Machine Learning - Challenge 1

Abdula Kalus

Svolgimento challenge 1, croso di Intelleigenza Artificiale, Universtià degli Studi di Trieste

Abstract

Lo scopo di questa challenge è studiare il dataset 'banknote-authentication', i dati sono stati estratti da immagini, prese da esemplari simili a banconote autentiche e contraffatte. Per la digitalizzazione è stata utilizzata una telecamera industriale solitamente utilizzata per l'ispezione della stampa. Le immagini finali hanno 400x400 pixel. Grazie alla lente dell'oggetto e alla distanza dall'oggetto esaminato sono state ottenute immagini in scala di grigi con una risoluzione di circa 660 dpi. Lo strumento Wavelet Transform è stato utilizzato per estrarre le features dalle immagini.

Analisi preliminare

Il dataset è composto da 4 variabili esplicative e 1372 osservazioni, tutte valide ovvero non presentano valori anoamli.

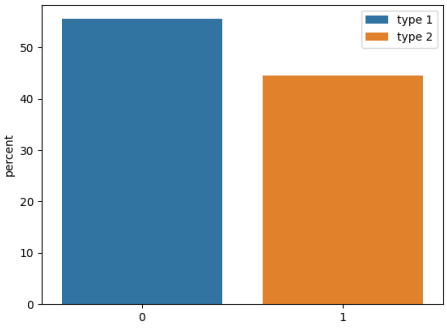

Fig. 1. Countplot

La figura 1 mostra che i due gruppi sono leggermente sbilanciati, con più del 50% delle osservazioni è etichettato con 0 ovvero contraffatte.

Adesso consiedriamo la tabella 1, dove si può osservare che la prima features ovvero **variance** ha una distribuzione di densità differente in base all'etichetta, se la banconotà è contraffatta mi aspetto un valore proveniente dalla densità aranacione, ragioanmento analaogo per l'altro caso. Questa differenza di densità suggerisce che magari sarà possibile identificare dei cluster in base alla densità dei due gruppi utilizzando **DBSCAN**. Per quanto riguarda le altre features non si osserva niente di rilevante.

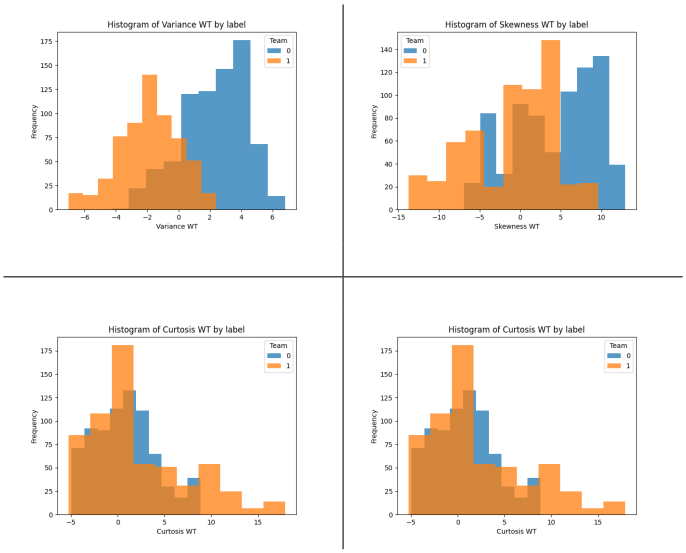

Table 1. Histogram by group

Per concludere l'analisi esplorativa osserviamo come sono correlate le variabili esplicative tra di loro

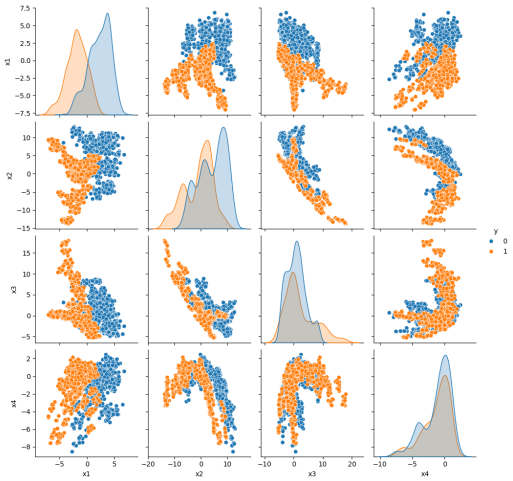

Fig. 2. Pair plot

Dalla figura 2 si può osservare che la variabile **x1** rispetto ad **x2** formino due gruppi, magari due clusetr, a differenza degli altri grafici. Tuttavia riguardo alle altre variabli non si puo dedurre molto.

Unsupervised Learning

L'unsupervised learning, è un metodo di analisi dei dati. Dove lo scopo è quello di riconoscere strutture all'interno dei valori di input. Tuttavia prima di porcedere con alcuni metodi di UL, cerchiamo di capire se tutte le nostre features sono importanti per studaire la struttura dei noi dati.

PCA. Per capire quali delle nostre features sono importanti, ovvero quali di loro ci aiutano a spiegare bene la struttura del modello, ci affidiamo ad una tecnica per la semplificazione dei dati che si chiama **PCA** (*principal component analysis*). La tecnica, ha lo scopo di ridurre il numero più o meno elevato di variabili che descrivono un insieme di dati a un numero minore di variabili latenti, limitando il più possibile la perdita di informazioni.

Applichiamo inanzitutto **PCA** su tutto il dataset, e usiamo la matrice di dispersione per visualizzare i nostri risultati, vedere la figura 3, solo che questa volta le nostre features sono le *componenti principali*, ordinate in base a quanta varianza dei dati sono capaci di spiegare.

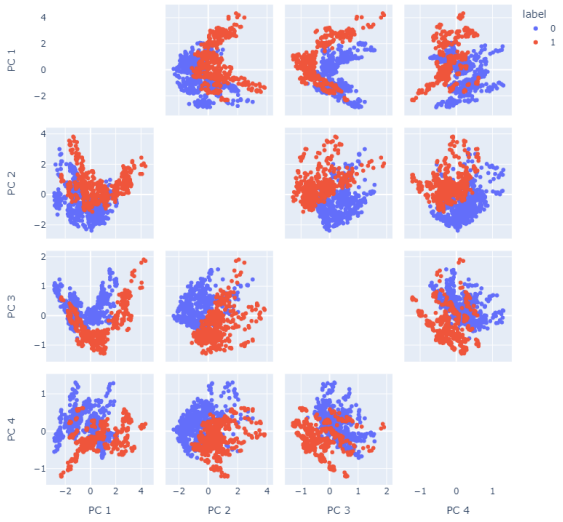

Fig. 3. Matrice dispersione PCA

L'importanza della varianza spiegata è dimostrata nella figura precedente 3. Il grafico tra **PC3** e **PC4** è chiaramente incapace di separare le due calassi, mentre il grafico tra **PC1** e **PC2** mostra una netta separazione.
Visualizziamo adesso un subset delle **PC** (*componenti principali*), qusto perchè quando si hanno molte fetures da visualizzare, forse siamo interessati solo a vedere le componenti più rilevanti. Dato che di solito queste catturano la maggior parte della varianza spiegata, la quale ci dice se queste componenti sono sufficenti per modellare il dataset.

Nell'esempio precedente abbaimo visto tutte le **PCs**, Adesso, ci concetreremo solo sulle prime due **PC**, riducendo la dimensione del dataset da 4 a 2, come si può vedere in figura 4.

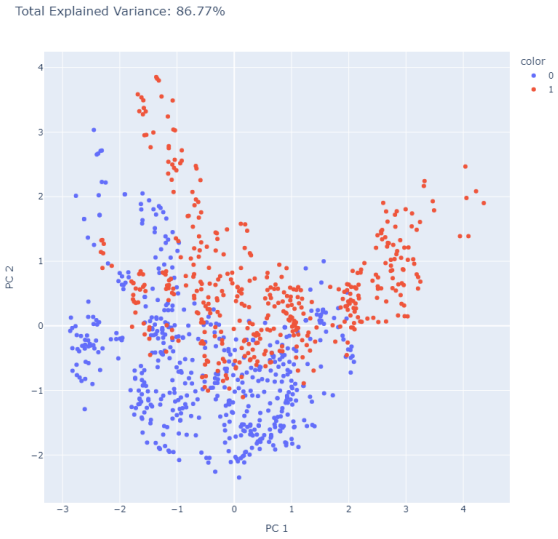

Fig. 4. Diagramma dispersione PC1 vs PC2

Adesso osserviamo quanta varianza PCA è in grado di spiegare man mano che aumenti il numero di componenti, al fine di decidere quante dimensioni mantenere o analizzare alla fine.
Con una varianza spiegata più elevata, puoi acquisire una maggiore variabilità nel tuo set di dati, il che potrebbe potenzialmente portare a prestazioni migliori durante l'addestramento del tuo modello.

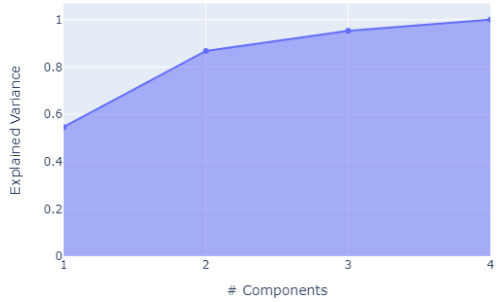

Fig. 5. Explained Variance

Dalla figura 5 possiamo vedere che già con solo due componenti siamo ingrado di spiegare circa il 90% della varaianza, ciò rende inutili le altre due componenti, le quali hanno pochissimo contributo.

kMeans. Per vedere se effettivamente si posssono riconoscere due gruppi all'interno di questo dataset, utlizziamo l'algoritmo di cluster kMeans, il cui obiettivo è di minimizzare la varianza totale intra-gruppo; ogni gruppo viene identificato mediante un centroide o punto medio.
Inanzitutto applichiamo **kMeans** sulle nuove features ottenute precedentemente con **PCA**

Fig. 6. PCA kMeans clustering

Possiamo osservare dal grafico in figura 6 i cluster trovati, rispettivamente quello blu e quello rosso. Essendo che il funzionamento di **kMeans** dipende dalla scelta dei centroidi e anche dalla dipsersione dei delle osservazioni, in questo se non fossimo a conoscenza delle vere etichette assegnate alle osservazioni, avremmo potuto pensare questi cluster siano 'buoni', tuttavia dalla figura 4 sappiamo che non sono i cluster corretti. Inoltre se ci sofferarmiamo sulla parola buona usata tra virgolette poco fa, questo buono è dato dal fatto che l'algoritmo è stato in grado di trovare due cluster, tuttavia potendo osservare un grafico 2D possiamo afferamre che con una colorazione uniforme ci sarebbe la presenza di un unico cluster.

Provando invece ad utilizzare **kmeans** sul dataset composto da tutte le features, otteniamo un risultato analogo, infatti osservadno la figura 7 e confrontandola con la figura precedente 3 possiamo osservare che anche in questo caso non siamo in grado di riconoscere i cluster corretti, inoltre senza la distinzione con i colori dei punti anche in questo caso, si vedrebbe una macchia omogenea per ogni grafico.

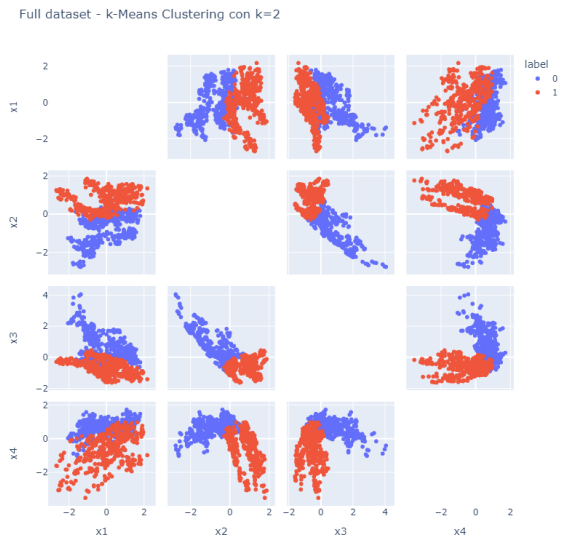

Fig. 7. full dimension kMeans clustering

t-SNE. *t-Dispersed Stochastic Neighbor* (**t-SNE**) è una strategia non diretta per la riduzione della dimensionalità che è particolarmente appropriata per la percezione di set di dati ad alta dimensione. La differenza sostanziale tra **t-SNE** e **PCA** è che quest'ultimo è una procedura numerica, invece **t-SNE** è una procedura probabilistica.

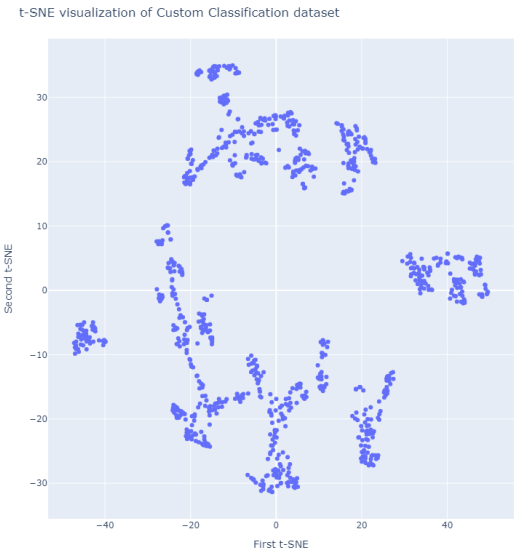

Fig. 8. t-SNE cluster

Osservando la figura 8 possiamo notare un risultato migliore rispetto a **kMeans** applicato con la riduzione di dimensionalità ottenuta con **PCA**, infatti qui possiamo chiaramente vedere più cluster e sotto-cluster distinti.

DBSCAN. Quando i dati richiedono lo sviluppo di cluster dalle forme arbitrarie o gruppi di cluster limitrofi e sovrapposti, le tecniche tradizionali faticano a ottenere buoni risultati, ad esmepio gli elementi all'interno del cluster potrebbero non condividere similarità sufficienti oppure potrebbero avere povere performance.

Le difficoltà non terminano qui, gli algoritmi tradizionali sono privi della nozione di *outliers*, il **DBSCAN** è invece in grado di riconoscerli evitando che questi finiscano assegnati a un cluster pur non appartenendo ad alcuno di essi: un'operazione fondamentale nel campo della anomaly detection.

Le osservazioni aberranti avvicinano a sé i centroidi dei cluster, rendendo più complessa la loro gestione, in contrasto, altri sistemi di clustering come il DBSCAN separano regioni ad alta e bassa densità. Evidentemente per densità ci riferiamo al numero di osservazioni entro uno specifico raggio. Uno dei meravigliosi vantaggi del DBSCAN è la capacità d'individuare cluster dalle forme arbitrarie senza rimanere influenzato dal rumore di fondo (noise).

Il funzionamento del DBSCAN si basa sul presupposto che se una particolare osservazione appartenesse a uno specifico cluster, dovrebbe essere vicina alla maggior parte dei samples appartenenti al medesimo cluster. In questo algoritmo due parametri giocano un ruolo chiave, detti *radius* e *minimum points*, definiamo poi dense area quella regione di spazio delimitata dalla circonferenza di raggio ***epsilon*** contenete un sufficiente numero di datapoints.
Iniziamo osservando la figura 9, dove si nota subito che è presente un unica nuvola di punti, nella quale i punti sono più o meno densi a seconda dell'area del grafico.

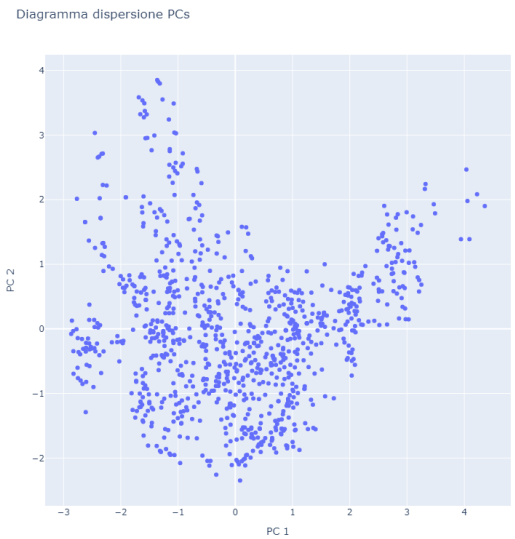

Fig. 9. diagramma dispersione PCs

Adesso proviamo ad utilizzare **DBSCAN**, impostando dei parametri inziali in modo casuale ma non troppo, impostando il *radius* a 0.5 e *minimum points* a 10.

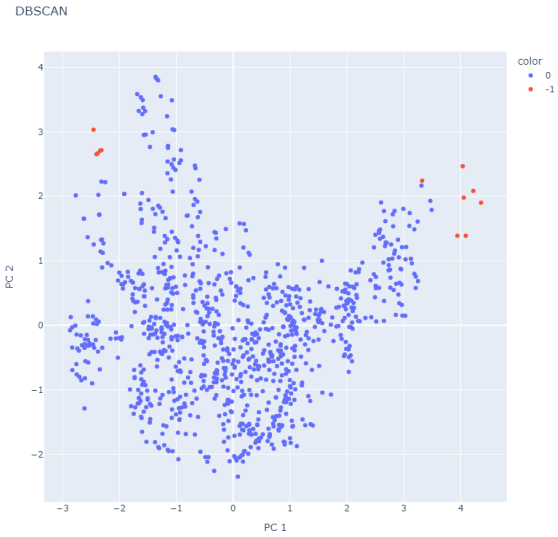

Fig. 10. DBSCAN *eps 0.5 - min*$_s mpl 10$

Dalla figura 10 notiamo la presenza di un cluster, con segnato in rosso gli *outliers*, dopo aver testato diverse impostazioni dei parametri, il miglior setting è *radius* a 0.3 e *minimum points* a 20.
Ottenendo il seguente rilsutato, vedere 11, dove sono presenti 5 cluster con circa 250 *outliers*.

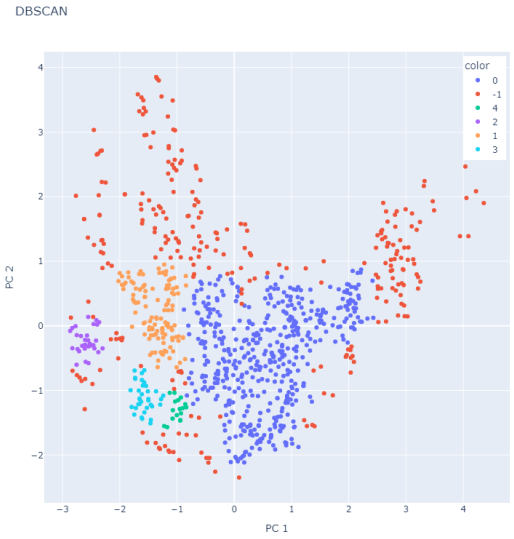

Fig. 11. DBSCAN *eps 0.3 - min*$_s mpl 20$

Supervised Learning

In questa sezione ci occuperemo di cerare un modello in grado di imparare anche dalle label che abbiamo a disposzione oltre che ai soli dati.

Logistic Regression. Costruiamo un modello di regressione lineare logistica binario, che ha lo scopo di imparare a classificare le nostre osservazioni nelle due classi. L'obbiettivo della classificazione è prevedere se una banconota è autentica o meno (variabile y). Autentica = 1 e Contraffatta = 0.
Inanzitutto consideriamo un modello senza regolarizzazione che denotiamo con **NRM**, il quale converge per tre diversi valori del *learning rate* come possiamo vedere dal grafico in figura 13

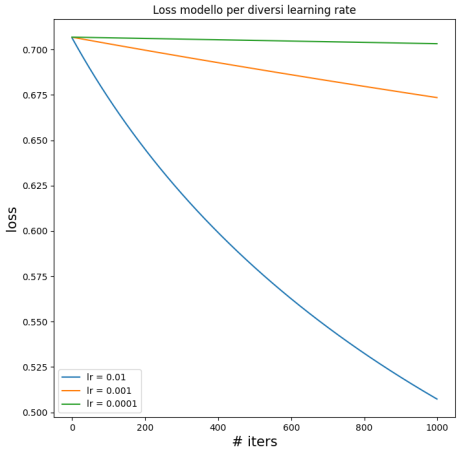

Fig. 12. Loss NRM

dato che il modello con *learning rate = 0.01* è quello che converge più velocemente, sarà quello preso in considerazione. Adesso confrontiamo la matrice di confusione del *base line model* **BLM**, tabelle 2 contro la matrice di confusione del modello **NRM**, 3

Totale = P + N = PP + PN, PP = Predetti Positivi, PN = Predetti Negativi, P = Positivi Veri, N = Negativi Veri

Totale	PP	PN
P	0	166
N	0	206

Table 2. BLM confusion matrix

Totale	PP	PN
P	120	46
P	7	199

Table 3. *NRM confusion matrix*

e infine consideriamo la loro bontà con le seguenti misure di performance

	Accuracy	Precision	Recall	F1-score
BLM	0.55	0	0	0
NRM	0.86	0.94	0.72	0.82

Table 4. *Precision measures*

confrontando le performance tra di loro abbiamo che il modello **NRM** è di gran lunga superiore al **BLM** quindi possiamo affermare che è un buon modello. Inoltre provando ad utilizzare **k-fold cross validation** con **K=5** otteniamo i seguenti riusltati 4, i quali ci mostrano che il modello trovato precedentemente riesce e spiegare bene i dati.

	Accuracy	Precision	Recall	F1-score
k-fold	0.84	0.93	0.70	0.80

Table 5. *K-fold*

Per migliorare il modello, inizialmente si è provatoa da aggiungere la regolarizzazione **RIDGE** la quale aiutà a eliminare i problemi di multicollinearità, tuttavia come si puo vedere dalla figura 2 le variabili non mostrano prblemi di multicollinearità, quindi escludiamo la possibilità di migliorare il modello in questo modo.

	Accuracy	Precision	Recall	F1-score
BLM	0.55	0	0	0
NRM	0.86	0.94	0.72	0.82
NB	0.85	0.84	0.83	0.83
DT	0.99	0.98	0.99	0.98
KNN	1	1	1	1

Table 6. NRM = LR no reg, NB = Naive-Bayes, DT = Decision Tree, KNN

Decision tree, Naive-bayes, k-NN. Come possiamo vedere dalla tabella 6, la quale riassume le performance di tutti i modelli, tutti riescono a modellare il problema molto bene, ottenendo anche un punteggio massimo con l'algorimto **KNN**

ROC curve

Per finire confrontiamo tutti i modelli con la ROC curve

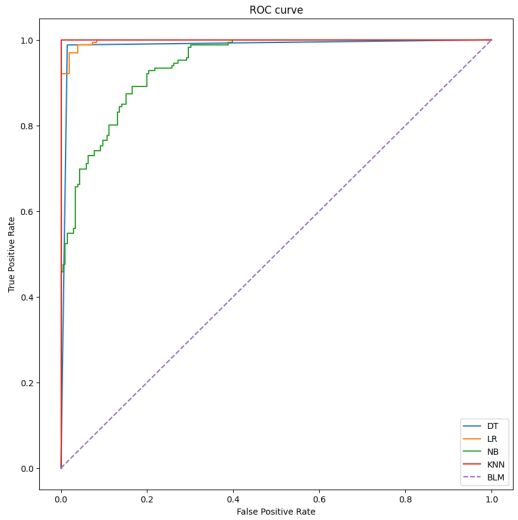

Fig. 13. ROC curve