PROJECTE 2:

ETIQUETADOR

Intel·ligència Artificial

Blanca Piñol Chacon, 1666134
Maria Muñoz Cabestany, 1668438
Berta Martí Boncompte, 1703200
David Ruiz Cáceres, 1672891

1. INTRODUCCIÓ

L'objectiu d'aquesta pràctica consisteix en la implementació d'algorismes d'aprenentatge no supervisat(K-means) i supervisat(KNN), amb la finalitat de classificar peces de roba segons les seves característiques de forma i color.

1.1 K-means

L'algorisme K-means s'utilitza per agrupar les imatges en clústers basats en les seves característiques o atributs. En aquesta ocasió, l'utilitzarem per agrupar les imatges de peces de roba en clústers segons les característiques dels seus píxels. D'aquesta manera, al finalitzar el procés, les imatges es classificaran en clústers segons els seus colors predominants, on cada un representa un conjunt d'imatges amb característiques de color similars.

El funcionament és el següent:

1. ***Inicialització de centroids:*** Abans d'aplicar cap tipus de millora, l''algorisme comença amb una inicialització determinista dels centroids dels clústers, consistent en la selecció dels primers K punts com centròids inicials. Aquests centroids són els colors inicials al voltant dels quals es formaran els clústers.
2. ***Assignació de clúster:*** En aquest pas, cada píxel de les imatges s'assigna al clúster amb el centroid més proper. Ho aconseguim calculant la distància entre cada punt i els centroids i assignant el píxel al clúster amb el centroid més proper. És a dir, s'assigna al clúster de color més similar.
3. ***Actualització dels centroids:*** Després d'assignar cada píxel a un clúster, es recalculen els centroids com la mitjana dels colors dels píxels assignats a cada clúster. D'aquesta manera es mouen els centroids per tal d'ajustar-se millor als colors predominants de cada imatge.
4. ***Convergència:*** Aquest procés es repeteix iterativament fins que els centroids ja no canvien significativament o s'arriba a un màxim d'iteracions marcat.

Havent executat aquest procés, cada imatge estarà assignada a un clúster basat en els seus colors predominants, permetent això classificar les imatges de peces de roba segons els colors que contenen.

1.2 KNN

L'algorisme KNN, o K-Nearest Neighbours, s'utilitza per classificar elements segons les característiques dels seus veïns més propers. En aquesta ocasió, l'utilitzarem per etiquetar les imatges de les peces de vestir segons la seva forma. Classificarem cada imatge segons la forma predominant que comparteix amb els seus veïns més propers al conjunt d'entrenament.

El funcionament és el següent:

1. ***Inicialització de la classe i preparació de les dades d'entrenament:*** El constructor de la classe rep les dades d'entrenament (train_data) i les etiquetes corresponents(labels), que s'assignen a les variables de la classe per un ús posterior. Seguidament, les imatges es transformen en una forma adequada pel KNN, en una matriu on cada fila representa una observació i cada columna una característica.
2. ***Càlcul dels veïns més propers:*** Es calculen les distàncies entre les dades de prova i les d'entrenament. A continuació, selecciona els K veïns més propers per cada punt de prova i guarda les seves etiquetes.
3. ***Determinació de la classe:*** Es determina la classe per cada punt de prova a partir d'una votació majoritària entre les etiquetes dels veïns més propers.

2. MÈTODES D'ANÀLISI IMPLEMENTATS

2.1 FUNCIONS D'ANÀLISI QUALITATIU

2.1.1. retrieval_by_color:

Retorna el conjunt d'imatges que contenen els colors desitjats.

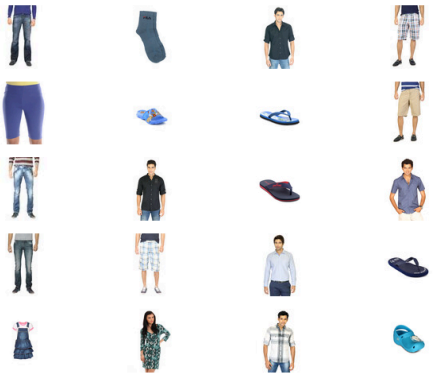
La funció rep tres paràmetres:

- llista d'imatges
- etiquetes del color obtingudes pel K-means (llista de llistes)
- llista que representa els colors que busquem

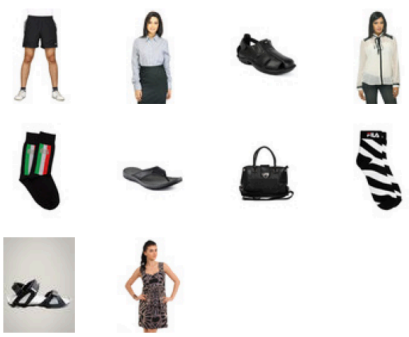
Amb aquesta informació es va iterant sobre les etiquetes i anem comprovant si coincideixen amb els colors buscats. Si hi coincideix ho afegim a una nova llista, que serà retornada per la funció.

RESULTATS:

```
color=retrieval_by_color(train_imgs,train_color_labels,["Blue", "White"])
visualize_retrieval(color, 20, info=None, ok=None, title='', query=None)
```



```
color=retrieval_by_color(train_imgs,train_color_labels,["Black"])
visualize_retrieval(color, 10, info=None, ok=None, title='', query=None)
```



CONCLUSIÓ:

Es pot veure com la funció agafa les imatge que continguin el color o els colors passats com a argument, sigui la quantitat que sigui, sense tenir en compte si hi ha també altres colors. Es podria arribar a millorar ordenant la llista retornada segons com de probable és que cadascun dels elements sigui correcte.

2.1.2. retrieval_by_shape:

Busca les imatges que continguin una determinada forma.

La funció rep tres paràmetres:

- llista d'imatges
- etiquetes de la forma obtingudes pel Knn (llista de llistes)
- string de la forma que busquem

Amb aquesta informació es va iterant sobre les etiquetes comprovant si coincideix amb la forma buscada. Si és així, s'afegeix a una llista nova. Al final, es retorna aquesta mateixa..

RESULTATS:

```
shape=retrieval _by_shape(train_imgs,train_class_labels,"Dresses")
visualize_retrieval(shape, 15, info=None, ok=None, title='', query=None)
```

```
shape=retrieval_by_shape(train_imgs,train_class_labels,"Flip Flops")
visualize_retrieval(shape, 7, info=None, ok=None, title='', query=None)
```

CONCLUSIÓ:

Aquí, en canvi, només retorna les imatge d'aquella forma en concret i no té l'inconvenient anterior. Això passa ja que les imatges només tenen una forma, i no poden ser una combinació de varies formes.

2.1.3.retrieval_combined:

Busca les imatges que continguin una determinada forma i uns determinats colors.
La funció rep cinc paràmetres:

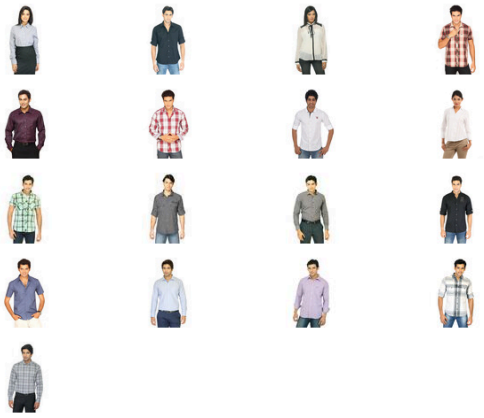
- llista d'imatges
- etiquetes del color obtingudes pel K-means (llista de llistes)
- etiquetes de la forma obtingudes pel Knn (llista de llistes)
- llista que representa els colors que busquem
- string de la forma que busquem

Amb aquest informació itera sobre l'índex de cada imatge, agafa les etiquetes de color i forma de cada una i comprova si coincideix amb les passades com a paràmetres. Si les dues coincideixen , ho afegim a una nova llista, la qual serà retornada al final.

RESULTATS:

```
shape_color=retrieval_combined(train_imgs,train_color_labels,train_class_labels,["Brown", "Yellow"],"Heels")
visualize_retrieval(shape_color, 10, info=None, ok=None, title='', query=None)
```

```
shape_color=retrieval_combined(train_imgs,train_color_labels,train_class_labels,["Orange", "White"],"Shirts")
visualize_retrieval(shape_color, 17, info=None, ok=None, title='', query=None)
```



CONCLUSIÓ:

Combinant la búsqueda del color i de la forma, el problema segueix estant en que el color o els colors que li demanem només han d'estar inclòs a la imatge, sense importar si són els únics. Es podria arribar a millorar de la mateixa manera que al retrieval _by_color.

2.2 FUNCIONS D'ANÀLISI QUANTITATIU

2.2.1. kmeans_statistics:

La funció rep 3 paràmetres:

- classe kmeans
- kmax
- conjunt d'imatges

Bucle des de k=2 fins a k=kmax on a cada iteració es crida a la funció fit, es calcula la heurística, el nombre d'iteracions i el temps que ha necessitat per convergir.

RESULTATS:

La imatge que hem utilitzat com exemple per cridar al kmeans_statistic ha estat la següent:

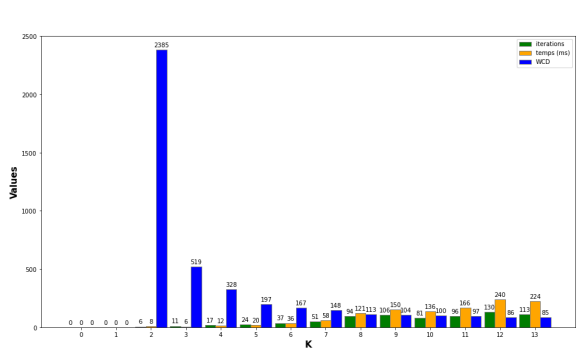
D'aquesta imatge hem calculat el valor de WCD, InterClass i Coeficient de Fisher per a tres inicialitzacions diferents dels centroides: "first", "random" i "Kmeans ++" entre K=2 i K=13.

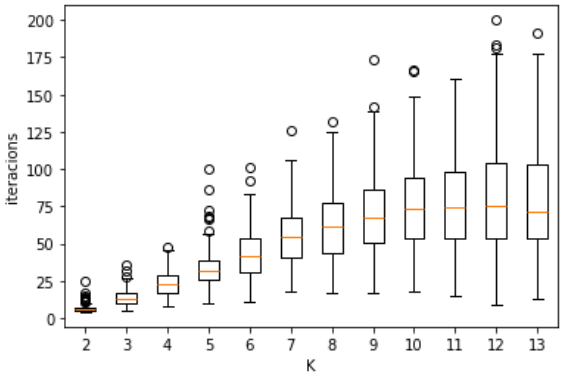
Per exemple, els resultats utilitzant FITTING = WCD amb les diferents inicialitzacions han estat els següents:

WITHIN CLASS DISTANCE:

Inicialització *'first'*:

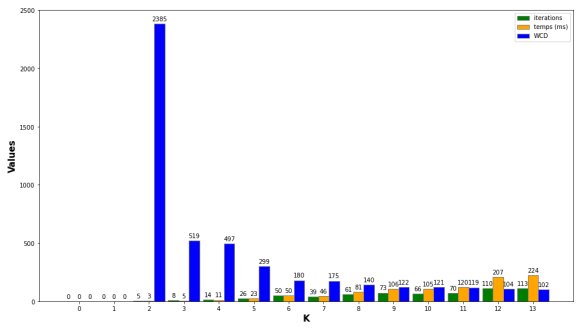
```
options = {
    'km_init': 'first' , 'fitting': 'WCD'
}
kmeans = KMeans(train_imgs[0], K=2, options=options)
Kmax = 13
kmean_statistics(kmeans, Kmax, train_imgs)
```

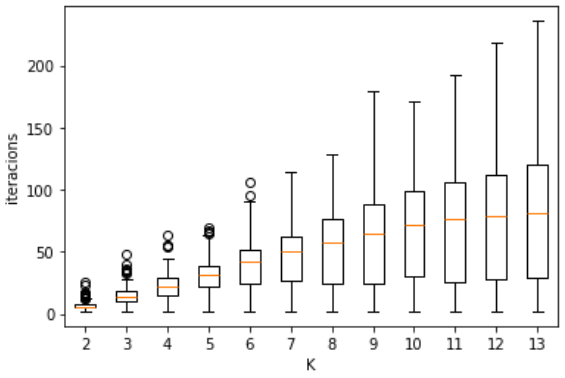




Inicialització *'random'*:

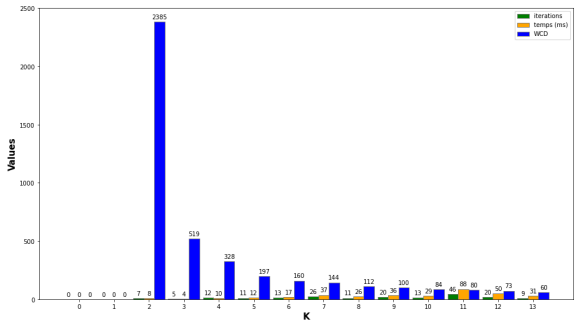
```
options = {
    'km_init': 'random' , 'fitting': 'WCD'
}
kmeans = KMeans(train_imgs[0], K=2, options=options)
Kmax = 13
kmean_statistics(kmeans, Kmax, train_imgs)
```

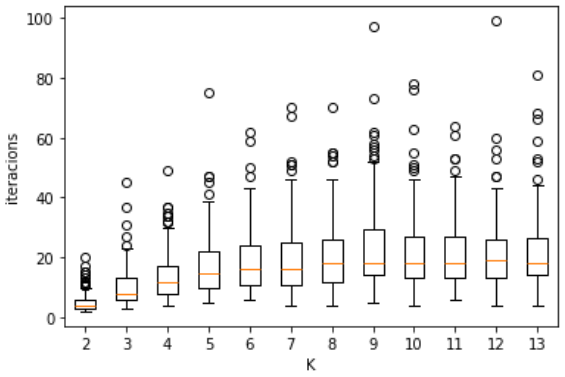




Inicialització *'kmeans++'*:

```
options = {
    'km_init': 'kmeans++' , 'fitting': 'WCD'
}
kmeans = KMeans(train_imgs[0], K=2, options=options)
Kmax = 13
kmean_statistics(kmeans, Kmax, train_imgs)
```



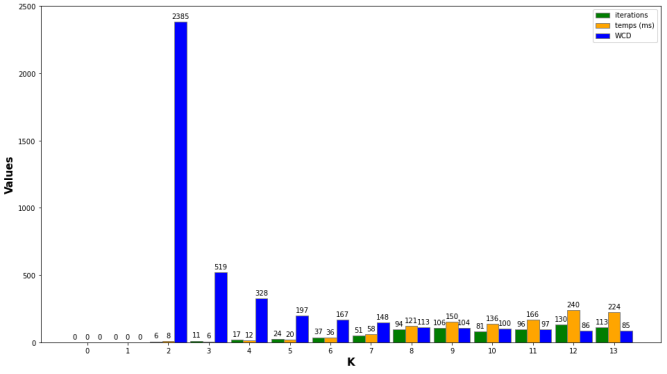


Podem observar que per molt que canviem la inicialització la tendència de l'heurística és la mateixa. Els boxplot són independents a l'heurística, només ens serveixen per la inicialització. Per tant, per molt que el gràfic també es pugui servir per estudiar les inicialitzacions ja que conté informació de les iteracions, ens resultarà més útil fer servir els gràfics per l'anàlisi d'heurístiques i els boxplot per l'anàlisi de les inicialitzacions.

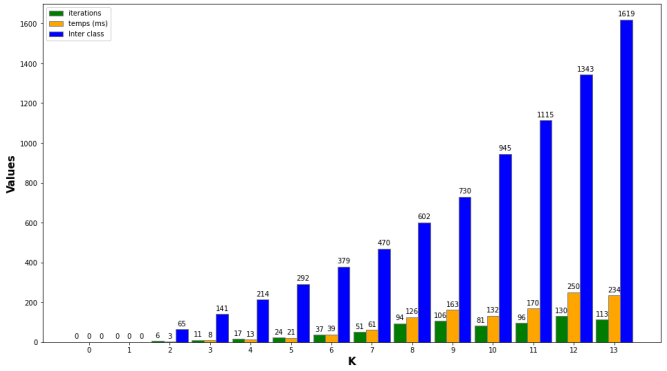
ANÀLISI DE RESULTATS SEGONS HEURÍSTIQUES

Per tal de determinar la K òptima per classificar aquest dataset hem avaluat el valor, les iteracions i el temps d'execució del K-means per a valors de K de 2 a 13. Ho hem realitzat utilitzant els tres mètodes heurístics implementats diferents: WCD (Within Class Distance), ICD (Inter Class Distance) i FD (Fisher Distance).

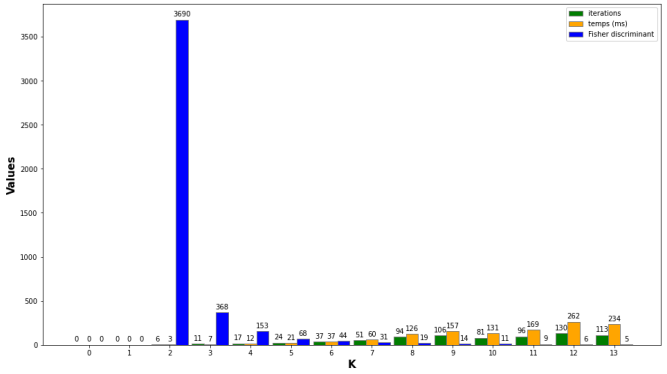
Per millorar la visualització dels resultats, hem realitzat una adaptació dels valors referents a les heurístiques per aconseguir una millor comprensió de la gràfica. Per a la mètrica del Inter Class Distance (ICD), els valors els hem dividit per cent, mentre que per a la mètrica de Fisher Discriminant (FD) els hem multiplicat per mil. D'aquesta manera solucionem el fet que aquests valors no s'adaptessin als valors de la resta d'elements. Amb aquesta adaptació, els resultats són més comprensibles i fàcils de comparar:



Within class distance(WCD)



Inter class distance(ICD)



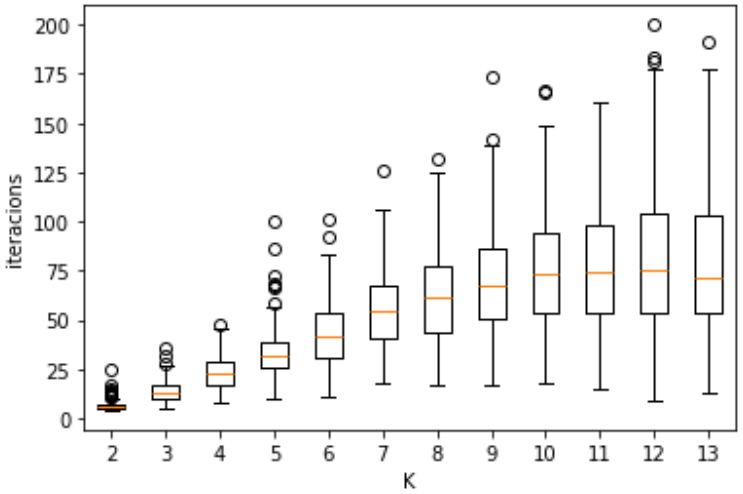
Fisher Discriminant (FD)

Podem observar com la distància intra-class i el discriminant de fisher disminueixen molt ràpidament, en canvi, la distància inter-class augmenta. Veiem que tant en el cas del WCD com en el de FD la millor K està al voltant del 3, ja que a partir d'aquí la disminució dels valors de l'heurística comença a ser menys significativa. En canvi, en el cas del IC, la disminució en la distància comença a ser menys significativa al voltant de K = 7.

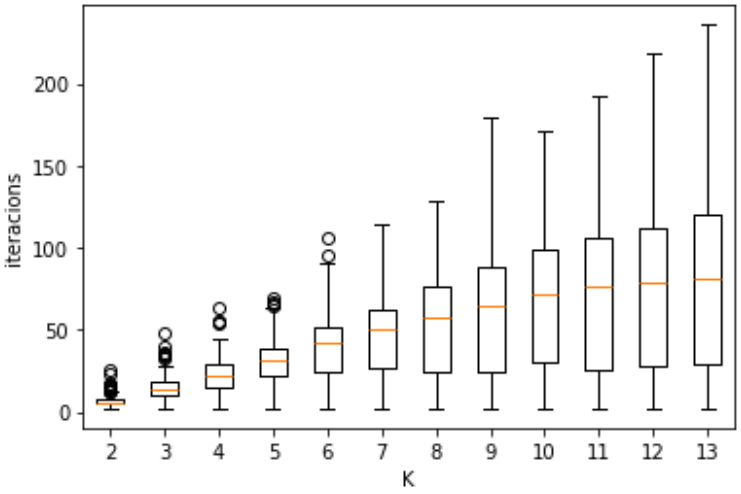
Amb el gràfic també podem concloure que el discriminant de Fisher és el més eficient a l'hora de decidir el valor de K, ja que la diferència és més dràstica. Això es deu a què combina tant el WCD com el ICD, que són dos paràmetres importants per agrupar clústers. És a dir, té en compte tant la distància intra-class(WCD) com la distància inter-class(ICD).

ANÀLISI DE RESULTATS SEGONS INICIALITZACIÓ

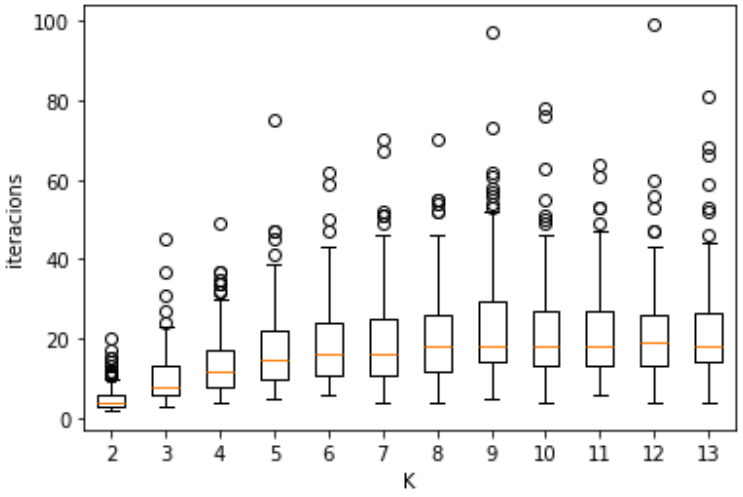
Per tal de decidir si les noves propostes d'inicialització suposen una millora hem estudiat les iteracions del Kmeans per a cada K començant per K=2 i fins a K=13. Per ajudar-nos, ens podem fixar en els gràfics utilitzats d'exemple amb el WCD, que parteixen de les diferents inicialitzacions i reflexen també el nombre d'iteracions, però principalment estudiarem els boxplot següents:



Boxplot per la inicialització 'first'



Boxplot per la inicialització 'random'



Boxplot per la inicialització 'kmeans++'

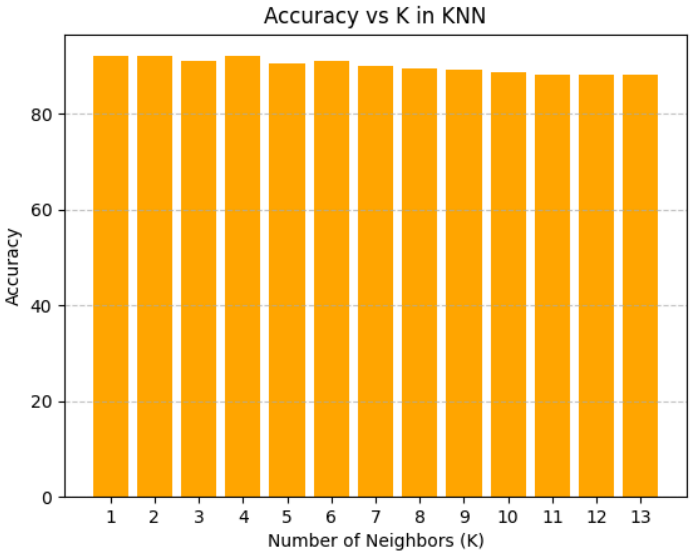
Com es pot observar en el primer i segon plot, la mitjana d'iteracions va augmentant conforme aumenta el nombre de clusters que es fan servir a la hora de fer la predicció. Tot i això, de mitja, la inicialització random fa servir més iteracions que la inicialització 'first'. En cap dels dos s'estabilitzen les iteracions, almenys per els nombres de klusters que hem provat.

A diferència de les altres dues, en el tercer plot veiem com la mitjana del nombre d'iteracions s'estabilitza per sota de les 20 iteracions. Cal esmentar que els punts per sota i per sobre de les rectes son els outliers, que corresponen als valors més extrems del nostre test.

Vists aquests tres mètodes podem dir sense dubte que la inicialització *Kmeans*++ és la millor de totes, ja que augmentar el nombre de klusters no augmenta la mitjana del nombre d'iteracions. A més, es pot observar que en el cas del kmeans++ té una variabilitat molt menor (fixant-nos en la longitud de la barra vertical) ja que el mateix percentatge de valors estan continguts en una franja molt més petita que en els altres dos casos.

2.2.2. get_shape_accuracy:

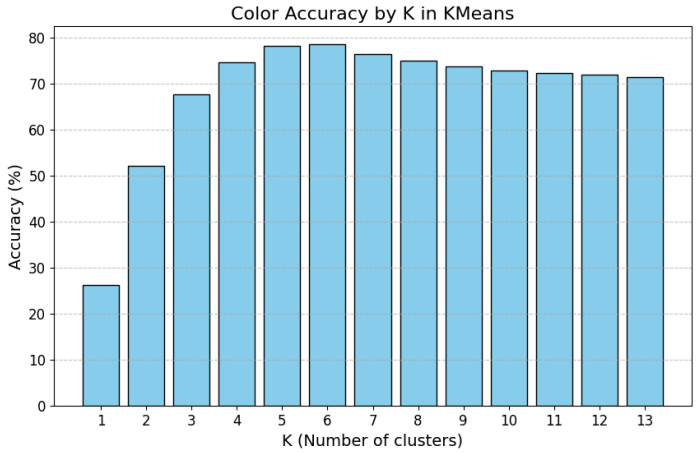
Compara les etiquetes del KNN amb les del Ground-Truth segons la seva forma



Es pot observar com, amb el KNN i el test fet per observar les accuracies obtingudes, les accuracies no tenen una gran variació en funció de la K (almenys fins que arriba a la K). Tot i això, és apreciable com la K = 4 és per poc la millor per al KNN.

2.2.3. get_color_accuracy:

Compara les etiquetes del Kmeans amb les del Ground-Truth segons el seu color



A mesura que es tenen en compte més clusters en el càlcul de l'exactitud segons el color, s'espera una millora en la precisió, ja que s'incorporen una gamma més àmplia de colors (centroides), facilitant una identificació més precisa de la imatge. El punt òptim al gràfic es produeix quan s'utilitzen 6 clústers, indicant que la identificació d'imatges amb 6 centroides o colors resulta en una representació visual més clara. No obstant això, a partir d'aquest punt, la precisió tendeix a disminuir, ja que un excés de clústers pot portar a un sobreajust, el que a la vegada pot afectar negativament la precisió del model.

3. MILLORES ALS MÈTODES DE CLASSIFICACIÓ

3.1 INICIALITZACIONS DE KMEANS

La funció init_centroids inicialitza els centroids segons l'opció triada. Inicialment només teníem en consideració l'opció first, però hem afegit les inicialitzacions random i kmeans++.

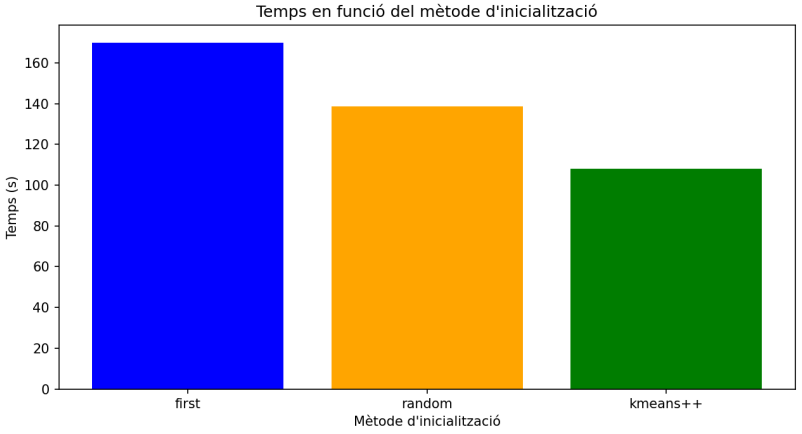
- **RANDOM**: els centroides s'inicialitzen escollint aleatòriament K punts.
- **KMEANS++:** selecciona el primer centroid de manera aleatòria del conjunt de dades i després selecciona els següents centroids de manera intel·ligent.

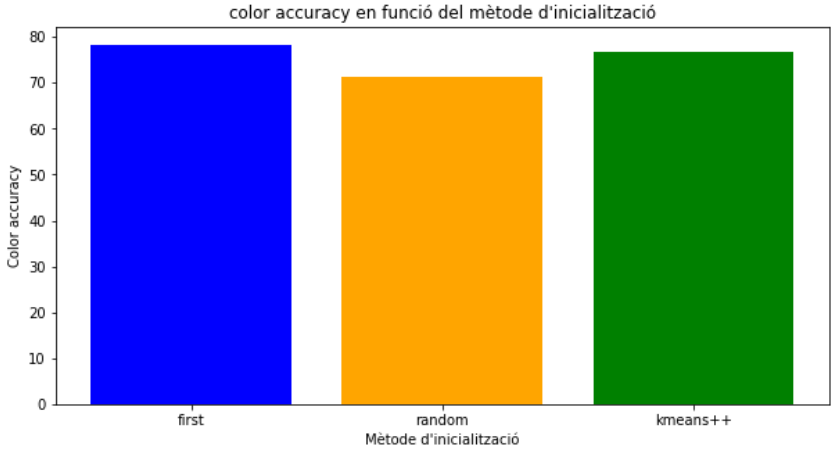
  La idea és assegurar-nos que els centroids estiguin inicialitzats de manera que estiguin el més separats possible entre ells. Per fer això, el primer centroide es selecciona aleatòriament de les dades. Després, en cada iteració, el següent

centroid es selecciona de manera que la probabilitat de triar aquell punt com centroid sigui proporcional a la distància al quadrat des del punt fins al centroid més proper actualment seleccionat. Això significa que els punts que estan més lluny dels centroids que s'estan considerant en aquella iteració determinada tenen més probabilitat de ser seleccionats com nous centroids.

RESULTATS:

A continuació, farem un anàlisi comparatiu de les inicialitzacions de centroids per determinar quina ofereix millors resultats. Avaluarem quin és més eficient en quant a rapidesa i quin és més exacte fent el test de color accuracy explicat a l'apartat anterior.





CONCLUSIÓ:

Després de comparar els temps d'execució i l'accuracy del color, podem observar que l'algorisme 'kmeans++' és el més ràpid i ofereix una precisió similar a l'opció 'first'. Pel que fa a l'opció 'random', hem observat que en les diferents proves el seu temps d'execució i el color accuracy varien. Aquest fet comporta que, pel que fa aquest aspecte, en algunes

ocasions pugui ser més eficient que el ‘kmeans++’. Tanmateix, no obté mai accuracies tan altes, no és tan precís. Per tant, podem concloure que la millor inicialització de centroids és utilitzant l'algorisme ‘kmeans++’.

3.2 DIFERENTS HEURÍSTIQUES PER BESTK

Hem creat dues noves funcions al kmeans per utilitzar diferents heurístiques a part de la ja implementada anteriorment, la intra-class.

- `intra_class:` distància entre punts d’un mateix clúster. És bo que sigui petita

Funcions noves:

- `inter_class:` distància entre els diferents centroides. És bo que sigui gran.

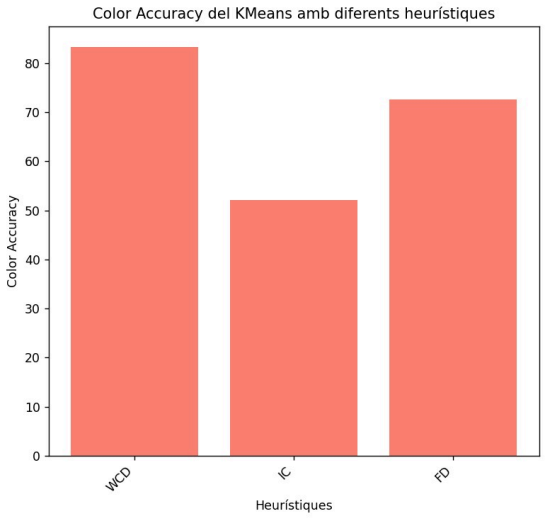
$$D(C_1, C_2) = \frac{1}{m_1 m_2} \sum_{j=1}^{m_1} \sum_{i=1}^{m_2} d(\vec{x}^i, \vec{x}^j) : \vec{x}^i \in C_1 : i : 1 \dots m_1, \vec{x}^j \in C_2 : j : 1 \dots m_2$$

- `fisher_disc:` Quocient entre la distància intra-class i la inter-class. És bo que sigui petit, ja que maximitza la distància inter-class i minimitza la distància intra-class.

$$\frac{d\_intra \cdot class}{d\_inter \cdot class} \rightarrow discriminant_k$$

RESULTATS:

Amb aquestes noves distàncies hem volgut comprovar amb quina s'aconsegueix el millor accuracy. Ho hem fet calculant-ho amb k=5.



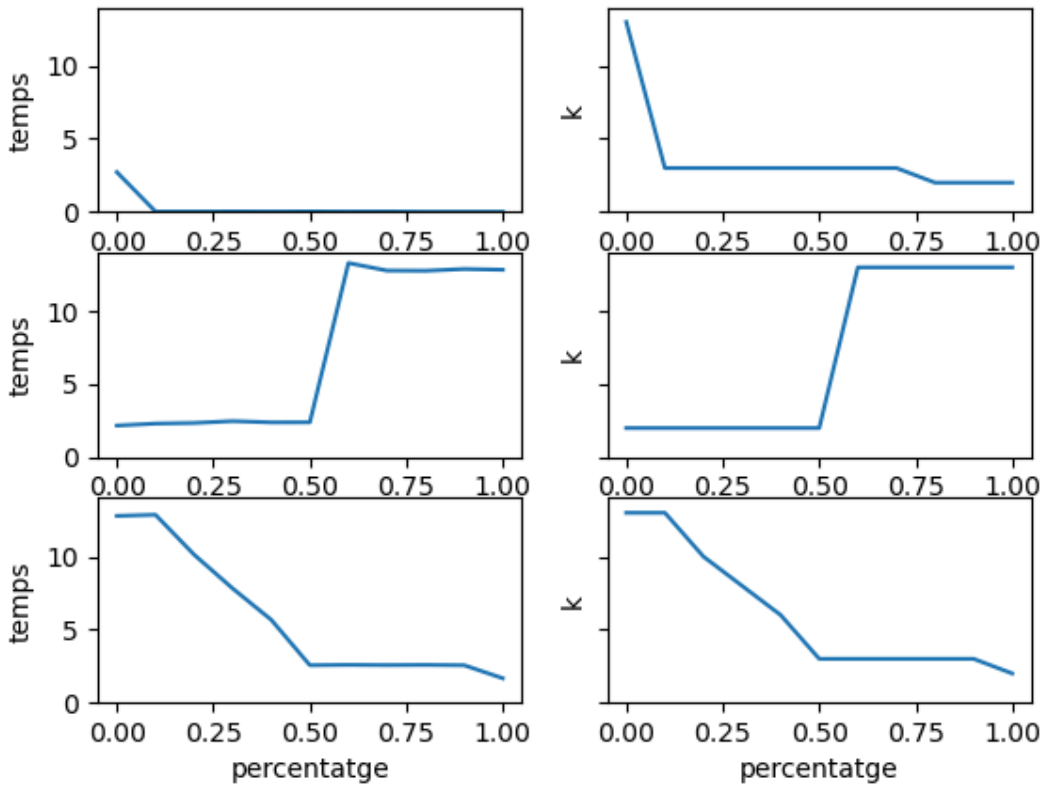
CONCLUSIÓ:

El gràfic revela que distància Intra-class (WCD) té una accuracy més alta i per tant influeix de manera més eficient en la determinació del valor de K. Això significa que els punts dintre de al classe estan més aprop uns dels altres en comparació amb altres punts del conjunt.
Sense necessitat de revisar la part relacionada amb el discriminant de Fisher, es pot inferir que aquesta heurística no és la més adequada en aquest cas específic. Això es deu al fet que la distància intra-classe és major que la distància inter-classe, resultant en un quocient no petit.
A l'apartat següent, les funcions es criden per trobar la bestK.

3.3 FIND_BESTK

S'encarrega d'avaluar amb cada heurística com evoluciona el temps d'execució i la K segons el percentatge preestablert, que es fa servir com a llindar a partir del qual es decideix si seguir o no. Aquest llindar va avançant segons el paràmetre factor.

RESULTATS:



CONCLUSIÓ:
El resultat del plot ens dona un conjunt de 6 subplots. Cada fila ens permet veure per tres heurístiques diferents (WCD, IC i FD respectivament) el temps d'execució i el valor de la K que es fa servir en funció del llindar prèviament esmentat.

En totes tres files es pot observar com hi ha una relació entre el comportament del temps i de la k segons el llindar. Amb la WCD i el FD, quan major és el llindar, menor el temps d'execució i la K, mentre que per la IC es pot observar el contrari.

Llavors podem concloure que per les heurístiques WCD i FD ens beneficiarem d'augmentar el llindar, mentre que per la IC ho farem en disminuir-ho, ja que així optimitzaríem el codi pel que fa a la complexitat en temps.

3.4 FEATURES FOR KNN

Per tal de comprovar quin tipus de distància és el més adient per calcular els k veïns més propers, hem fet una modificació en la funció get_k_neighbours de manera que pugui calcular-los usant diferents distàncies. D'aquesta manera, podem veure si modificant l'elecció dels veïns propers obtenim una millor shape accuracy en aplicar el knn.

A l'hora de l'elecció de les distàncies , hem tingut en compte aquelles implementades en la funció cdist de la llibreria Scipy i ens hem quedat amb algunes de les més usades (d'entre les que poguessin ser aplicades en aquest problema) i aquelles que ens han donat uns resultats més significatius (millorant o empitjorant clarament els resultats obtinguts amb la distància euclidiana amb la qual ho havíem calculat inicialment).
Distàncies contemplades:

- Manhattan: suma de les distàncies horitzontals i verticals entre els dos punts. $\sum_{i=1}^{n} |x_i - y_i|$
- Chebyshev: distància entre dos punts com la màxima diferència entre les seves coordenades.
  $max_i\,(|p_i - q_i|)$
- Cosine: producte de punts dels vectors dividit per la norma dels vectors de manera que distància 1 indica que els vectors són idèntics en direcció, mentre que 0 indica que són ortogonals

- Minkowski: generalització de les distàncies de Manhattan i Euclidiana utilitzant un paràmetre de potència, que determina la forma de la distància on quan el paràmetre de potència és 1 és la distància de Manhattan mentre que quan és 2 és l'euclidiana

  $$\sum_{i=1}^{n} (|x_i - y_i|^p)^{1/p}$$

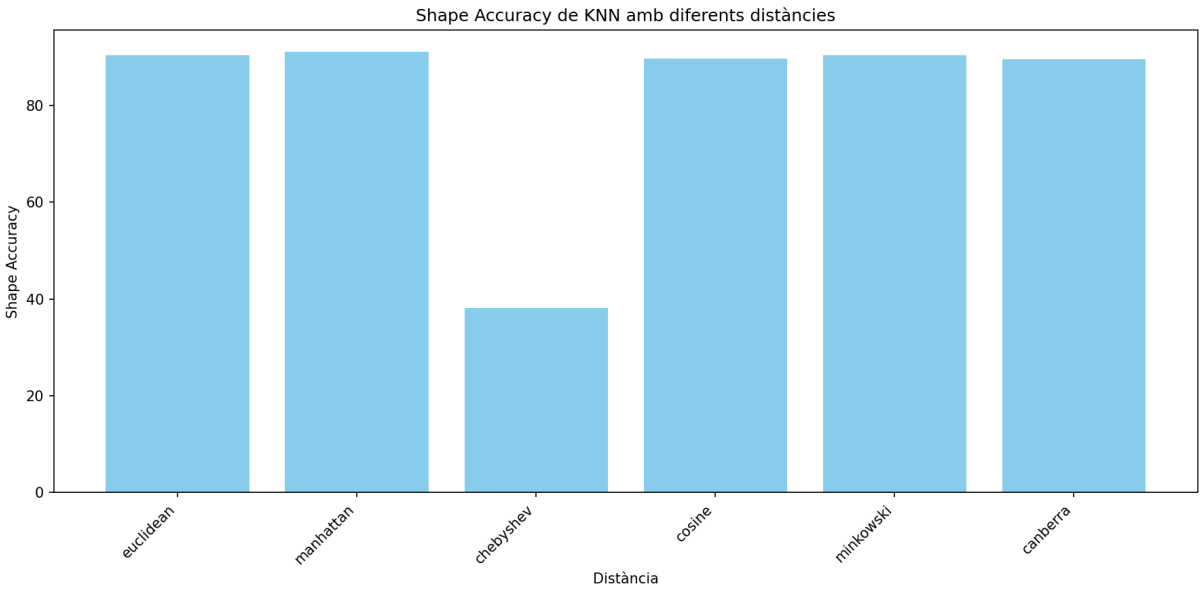
- Canberra: versió ponderada de la distància de manhattan útil quan les variables tenen valors similars

  $$\sum_{k=1}^{n} \frac{|x_{ik} - x_{jk}|}{|x_{ik}| + |x_{jk}|}$$

Hem descartat usar altres distàncies d'aquesta funció ja que no tenien aplicabilitat en el problema (com seria per exemple la distància de hamming).

RESULTATS:

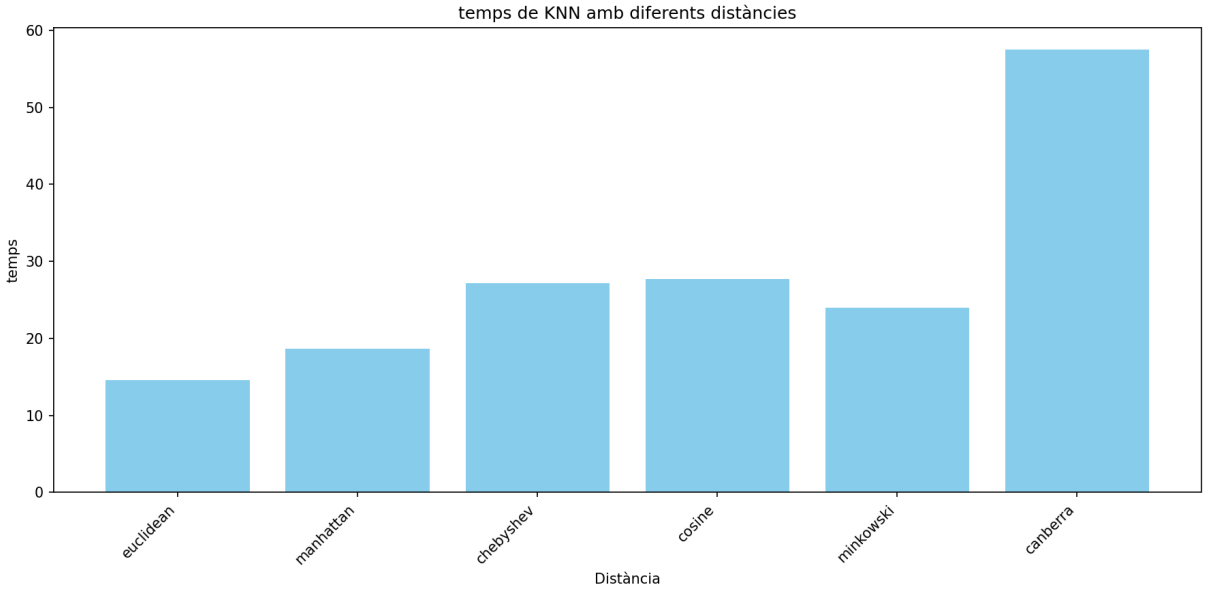
Per poder valorar quina és la millor opció, hem comprovat el shape accuracy (usant la funció anteriorment esmentada que ens fa un anàlisis quantitatiu de les proves) per diferents tipus de distància. En la següent gràfica podem veure una comparativa amb els diferents resultats obtinguts.



Així podem veure com usar la distància de manhattan o la de minkowski per trobar els k veïns més propers ens dona una millora del resultat per la shape accuracy respecte la distància euclidiana usada originalment. El fet que aquestes dues obtinguin la mateixa accuracy és coherent ja que aquesta última és una generalització de la primera. No obstant,

la millora és molt lleu respecte la distància euclidiana, per tant, aquesta no és una gran millora en termes de precisió.

Per aquest motiu, hem decidit també fer un anàlisis del temps d'execució per determinar si hi ha hagut una millora en aquest àmbit en les distàncies esmentades.



CONCLUSIÓ:

Podem observar com tot i que manhattan i minkowski ofereixen una mica més de precisió que a l'usar distància euclidiana, aquestes triguen més temps a resoldre el problema i per tant pot ser més interessant usar la distància euclidiana si el conjunt de dades és molt gran. També podem observar com canberra ens donava un accuracy similar a la distància euclidiana però és significativament més lent i per tant no és convenient usar-ho.

4. CONCLUSIÓ DEL TREBALL

4.1 KMeans

Després de totes les millores aplicades al KMeans i els tests que hem fet per tal de visualitzar i analitzar aquestes, hem arribat a les següents conclusions:

1. En el cas del *retrieval_by_color*, es podria millorar el sistema per garantir que les imatges recuperades continguin la major quantitat possible del color sol·licitat, de manera similar es podria fer en el *retrieval_combined*
2. L'aplicació de la funció *kmeans_statistics* ens porta a la conclusió que, pel que fa a heurístiques, la del **Discriminant de Fisher** és la més eficient a l'hora de decidir el valor de K. En el cas de les inicialitzacions, el **k-means++** és la millor opció pel que fa a iteracions.

3. Entre les tres formes d'inicialització que hem implementat, el mètode **k-means++** es presenta com la millor opció possible. Encara que pugui mostrar una precisió lleugerament inferior en comparació amb el mètode *first*, aquesta discrepància és tan insignificant que el temps estalviat en el procés de convergència compensa àmpliament qualsevol diferència en precisió.
4. També hem fet servir tres heurístiques diferents (Within Class Distance, Inter-Class Distance i Discriminant de Fisher). Després de fer plots de gràfics de barres per tal de comparar-les, vam poder afirmar que la distància Intra-class (WCD) ens proporciona una major accuracy per les prediccions dels colors.
5. Utilitzant les tres heurístiques mencionades i ajustant el llindar corresponent, hem determinat que la millor K per al KMeans amb el dataset proporcionat és 6, donant com a resultat una precisió aproximada del 80%.

4.2 KNN

Després de totes les millores aplicades al KNN i els tests que hem fet per tal de visualitzar i analitzar aquestes, hem arribat a les següents conclusions:

1. Al retrieval_by_shape les imatges retornades coincideixen notablement amb la forma sol·licitada.
2. Hem aplicat diferents distàncies per calcular els veïns més propers i hem fet un anàlisis dels resultats respecte l'accuracy i el temps on hem obtingut que pel primer és millor usar manhattan (o equivalentment minkowsky) mentre que pel segon és millor l'euclidiana. Generalment usarem la distància euclidiana ja que és la que ens ofereix millors resultats tenint en compte ambdós criteris.
3. Amb el mateix test que l'apartat anterior, hem pogut veure que els pitjors valors per les accuracies de la shape s'obtenen fent servir la distància de chebyshev mentre que el knn més lent és l'implementat amb distància canberra.
4. Per l'accuracy del shape, la millor K segons els tests que hem fet és el valor de k=4, resultant en un 90% aproximadament d'encerts.