Relazione progetto

Data analysis di immagini

Progetto assegnato: **D**

Andrea Fazio 0729956
Giuseppe Rosa 0730875

Professore: Domenico Garlisi

Corso: Fondamenti di Scienza dei Dati
A.A.:2022/2023

Università degli Studi di Palermo

Sommario

Introduzione 3
Data Collection 3
Data Cleaning 3
Analisi delle Componenti Principali (PCA) 4
Clustering con K-Means 5
Clustering con PCA Variabile 6

Introduzione

In questo progetto, abbiamo condotto un'analisi su un dataset contenente immagini di numeri scritte a mano. Le immagini sono date come matrice 8x8 di pixel, mentre l'ultima cifra rappresenta il target.
L'obiettivo principale del progetto è stato ridurre la dimensionalità di questo dataset utilizzando l'Analisi delle Componenti Principali (PCA) e successivamente eseguire il clustering dei dati tramite algoritmo K-Means, con $k = 10$ cluster.
Abbiamo inoltre eseguito il clustering su proiezioni con un numero variabile di componenti principali (da 2 a 6), per studiare l'effetto della dimensionalità sulla classificazione.

Data Collection

Il dataset è stato caricato utilizzando la libreria sklearn, dividendo tra la matrice dei dati e quella del target.

```
data_set = load_digits(as_frame=True)
data = data_set.data
target = data_set.target
```

Specifichiamo il parametro `as_frame=True` per assicurarci che il dataset venga caricato come dataframe di pandas e agevolare l'analisi.

Visualizzando la dimensione del dataset, e qualche riga casuale, vediamo la struttura del set. Risultano 1797 osservazioni, ognuna delle quali con 64 colonne (i pixel).
Utilizzando la libreria matplotlib abbiamo infine visualizzato un'immagine

Data Cleaning

La fase di cleaning è iniziata verificando la presenza di dati mancanti.

```
missing_values = data.isnull().sum().sum()
```

La funzione `isnull()` restituisce un dataframe con gli elementi a `True` se tale elemento è nullo, i `sum()` servono per sommare tutti i valori della tabella.

Riscontriamo l'assenza di valori nulli nel dataset.

Abbiamo proceduto verificando la distribuzione dei valori, notando, tramite boxplot della libreria sns, che ogni colonna (pixel) del dataset, si muove entro il range [0,16].

Non risulta dunque necessario normalizzare.

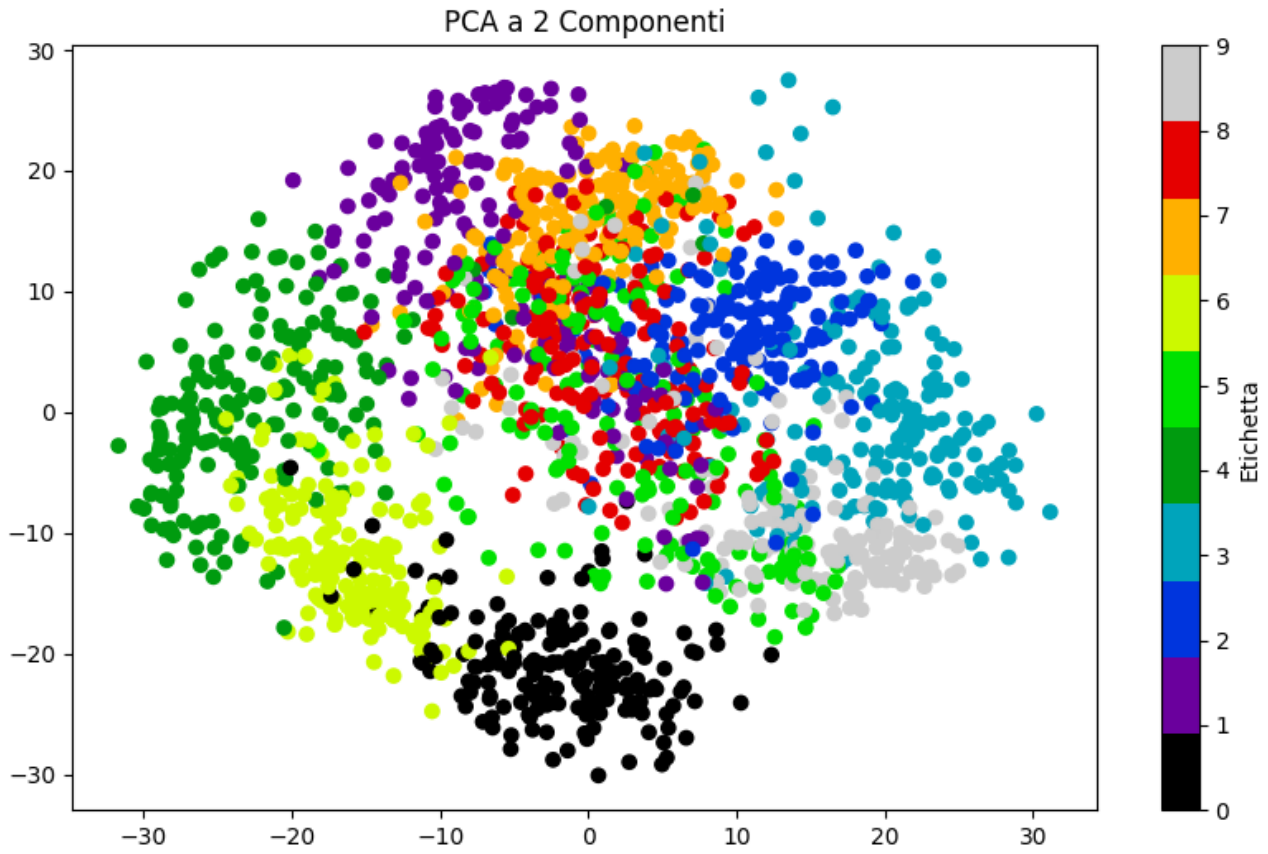
Analisi delle Componenti Principali (PCA)

Per realizzare la PCA abbiamo utilizzato la classe PCA da sklearn.decomposition.

```
pca = PCA(n_components=2)
data_pca = pca.fit_transform(data)
```

La funzione `fit_transform()` centra i dati (supposti normalizzati), per poi svolgere la PCA tramite SVD. In questo modo proiettiamo le 64 dimensioni originali su sole 2 componenti principali.
Moltiplichiamo ogni punto del dataset per -1 in modo da invertire gli assi e ottenere lo stesso risultato che si otterrebbe svolgendo la PCA "manualmente" (sklearn usa un'implementazione diversa).

Successivamente visualizziamo tramite scatter plot il dataset proiettato sulle 2 componenti, etichettando in base al target, ottenendo il seguente plot:



Notiamo come nonostante si abbiano solo due componenti, diversi gruppi di dati risultano ben distinguibili. Sulla parte centrale del plot si riscontra invece un "mescolamento" di dati con etichette diverse, dovuto all'inevitabile perdita di informazioni.

Mostriamo infine la varianza spiegata dalle due componenti rispetto al dataset originale, e a quello ottenuto dopo la PCA. La prima viene data dall'attributo della classe PCA `explained_variance_ratio_`, mentre la seconda la abbiamo calcolata banalmente dividendo la varianza spiegata dalla componente i-esima con la somma delle varianze spiegate dalle 2 componenti.

```
pca.explained_variance_[i]/np.sum(pca.explained_variance_.sum()
```

Clustering con K-Means

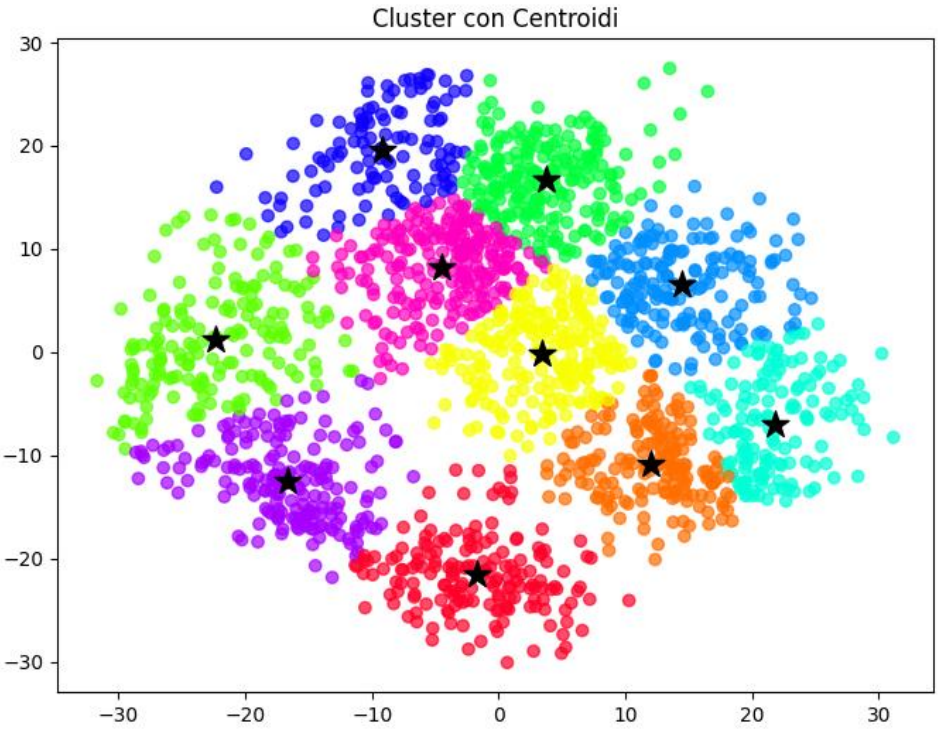
Il clustering dei dati è stato eseguito utilizzando l'algoritmo K-Means.
Tale algoritmo calcola i centroidi tramite media dei valori associati (i più vicini), assegnando a ogni iterazione i punti al centroide più vicino, per poi ricalcolare le medie. Si conclude quando i centroidi non cambiano più (quindi l'algoritmo converge).

Abbiamo utilizzato la classe KMeans da sklearn.cluster con `n_clusters=10` per raggruppare le immagini in 10 cluster distinti, rappresentativi dei target.

```
kmeans = KMeans(n_clusters=10, random_state=0)
cluster_labels = kmeans.fit_predict(data_pca)
```

Il clustering viene svolto dalla funzione `fit_predict()` che utilizza l'algoritmo (di Lloyd) sopra descritto.

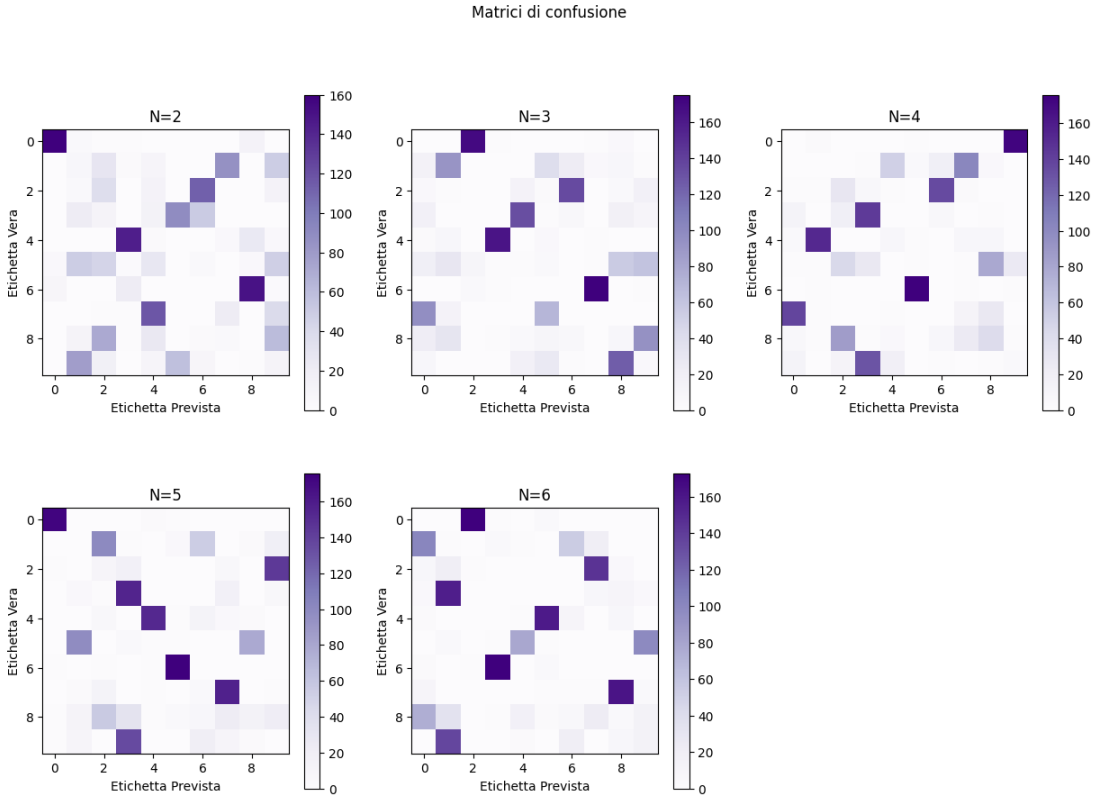
Visualizziamo infine i cluster e i relativi centroidi tramite scatter plot.



Come ci aspettavamo, l'algoritmo svolge un buon lavoro dove i punti erano ben raggruppati e distinti rispetto al target, ma si perde accuratezza nella parte centrale a causa del mescolamento dei punti.

Clustering con PCA Variabile

Abbiamo eseguito il clustering su proiezioni PCA con un numero variabile di componenti principali `n_components` (da 2 a 6).
Ad ogni iterazione, abbiamo utilizzato la PCA per proiettare i dati in N dimensioni e successivamente eseguito il clustering K-Means.
Per ogni clustering si sono calcolate le matrici di confusione per valutarne l'accuratezza rispetto all'etichettatura iniziale del dataset.



I risultati hanno mostrato che aumentando il numero di componenti principali, l'accuracy nella classificazione tende a migliorare, raggiungendo un picco per N=5, a N=6 infatti riscontriamo un grosso calo dell'accuratezza.

Ciò può essere anche constatato usando la funzione `accuracy_score()` di sklearn per classificare la qualità delle previsioni usando le etichette fornite inizialmente (i target).