del segundo conjunto, usando la primera extracción:

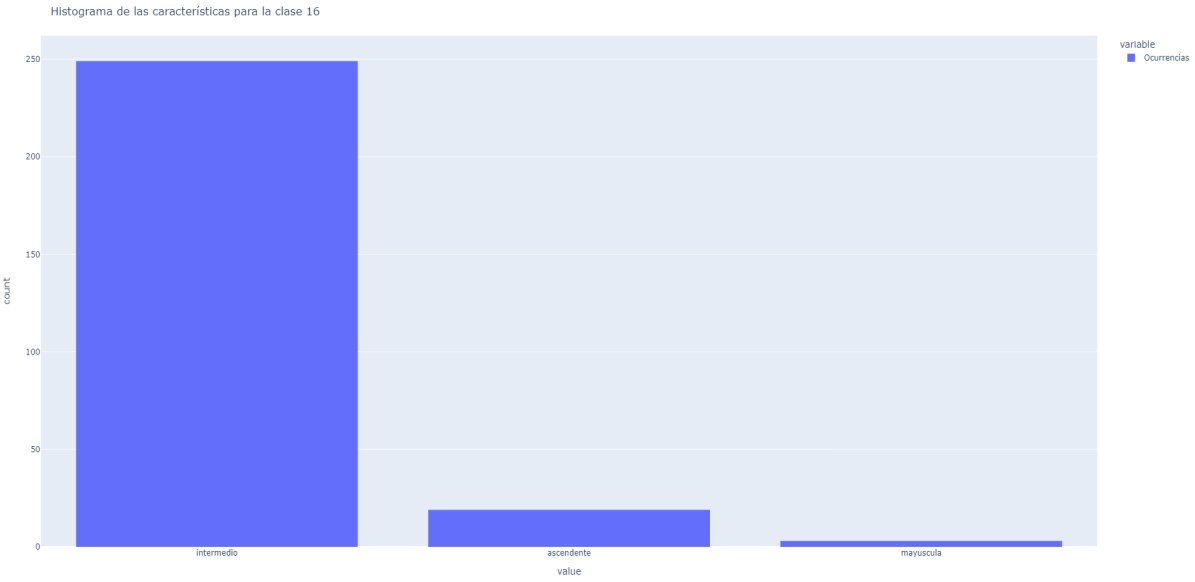

Figura A.13: Histograma que muestra la distribución de las características usando la primera extracción en el segundo conjunto de clases. Clase 16

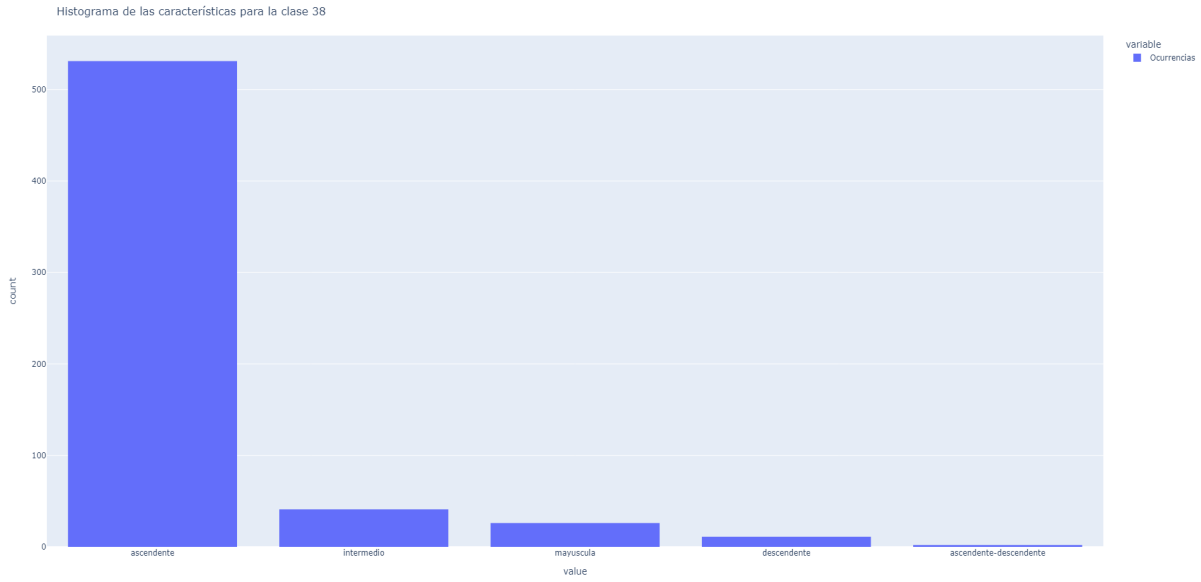

Figura A.14: Histograma que muestra la distribución de las características usando la primera extracción en el segundo conjunto de clases. Clase 38

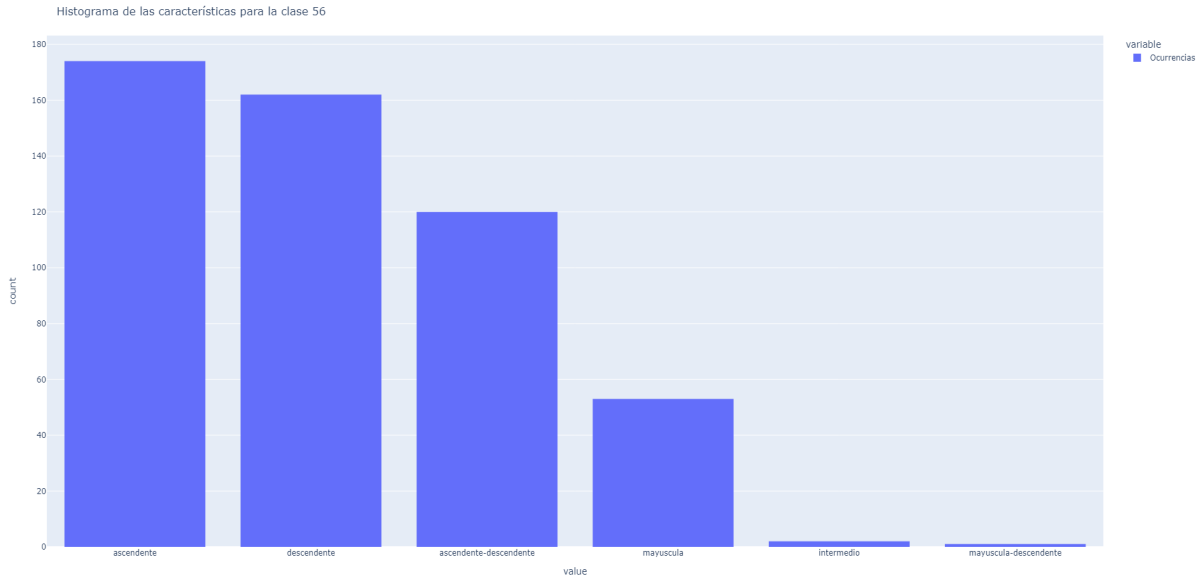

Figura A.15: Histograma que muestra la distribución de las características usando la primera extracción en el segundo conjunto de clases. Clase 56

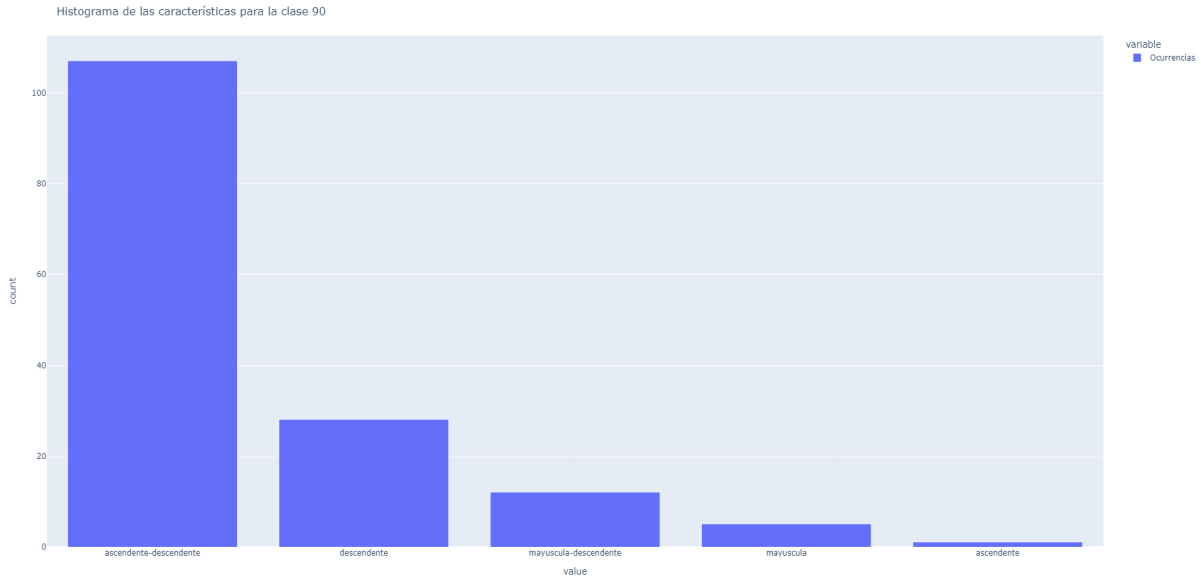

Figura A.16: Histograma que muestra la distribución de las características usando la primera extracción en el segundo conjunto de clases. Clase 90

Se muestran los resultados de usar la segunda extracción en las Figuras A.17 a A.20

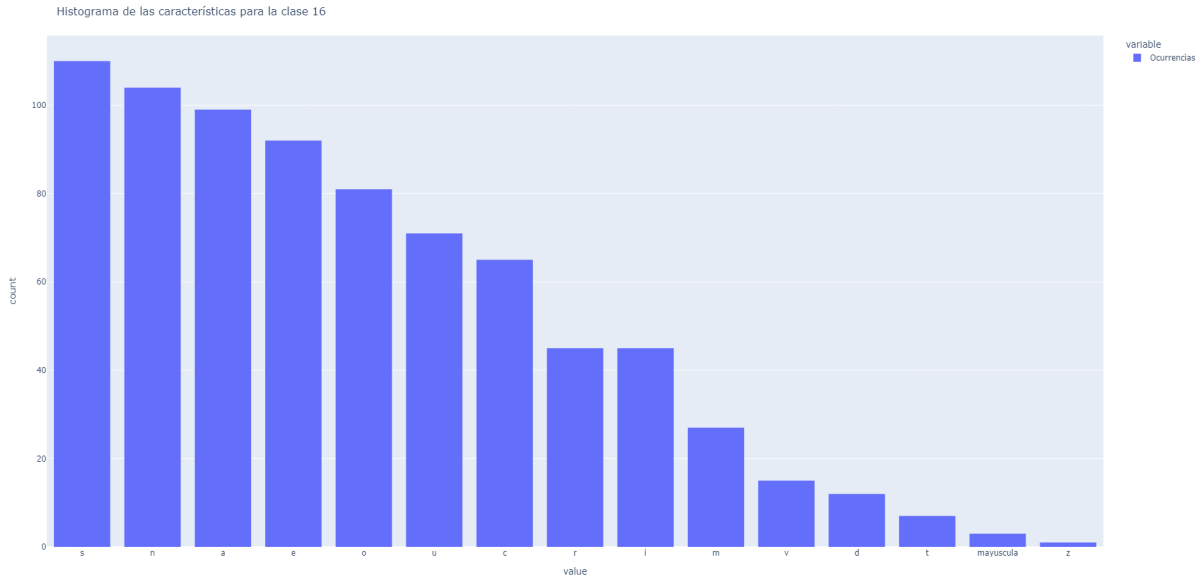

Figura A.17: Histograma que muestra la distribución de las características usando la segunda extracción en el segundo conjunto de clases. Clase 16

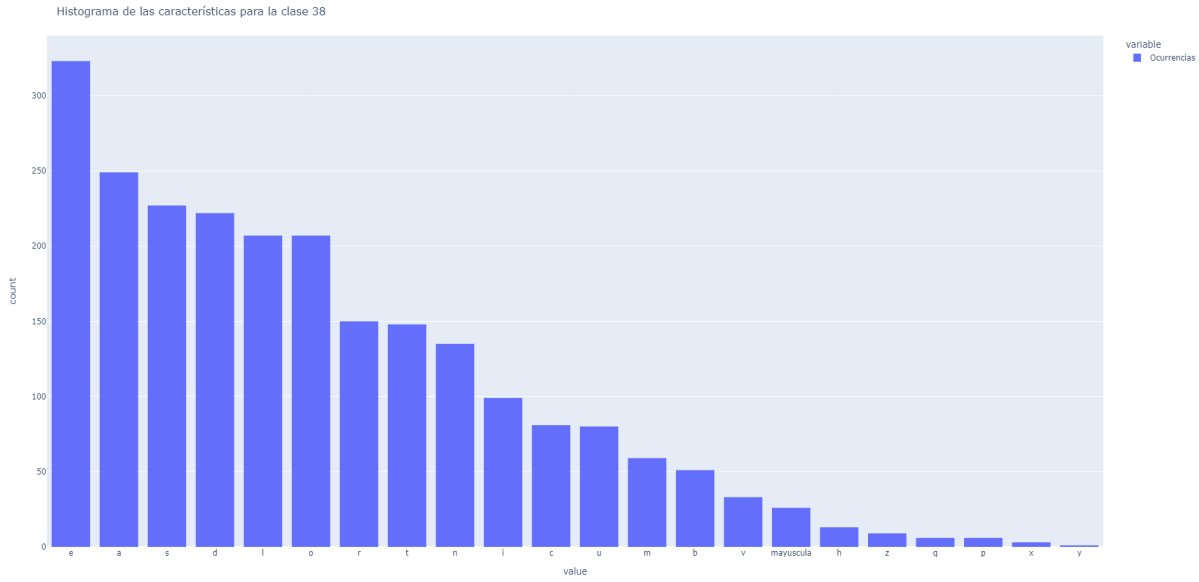

Figura A.18: Histograma que muestra la distribución de las características usando la segunda extracción en el segundo conjunto de clases. Clase 38

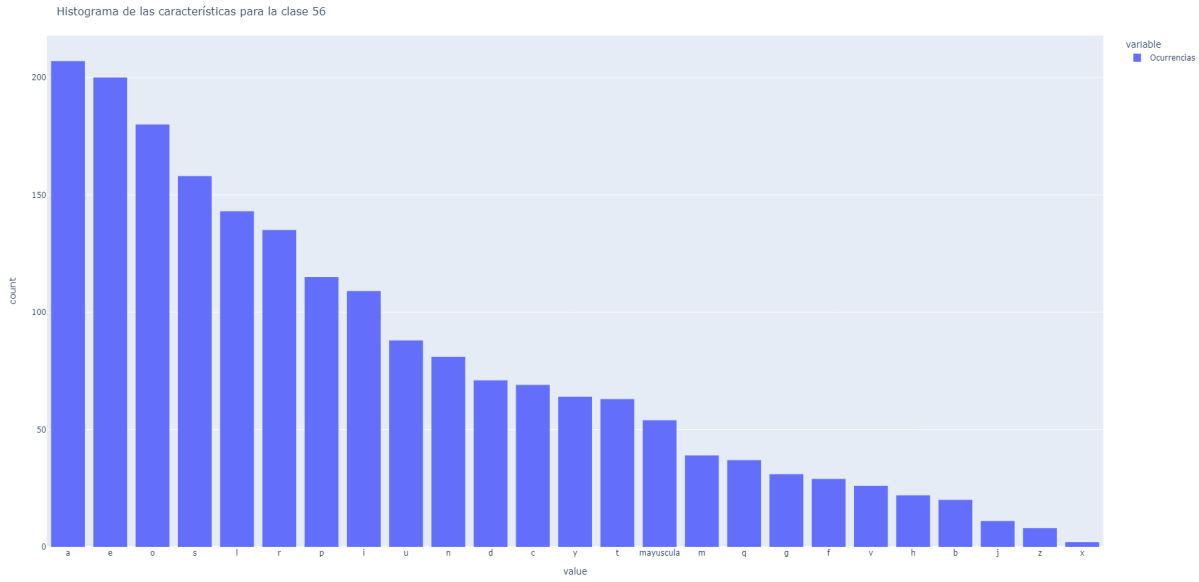

Figura A.19: Histograma que muestra la distribución de las características usando la segunda extracción en el segundo conjunto de clases. Clase 56

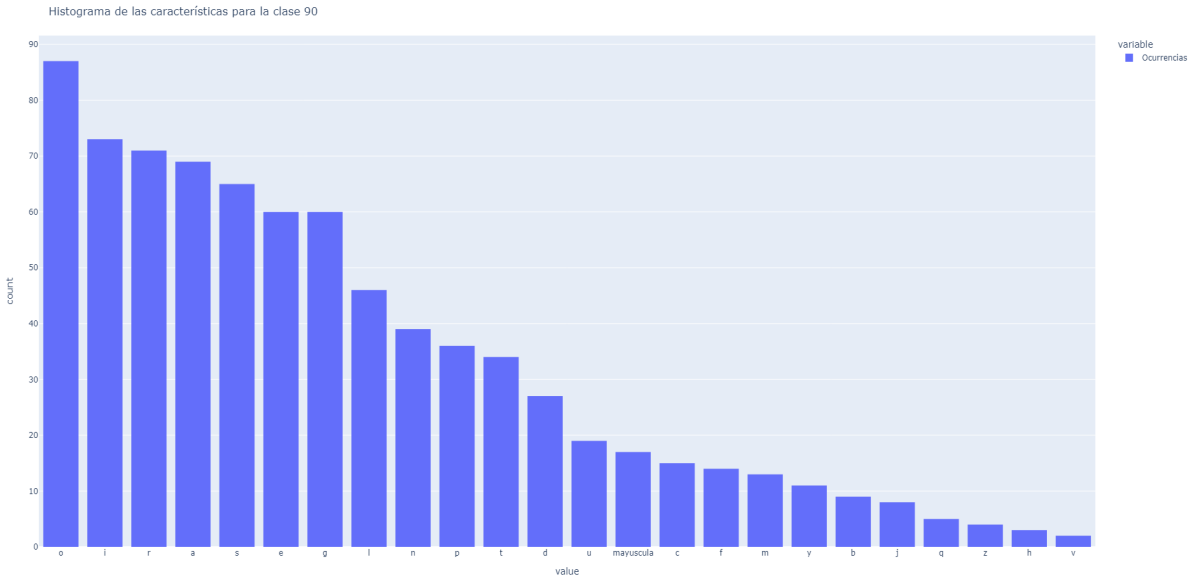

Figura A.20: Histograma que muestra la distribución de las características usando la segunda extracción en el segundo conjunto de clases. Clase 90

Usando la tercera extracción, los resultados son los mostrados en las Figuras A.21 a A.24

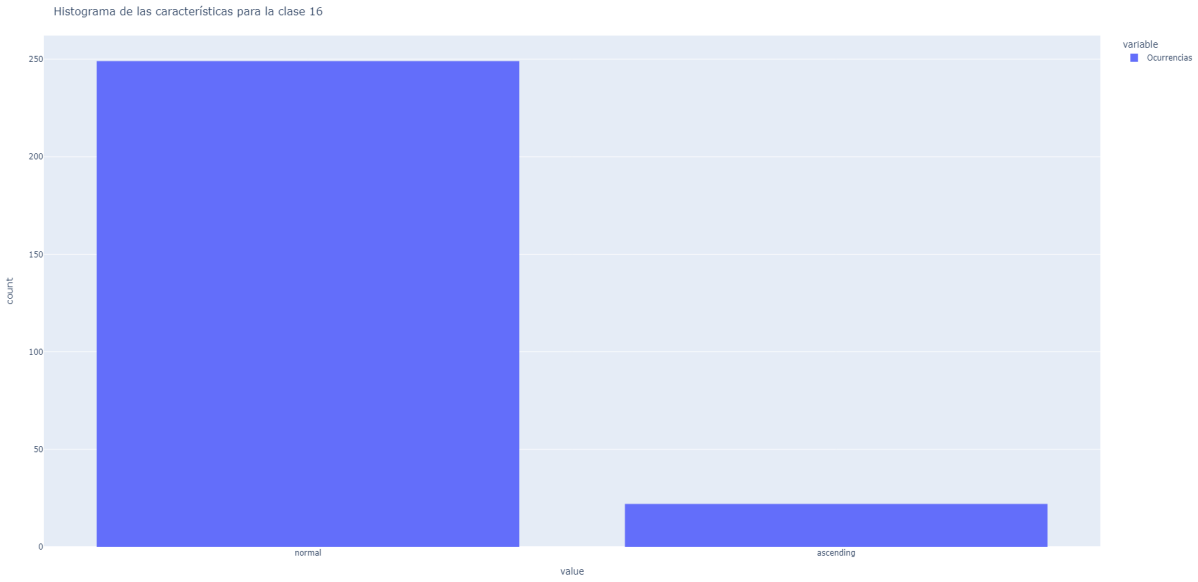

Figura A.21: Histograma que muestra la distribución de las características usando la tercera extracción en el segundo conjunto de clases. Clase 16

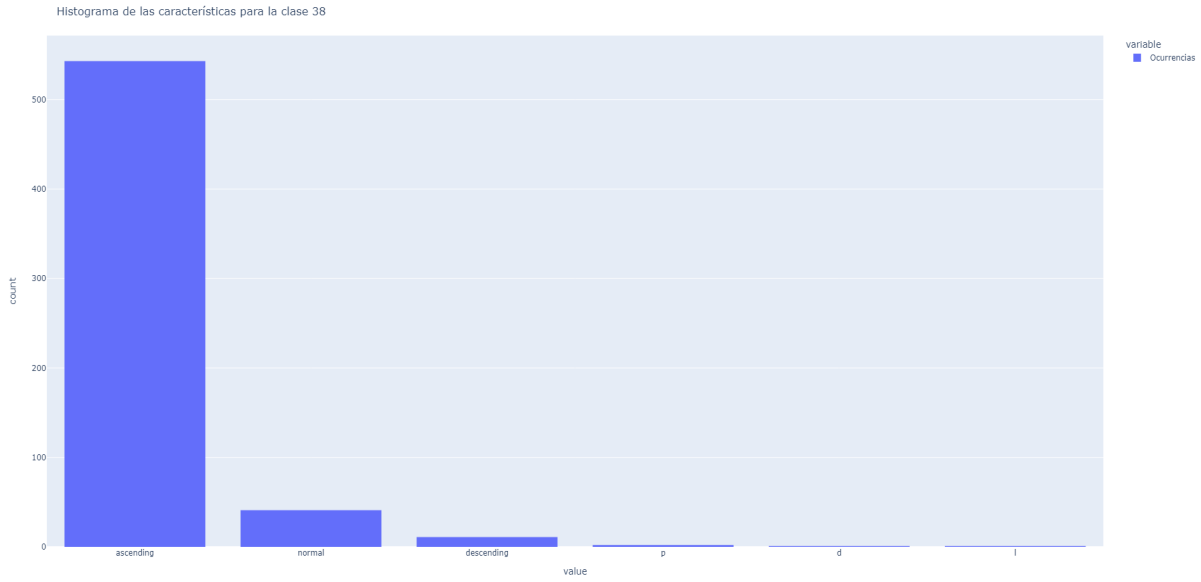

Figura A.22: Histograma que muestra la distribución de las características usando la tercera extracción en el segundo conjunto de clases. Clase 38

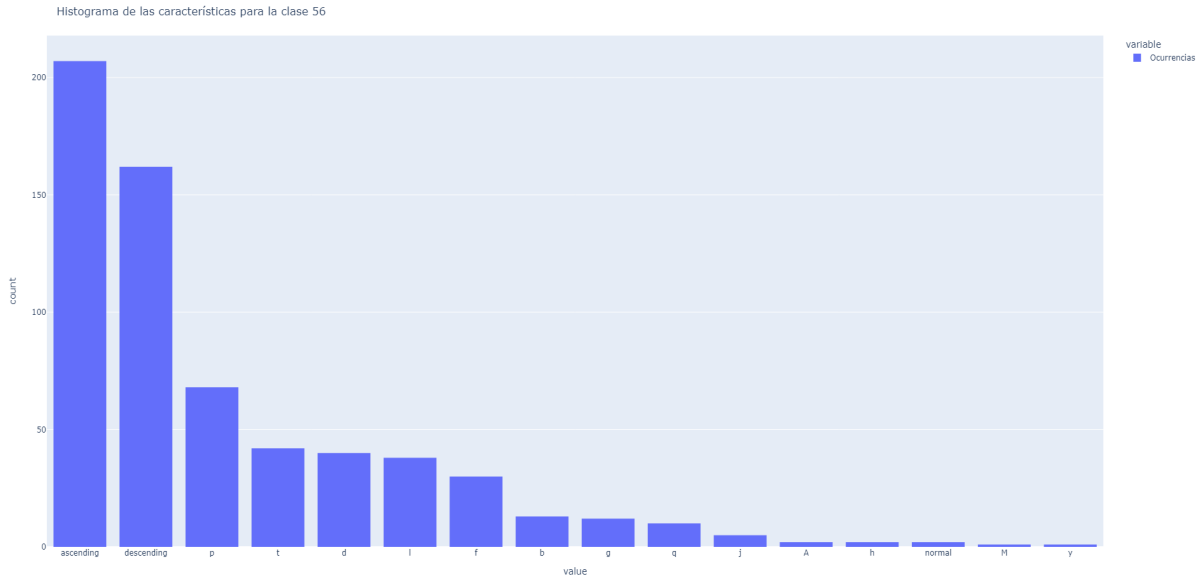

Figura A.23: Histograma que muestra la distribución de las características usando la tercera extracción en el segundo conjunto de clases. Clase 56

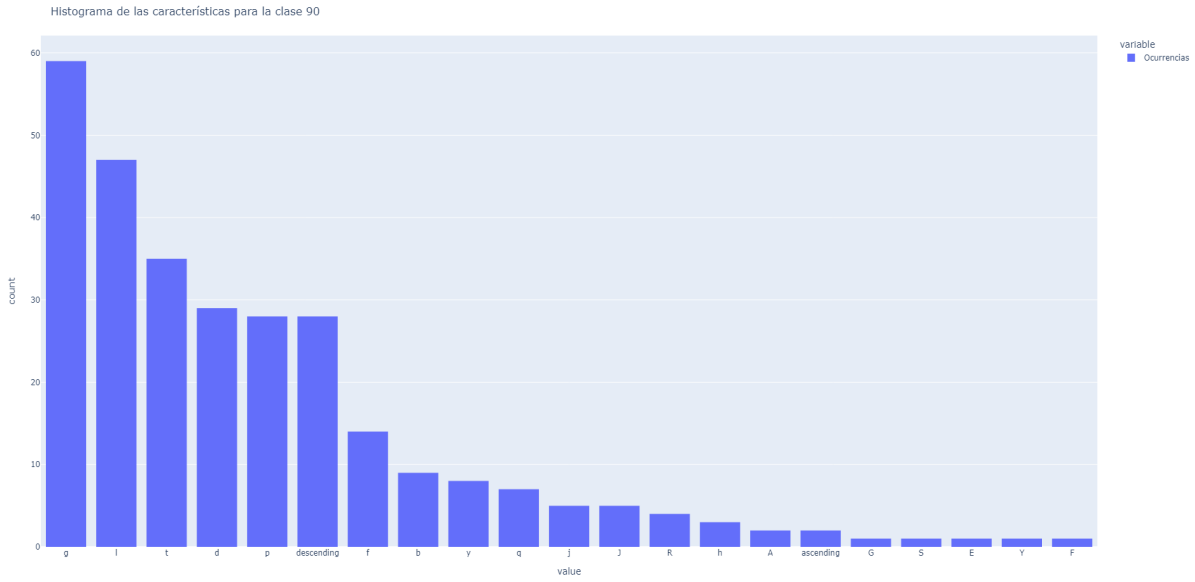

Figura A.24: Histograma que muestra la distribución de las características usando la tercera extracción en el segundo conjunto de clases. Clase 90

A.2 Anchura

Se han omitido histogramas repetidos para los siguientes conjuntos después del primero.

A.2.1. Primer conjunto

Se muestran los histogramas en las Figuras A.25 a A.30

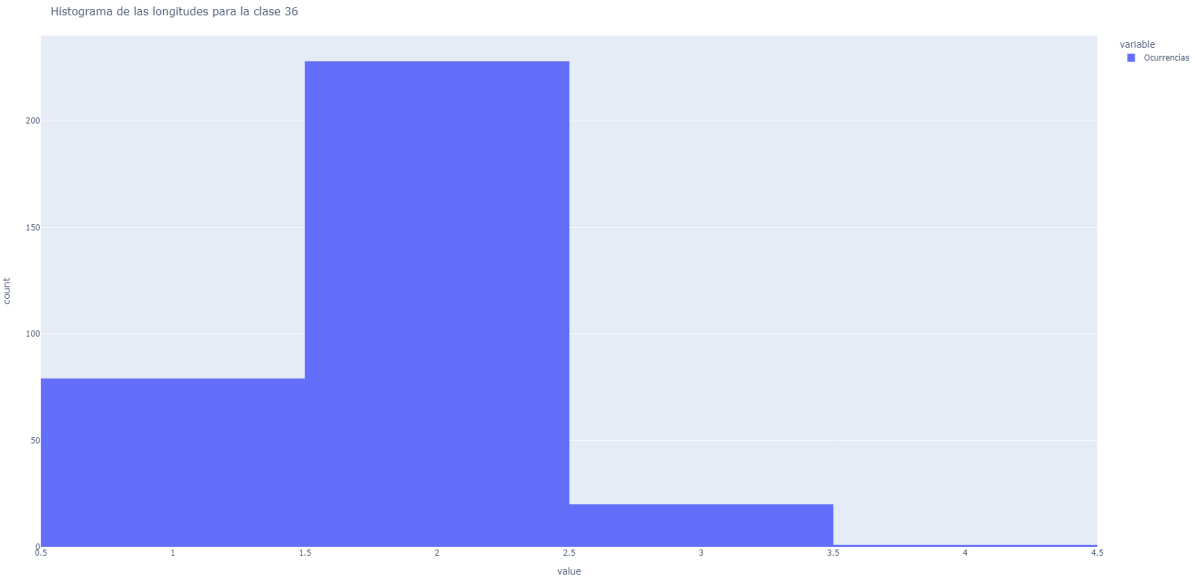

Figura A.25: Histograma para mostrar la longitud de las palabras por clase en el primer conjunto. Clase 36

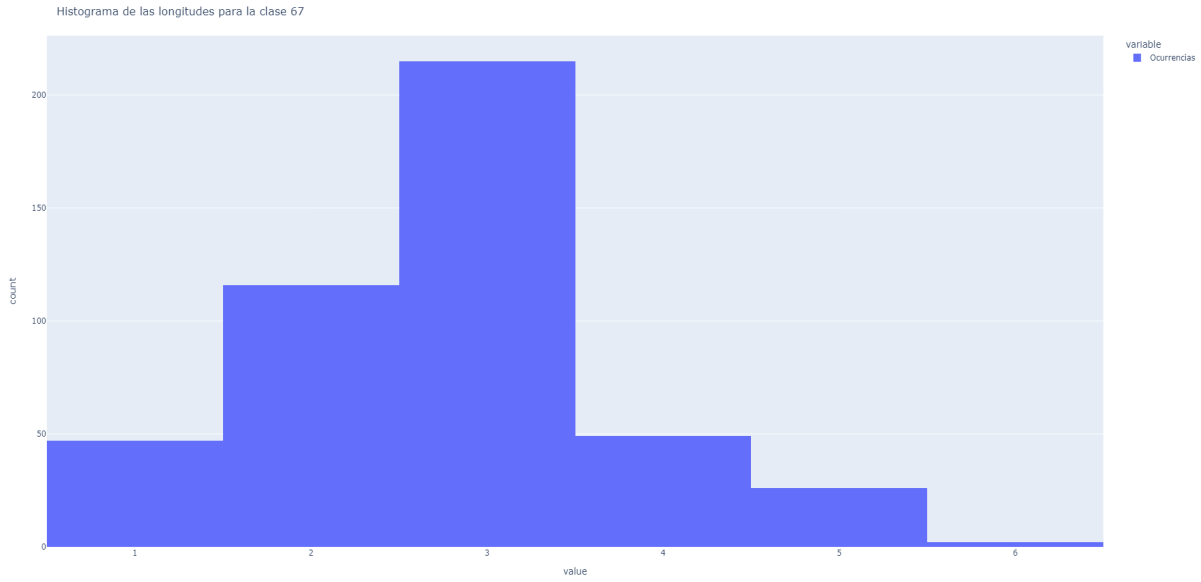

Figura A.26: Histograma para mostrar la longitud de las palabras por clase en el primer conjunto. Clase 67

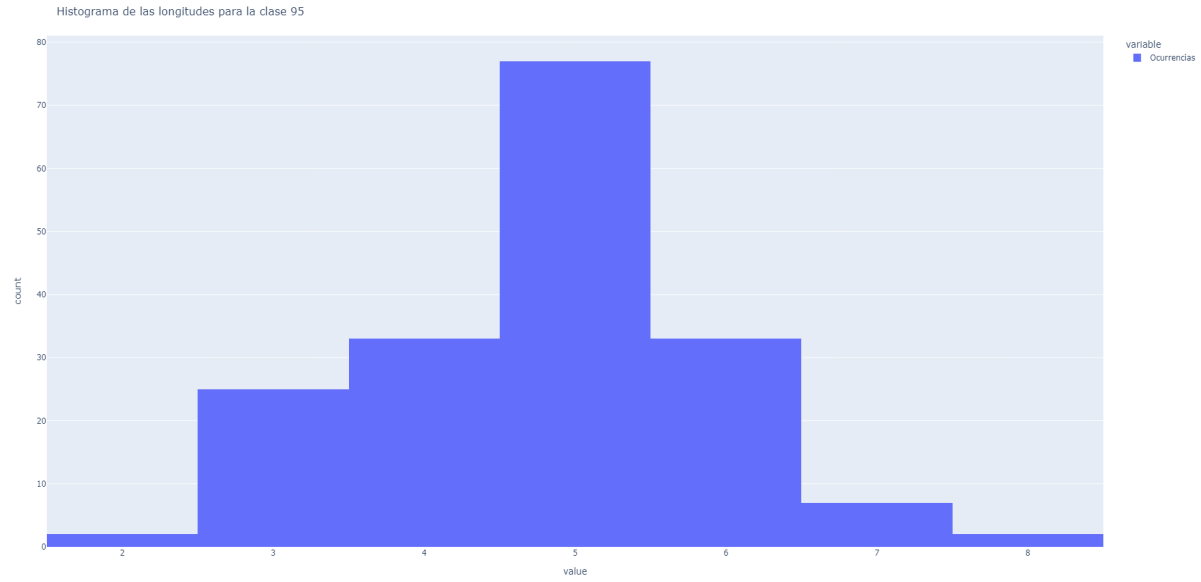

Figura A.27: Histograma para mostrar la longitud de las palabras por clase en el primer conjunto. Clase 95

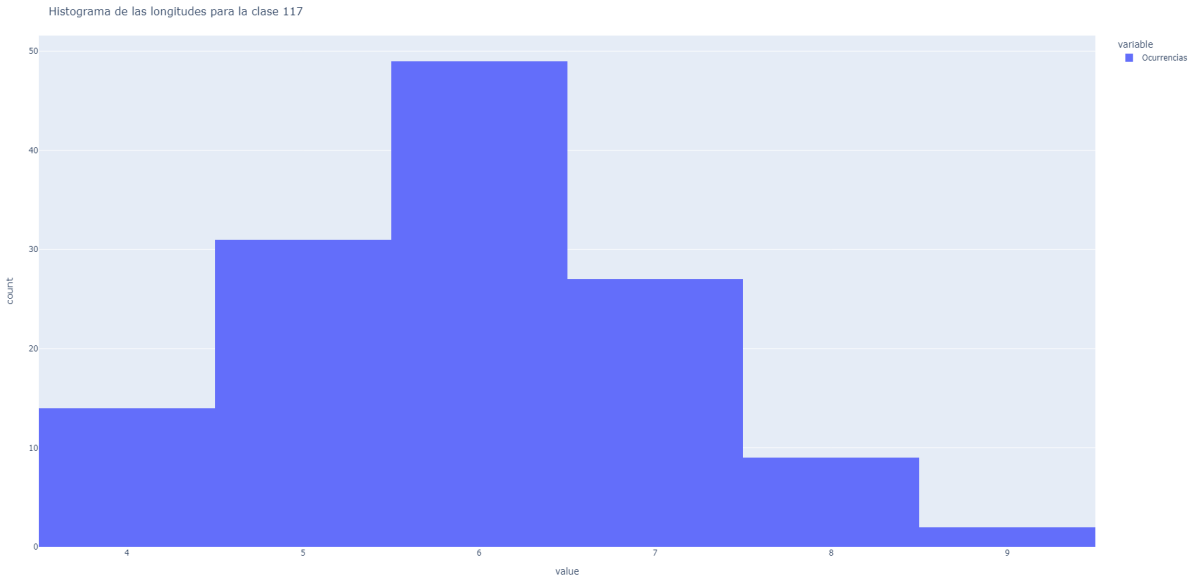

Figura A.28: Histograma para mostrar la longitud de las palabras por clase en el primer conjunto. Clase 117

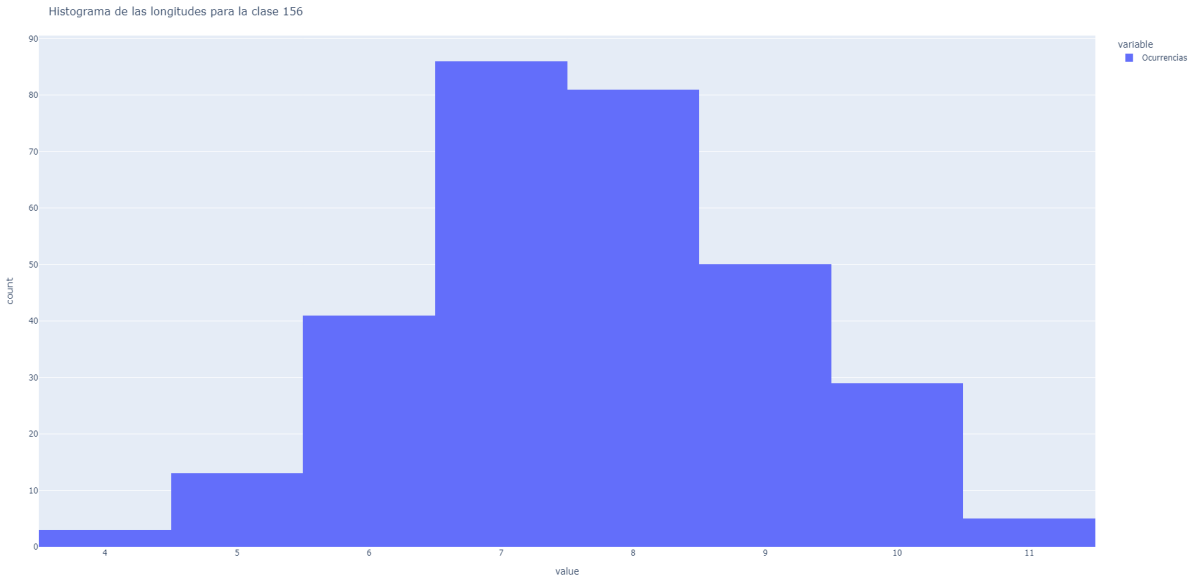

Figura A.29: Histograma para mostrar la longitud de las palabras por clase en el primer conjunto. Clase 156

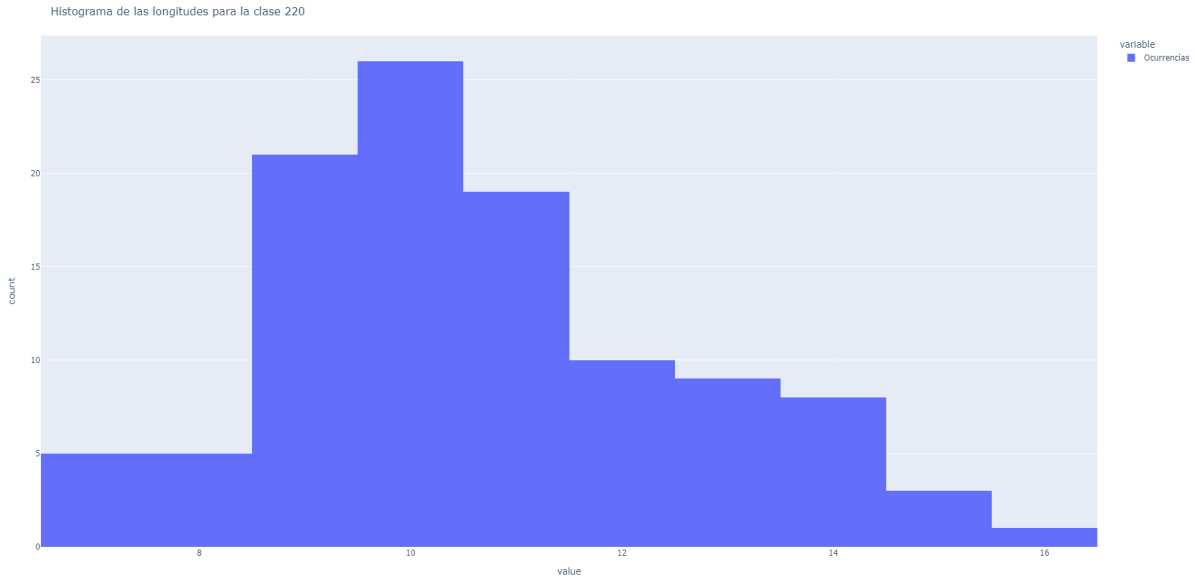

Figura A.30: Histograma para mostrar la longitud de las palabras por clase en el primer conjunto. Clase 220

A.2.2. Segundo conjunto

Se muestran los histogramas en las Figuras A.31 a A.34

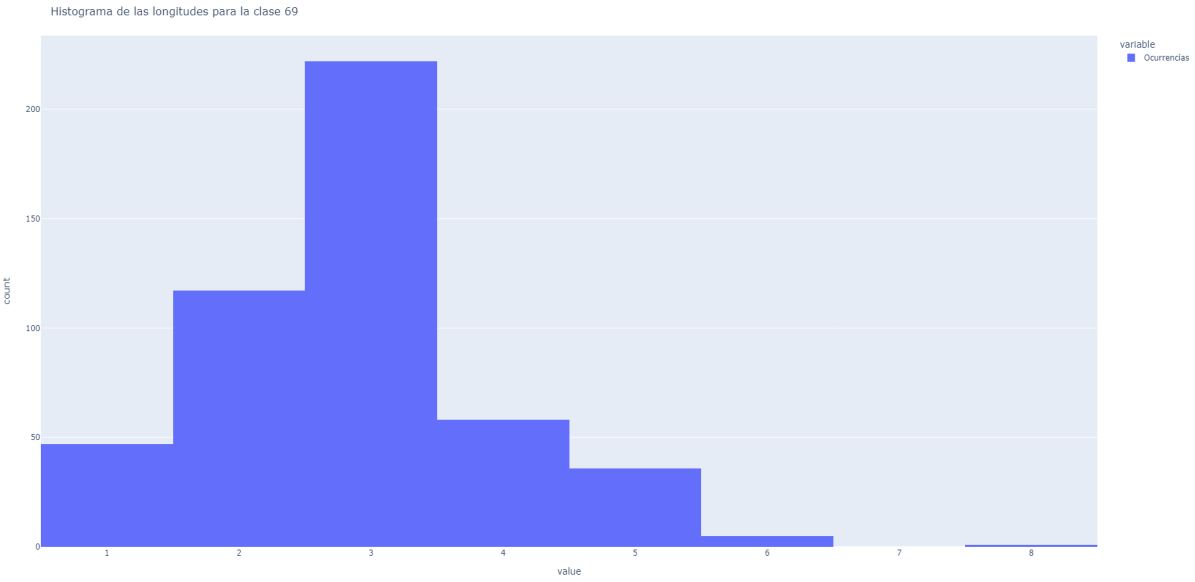

Figura A.31: Histograma para mostrar la longitud de las palabras por clase en el segundo conjunto. Clase 69

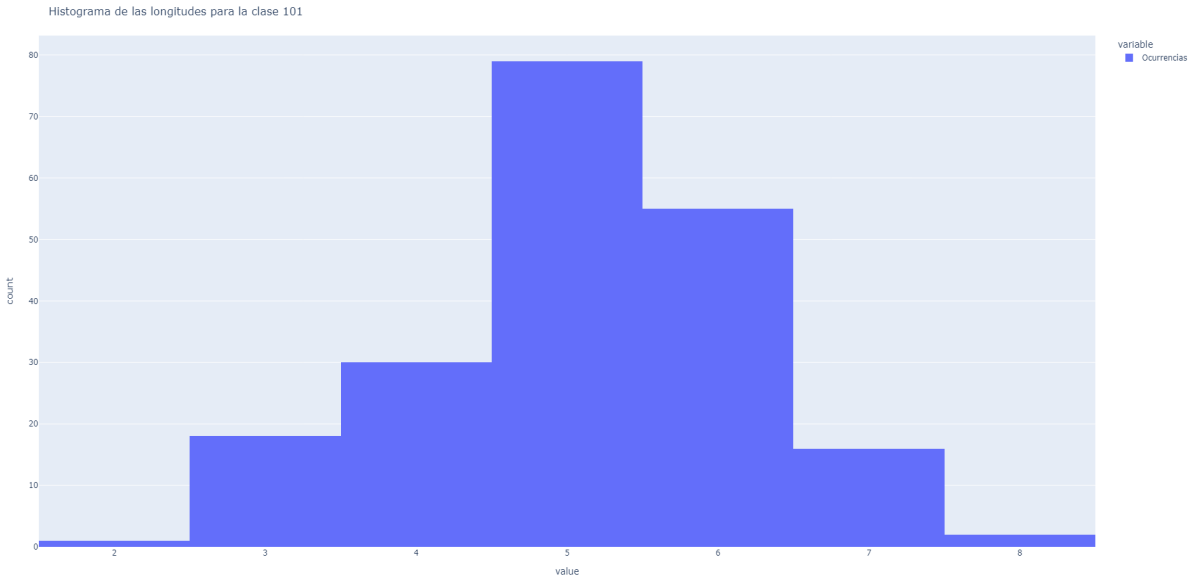

Figura A.32: Histograma para mostrar la longitud de las palabras por clase en el segundo conjunto. Clase 101

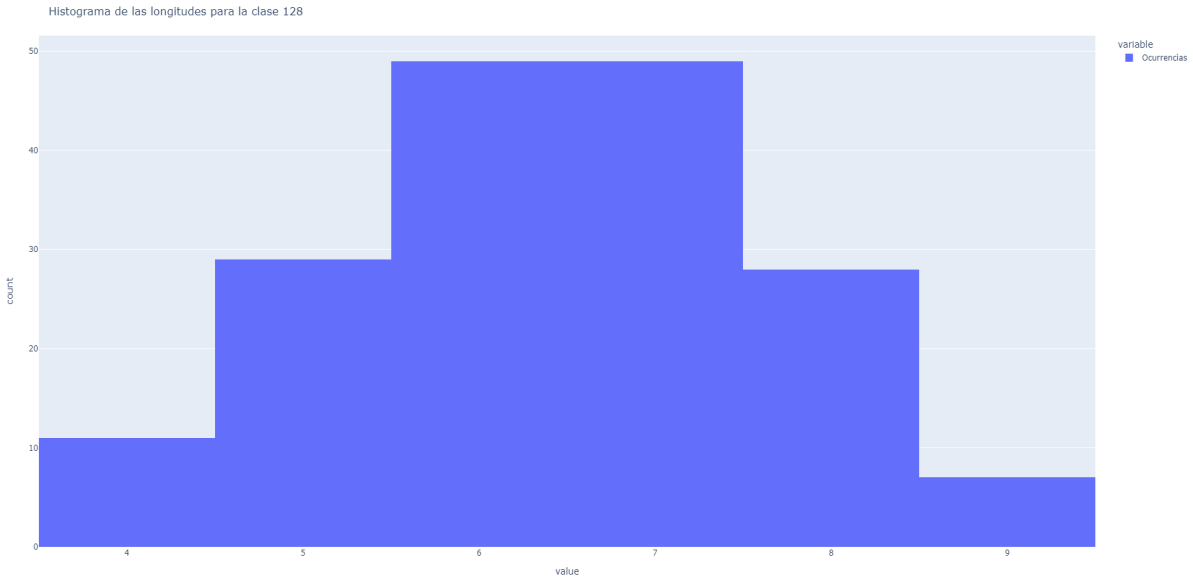

Figura A.33: Histograma para mostrar la longitud de las palabras por clase en el segundo conjunto. Clase 128

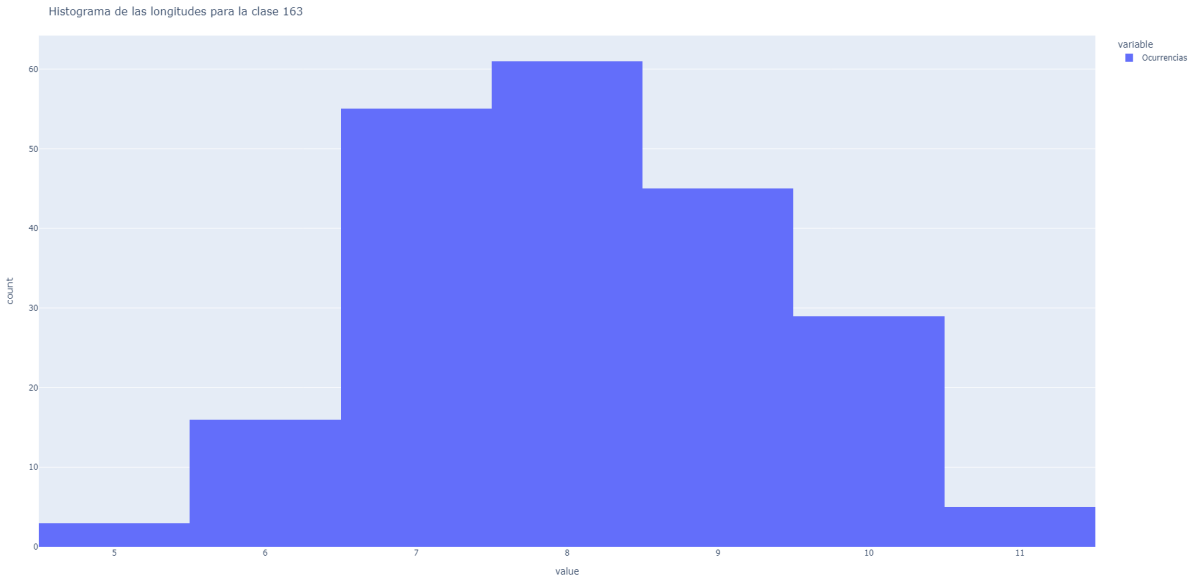

Figura A.34: Histograma para mostrar la longitud de las palabras por clase en el segundo conjunto. Clase 163

A.2.3. Tercer conjunto

Se muestran los histogramas en las Figuras A.35 y A.36

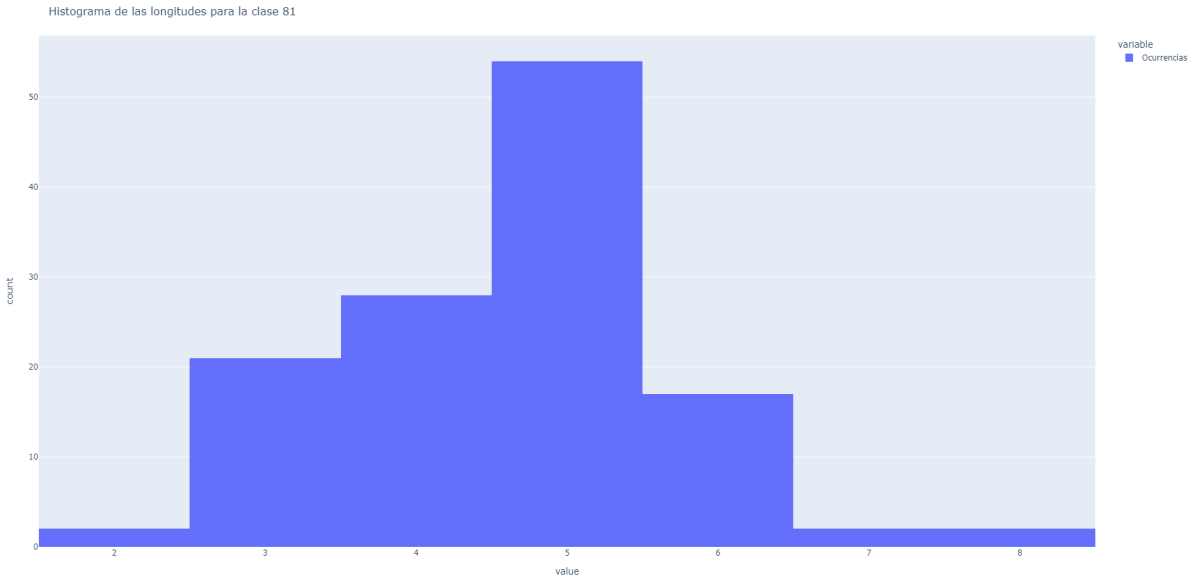

Figura A.35: Histograma para mostrar la longitud de las palabras por clase en el segundo conjunto. Clase 81

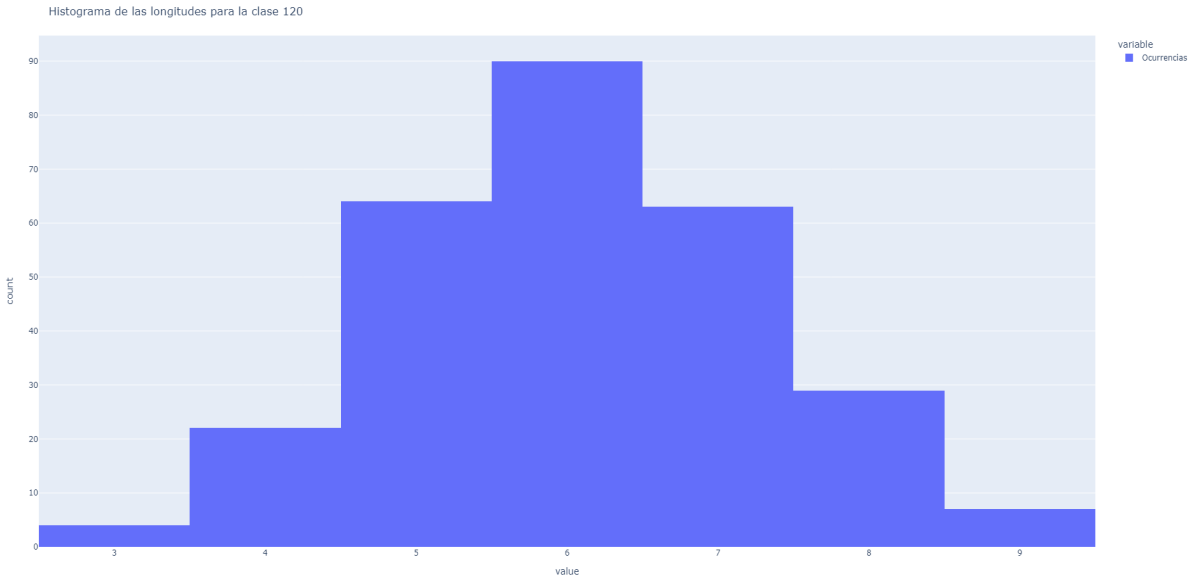

Figura A.36: Histograma para mostrar la longitud de las palabras por clase en el segundo conjunto. Clase 120

ANEXO

OBJETIVOS DE DESARROLLO SOSTENIBLE

Grado de relación del trabajo con los Objetivos de Desarrollo Sostenible (ODS).

Objetivos de Desarrollo Sostenibles	Alto	Medio	Bajo	No Procede
ODS 1. **Fin de la pobreza.**				X
ODS 2. **Hambre cero.**				X
ODS 3. **Salud y bienestar.**				X
ODS 4. **Educación de calidad.**			X	
ODS 5. **Igualdad de género.**				X
ODS 6. **Agua limpia y saneamiento.**				X
ODS 7. **Energía asequible y no contaminante.**				X
ODS 8. **Trabajo decente y crecimiento económico.**				X
ODS 9. **Industria, innovación e infraestructuras.**			X	
ODS 10. **Reducción de las desigualdades**.				X
ODS 11. **Ciudades y comunidades sostenibles**.				X
ODS 12. **Producción y consumo responsables.**				X
ODS 13. **Acción por el clima.**				X
ODS 14. **Vida submarina.**				X
ODS 15. **Vida de ecosistemas terrestres.**				X
ODS 16. **Paz, justicia e instituciones sólidas.**			X	
ODS 17. **Alianzas para lograr objetivos.**				X

Reflexión sobre la relación del TFG/TFM con los ODS y con el/los ODS más relacionados.

“Estimación del tamaño de las bounding box de palabras en texto manuscrito a partir de su transcripción” es el título del trabajo de fin de grado presentado. Está relacionado con los objetivos de desarrollo sostenible de forma que tiene una conexión tenue, es decir, mantiene una relación lejana pues es un trabajo centrado principalmente más en el área de un estudio o investigación muy cercana a la rama de computación que en un área práctica o como solución directa de los problemas planteados en los objetivos de desarrollo sostenible.

De todas formas, si hubiera que seleccionar algunos, unos pocos, de los objetivos de desarrollo sostenible los tres más cercanos serían “ODS 4 Educación de calidad”, “ODS 9 Industria, innovación e infraestructuras” y “ODS 16 Paz, justicia e instituciones sólidas” y por tanto serían las opciones más apropiadas.

Se puede ver la relación con el ODS 4 Educación de calidad pues estimar el tamaño de las bounding box tiene aplicaciones en la transcripción y recuperación de la información de documentos históricos manuscritos, que no dejan de ser cultura y pueden llegar a formar parte de los conocimientos necesarios en una educación de calidad.

El ODS 9 Industria, innovación e infraestructuras se relaciona pues el trabajo es, expuesto de forma simple, un estudio centrado entorno a la posibilidad de estimar el tamaño de las palabras a partir de su transcripción y el posible rendimiento de esta aproximación, para ver si es viable o no y que sirva de base para que otras aplicaciones en el área de la visión por computador o fuera de ella puedan usarlo. Teniendo en cuenta lo escrito en este trabajo, sabemos que la transcripción de documentos y búsqueda de palabras en los mismos son problemas difíciles que requieren de bastante esfuerzo y por tanto aportaciones que ayuden a aligerar la carga y mejorar los resultados son siempre bienvenidas.

El ODS 16 Paz, justicia e instituciones sólidas está relacionado dado que este trabajo tiene aplicaciones clave en recuperación de la información y reconocimiento de entidades nombradas en texto manuscrito y por tanto puede tener resultados importantes en la búsqueda de muchos de los documentos manuscritos que a día de hoy todavía tienen las administraciones de muchas instituciones a lo largo del mundo, haciendo algo más eficiente la gestión de éstos. Viendo cómo las administraciones tienen problemas a adaptarse a las nuevas tecnologías esto podría ser otro de los pequeños pasos necesarios antes de acabar de dar el gran salto. Además, el fácil acceso a una búsqueda en documentos manuscritos digitalizados podría permitir la resolución de algunos trámites que quedaron estancados en el pasado.

Con esto cabe concluir que aunque este trabajo de fin de grado no tiene una relación directa con los objetivos de desarrollo sostenible, pues está más centrado en la realización de un estudio y sus resultados, sus aplicaciones más cercanas, tanto pasadas como futuras sí tienen una relación aunque sea también de bajo o medio impacto. Es por esto que se concluye que el trabajo tiene relaciones de bajo impacto en los objetivos de desarrollo sostenible.