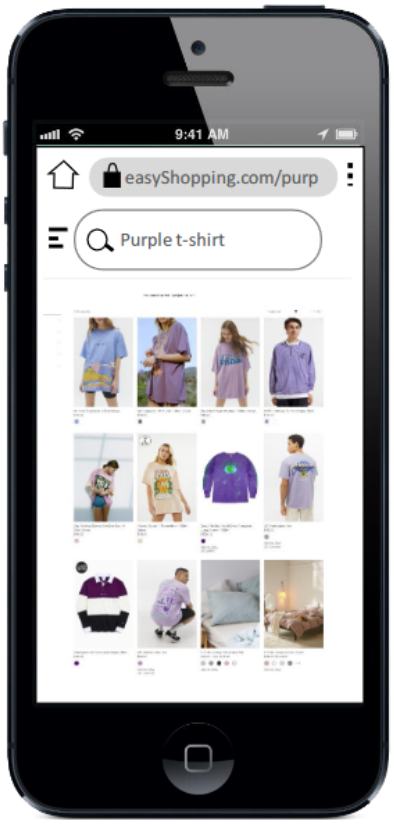


PROJECTE 2:

Etiquetador

Intel·ligència Artificial

Universitat Autònoma de Barcelona

Javier Comes Tello - NIU: 1631153
Lucas Dalmau Garcés - NIU: 1636290
Cristina Soler Bigorra - NIU: 1636589

INTRODUCCIÓ A LA PRACTICA

Sens planteja resoldre un problema simple d'etiquetatge d'imatges. Donat un conjunt d'imatges d'un catàleg de roba desenvoluparíem els algorismes que permetessin aprendre a etiquetar automàticament imatges per tipus de peça i per color. Com que el problema podia ser molt complex es va limitar a paraules en angles, 8 tipus de roba i 11 colors basics a més de treballar amb imatges de baixa resolució.

Es plantegen 3 problemes a resoldre:

- **Etiquetatge automàtic de color (no supervisat)**

  Es va decidir realitzar un agrupament no supervisat de punts utilitzant l'algorisme *K-means,* on es van implementar 10 funcions en les quals destacaven: fit(), withinClassDistance(), find_best_K(max_K), i get_colors(centroids).

- **Etiquetatge automàtic de forma (supervisat)**

  Per a organitzar el tipus de roba, es va decidir utilitzar l'algorisme *KNN,* on es van implementar 3 funcions per a agrupar-les en classes.

- **Implementació del cercador**

  Una vegada els mètodes son implementats es combinarien per a crear diferents test d'anàlisis i arribar a tenir una interfície similar a la d'una botiga online.

A continuació es parla amb més detall de cada una d'aquests problemes.

METODES D'ANALISI IMPLEMENTATS

METODES D'ANALISIS QUALITATIUS

Els mètodes d'anàlisis qualitatius son funcions que ens permeten avaluar d'una manera visual els nostres classificadors. Sens va donar tres funcions a fer de les quals es va decidir fer les tres: **Retrieval_by_color**, **Retrieval_by_shape** i **Retrival_combined**. Aquestes funciones retornaran les imatges que contenen les etiquetes d'una pregunta que es passa per paràmetre.

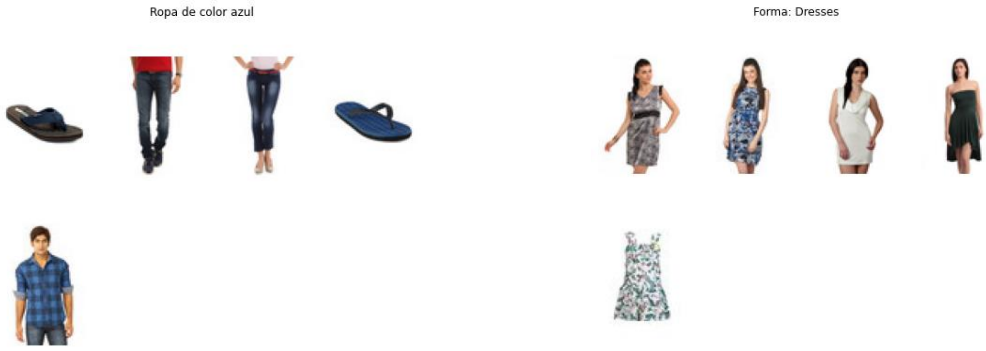
En la primera classe de practiques de la tercera part es va comentar que si es tenia una funció es tenien totes. Amb aquesta primera idea, es va començar a fer un esborrany de la primera funció: *'Retrieval_by_color'*. No va costar arribar a una possible solució.

Es va fer la resta de funcions trobant una diferencia en cada una de les funcions:

- **Retrieval_by_color:** Es va posar: 'if pregunta in clase:', perquè si s'hagués posat un == , al tractar-se d'una cadena, aquesta hauria fallat.
- **Retrieval_by_shape:** Al contrari que la funció anterior, en aquesta era necessària un == al tractar-se d'una única etiqueta.
- **Retrieval_combined:** Com el seu nom indica, es va combinar les dos funcions en una sola, amb dos condicions *(forma i color)* dins un bucle for.

Les funcions no van suposar un problema.

Per comprovar que funcionés, un membre del grup va decidir fer els seus propis tests per a cada una de les tres funcions. Per fer-ho, es va informar de les funcions a utils_data *(visualize_retrieval),* llegit atentament el pdf de presentació de la part i va tomar inspiració dels tests proporcionats pels professors *(TestCases_kmeans y TestCases_knn).* Agafant test_color_labels per Retrival_by_color y test_class_labels per Retrieval_by_shape, el test va donar un resultat correcte en base al que se li demana.



Imatges que retorna el test color y shape amb les etiquetes predeterminades

El problema va ser amb Retrieval_Combined el qual no mostrava cap imatge i imprimia aquest missatge a la terminal.

```
In [4]: runfile('C:/Users/jacot/Downloads/Etiquetador(1)/Etiquetador/my_labeling.py',
wdir='C:/Users/jacot/Downloads/Etiquetador(1)/Etiquetador')
Reloaded modules: utils_data, utils, Kmeans
<Figure size 576x432 with 0 Axes>
<Figure size 576x432 with 0 Axes>
<Figure size 576x432 with 0 Axes>
```

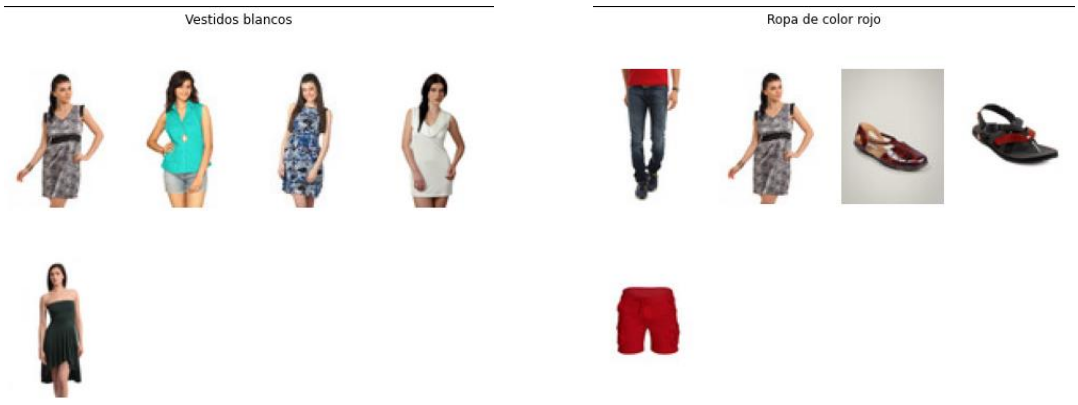
El problema es trobava en la con

dició del color, que només retornava les imatges correctes si es posava amb in, al contrari que la seva part per separat.

Imatge que retorna el test combined amb les etiquetes predeterminades

En un principi, els resultats donen correctament amb la primera implementació, però més tard es va adonar que no s'estava utilitzant les etiquetes implementades pel nostre Kmeans i Knn, i només s'estava comprovant que la funció fos correcta. El membre del grup que va fer els test va crear una funció (my_labels) on, en un bucle iterava les diferents imatges del conjunt, calculant les etiquetes de color i en la mateixa funció, calcular les etiquetes de forma però fora del bucle ja que ja havia un bucle implementat en la mateixa funció del Knn.

El resultat va donar imatges diferents però no mostrava signes de fallar o de donar un resultat inesperat.



Imatges que retorna el test color amb la funció pròpia.

Observació: *Existeix la possibilitat de que el color no sigui l'indicat ja sigui per un píxel que el programa detecta com del color que es busca o que es detecta el color de la peça de roba externa que no ha sigut eliminada a l'hora de fer la retallada.*

METODES D'ANALISIS QUANTITATIUS

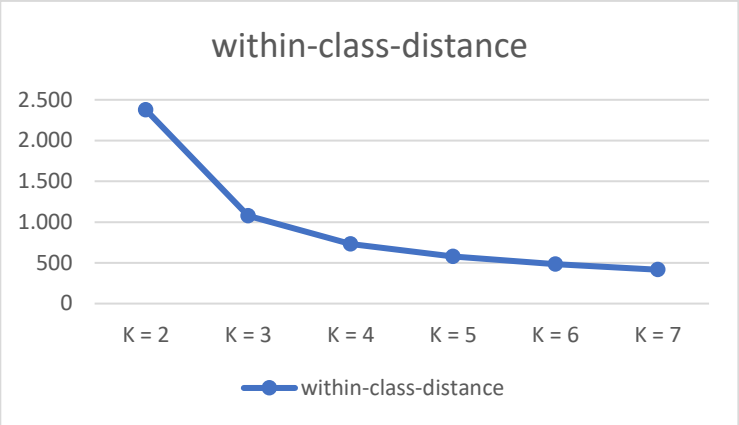
Els mètodes quantitatius s'encarreguen d'observar y treure resultats per a poder qualificar el treball fet per els algorismes implantats a la practica. Per a aquesta part sens ofereix realitzar 3 mètodes: Kmean_statistics, Get_shape_accuracy, Get_color_accuracy.

De moment tenim implementada la funció **Kmean_statistics**, amb la qual hem pogut observar la distancia entre classes (WCD) així com el temps necessari per a convergir per a cada valor de K que ve des de K = 2 fins a Kmax, el qual hem escollit que sigui K = 7. (6 nombres de 'K' diferents)

Els resultats els dividirem en dues parts, primer analitzarem la **distancia intra-class**.

WithinClassDistance: [2377.4794200537094, 1075.0367413718914, 729.7420840406991, 577.6647183568346, 484.0979100647066, 415.5614643165185]

Com podem observar al gràfic, la WCD cau dràsticament quan el numero de classes 'K' va pujant, això es a causa de que contra mes opcions de classificació tenim, els punts es van dispersant mes entre les diferents classes y que, per tant, les distancies entre classes cada vegada siguin mes curtes.



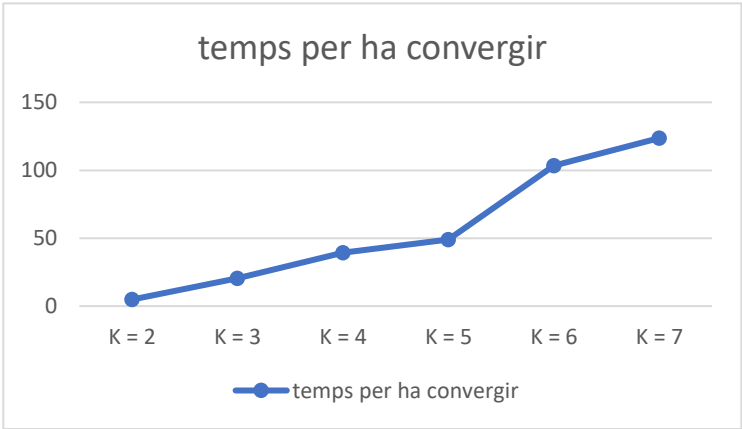
D'altra banda, tenim l'anàlisi del **temps necessari per convergir** en funció del nombre de classes:

Time: [4.844397783279419, 20.479047298431396, 39.41282272338867, 48.98620820045471, 103.39242196083069, 123.68818950653076]

Amb aquesta gràfica podem observar com a mesura que augmentem el numero de classes, el temps per a l'execució del k-means va augmentant també. Per entendre millor perquè passa això podem observar la formula de la complexitat del algorisme el qual es:

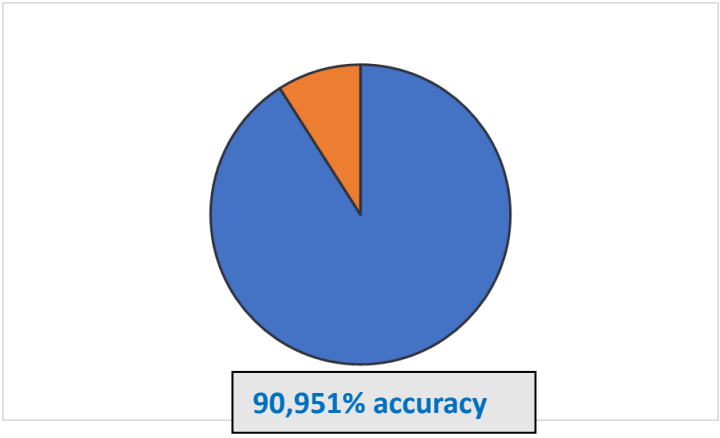
Complexitat del k-means: $O(t \times n \times k)$

t: *núm d'iteracions* (*)
n: *núm de punts*
k: *núm de classes*



Com el que variem es el nombre de classes, inevitablement això farà que necessitem mes iteracions per arribar a la convergència, per tant, cada vegada que incrementem K, estem augmentat dues de les tres variables que conformen la complexitat, deixant com a resultat un increment substancial del temps.

El segon mètode quantitatiu implementat ha estat El **get_shape_acuracy**, aquest ens indica a partir del conjunt de prova, totes les etiquetes de forma que ha estat capaç l' algoritme de classificar correctament. Com podem observar al gràfic, el percentatge d'encerts ha estat d'un 90,95%. És a dir, 9 de cada 10 etiquetes han estat assignades correctament.



MILLORES SOBRE EL KMEANS I KNN

S'han implementat varies millores sobre el Kmeans i el KNN.

KMEANS:

Primerament, en el Kmeans, en la funció *init_centroids* es pot inicialitzar de tres formes:

- Amb el mètode '**first'** que inicialitza els centroides assignant-los els primers K punts de la imatge.
- Amb el mètode '**random'** fa el mateix però amb punts aleatoris.
- Amb el mètode '**custom'** permet a l'usuari escollir una estratègia d'inicialització.

En la primera part de la practica ja es comentava aquestes inicialitzacions i encara que no es demanaven, un membre del grup les va fer, però no es va fer la comprovació fins ara. Els paràmetres d'entrada no canvien però donen diferents resultats ja que els K punts són els mateixos però en ordre diferent.

En un principi només teníem l'heurística *WithinClassDistance*...

```
WithinClassDistance: [2386.149308560523, 1102.5032745998456]
clusters: [2, 3]
time: [14.480414867401123, 53.50222587585449]
```

...però ara s'han implementat dues més:

- **InterClassDistance**, que calcula la distancia entre classes. Tindrà el paràmetre d'entrada self i en canvi de fer els càlculs per calcular la distancia entre la mateixa classe es faran els de calcular la distancia entre diferents classes.

```
InterClassDistance: [80021.33223922743, 56976.553229982725]
clusters: [2, 3]
time: [14.718581199645996, 53.109251976013184]
```

- **CoeficientFisher** que calcula la distància de Fisher. Al principi es va fer una funció per calcular el numerador i denominador, afegint dos paràmetres a la inicialització de la classe Kmeans: N i centroide global, però fer-ho resultava en un error ja que no es tenía certesa el com calcular-ho correctament fins que es va arribar a una solució molt fàcil: fer una divisió (WCD / ICD). Amb diferencia, aquest es l'heurística que més tarda.

```
CoeficientFisher: [0.029818915054138573, 0.01935012232399618]
clusters: [2, 3]
time: [22.36840796470642, 120.06999015808105]
```

S'ha millorat també **find_best_K**.

La funció segueix igual excepte que s'ha modificat el valor del llindar. D'aquesta forma els càlculs tenen més exactitud i es troba una millor K. Per tant, modificar el llindar fa que el resultat millori o empitjori depenent de quina K troba.

Aquesta imatge mostra els resultats d'alterar el llindar en 20, 10 i 50. Com es pot observar hi ha un gran canvi, sobretot amb un llindar=50 respecte als dos altres.

KNN:

Per obtenir millors resultats sobre KNN es va canviar la distancia que utilitza.

Durant la part 2 del projecte es deia que s'utilitzés la distancia euclidiana. En aquesta part es va modificar aquesta distancia, buscant una distancia que donés un KNN correcte y que sigui millor respecte l'anterior. Es volia que aquesta nova distancia fos més precisa que la distancia euclidiana.

```
The distance metric to use. If a string, the distance function can be
'braycurtis', 'canberra', 'chebyshev', 'cityblock', 'correlation',
'cosine', 'dice', 'euclidean', 'hamming', 'jaccard', 'jensenshannon',
'kulczynski1', 'mahalanobis', 'matching', 'minkowski',
'rogerstanimoto', 'russellrao', 'seuclidean', 'sokalmichener',
'sokalsneath', 'sqeuclidean', 'yule'.
```

De tots aquestes distancies en base al nostre codi només donava un valor correcte "minkowski", i "sqeuclidian", ho hem comprovat mitjançant 'TestCases_knn', proporcionat durant la part 2.

CONCLUSIÓ GLOBAL

Durant la realització del projecte es va desenvolupar e implementar algorismes per el etiquetatge automàtic d'imatges de peces de roba per tipus i color, es van obtenir resultats satisfactoris en la classificació i es van realitzar millores en els algorismes utilitzats. Dit això, la realització va ser afavorida amb un nivell de treball equilibrat, potser no tant en la primera part de l'entrega considerant que el temps que es proporcionava era una mica curt, però, un cop s'adapta, no suposa un problema.

Parlant de problemes, en cada part hi ha hagut alguna implementació que no es podia resoldre tenint que acudir al professorat per a resoldre alguns dubtes com va ser en el cas de find_best_K en Kmeans o get_class en Knn.

En conclusió, com ja s'ha mencionant anteriorment, en aquesta part del treball s'han implementat noves funcions per poder analitzar el treball fet fins ara, així que ha sigut útil per tenir una visió gràfica del projecte. A més al optimitzar el codi respecte a les versions anteriors, s'ha pogut observar com es podia millorar un codi ja treballat en altres parts del projecte. Per tant, en altres paraules, aquesta última part del treball ens ha servit sobretot per acabar d'assolir un treball ja fet prèviament i millorar-lo una mica.