Documentazione Caso di Studio Ingegneria della Conoscenza LAPTOP CLASSIFICATION

Gruppo di lavoro

Francesco Dinapoli, 726267, f.dinapoli5@studenti.uniba.it

Repository GitHub

https://github.com/Francesco-din/IconProg

AA 2022-23

Sommario

Introduzione

Il software per la classificazione dei PC in tipologie mira ad assegnare automaticamente una tipologia specifica a ciascun PC in base alle sue caratteristiche, semplificando l'identificazione degli stessi.

Preprocessing

Strumenti utilizzati

Il Dataset è composta da un file csv, al fine di manipolare e leggere i dati è stata utilizzata la libreria "*pandas*"

Decisioni di progetto

Il dataset iniziale "laptop_price.cvs" contiene informazioni relative a diversi PC.

Per ogni PC sono presenti le seguenti informazioni:

- un ID identificativo all'interno del dataset (laptop_ID)
- la compagnia produttrice del PC (Company)
- il nome del modello (Product)
- il tipo di PC (TypeName)
- la dimensione dello schermo in pollici (Inches)
- la risoluzione dello schermo (ScreenResolution)
- il nome della cpu (Cpu)
- la grandezza della ram (Ram)
- la quantità di memoria (Memory)
- il nome della gpu (Gpu)
- il sistema operativo nativo (OpSys)
- il peso del PC (Weight)
- il prezzo in euro (Price_euros)

È stata formattata la colonna "Price_euros" andando a rendere gli elementi contenuti nel formato corretto per il tipo di dato.

Nella colonna denominata "Product" erano presenti informazioni aggiuntive oltre al nome del prodotto e sono stati eliminate.

57	HP	15-bs002nv (i3-6006U/4GB/128GB/FHD/W10)	I
58	Asus	VivoBook Max	I
59	MSI	GS73VR 7RG	(
60	Asus	X541UA-DM1897 (i3-6006U/4GB/256GB/FHD/Linux)	I
61	Dell	Inspiron 5770	I
62	Dell	Vostro 5471	I
63	Lenovo	IdeaPad 520S-14IKB	I
64	Asus	UX410UA-GV350T (i5-8250U/8GB/256GB/FHD/W10)	I

57	HP	15-bs002nv
58	Asus	VivoBook Max
59	MSI	GS73VR 7RG
60	Asus	X541UA-DM1897
61	Dell	Inspiron 5770
62	Dell	Vostro 5471
63	Lenovo	IdeaPad 520S-14IKB
64	Asus	UX410UA-GV350T

È stata poi modificata la colonna “ScreenResolution” andando mantenere solo la risoluzione numerica e andando a creare una nuova colonna chiamata “TouchScreen” che presenta il valore 0 se il computer non è touchscreen, e il valore 1 in caso in cui lo sia.

All’interno delle colonne “Weight, Ram, Price_euros” sono state rimosse le unità di misura.

È stata modificata la colonna “OpSys” andando a sostituire i nomi dei sistemi operativi con un numero identificativo, ogni numero corrisponde ad un sistema operativo seguendo lo schema sottostante:

1: macOS 2: No OS 3: Windows 10 4: Mac OS X 5: Linux

6: Android 7: Windows 10 S 8: Chrome OS 9: Windows 7

Tutte queste modifiche sono state salvate in un nuovo file con il nome di “*newLaptopPrice.csv*”

Knowledge Base

Una knowledge base è una raccolta di informazioni organizzata in modo tale che un programma o un sistema possa utilizzarle per poter rispondere a domande o prendere delle decisioni.

Strumenti utilizzati

La knowledge base è stata creata utilizzando la libreria “*pyswip”* la quale permette di creare basi di conoscenza, interrogarle e creare fatti e regole per manipolare conoscenze a SWI-Prolog programmando in Python.

Fatti

In questa sezione sono analizzati i fatti che sono stati creati nel file “*Knowledgebase.py*” seguendo lo schema sottostante:

prop(identificatore, proprietà, valore)

di seguito sono presentati alcuni esempi, nella prima immagine è possibile vedere come i fatti sono stati implementati, nella seconda è possibile vedere come risultano (nel file *kb.pl*).

```
prolog_file.write(f"prop({row['laptop_ID']}, 'Dimensioni schermo', {row['Inches']}).\n")
prolog_file.write(f"prop({row['laptop_ID']}, 'Cpu', '{row['Cpu']}').\n")
prolog_file.write(f"prop({row['laptop_ID']}, 'Ram', '{row['Ram']}').\n")
prolog_file.write(f"prop({row['laptop_ID']}, 'Memoria', '{row['Memory']}').\n")
prolog_file.write(f"prop({row['laptop_ID']}, 'Sistema Operativo', {row['OpSys']}).\n")
prolog_file.write(f"prop({row['laptop_ID']}, 'Peso', {row['Weight']}).\n")
prolog_file.write(f"prop({row['laptop_ID']}, 'Prezzo (euro)', {row['Price_euros']}).\n")
prolog_file.write(f"prop({row['laptop_ID']}, 'Touchscreen', {row['TouchScreen']}).\n")
prolog_file.write(f"prop({row['laptop_ID']}, 'tipo', '{row['TypeName']}').\n")
```

```
prop(1, 'Peso', 1.37).
prop(1, 'Prezzo (euro)', 1340.09).
prop(1, 'Touchscreen', 0).
prop(1, 'tipo', 'Ultrabook').
prop(2,'larghezza Risoluzione',1440).
prop(2,'altezza Risoluzione',900).
prop(2, 'Modello', 'Macbook Air').
prop(2, 'Marca', 'Apple').
prop(2, 'Dimensioni schermo', 13.3).
prop(2, 'Cpu', 'Intel Core i5 1.8GHz').
prop(2, 'Ram', '8').
prop(2, 'Memoria', '128GB Flash Storage').
prop(2, 'Sistema Operativo', 1).
```

Regole

In seguito la base di conoscenza è stata arricchita con diverse regole, aggiungendole nel file "*rules.pl*", le regole aggiunte sono le seguenti:

- pcPesante(ID,Peso) :- prop(ID, 'Peso', Peso), Peso >= 3.0.
- pcLeggero(ID,Peso) :- prop(ID, 'Peso', Peso), Peso =< 1.0.
- pcEconomico(ID,Prezzo) :- prop(ID, 'Prezzo (euro)', Prezzo), Prezzo =< 300.
- pcPrezzoMedio(ID,Prezzo) :- prop(ID, 'Prezzo (euro)', Prezzo), (Prezzo =< 300,Prezzo>=1500).
- pcCostoso(ID,prezzo) :- prop(ID, 'Prezzo (euro)', Prezzo), Prezzo >= 2000.
- ris_4k(ID) :- prop(ID, 'larghezza Risoluzione', 3840), prop(ID, 'altezza Risoluzione', 2160).
- tot_pc(Tot) :- findall(ID, prop(ID, 'larghezza Risoluzione' , _), PCList), length(PCList, Tot).
- count_qk(Num) :- findall(ID, ris_4k(ID), QKList), length(QKList, Num).
- perc_4k(Perc) :- tot_pc(Tot), findall(ID, ris_4k(ID), QKList), length(QKList, QKCount),

  Perc is (QKCount / Tot) * 100.
- max_Price(Prezzo) :- findall(Val, prop(_,'Prezzo (euro)',Val), Prlist), max_list(Prlist, Prezzo).
- min_Price(Prezzo) :- findall(Val, prop(_,'Prezzo (euro)',Val), Prlist), min_list(Prlist, Prezzo).
- pc_apple_schermo_piu_piccolo(ID, MinDimensioni) :- prop(ID, 'Marca', 'Apple'),

  findall(Dimensioni, prop(ID, 'Dimensioni schermo', Dimensioni), ListaDimensioni),

  min_list(ListaDimensioni, MinDimensioni), prop(ID, 'Dimensioni schermo',

  MinDimensioni), \+ (prop(AltroID, 'Marca', 'Apple'), AltroID \= ID, prop(AltroID,

  'Dimensioni schermo', AltroDimensioni), AltroDimensioni < MinDimensioni).

- perc_ultrabook(Perc) :- tot_pc(Tot), findall(_, prop(_, 'tipo', 'Ultrabook'),UltList),

  length(UltList, Num), Perc is (Num / Tot) * 100.

- count_lenovo(Num) :- findall(_, prop(_,'Marca','Lenovo'), LenList), length(LenList, Num).

- pcApple_MacOSX(ID) :- prop(ID, 'Marca', 'Apple'), prop(ID, 'Sistema Operativo', 4).

Il significato di queste regole è facilmente intuibile dal nome di esse.
All'intero delle regole sono stati utilizzati alcuni predicati incorporati come:

- findall(template, goal, list) crea una lista di istanze di template che soddisfano il goal e la restituisce.
- length(list, lunghezza) restituisce la lunghezza della lista "list".
- min_list(list, minimo) restituisce l'elemento più piccolo della lista "list".
- max_list(list, massimo) restituisce l'elemento maggiore della lista "list".
- " \+ " viene usato in Prolog per negare il successo di un predicato.

Per quanto riguarda l'interrogazione alla base di conoscenza essa viene fatta nel file "*kbul.py*" e di seguito sono presentati alcuni esempi di interrogazione.

```
file.write('-' * 50 + '\n')
file.write("I seguenti pc hanno Prezzo inferiore a 300 euro:\n")
for result in prolog.query("pcEconomico(ID, Prezzo)"):
    file.write(f"pcID: {result['ID']}, prezzo: {result['Prezzo']} euro\n")
```

```
file.write('-' * 50 + '\n')
for result in prolog.query("perc_ultrabook(Perc)"):
   file.write("percentuale di Ultrabook presente: {:.2f} % \n".format(result['Perc']))
```

```
file.write('-' * 50 + '\n')
for result in prolog.query("pcApple_MacOSX(ID)"):
    file.write(f"pc Apple con MacOSX: {result['ID']} \n")
```

Nel file "*output.txt*" è possibile vedere il risultato di tali interrogazioni.

Apprendimento supervisionato

Preprocessing

Prima di iniziare la parte di apprendimento è stata effettuata un ulteriore fase di preprocessing in quanto i modelli applicati richiedevano che le colonne non contenessero stringhe, quindi è stata applicata la funzione "fit_transform" per trasformare le colonne "TypeName, Company, Product, Cpu, ScreenResolution, Memory, Gpu".

Strumenti utilizzati

Per la realizzazione dell'apprendimento supervisionato sono stati impiegati diversi modelli di classificazione presi dalla libreria "*Scikit Learn*", ovvero:

- KNN, tramite la classe KNeighborsClassifier.
- DecisionTree, tramite la classe DecisionTreeClassifier.
- Random Forest, tramite la classe RandomForestClassifier.
- AdaBoostClassifier, tramite la classe omonima.
- GradientBoostClassifier, tramite la classe omonima.

Per quanto riguarda la gestione del dataset è stato utilizzata la libreria "*Pandas*" .

Per utilizzare la tecnica "Grid_Search" è stata utilizzata la classe "GridSearchCV" della libreria "*Scikit Learn*".

Per l'utilizzo della tecnica "k-fold cross-validation" è stata utilizzata la classe "KFold" della libreria "*Scikit Learn*".

Modelli

Tutti i modelli sono stati valutati, nel file "*classifier.py*", tramite:

- Precison (misura quanto sia preciso un classificatore facendo il rapporto tra veri positivi e la somma tra veri positivi e falsi positivi)
- Recall (dato dal rapporto tra i veri positivi e la somma di veri positivi e falsi negativi)
- F1 score (media armonica di precision e recall, utile per trovare un equilibro tra entrambi)
- Macro Avg (medi aritmetica delle metriche, assegna lo stesso peso ad ogni classe indipendentemente dalla dimensione)
- Weighted Avg (media ponderate delle metriche in cui le classi con maggior supporto hanno peso maggiore)
- Support (numero totale di campioni di ciascuna classe nel dataset)
- Confusion Matrix (strumento utilizzato per valutare le prestazione di un modello di classificazione che mostra il numero di veri positivi, falsi positivi, veri negativi e falsi negativi)

Su tutti i modelli scelti è stata effettuata, nel file "*Kfold_grid.py*", la "K-fold-CrossValidation" e la tecnica "Grid_Search" per ridurre il rischio di overfitting o underfitting e per selezionare gli iperparametri migliori per addestrare ogni modello.

Knn

Attraverso la "Grid_Search" sono state testate le diverse combinazione dei seguenti parametri:

- n_neighbors : ovvero il numero di vicini da considerare quando si effettua una previsione, i valori considerati sono (3, 4, 5, 6)
- metric : specifica la metrica per misurare la distanza tra i punti, le opzioni prese in considerazione sono ('euclidean', 'manhattan', 'minkowski')

la miglior combinazione risultante è stata “metric = ‘manhattan’ e n_ neighbors = 5 ”.
i risultati ottenuti sono i seguenti:

```
KNN Classification Report:
              precision    recall  f1-score   support

           0       0.67      0.20      0.31        30
           1       0.53      0.59      0.56        39
           2       0.00      0.00      0.00         1
           3       0.76      0.86      0.81       153
           4       0.45      0.42      0.44        33
           5       1.00      0.40      0.57         5

    accuracy                           0.68       261
   macro avg       0.57      0.41      0.45       261
weighted avg       0.68      0.68      0.66       261

F2 Score: 0.6781609195402298

Confusion Matrix :
[[  6   3   0  18   3   0]
 [  0  23   0  10   6   0]
 [  0   0   0   1   0   0]
 [  1  11   2 132   7   0]
 [  2   5   1  11  14   0]
 [  0   1   0   1   1   2]]
```

Osservando l’accuratezza pari a 0.68 possiamo capire che questo modello non mostra una capacità di classificazione buona per il nostro scopo, infatti analizzando più nello specifico possiamo notare come sono presenti classi aventi prestazione buone (la classe ‘3’), mentre ci sono altre classi con precision e recall aventi valori bassi (classi ‘0’ e ‘2’).
Osservando la matrice di confusione salta subito all’occhio che con questo modello :

- la classe ‘0’ ha classificato correttamente 6 istanze , ha fatto 3 falsi positivi e 18 falsi negativi, quindi potrebbe risultare difficile da predire correttamente
- la classe ‘1’ ha classificato correttamente 23 istanze, facendo 10 falsi positivi e 6 falsi negativi
- la classe ‘2’ non ha classificato nessuna istanza e ha fatto solo un falso negativo (da notare che il support è pari ad 1)
- la classe ‘3’ ha buone prestazioni avendo trovato 132 veri positivi e 7 falsi negativi
- la classe ‘4’ ha classificato correttamente 14 istanze facendo 5 falsi positivi e 11 falsi negativi
- la classe ‘5’ ha classificato correttamente 2 istanze della classe 5, ma ha fatto 1 falso positivo e 1 falso negativo (anche questa classe ha un supporto basso)

Decision Tree

Attraverso la “Grid_Search” sono state testate le diverse combinazione dei seguenti parametri:

- criterion : Indica la funzione di misura della qualità di una divisione, i valori considerati sono (‘gini’, ‘entropy’)
- max_depth : indica la massima profondità dell’albero decisionale, i valori considerati sono (none, 5, 10, 15)
- min_samples_split : specifica il numero minimo di campioni richiesti per suddividere un nodo interno, i valori considerati sono (2, 5, 10)
- min_samples_leaf : indica il numero minimo di campioni richiesti in una foglia, i valori considerati sono (1, 2, 4)

la miglior combinazione risultante è stata “criterion= 'gini', max_depth= 10, min_samples_leaf= 4, min_samples_split= 2”.
i risultati ottenuti sono i seguenti:

```
Decision Tree Classification Report:
              precision    recall  f1-score   support

           0       0.88      0.73      0.80        30
           1       0.88      0.97      0.93        39
           2       0.33      1.00      0.50         1
           3       0.92      0.92      0.92       153
           4       0.71      0.73      0.72        33
           5       1.00      0.80      0.89         5

    accuracy                           0.88       261
   macro avg       0.79      0.86      0.79       261
weighted avg       0.88      0.88      0.88       261

F2 Score: 0.8773946360153256

Confusion Matrix :
[[ 22   1   0   5   2   0]
 [  0  38   0   1   0   0]
 [  0   0   1   0   0   0]
 [  1   3   1 140   8   0]
 [  2   0   1   6  24   0]
 [  0   1   0   0   0   4]]
```

Osservando questi dati possiamo notare che il modello ha prestazioni migliori rispetto al KNN, dal valore dell’ F2 score possiamo capire che il modello sta prestando particolare attenzione al recall (ovvero alla capacità di trovare correttamente esempi positivi).
Osservando la matrice di confusione salta subito all’occhio che con questo modello:

- la classe ’0’ ha classificato correttamente 22 istanze, facendo 1 falso positivo e 5 falsi negativi
- la classe ‘1’ ha classificato correttamente 38 istanze senza falsi positivi o falsi negativi
- la classe ‘2’ ha classificato correttamente 1 istanza
- la classe ‘3’ ha classificato correttamente 140 istanze, con 1 falso positivo e 3 falsi negativi
- la classe ‘4’ ha classificato correttamente 24 istanze facendo 2 falsi positivi e 6 falsi negativi
- la classe ‘5’ ha classificato correttamente 4 istanze con un falso positivo e un falso negativo

Random forest classifier

Attraverso la “Grid_Search” sono state testate le diverse combinazione dei seguenti parametri:

- n_estimators : Indica il numero di alberi decisionali da utilizzare nel modello, i valori considerati sono (50, 100, 150)
- max_depth : indica la massima profondità di ciascun albero decisionale nel modello, i valori considerati sono (none, 10, 20)
- min_samples_split : specifica il numero minimo di campioni richiesti per suddividere un nodo interno di ciascun albero, i valori considerati sono (2, 5, 10)
- min_samples_leaf : indica il numero minimo di campioni richiesti in una foglia di ciascun albero, i valori considerati sono (1, 2, 4)
- criterion : Indica la funzione di misura della qualità di una divisione, i valori considerati sono (‘gini’, ‘entropy’)
- max_features indica il numero massimo di caratteristiche da considerare per la suddivisione di un nodo, i valori considerati sono (‘sqrt’, ‘log2’, none)

la miglior combinazione risultante è stata “criterion = 'entropy', max_depth = None, max_features = 'sqrt', min_samples_leaf = 1, min_samples_split = 2, n_estimators = 50”.
i risultati ottenuti sono i seguenti:

```
Random Forest Classification Report:
              precision    recall  f1-score   support

           0       0.95      0.70      0.81        30
           1       0.91      1.00      0.95        39
           2       0.33      1.00      0.50         1
           3       0.90      0.94      0.92       153
           4       0.85      0.70      0.77        33
           5       0.83      1.00      0.91         5

    accuracy                           0.89       261
   macro avg       0.80      0.89      0.81       261
weighted avg       0.90      0.89      0.89       261

F2 Score: 0.89272030651341
Confusion Matrix :
[[ 21   1   0   7   1   0]
 [  0  39   0   0   0   0]
 [  0   0   1   0   0   0]
 [  1   3   1 144   3   1]
 [  0   0   1   9  23   0]
 [  0   0   0   0   0   5]]
```

Osservando questi dati notiamo che questo modello ha prestazioni maggiori rispetto al Knn e simili al modello precedente (Decision Tree) con il quale ha molti dati quasi in comune, possiamo notare però alcune differenze:

- per le classi ‘0’, ‘1’ e ‘4’ questo modello ha una precision e un recall maggiore
- l’F1 score di questo modello è leggermente più elevato ciò indica che ha una migliore combinazione di precision e recall

possiamo notare una similitudine tra i due modelli anche osservando la matrice di confusione, grazie alla quale notiamo quasi gli stessi valori visualizzati nel modello precedente.

Ada Boost Classifier

Attraverso la “Grid_Search” sono state testate le diverse combinazione dei seguenti parametri:

- n_estimators : Indica il numero di modelli di base, i valori considerati sono (50, 100, 200)
- learning_rate : rappresenta la quantità con cui contribuiscono i singoli modelli di base al modello, i valori considerati sono (0.1, 0.5, 1.0)
- algorithm : specifica l’algoritmo utilizzato per il boosting, i valori considerati sono (‘SAMME’, ‘SAMME.R’)
- estimator : indica il tipo di stimatore di base utilizzato per il modello, i valori considerati sono (‘DecisionTreeClassifier(max_depth=4)’, ‘DecisionTreeClassifier(max_depth=5)’, ‘DecisionTreeClassifier(max_depth=6)’, ‘DecisionTreeClassifier(max_depth=7)’)
- random_state : fornisce un seme per la riproducibilità dell’addestramento, è stato posto a ‘42’

la miglior combinazione risultante è stata “estimator = ‘DecisionTreeClassifier(max_depth=7)’, algorithm ='SAMME', learning_rate = 1.0, n_estimators = 100”.

i risultati ottenuti sono i seguenti:

```
Ada Boost Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.80      0.87        30
           1       0.95      0.95      0.95        39
           2       0.25      1.00      0.40         1
           3       0.89      0.92      0.91       153
           4       0.76      0.67      0.71        33
           5       0.83      1.00      0.91         5

    accuracy                           0.88       261
   macro avg       0.77      0.89      0.79       261
weighted avg       0.89      0.88      0.88       261

F2 Score: 0.8812260536398467
Confusion Matrix :
[[ 24   0   0   6   0   0]
 [  0  37   0   2   0   0]
 [  0   0   1   0   0   0]
 [  1   2   1 141   7   1]
 [  0   0   2   9  22   0]
 [  0   0   0   0   0   5]]
```

Osservando i dati possiamo notare che questo modello mostra precision e recall superiori agli altri modelli per molte classi, avendo un accuracy uguale al DecisionTree ma di poco inferiore al Random forest.
Osservando la matrice di confusione si può notare che:

- la classe ’0’ ha classificato correttamente 24 istanze, senza avere falsi positivi o negativi
- la classe ‘1’ ha classificato correttamente 37 istanze, con 2 falsi negativi
- la classe ‘2’ ha classificato correttamente 1 istanza
- la classe ‘3’ ha classificato correttamente 141 istanze, con 7 falsi positivi e 2 falsi negativi
- la classe ‘4’ ha classificato correttamente 22 istanze facendo 2 falsi positivi e 8 falsi negativi
- la classe ‘5’ ha classificato correttamente 5 istanze.

Gradient Boosting Classifier

Attraverso la “Grid_Search” sono state testate le diverse combinazione dei seguenti parametri:

- n_estimators : Indica il numero di modelli di base, i valori considerati sono (50, 100, 150)
- learning_rate : rappresenta il tasso di apprendimento che controlla la dimensione dei passi che vengono fatti durante l'ottimizzazione, i valori considerati sono (0.01, 0.1, 0.2)
- max_depth : indica la massima profondità di ciascun albero decisionale nel modello, i valori considerati sono (3, 4, 5)
- min_samples_split : specifica il numero minimo di campioni richiesti per suddividere un nodo interno di ciascun albero decisionale debole, i valori considerati sono (2, 5, 10)
- min_samples_leaf : indica il numero minimo di campioni richiesti in una foglia di ciascun albero decisionale debole, i valori considerati sono (1, 2, 4)

la miglior combinazione risultante è stata “learning_rate = 0.1, max_depth = 5, min_samples_leaf = 4, min_samples_split = 5, n_estimators = 100”.

i risultati ottenuti sono i seguenti:

```
Gradient Boosting Classifier Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.83      0.89        30
           1       0.88      0.97      0.93        39
           2       0.25      1.00      0.40         1
           3       0.93      0.92      0.92       153
           4       0.80      0.73      0.76        33
           5       0.83      1.00      0.91         5

    accuracy                           0.90       261
   macro avg       0.78      0.91      0.80       261
weighted avg       0.90      0.90      0.90       261

F2 Score: 0.896551724137931
Confusion Matrix :
[[ 25   1   0   3   1   0]
 [  0  38   0   1   0   0]
 [  0   0   1   0   0   0]
 [  1   4   1 141   5   1]
 [  0   0   2   7  24   0]
 [  0   0   0   0   0   5]]
```

Osservando i dati si può notare che l'accuracy è pari a 0.90 che è il valore più tra quelli registrati nei modelli valutati fino ad ora, questo indica che tra tutti questo modello è il più accurato.
Osservando l'F2 score anch'esso il più alto tra tutti si può capire che questo modello ha una buona capacità di individuare correttamente i veri postivi.
Osservando La matrice di confusione si può notare che:

- la classe '0' ha classificato correttamente 25 istanze, con 1 falso positivo e 4 falsi negativi
- la classe '1' ha classificato correttamente 39 istanze
- la classe '2' ha classificato correttamente 1 istanza
- la classe '3' ha classificato correttamente 141 istanze, con 1 falso positivo e 7 falsi negativi
- la classe '4' ha classificato correttamente 22 istanze facendo 2 falsi positivi e 8 falsi negativi
- la classe '5' ha classificato correttamente 5 istanze.

Conclusioni finali

Osservando i risultati di tutti i modelli e prendendo in considerazione l'accuracy possiamo dire che ai fini del progetto il modello che ha avuto prestazioni migliori è, seppur di poco, il Gradient Boost Classifier con un valore di 0.90; anche tenendo conto delle altre metriche notiamo dei valori quasi sempre maggiori per questa metrica.

Possibili miglioramenti alla classificazione si potrebbero avere ampliando il dataset di partenza aggiungendo altri elementi appartenenti ad alcune classi che ne presentano pochi come la classe '2' e '5'.