Analisi dei test effettuati

Test di Stefano Biddau su F-MNIST con Percettrone multinastro

Cenni teorici Percettrone multinastro

DEFINIZIONE:

La **Rete Neurale Artificiale** è un sottoinsieme del machine-learning e costituisce il nucleo degli algoritmi di deep-learning. Il suo nome e la sua struttura è ispirata al cervello umano e imita il modo in cui i neuroni biologici si scambiano segnali.
Un neurone artificiale è una funzione matematica che riceve un input, elabora quell'informazione e restituisce un output.
Il percettrone multinastro è un **architettura** di rete neurale, costituita da una rete di neuroni (chiamati anche nodi) distribuiti su più layers:

- **Un layer di input**: in cui il numero di neuroni corrisponde al numero di proprietà del nostro datate;
- **Un layer di output**: in cui il numero di neuroni corrisponde al numero di classi;
- **Uno o più hidden layers**: livelli intermedi che utilizzano l'output del lager precedente per apprendere nuove proprietà;

Una rete neurale con un **singolo hidden layer** è anche definita **vanilla neural network**, mentre una rete **con due o più hidden layers** è anche definita **deep neurale network** (rete neurale profonda).

FUNZIONAMENTO

Ogni neurone di un hidden layer corrisponde ad un percettrone e l'attivazione di ognuno di essi è data da una funzione di attivazione non lineare:

PRINCIPALI IPER-PARAMETRI DEL MODELLO

Ci sono diversi iper-parametri che caratterizzano una ANN tra questi i più importanti dal punto di vista di incidenza sull'accuratezza dei risultati sono:
Il principale è sicuramente la **funzione di attivazione per il livello nascosto**, che determina se attivare il neurone. Ciò significa utilizzare semplici operazioni matematiche per determinare se l'input di un neurone nella rete è rilevante per il processo di previsione. Lo scopo di una funzione di attivazione è quello di introdurre una non linearità in una rete neurale artificiale, producendo un output da una raccolta di valori di input dati allo strato. Principalmente teniamo in considerazione due tipologie di funzione di attivazione:

- **Relu (unità lineare rettificata)**: una funzione facile da calcolare, e dato che tutti i neuroni non sono attivati, questo crea scarsità nella rete e quindi sarà veloce ed efficiente. Tuttavia tale funzione non è centrata su zero e può uccidere alcuni neuroni per sempre poiché da sempre 0 per valori negativi;
- **Tanh (tangente iperbolica)**: è una funzione centrata su zero ed ha un intervallo compreso tra -1 e +1 e questi ne caratterizzano i suoi vantaggi, anche se è computazionalmente costosa.

Un altro parametro importante e il **risolutore** per l'ottimizzazione del peso, nel nostro caso considereremo il risolutore **adam perché funziona abbastanza bene su set di dati relativamente grandi.**

Associato al peso abbiamo un'altro parametro che è il **learning_rate**, che è il programma del tasso di apprendimento per gli aggiornamenti del peso.
Principalmente si considerano due tipologie ossia:

- **constant**: rappresenta una velocità di apprendimento costante
- **adaptive**: mantiene la velocità di apprendimento costante finché la perdita di allenamento continua a diminuire.

Poi abbiamo imax_iter, che nel caso in cui il risolutore è Adam rappresenta il numero di epoche, ossia quante volte verrà utilizzato ciascun punto dati.

Ultimo ma non meno importante è il valore **hidden_layer_sizes** che ci permette di settore il numero di hidden layers che costituirà la nostra rete e la quantità di neuroni nascosti per ogni livello. Avere più livelli intermedi e un gran numero di neuroni permette al modello di separare i dati in modo non lineare. Quindi maggiori saranno i neuroni, maggiore sarà la complessità che tale rete può gestire.

PRO E CONTRO DELLA RANDOM FOREST

PRO

- Capacità di classificare pattern complessi e non lineari come le immagini, video, suoni e testi;
- Capacità di lavorare in parallelo;
- Tolleranza a agli errori e al rumore;
- Alta precisione;
- Capacità di generalizzazione;

CONTRO

- Servono per risolvere problemi specifici;
- Occorrono dataset di una certa dimensione;
- Possono richiedere tempi di training molto lunghi;

TEST EFFETTUATI

F-MNIST

Dai grafici di accuratezza generati dall'esecuzione dello script per l'ottimizzazione degli Iper parametri notiamo che la complessità del dataset fmnist rende inadatto il numero di neuroni per layer utilizzato per classificare invece il dataset mnist.
Per questo abbiamo testato la rete prima con hidden layer da 250 neuroni e poi con 512.
I risultati ottenuti hanno mostrato diverse combinazioni che permettono alla rete di performare molto bene sul dataset.

I primi 4 blocchi di test verranno effettuati con una rete a diversi hidden layer ognuno composto da 250 neuroni.

1° Blocco di test (Hidden Layers crescenti , a: Tahn, lr: Adaptive)

Indice test	Hidden Layers	Funz att	Learning rate	Acc Train	Acc Test	Differenza MSE	Over-fit
1	1	Tanh	Adaptive	99,74%	88,78%	1,55	NO
2	**2**	**Tanh**	**Adaptive**	**99,66%**	**89,54%**	**1,44**	**NO**
3	3	Tanh	Adaptive	88,26%	85,21%	0,33	NO
4	4	Tanh	Adaptive	98,94%	88,84%	1,44	NO

Da questi primi 4 test si evince che all'aumentare degli hidden layers che rendono la rete sempre più profonda, l'accuratezza sul set di test e di train aumenta e sembra stabilizzarsi intorno ai valori del 99% per il set di train e del 89 % invece per il set di test.
Il test 3 stona con gli altri risultati, quindi ci mostra che con 3 hidden layers la rete performa in mano modo. In contrapposizione abbiamo la rete con 2 hidden layers del **test 2** che mostra i migliori risultati si aia termini di accuratezza sul set di test e di train sia per quanto riguarda la differenza di MSE. Non sembra esserci overfitting o underfitting.

Altro aspetto che notiamo è che il test 2 (il migliore) propone un architettura di rete il cui tempo di addestramento è il più basso rispetto a tutti gli altri :

- Vanilla neural network -> 4 min
- **ANN con 2 hidden layers -> 3 min**
- ANN con 3 hidden layers -> 5 min
- ANN con 4 hidden layers -> 5 min

Adesso cambiamo il Learning rate Adattive -> Constant è vediamo la nostra rete come si comporta

2° Blocco di test (Hidden Layers crescenti , a: Tahn, lr: Constant)

Indice test	Hidden Layers	Funz att	Learning rate	Acc Train	Acc Test	Differenza MSE	Over-fit
1	1	Tanh	Constant	99,74%	88,78%	1,55	NO
2	**2**	**Tanh**	**Constant**	**99,66%**	**89,54%**	**1,44**	**NO**
3	3	Tanh	Constant	88,26%	85,21%	0,33	NO
4	4	Tanh	Constant	98,94%	88,84%	1,44	NO

Da questi 4 test si evince che il cambio del larning rate non apporta cambiamenti ne all'accuratezza ne alla differenza di MSE, infatti abbiamo esattamente gli stessi risultati del 1° Blocco di test.
L'unica differenza che possiamo riscontrate è il tempo di addestramento della rete che risulta maggiore per tutte le strutture proposte:

- Vanilla neural network -> 4 min
- **ANN con 2 hidden layers -> 4 min**
- ANN con 3 hidden layers -> 4 min
- ANN con 4 hidden layers -> 6 min

Ripetiamo i due blocchi di test effettuati cambiando la funzione di attivazione Tanh -> Relu e vediamo se otteniamo risultati migliori

3° Blocco di test (Hidden Layers crescenti , a: Relu, lr: Adaptive)

Indice test	Hidden Layers	Funz att	Learning rate	Acc Train	Acc Test	Differenza MSE	Over-fit
1	1	Relu	Adaptive	99,02%	89,25%	1,44	NO
2	**2**	**Relu**	**Adaptive**	**99,24%**	**89,68%**	**1,36**	**NO**
3	3	Relu	Adaptive	99,32%	89,52%	1,41	NO
4	4	Relu	Adaptive	99,30%	89,30%	1,37	NO

Da questi 4 test si evince che la funzione di attivazione **Relu** permette di abbassare il tasso di accuratezza sul set di train (chiaramente di poche frazioni di punti percentuali) e di alzare la percentuale di accuratezza sul set di test permettendo così di ridurre la differenza di MSE.
I test mostrano un andamento costante con l'accuratezza sul set di train e di test che oscillano rispettivamente nel 99% e nell'89%.

I migliori risultati si ottengono con il **test 2** che mostra la miglior accuratezza sul set di test fino a questo momento ma anche la migliore difenda di MSE.

Un' altra differenza che possiamo notare sta nel tempo di addestramento della rete che con la funzione di attivazione Relu aumenta considerevolmente rispetto ai test di addestramento precedenti:

- Vanilla neural network -> 5 min
- **ANN con 2 hidden layers -> 7 min**
- ANN con 3 hidden layers -> 5 min
- ANN con 4 hidden layers -> 8 min

4° Blocco di test (Hidden Layers crescenti , a: Relu, lr: Constant)

Indice test	Hidden Layers	Funz att	Learning rate	Acc Train	Acc Test	Differenza MSE	Over-fit
1	1	Relu	Constant	99,02%	89,25%	1,44	NO
2	**2**	**Relu**	**Constant**	**99,24%**	**89,68%**	**1,36**	**NO**
3	3	Relu	Constant	99,32%	89,52%	1,41	NO
4	4	Relu	Constant	99,30%	89,30%	1,37	NO

Da questi 4 test si evince che il cambio del larning rate non apporta cambiamenti ne all'accuratezza ne alla differenza di MSE, infatti abbiamo esattamente gli stessi risultati del 3° Blocco di test.
L'unica differenza che possiamo riscontrate è il tempo di addestramento della rete che risulta maggiore per tutte le strutture proposte:

- Vanilla neural network -> 6 min
- **ANN con 2 hidden layers -> 7 min**
- ANN con 3 hidden layers -> 5 min
- ANN con 4 hidden layers -> 8 min

Adesso reseguiamo i 4 blocchi di test precedenti, solamente aumentiamo il numero di neuroni da 250 -> 512

5° Blocco di test (Hidden Layers crescenti , a: Tahn, lr: Adaptive)

Indice test	Hidden Layers	Funz att	Learning rate	Acc Train	Acc Test	Differenza MSE	Over-fit
1	**1**	**Tanh**	**Adaptive**	**99,72%**	**89,72%**	**1,40**	**NO**
2	2	Tanh	Adaptive	98,91%	89,07%	1,33	NO
3	3	Tanh	Adaptive	98,95%	89,22%	1,37	NO
4	4	Tanh	Adaptive	98,63%	89,10%	1,31	NO

Da questi primi 4 test si evince che il numero di neuroni aumenta l'accuratezza sul set di test, che mostra i sui risultati migliori subito con il test 1. Dopodiché il valore di accuratezza sembra mantenersi stabile sull' 89% per il set di test e sul 98% per il set di train.
Lievita notevolmente il tempo di addestramento delle reti che ora hanno dei livelli composti da più del doppio dei neuroni:

- **Vanilla neural network -> 6 min**
- ANN con 2 hidden layers -> 7 min
- ANN con 3 hidden layers -> 9 min
- ANN con 4 hidden layers -> 10 min

Adesso cambiamo il Learning rate Adattive -> Constant è vediamo la nostra rete come si comporta

6° Blocco di test (Hidden Layers crescenti , a: Tahn, lr: Constant)

Indice test	Hidden Layers	Funz att	Learning rate	Acc Train	Acc Test	Differenza MSE	Over-fit
1	**1**	**Tanh**	**Constant**	**99,72%**	**89,72%**	**1,40**	**NO**
2	2	Tanh	Constant	98,91%	89,07%	1,33	NO
3	3	Tanh	Constant	98,95%	89,22%	1,37	NO
4	4	Tanh	Constant	98,63%	89,10%	1,31	NO

Da questi 4 test si evince che il cambio del larning rate non apporta cambiamenti ne all'accuratezza ne alla differenza di MSE, infatti abbiamo esattamente gli stessi risultati del 3° Blocco di test.
L'unica differenza che possiamo riscontrate è il tempo di addestramento della rete che risulta maggiore per alcune strutture proposte proposte (rete con tre hidden layers):

- **Vanilla neural network -> 6 min**
- ANN con 2 hidden layers -> 8 min
- ANN con 3 hidden layers -> 9 min
- ANN con 4 hidden layers -> 12 min

Ripetiamo i due blocchi di test effettuati cambiando la funzione di attivazione Tanh -> Relu e vediamo se otteniamo risultati migliori

7° Blocco di test (Hidden Layers crescenti , a: Relu, lr: Adaptive)

Indice test	Hidden Layers	Funz att	Learning rate	Acc Train	Acc Test	Differenza MSE	Over-fit
1	1	Relu	Adaptive	99,54%	89,35%	1,40	NO
2	2	Relu	Adaptive	99,19%	89,57%	1,40	NO
3	**3**	**Relu**	**Adaptive**	**99,53%**	**89,69%**	**1,44**	**NO**
4	4	Relu	Adaptive	98,82%	89,35%	1,37	NO

Da questi 4 test si evince che

8° Blocco di test (Hidden Layers crescenti , a: Relu, lr: Constant)

Indice test	Hidden Layers	Funz att	Learning rate	Acc Train	Acc Test	Differenza MSE	Over-fit
1	1	Relu	Constant	99,54%	89,35%	1,40	NO
2	2	Relu	Constant	99,19%	89,57%	1,40	NO
3	**3**	**Relu**	**Constant**	**99,53%**	**89,69%**	**1,44**	**NO**
4	4	Relu	Constant	98,82%	89,35%	1,37	NO

Da questi 4 test si evince che

Possiamo stilare una lista di 3 test, quelli con i migliori risultati ottenuti in termini di percentuale di accuratezza sul set di test:

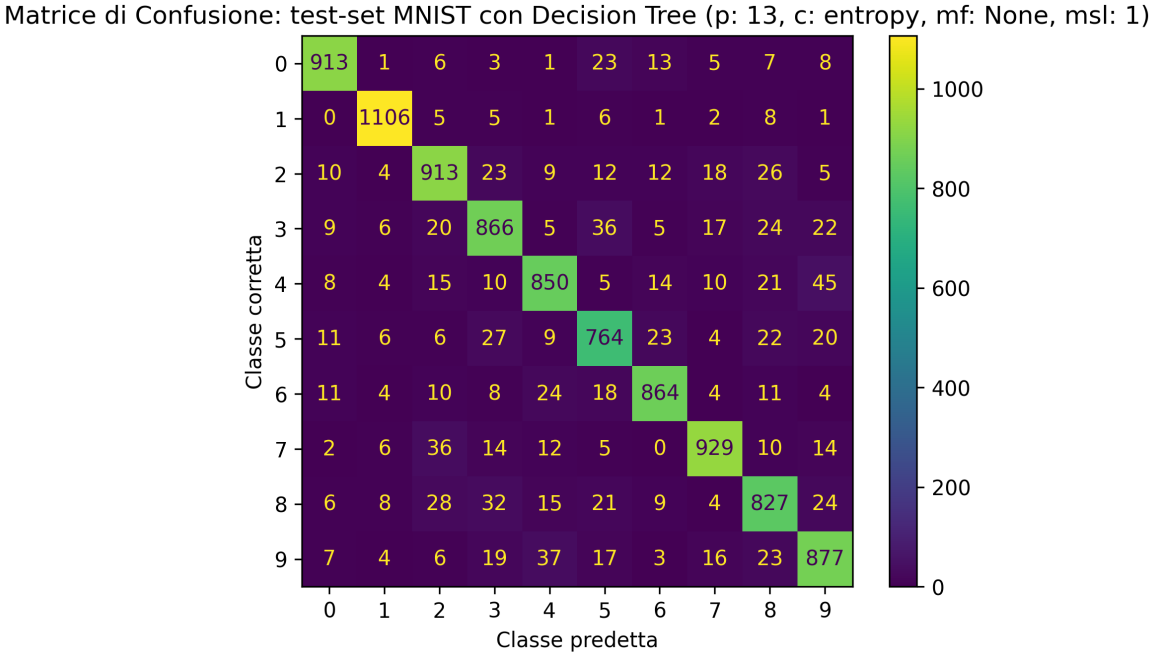
1) Blocco 7 - Test 2: Test del 2022-10-31 12:46:39
2) Blocco 4 - Test 4: Test del 2022-10-30 17:34:43
3) Blocco 10 - Test 3: Test del 2022-11-01 17:24:20

1) **Test del 2022-10-31 12:46:39 MNIST con Decision Tree (profondità: 13, c: entropy, mf: None, msl:1)**

Report di classificazione

	precision	recall	f1-score	support
0	0.93	0.93	0.93	980
1	0.96	0.97	0.97	1135
2	0.87	0.88	0.88	1032
3	0.86	0.86	0.86	1010
4	0.88	0.87	0.87	982
5	0.84	0.86	0.85	892
6	0.92	0.90	0.91	958
7	0.92	0.90	0.91	1028
8	0.84	0.85	0.85	974
9	0.86	0.87	0.86	1009
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

Dal report di classificazione notiamo che il nostro classificatore ottiene i migliori risultati classificando la **classe 1-> cifra 1** e i peggiori risultati di classificazione con la **classe 8 -> cifra 8**



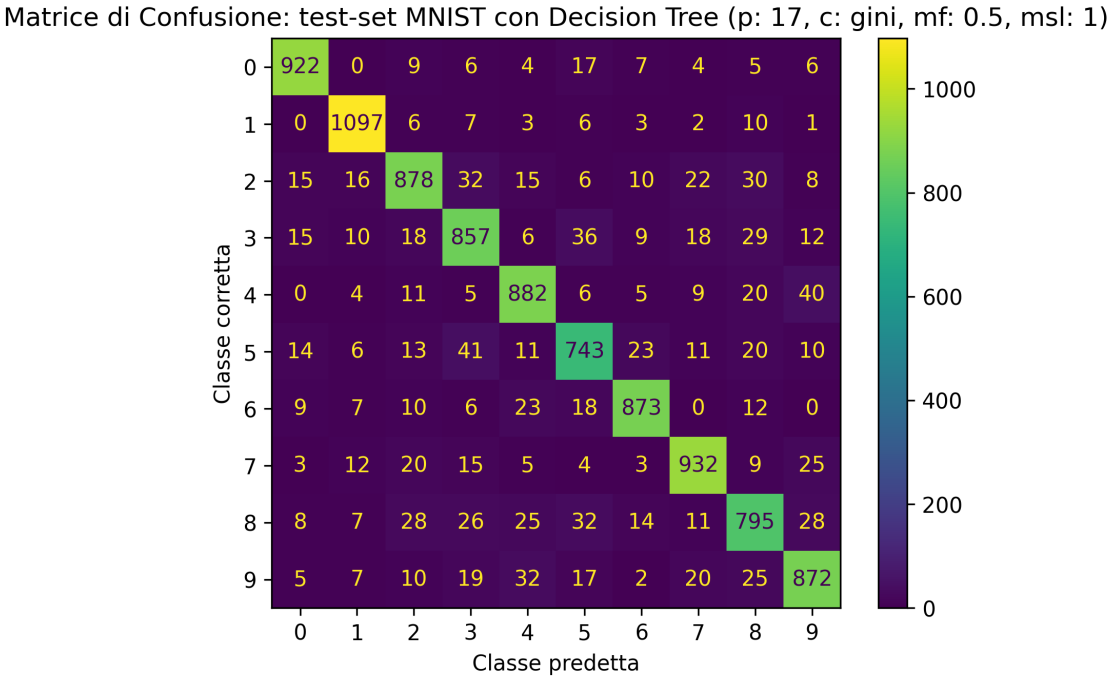
La matrice di confusione conferma ciò che viene mostrato nel report di classificazione la classe che corrisponde alla **cifra 1** mostra su circa **1135** campioni di **cifra 1 1106** vengono classificati correttamente, analogamente la classe che corrisponde alla **cifra 8** è quella che ottiene i peggiori risultati infatti su **974** campioni disponibili solo **827** vengono classificati correttamente. Risaltano all'occhio **32** campioni di **cifra 8** classificati come **cifra 3** e **28** campioni di **cifra 8** classificata come **cifra 2**

2) **Test del 2022-10-30 17:34:43 MNIST con Decision Tree (profondità: 17, c: gini, mf: 0.5, msl: 1)**

Report di classificazione

	precision	recall	f1-score	support
0	0.93	0.94	0.94	980
1	0.94	0.97	0.95	1135
2	0.88	0.85	0.86	1032
3	0.85	0.85	0.85	1010
4	0.88	0.90	0.89	982
5	0.84	0.83	0.84	892
6	0.92	0.91	0.92	958
7	0.91	0.91	0.91	1028
8	0.83	0.82	0.82	974
9	0.87	0.86	0.87	1009
accuracy			0.89	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.89	0.88	10000

Dal report di classificazione notiamo che il nostro classificatore ottiene i migliori risultati classificando la **classe 1-> cifra 1** e i peggiori risultati di classificazione con la **classe 8 -> cifra 8**



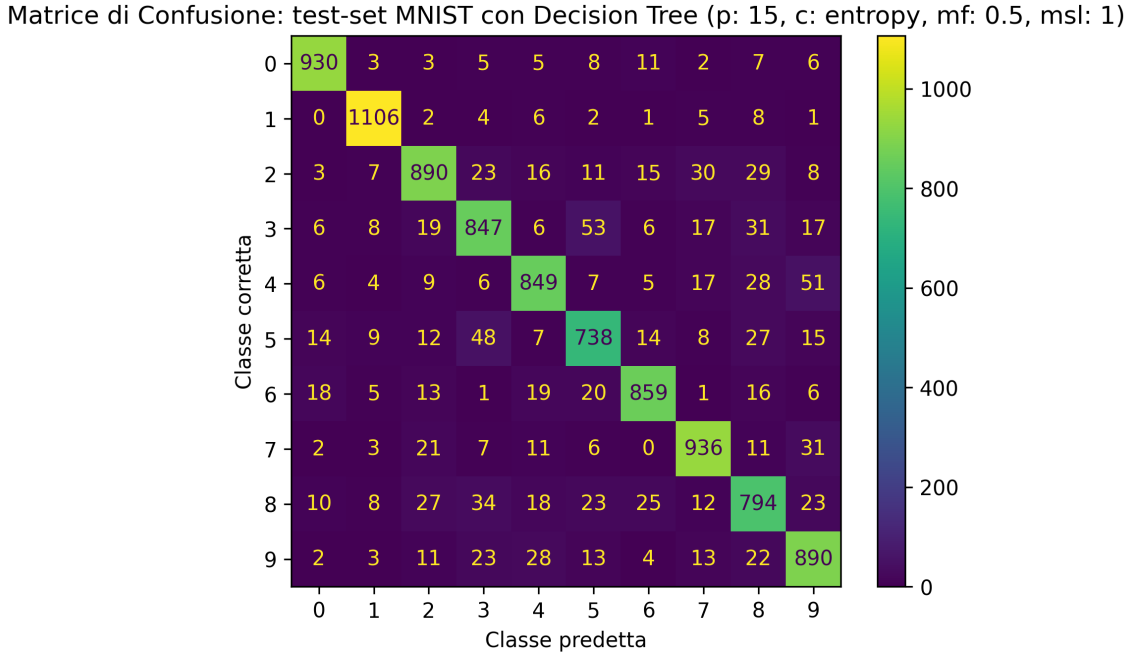
La matrice di confusione conferma ciò che viene mostrato nel report di classificazione la classe che corrisponde alla **cifra 1** mostra su circa **1135** campioni di **cifra 1 1097** vengono classificati correttamente, analogamente la classe che corrisponde alla **cifra 8** è quella che ottiene i peggiori risultati infatti su **974** campioni disponibili solo **795** vengono classificati correttamente. Risaltano all'occhio **32** campioni di **cifra 8** classificati come **cifra 5, 28** campioni di **cifra 8** classificata come **cifra 2** e **26** campioni di **cifra 8** classificati come **cifra 3**

3) **Test del 2022-11-01 17:24:20 MNIST con Decision Tree (profondità: 15, c: entropy, mf: 0.5, msl: 1)**

Report di classificazione

	precision	recall	f1-score	support
0	0.94	0.95	0.94	980
1	0.96	0.97	0.97	1135
2	0.88	0.86	0.87	1032
3	0.85	0.84	0.84	1010
4	0.88	0.86	0.87	982
5	0.84	0.83	0.83	892
6	0.91	0.90	0.91	958
7	0.90	0.91	0.90	1028
8	0.82	0.82	0.82	974
9	0.85	0.88	0.87	1009
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000

Dal report di classificazione notiamo che il nostro classificatore ottiene i migliori risultati classificando la **classe 1-> cifra 1** e i peggiori risultati di classificazione con la **classe 8 -> cifra 8**



La matrice di confusione conferma ciò che viene mostrato nel report di classificazione la classe che corrisponde alla **cifra 1** mostra su circa **1135** campioni di **cifra 1 1106** vengono classificati correttamente, analogamente la classe che corrisponde alla **cifra 8** è quella che ottiene i peggiori risultati infatti su **974** campioni disponibili solo **794** vengono classificati correttamente. Risaltano all'occhio **34** campioni di **cifra 8** classificati come **cifra 3, 27** campioni di **cifra 8** classificata come **cifra 2** e **25** campioni di **cifra 8** classificati come **cifra 5**