Analisi dei test effettuati

Test di Stefano Biddau su MNIST con Foreste randomiche

Cenni teorici Random Forest

DEFINIZIONE:

Uno degli algoritmi **più utilizzati nel Machine Learning** è proprio quello denominato **Random Forest**. Esso è un **algoritmo di tipo supervisionato** che può essere usato sia per task di **classificazione** che di **regressione**.
Caratteristica di questo algoritmo è quella di essere un modello "**ensemble**" ovvero un modello che al proprio interno mette **insieme altri modelli** più semplici. In particolare la tipologia di ensemble rappresentata dal random forest è quella denominata "**bagging**" che sta per **Boostrap aggregating**.

FUNZIONAMENTO

Quando si decide di utilizzare un Random Forest, l' obiettivo sarà quello di **creare tanti classificatori deboli** che uniti insieme possono portare ad un risultato eccellente. La caratteristica di questo algoritmo è che **gli alberi sono indipendenti**, non si condizionano a vicenda. Questo permette di evitare alcuni degli errori più comuni degli alberi, infatti gli alberi rischiano di essere molto instabili a causa del loro modo intrinseco di fare prediction, ovvero il metodo verticale ad albero.
Altra importante peculiarità è quella di scegliere ogni volta un **campione casuale di variabili** da utilizzare per il train interno, il che rende ancora di più indipendenti e differenti gli alberi, permettendo ad ognuno di loro una diversa specializzazione. Il Bagging inoltre fa riferimento alla **tecnica boostrap**, o in italiano **campionamento casuale con rimpiazzo**. Questa tecnica implica che alcuni dati possono comparire contemporaneamente in più modelli mentre altri potrebbero non comparire mai.
Ogni albero vale per uno, e nessuno pesa più degli altri, questo come abbiamo detto è un punto di forza, ma bisogna comunque arrivare ad una qualche conclusione. Possiamo immaginare il **Random Forest come un parlamento**: il modo di decidere è quello del voto, nel caso della classificazione. Se si avranno 5 alberi nel modello e tre ottengono come risultato per una certa osservazione il target 1 e due il target 0, il risultato per maggioranza andrà al target 1.

PRINCIPALI IPER-PARAMETRI DEL MODELLO

Vediamo quali sono (per la mia esperienza) i principali parametri o iper-parametri di un modello Random Forest, prendendo in considerazione l'implementazione della libreria scikit-learn in Python.

Tra i parametri da tenere fortemente in considerazione c'è la metrica di splitting, che, misurando la qualità dello split, separerà i dati. Questo parametro si chiama **criterion** e dipende dal task che si vuole svolgere.
Per quanto riguarda il problema della classificazione, può avere "gini" e "entropy" come possibili valori. "Gini" fa riferimento all'impurità di Gini, mentre "entropy" all'informazione gain, basato sul concetto di entropia nella teoria dell'informazione.

Un altro parametro da controllare è quello che si chiama **max_depth**: questo indica quanto profondo sarà l'albero. Più profondo sarà l'albero, più split farà e più sarà preciso sui dati di addestramento. Questo però non vuol dire che se è più preciso sui dati di train il modello sarà in

grado di generalizzarli bene, anzi più sarà profondo più sarà profondo più preciso sarà sui dati di train e mono probabilmente sarà in grado di generalizzare su dati nuovi.
Una maggior profondità quindi rischia di far cadere in quel fenomeno chiamato overfitting ovvero un'ottima capacità sul train ma una pessima sul test che si traduce in un incapacità di generalizzare su nuovi dati.

Un altro parametro considerato è **min_samples_split**, ovvero il numero minimo di campioni necessari per dividere un nodo intero. Nel nostro caso sarà un valore intero.

Altro iper-parametro importante è **n_estimators**. Questo parametro ci da la possibilità di scegliere in numero di alberi nella che faranno parte della foresta

PRO E CONTRO DELLA RANDOM FOREST

PRO
- Modello estremamente performante;
- Poca probabilità di incorrere in overfitting;
- Utile sia per classificazione che per regressione;
- Abbastanza facile da interpretare, essendo formato da algoritmi molto semplici;

CONTRO
- Tempo di esecuzione (solo se rapportato all'albero decisionale, stiamo parlando comunque di una manciata di secondi)

TEST EFFETTUATI

MNIST

Dai grafici di accuratezza generati dall'esecuzione dello script per l'ottimizzazione degli Iper parametri notiamo che l'indice di accuratezza si stabilizza al raggiungimento di queste soglie:
- Popolazione: almeno 100 alberi
- Profondità albero: almeno 13

Eseguendo lo script per l'ottimizzazione degli iper parametri salta all'occhio che nelle prime 10 posizioni delle simulazioni effettuate con i migliori responsi in termini di accuratezza troviamo principalmente una sola profondità ottimale degli alberi che è quella limite ossia 21, tutavia ciò che varia sembra essere il numero di alberi che compongono la foresta.

Testiamo perciò il comportamento del classificatore con profondità per gli alberi: (21) e popolazione (100,120,140,160,180,200). Inoltre per un controllo eventuale dell' overfitting teniamo in considerazione per alcuni blocchi di test i valori 25 e 220 che sforano i range rispettivamente per profondità e popolazione.

1° Blocco di test (Aumento popolazione, profondità: 21, c: Gini, mss: 2)

Indice test	Popolazione massima	Profondità massima	Criterio	Min samples split	Acc Train	Acc Test	Differenza MSE	Over-fit
1	100	21	Gini	2	99,85%	96,84%	0,54	NO
2	120	21	Gini	2	99,85%	96,87%	0,55	NO
3	140	21	Gini	2	99,85%	96,88%	0,53	NO
4	160	21	Gini	2	99,85%	96,86%	0,54	NO
5	**180**	**21**	**Gini**	**2**	**99,86%**	**96,91%**	**0,55**	**NO**
6	200	21	Gini	2	99,85%	96,86%	0,55	NO

Da questi primi 6 test si evince che l'aumento della popolazione massima della foresta incide sul livello di accuratezza del set di test in modo molto lieve, mostrando un andamento di accuratezza in termini di percentuale altalenante ma che si mantiene in linea col valore del miglior test (**test 5**).
Da notare che mentre il tasso di accuratezza del set di test cresce all'aumentare della popolazione, l'accuratezza sul set di train rimane stabile.
Ciò che cambia è il tempo di addestramento del modello:

- 100 -> 37 sec
- 120 -> 46 sec
- 140 -> 53 sec
- 160 -> 62 sec
- 180 -> 68 sec
- 200 -> 74 sec

Adesso per i prossimi blocchi di test proviamo a cambiare il valore min samples split e vediamo il classificatore come si comporta.

2° Blocco di test (Aumento popolazione, profondità: 21, c: Gini, mss: 5)

Indice test	Popolazione massima	Profondità massima	Criterio	Min samples split	Acc Train	Acc Test	Differenza MSE	Over-fit
1	100	21	Gini	5	99,08%	96,25%	0,51	NO
2	120	21	Gini	5	99,08%	96,20%	0,53	NO
3	140	21	Gini	5	99,10%	96,33%	0,53	NO
4	160	21	Gini	5	99,11%	96,34%	0,51	NO
5	180	21	Gini	5	99,12%	96,37%	0,52	NO
6	**200**	**21**	**Gini**	**5**	**99,13%**	**96,46%**	**0,51**	**NO**

Per questi 7 test effettuati raccogliamo dei risultati analoghi al blocco precedente soltanto con valori in percentuale di accuratezza, sul set di test e sul set di train, leggermente più bassi.
Il **test 6** in questo caso è quello che fornisce i migliori risultati sia in termini di accuratezza che differenza di MSE. Un' altra considerazione che possiamo fare è quella relativa al tempo di addestramento del modello che come per il blocco precedente aumenta all'aumentare della popolazione di alberi della foresta, tuttavia il valore nuovo di **min_samples_split** lo rende più basso rispetto al blocco precedente anche se di pochi secondi:

- 100 -> 36 sec
- 120 -> 43 sec
- 140 -> 52 sec
- 160 -> 58 sec
- 180 -> 64 sec
- 200 -> 74 sec

3° Blocco di test (Aumento popolazione, profondità: 21, c: Gini, mss: 10)

Indice test	Popolazione massima	Profondità massima	Criterio	Min samples split	Acc Train	Acc Test	Differenza MSE	Over-fit
1	100	21	Gini	10	97,83%	95,86%	0,36	NO
2	120	21	Gini	10	97,82%	95,87%	0,36	NO
3	140	21	Gini	10	97,85%	95,93%	0,35	NO
4	160	21	Gini	10	97,88%	95,96%	0,35	NO
5	**180**	**21**	**Gini**	**10**	**97,88%**	**96,00%**	**0,37**	**NO**
6	**200**	**21**	**Gini**	**10**	**97,90%**	**96,00%**	**0,36**	**NO**

Per questi 6 test effettuati raccogliamo dei risultati analoghi ai due blocchi precedenti con valori in percentuale di accuratezza, sul set di test e sul set di train, ancora più bassi.
Il **test 5 insieme al test 6** in questo caso sono quelli che forniscono i migliori risultati in termini di accuratezza e in termini di differenza MSE (test 6).
Un' altra considerazione che possiamo fare è quella relativa al tempo di addestramento del modello che come per il blocco precedente aumentano all'aumentare della popolazione di alberi della foresta, tuttavia il valore nuovo di **min_samples_split** li rende più alti rispetto ai blocchi precedenti anche se di pochi secondi:

- 100 -> 34 sec
- 120 -> 51 sec
- 140 -> 60 sec
- 160 -> 68 sec
- 180 -> 76 sec
- 200 -> 85 sec

*Ora ripetiamo questi 3 blocchi di test cambiando il **criterio** da **Gini** -> **Entropy** e vediamo se otteniamo migliori risultati*

4° Blocco di test (Aumento popolazione, profondità: 21, c: Entropy, mss: 2)

Indice test	Popolazione massima	Profondità massima	Criterio	Min samples split	Acc Train	Acc Test	Differenza MSE	Over-fit
1	100	21	Entropy	2	99,94%	96,89%	0.57	NO
2	**120**	**21**	**Entropy**	**2**	**99,94%**	**96,93%**	**0,57**	**NO**
3	140	21	Entropy	2	99,94%	96,87%	0,59	NO
4	160	21	Entropy	2	99,94%	96,85%	0,58	NO
5	180	21	Entropy	2	99,95%	96,89%	0,58	NO
6	200	21	Entropy	2	99,95%	96,86%	0,60	NO

Da questi 6 test si può evincere di come il criterio Entropia faccia lievitare anche se di pochi punti percentuali i tassi di accuratezza sia sui set di test che di train, i quali mostrano valori altalenanti che raggiungono il loro massimo picco al **test 2** in cui oltre ad ottenere il maggior tasso di accuratezza sul set di test otteniamo anche la **più piccola differenza di MSE.**

Rispetto ai test del **1° blocco** i valori sono lievemente più alti.

- 100 -> 56 sec
- 120 -> 65 sec
- 140 -> 75 sec
- 160 -> 86 sec
- 180 -> 97 sec
- 200 -> 109 sec

5° Blocco di test (Aumento popolazione, profondità: 21, c: Entropy, mss: 5)

Indice test	Popolazione massima	Profondità massima	Criterio	Min samples split	Acc Train	Acc Test	Differenza MSE	Over-fit
1	100	21	Entropy	5	99,20%	96,33%	0,52	NO
2	120	21	Entropy	5	99,22%	96,45%	0,50	NO
3	140	21	Entropy	5	99,22%	96,50%	0,49	NO
4	160	21	Entropy	5	99,23%	96,53%	0,49	NO
5	**180**	**21**	**Entropy**	**5**	**99,22%**	**96,54%**	**0,50**	**NO**
6	200	21	Entropy	5	99,22%	96,48%	0,50	SI
7	220	25	Entropy	5	99,25%	96,49%	0,52	SI

Per questi 7 test effettuati raccogliamo dei risultati quasi analoghi al blocco corrispondente con **criterio Gini**, tuttavia in questo caso otteniamo dei valori di accuratezza sul set di test molto più alti, ma anche un avvicinamento all'overfitting cosa che non accadeva con il blocco di test analogo . Infatti in questi test effettuati notiamo che l'accuratezza sul set di test dopo aver raggiunto i migliori risultati nel **test 5** da una popolazione di 180 alberi in poi rende il modello meno efficiente, cosa che non avveniva affatto con il **2° Blocco** di test, in realtà con nessuno.
I tempi di addestramento del modello inoltre sono leggermente più alti:

- 100 -> 52 sec
- 120 -> 62 sec
- 140 -> 72 sec
- 160 -> 83 sec
- 180 -> 96 sec
- 200 -> 103 sec

6° Blocco di test (Aumento popolazione, profondità: 21, c: Entropy, mss: 10)

Indice test	Popolazione massima	Profondità massima	Criterio	Min samples split	Acc Train	Acc Test	Differenza MSE	Over-fit
1	100	21	Entropy	10	97,96%	95,97%	0,34	NO
2	120	21	Entropy	10	98,00%	95,96%	0,35	NO
3	**140**	**21**	**Entropy**	**10**	**98,03%**	**96,03%**	**0,37**	**NO**
4	160	21	Entropy	10	98,03%	95,99%	0,37	NO
5	180	21	Entropy	10	98,04%	96,02%	0,36	NO
6	**200**	**21**	**Entropy**	**10**	**98,05%**	**96,03%**	**0,37**	**NO**

Per questi 6 test notiamo un comportamento analogo con il **3° Blocco** di test con criterio **Gini** in cui all'aumentare della popolazione abbiamo un in concomitanza un aumento della percentuale di accuratezza sul set di test e di train il cui picco viene raggiunto sia al **test 3** che al **test 6** con valori identici sia per quanto riguarda la percentuale di accuratezza sul set di test che per differenza di MSE anche se il **test 3** ha una percentuale di accuratezza leggermente più bassa del set di train.

Per quanto riguarda i tempi di addestramento del modello essi sono in linea con quelli del blocco analogo.

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

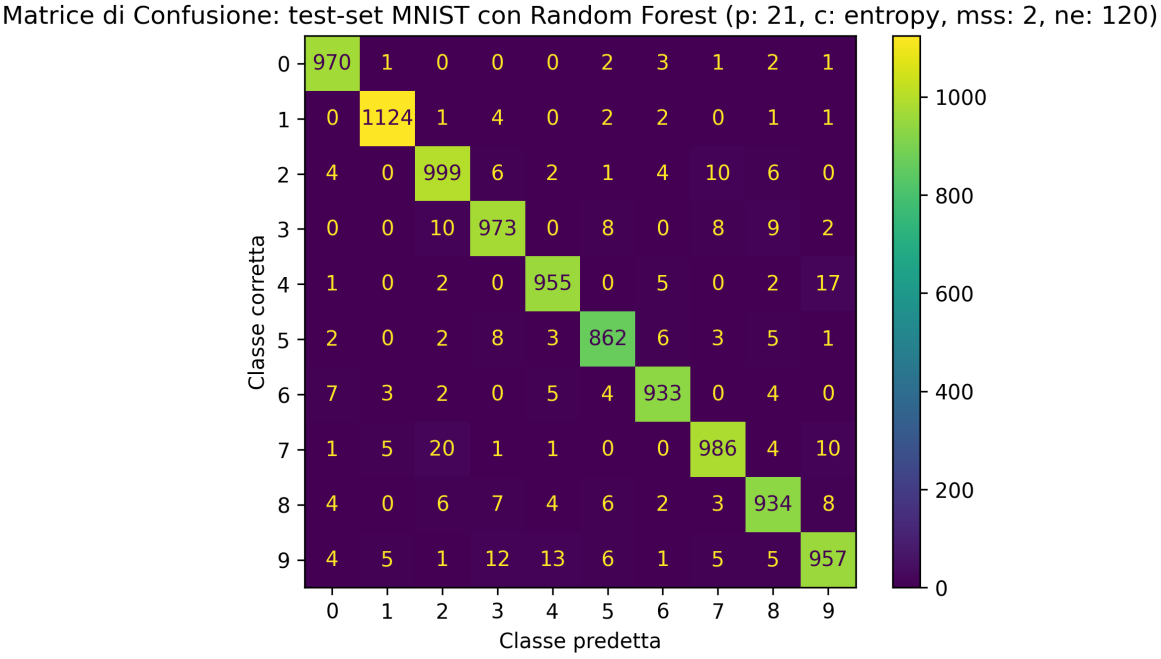
1) Blocco 4 - Test 2: Test del 2022-11-06 17:55:10
2) Blocco 1 - Test 5: Test del 2022-11-06 16:08:09
3) Blocco 5 - Test 5: Test del 2022-11-06 19:21:15

1) **Test del 2022-11-06 17:55:10 MNIST con Random Forest (profondità: 21, c: entropy, mss: 2, ne: 120)**

Report di classificazione

	precision	recall	f1-score	support
0	0.98	0.99	0.98	980
1	0.99	0.99	0.99	1135
2	0.96	0.97	0.96	1032
3	0.96	0.96	0.96	1010
4	0.97	0.97	0.97	982
5	0.97	0.97	0.97	892
6	0.98	0.97	0.97	958
7	0.97	0.96	0.96	1028
8	0.96	0.96	0.96	974
9	0.96	0.95	0.95	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	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 9 -> Cifra 9**



La matrice di confusione conferma ciò che viene mostrato nel report di classificazione la classe 1 che corrisponde alla **cifra 1** mostra su circa **1135** campioni **1124** vengono classificati correttamente. Analogamente la **classe 9** che corrisponde alle **cifra 9** è quella che ottiene i

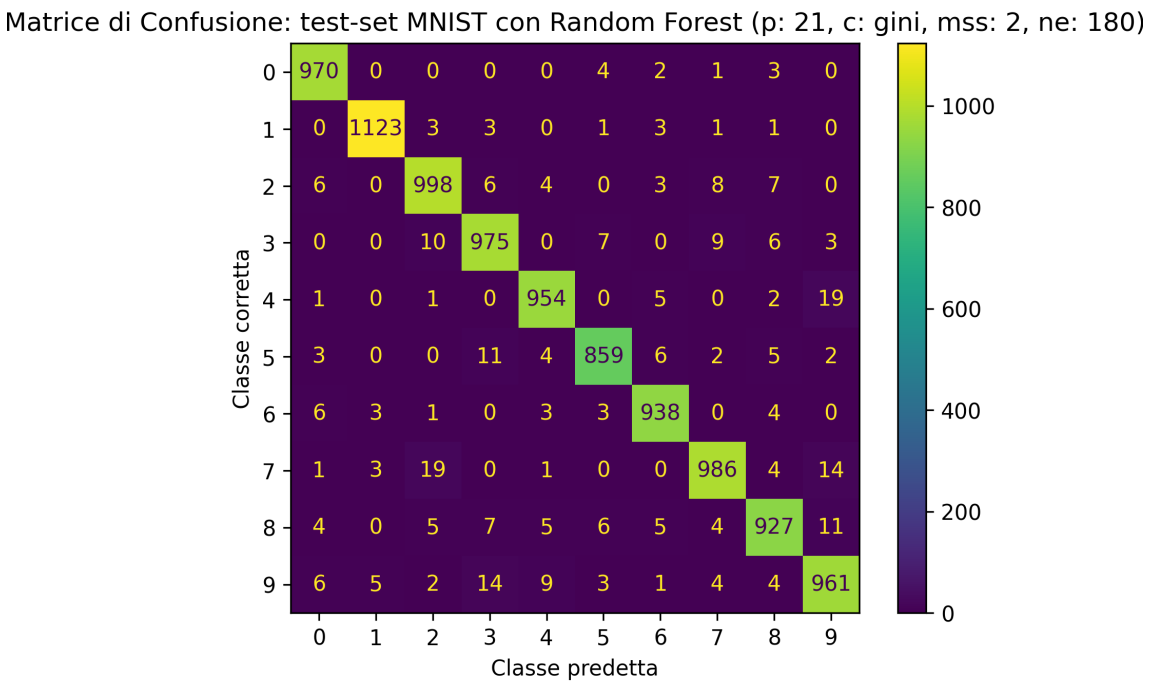
peggiori risultati infatti su **1009** campioni disponibili solo **957** vengono classificati correttamente. Chiaramente questo risultato è semplicemente il meno ottimale ma comunque un buonissimo risultato infatti la classe 9 ha un accuratezza di circa 96% che tra le altre è quella leggermente più bassa.

2) **Test del 2022-11-06 16:08:09 MNIST con Random Forest (profondità: 21, c: gini, mss: 2, ne: 180)**

Report di classificazione

	precision	recall	f1-score	support
0	0.97	0.99	0.98	980
1	0.99	0.99	0.99	1135
2	0.96	0.97	0.96	1032
3	0.96	0.97	0.96	1010
4	0.97	0.97	0.97	982
5	0.97	0.96	0.97	892
6	0.97	0.98	0.98	958
7	0.97	0.96	0.97	1028
8	0.96	0.95	0.96	974
9	0.95	0.95	0.95	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	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 9 -> Cifra 9**



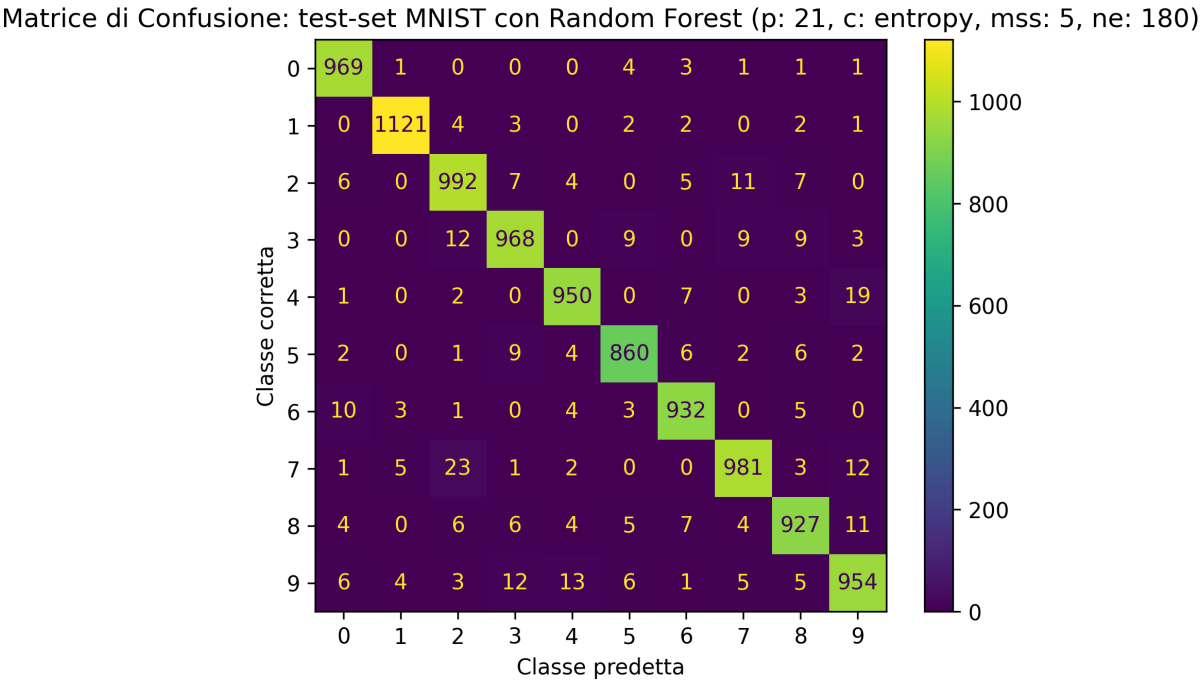
La matrice di confusione conferma ciò che viene mostrato nel report di classificazione la classe 1 che corrisponde alla **cifra 1** mostra su circa **1135** campioni **1123** vengono classificati correttamente. Analogamente la **classe 9** che corrisponde alle **cifra 9** è quella che ottiene i peggiori risultati infatti su **1009** campioni disponibili solo **927** vengono classificati correttamente. Chiaramente questo risultato è semplicemente il meno ottimale ma comunque un buonissimo risultato infatti la classe 9 ha un accuratezza di circa 95% che tra le altre è quella leggermente più bassa.
Guardando la matrice di confusione non facciamoci tranne in inganno dai valori ottenuti dalla **classe 5, che sicuramente mostra il numero di campioni predetti correttamente più basso ma ha anche il numero di campioni disponibili più basso di tutti ossia 892.**
Per questo motivo avendo soltanto **892** campioni disponibili e di questi **859** vengono classificati correttamente sono sintomo di un ottimo risultato, infatti il report parla di una percentuale di accuratezza sulla **classe 5** di circa il **97%.**

3) **Test del 2022-11-06 19:21:15 MNIST con Random Forest (profondità: 21, c: entropy, mss: 5, ne: 180)**

Report di classificazione

	precision	recall	f1-score	support
0	0.97	0.99	0.98	980
1	0.99	0.99	0.99	1135
2	0.95	0.96	0.96	1032
3	0.96	0.96	0.96	1010
4	0.97	0.97	0.97	982
5	0.97	0.96	0.97	892
6	0.97	0.97	0.97	958
7	0.97	0.95	0.96	1028
8	0.96	0.95	0.95	974
9	0.95	0.95	0.95	1009
accuracy			0.97	10000
macro avg	0.97	0.97	0.97	10000
weighted avg	0.97	0.97	0.97	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 9 -> Cifra 9**



La matrice di confusione conferma ciò che viene mostrato nel report di classificazione la classe 1 che corrisponde alla **cifra 1** mostra su circa **1135** campioni **1121** vengono classificati correttamente. Analogamente la **classe 9** che corrisponde alle **cifra 9** è quella che ottiene i peggiori risultati infatti su **1009** campioni disponibili solo **954** vengono classificati correttamente. Chiaramente questo risultato è semplicemente il meno ottimale ma comunque un buonissimo risultato infatti la classe 9 ha un accuratezza di circa 95% che tra le altre è quella leggermente più bassa.
Guardando la matrice di confusione non facciamoci tranne in inganno dai valori ottenuti dalla **classe 5, che sicuramente mostra il numero di campioni predetti correttamente più basso ma ha anche il numero di campioni disponibili più basso di tutti ossia 892.**
Per questo motivo avendo soltanto **892** campioni disponibili e di questi **860** vengono classificati correttamente sono sintomo di un ottimo risultato, infatti il report parla di una percentuale di accuratezza sulla **classe 5** di circa il **97%.**