Esame di Machine Learning and Data Mining

Documento di progetto

Page Blocks Classification Dataset

Elisa Spigarelli

Matricola: 327125

Anno Accademico 2019/2020

Sommario

Capitolo 1

Descrizione del Problema

Il Dataset *Page Blocks Classification* proviene da una ricerca condotta da Donato Malerba del Dipartimento di Informatica dell'Università di Bari ed è stato utilizzato inizialmente per provare differenti metodi di semplificazione per gli alberi di decisione.
Il problema consiste nella classificazione di tutti i blocchi del layout di una pagina di un documento, ottenuti attraverso un processo di segmentazione.
Questo problema è un passo essenziale nell'analisi di documenti che permette di separare il testo dalla parte grafica. In particolare, la classificazione avviene attraverso le seguenti 5 classi:

(1) testo
(2) linee orizzontali
(3) immagini
(4) linee verticali
(5) grafici

Capitolo 2

Descrizione del Dataset

Il Dataset si compone di 5473 campioni (*samples*) che provengono da 54 documenti differenti. Ogni osservazione riguarda un blocco. Per descrivere i campioni sono utilizzati 10 attributi di tipo numerico che esprimono le caratteristiche dei blocchi:

Features	*Tipo dell'attributo*	*Significato*
height	numero intero	Altezza del blocco
lenght	numero intero	Lunghezza del blocco
area	numero intero	Area del blocco (height* length)
eccen	numero continuo	Eccentricità del blocco (height/length)
p_black	numero continuo	Percentuale di pixel neri all'interno del blocco (blackpix/area)
p_and	numero continuo	Percentuale di pixel neri dopo l'applicazione del Run Length Smoothing Algorithm (RLSA) (blackand/area);
mean_tr	numero continuo	Numero medio di transizioni bianco-nero (blackpix/wb_trans)
blackpix	numero intero	Numero totale di pixel neri nella bitmap originale del blocco
blackand	numero intero	Numero totale di pixel neri nella bitmap del blocco dopo l'RLSA
wb_trans	numero intero	Numero di transizioni bianco-nero nella bitmap originale del blocco

Il Dataset è accompagnato da una documentazione che descrive caratteristiche e proprietà e informa che i dati sono stati controllati in modo da non presentare attributi nulli. Le proprietà statistiche relative alle differenti features sono riassunte nella tabella:

Features	*Mean*	*Std Dev*	*Minimum*	*Maximum*	*Correlation*
height	10.47	18.96	1	804	.3510
lenght	89.57	114.72	1	553	-.0045
area	1198.41	4849.38	7	143993	.2343
eccen	13.75	30.70	.007	537.00	.0992
p_black	.37	.18	.052	1.00	.2130
p_and	.79	.17	.062	1.00	-.1771
mean_tr	6.22	69.08	1.00	4955.00	.0723
blackpix	365.93	1270.33	7	33017	.1656
blackand	741.11	1881.50	7	46133	.1565
wb_trans	106.66	167.31	1	3212	.0337

Inoltre nel Dataset è presente un ulteriore attributo **Class**, che descrive la classe a cui appartiene il blocco (assume un valore da 1 a 5).
La distribuzione delle classi, ovvero della variabile target, è schematizzata nella tabella:

Class	*Frequenza*	*Percentuale*
(1) text	4913	89.8 %
(2) horiz. line	329	6.0 %
(3) graphic	28	0.5 %
(4) vert. line	88	1.6 %
(5) picture	115	2.1 %
TOTALE	5473	100.0 %

Capitolo 3

Exploratory Data Analysis

Per risolvere il problema di classificazione innanzitutto è necessario eseguire l'analisi del Dataset che viene gestito sfruttando la libreria *Pandas*.
Nello specifico, il Dataset viene convertito in un oggetto *DataFrame* con 5473 righe e 11 colonne che vengono rinominate con il relativo nome dell'attributo che rappresentano.
Attraverso i metodi offerti dalla libreria si visualizzano le caratteristiche dell'oggetto e si verificano le informazioni di base del dataset riportate nella documentazione associata descritta nel capitolo precedente.
L'immagine seguente, attraverso un istogramma, visualizza la distribuzione dei dati del dataset nelle diverse classi.

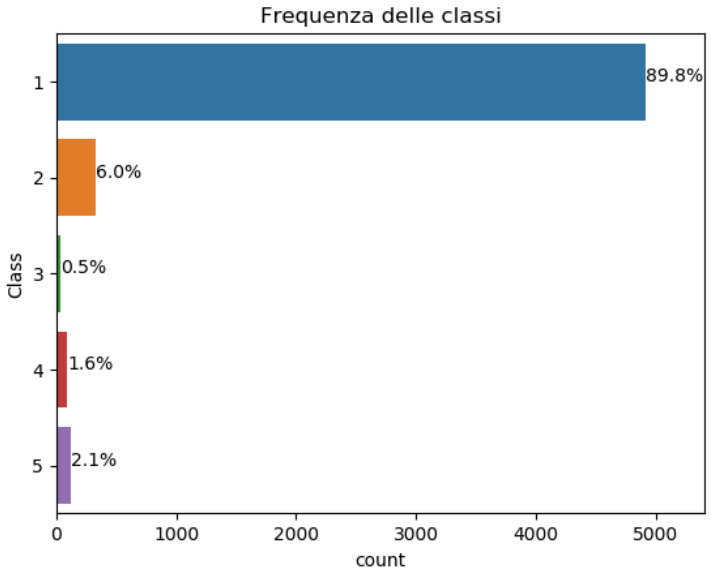

Fig 1. Distribuzione dei samples del dataset nelle classi

Come già preannunciato, non tutte le classi contengono lo stesso numero di samples; ad esempio la classe *(1) text* contiene molti più campioni della classe *(3) graphic*.
Il caso in questione è perciò un problema di classificazione multiclasse con classi sbilanciate. Questa osservazione è fondamentale per la scelta degli algoritmi e delle metriche di valutazione ad essi correlate che devono essere applicati.
Per poter iniziare l'elaborazione, il dataset viene diviso in dati di input e di output: le prime 10 colonne costituiscono la matrice delle features, detta *Design matrix*, mentre l'ultima colonna, denominata *Class*, il vettore delle etichette che rappresenta la classe di appartenenza di ciascun dato in input.
Per ottenere maggiori informazioni sugli attributi che descrivono i dati si analizza la Design matrix, rappresentando attraverso il grafico boxplot le caratteristiche statistiche e i range di appartenenza di ciascuna features.

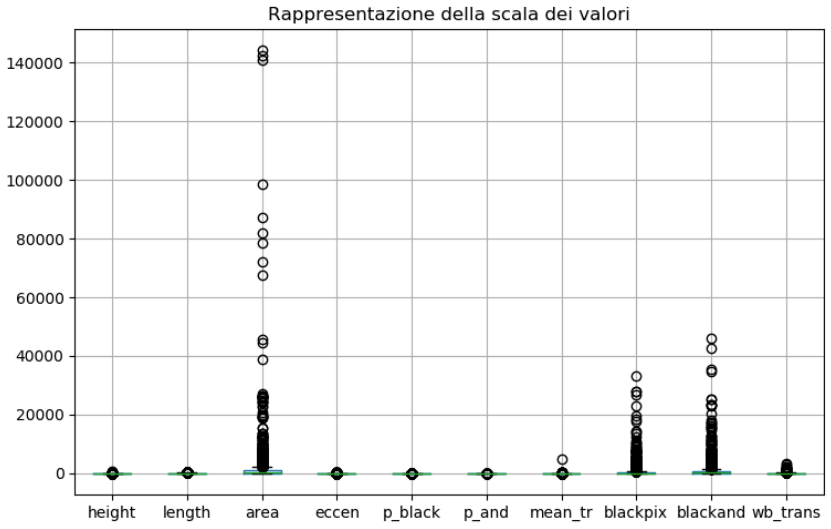

Fig2. Boxplot con i valori assunti dalle features

Come è mostrato nell'immagine precedente, le features assumono scale di valori molto differenti tra loro, in particolare si osservi come la feature '*area*' è caratterizzata da valori molto più grandi rispetto alle altre. Si può notare come le variabili, rappresentando grandezze reali, assumano sempre valori maggiori di zero.

Per eseguire un'analisi più completa si visualizza la distribuzione di ciascuna feature sfruttando la funzione *displot*. A scopo illustrativo, vengono riportati solo alcuni di questi grafici, scegliendo quelli relativi alle variabili '*area*' e '*p_black*'; in modo chiaro si possono osservare le caratteristiche sopra descritte e la differenza dei range di valori assunti.

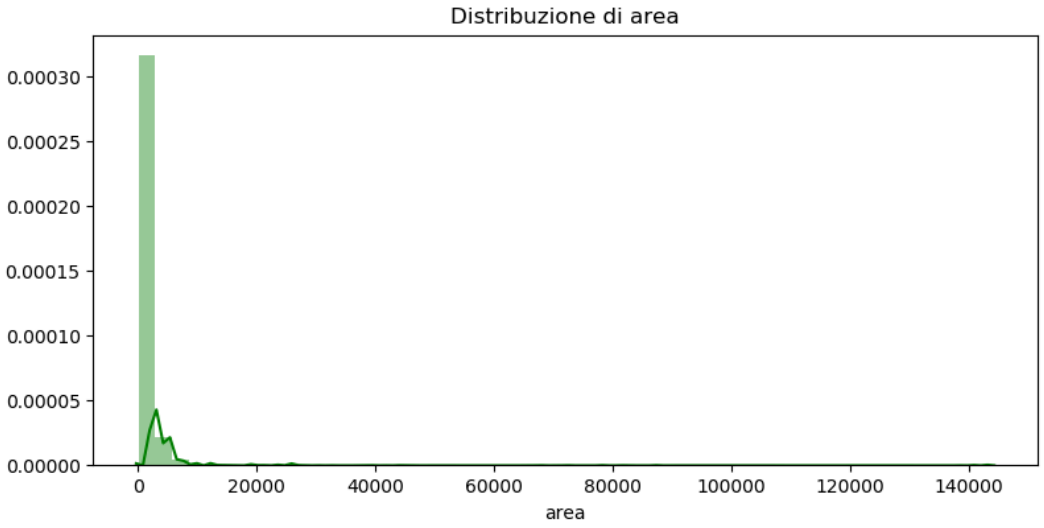

Fig3. Distribuzione della variabile 'area'

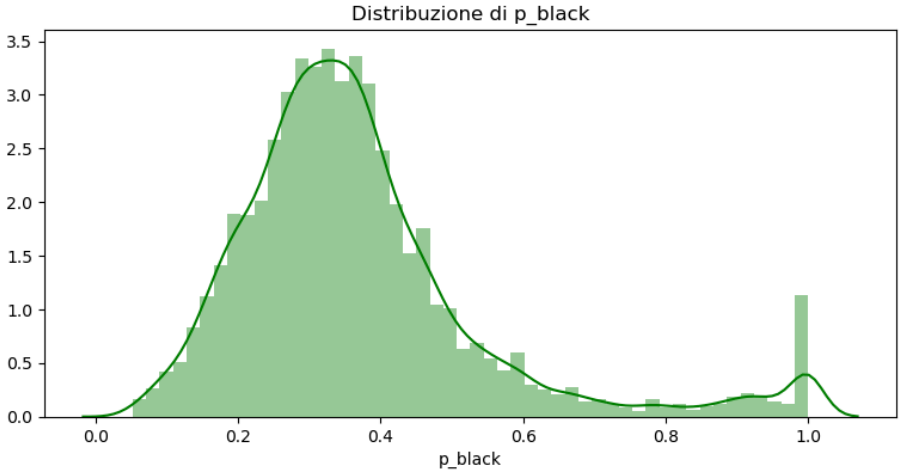

Fig4. Distribuzione della variabile 'p_black'

Capitolo 4

Preprocessing

L'analisi dei dati è seguita da una fase di processamento che permette di applicare gli algoritmi di apprendimento nel modo corretto. Infatti è importante analizzare le features che vengono utilizzate, perché è necessario che queste rappresentino nel miglior modo la variabile d'uscita.

4.1 Eliminazioni delle variabili collineari

Si definiscono variabili collineari due features del dataset che presentano una forte correlazione lineare. Quando questo accade il modello è meno interpretabile in quanto l'effetto di queste due features sull'output risulta indistinguibile ed inoltre, i parametri associati a queste variabili non sono univocamente determinati. Perciò è importante individuare queste features ed eliminare una tra le due.
Per condurre un'analisi di questo tipo si calcola la matrice di correlazione tra tutte le features e si analizza. L'immagine seguente rappresenta la matrice di correlazione del dataset ottenuta con la funzione *heatmap*, in cui i valori vengono rappresentati con una scala di colori.

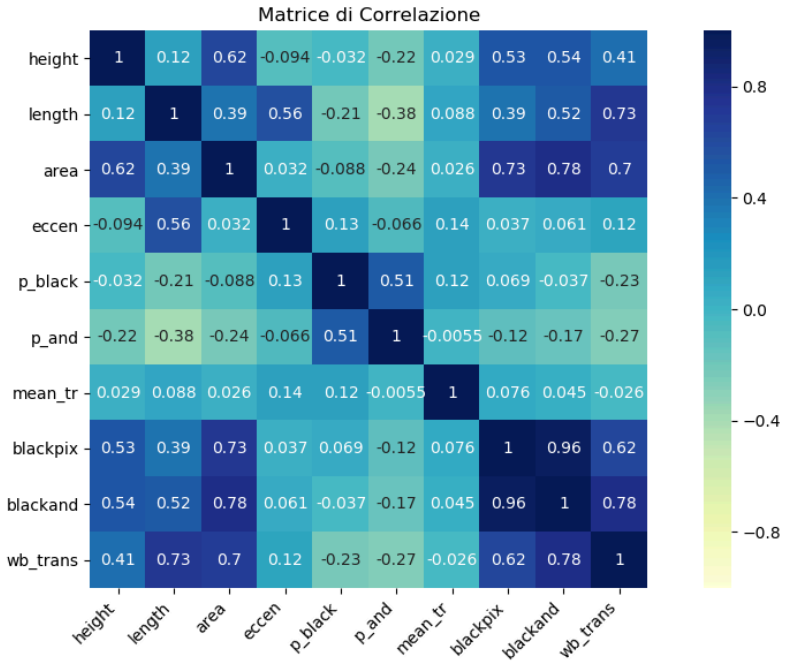

Fig5. Matrice di correlazione delle features

Come si può notare la matrice è simmetrica quindi si consideri la triangolare superiore. I valori contenuti in ogni cella della matrice rappresentano il grado di correlazione tra le due variabili: più il valore in modulo è prossimo ad uno, più le variabili sono correlate linearmente.

In particolare si nota una forte correlazione tra le variabili '*blackand*' e '*blackpix*' che riportano un valore nella matrice pari a 0.96. La variabile '*blackand*' ha anche una correlazione pari a 0.78 sia con '*area*' che con '*wb_trans*'.
Per mantenere il maggior numero di informazioni si sceglie di procedere all'eliminazione delle variabili altamente correlate, considerando un valore di soglia:

$$Corr(X,Y)>0.95$$

Imponendo tale vincolo le variabili individuate risultano pertanto '*blackand*' e '*blackpix*'.
Per decidere quali delle due variabili eliminare si esegue un'analisi ulteriore: si utilizza il *RandomForestClassifier* dalla libreria Scikit-Learn per determinare l'importanza delle features.
I risultati di questa analisi vengono mostrati nel grafico seguente in cui le features vengono riportate in ordine decrescente di importanza.

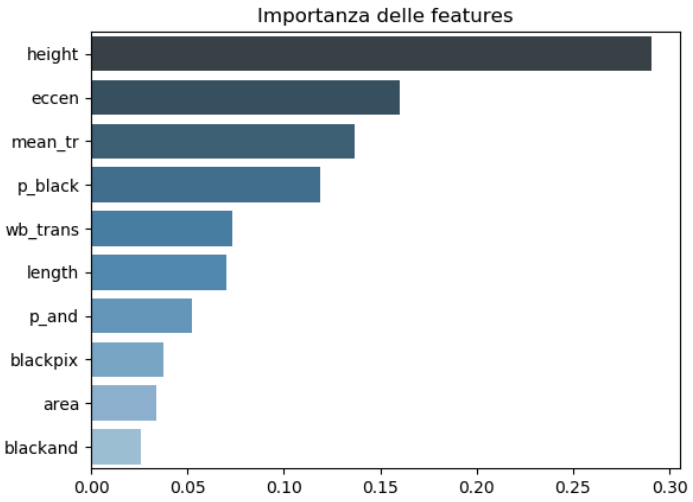

Fig6. Importanza delle features con Random Forest

Sulla base di questi risultati pertanto è stata eliminata la feature '*blackand*' che risulta avere l'importanza minore.

4.2 Standardizzazione

L'analisi del dataset condotta nel capitolo precedente ha evidenziato un andamento diverso nella distribuzione delle features e dei valori di scala.
È importante considerare questi aspetti perché l'applicazione di alcuni algoritmi di apprendimento su dati che presentano diversi range di valori può portare a risultati non ottimali. Questo è il caso ad esempio del modello KNN che realizza la classificazione sulla base della distanza dei punti nel dataset. Per risolvere questo problema si ricorre perciò alla normalizzazione dei dati, che può essere applicata con diversi criteri.
Nel caso in esame viene applicata la standardizzazione secondo la formula:

$$\hat{x} = \frac{x-\mu}{\sigma}$$

dove con μ si indica la media relativa alla variabile x e con σ la rispettiva varianza.
Attraverso questa trasformazione i valori delle features assumono quindi media nulla e varianza unitaria; si esegue nuovamente la rappresentazione attraverso il boxplot delle features e si confrontata con la Fig2.

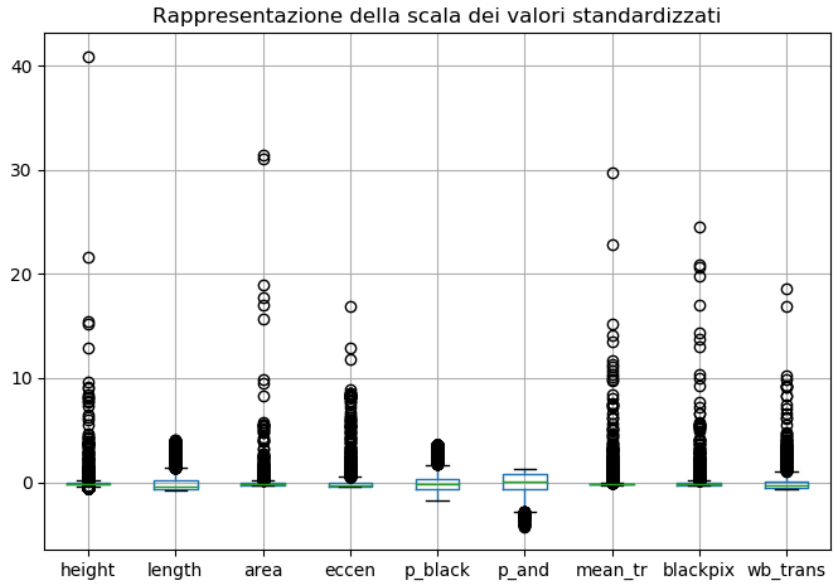

Fig7. Boxplot con i valori assunti dalle features dopo la standardizzazione

Vengono riportati inoltre, i grafici delle distribuzioni delle due variabili '*area*' e '*p_black*' dopo la standardizzazione, evidenziando il confronto con la Fig3 e Fig4.

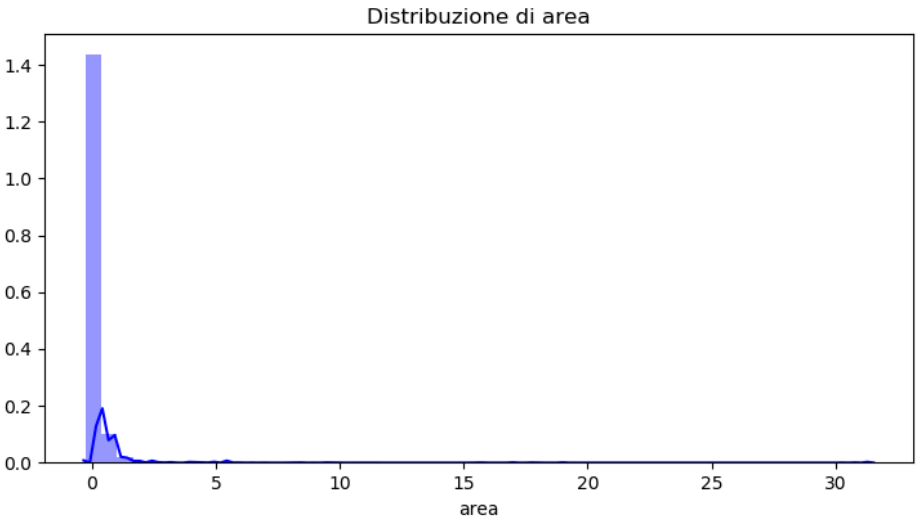

Fig8. Distribuzione della variabile 'area' standardizzata

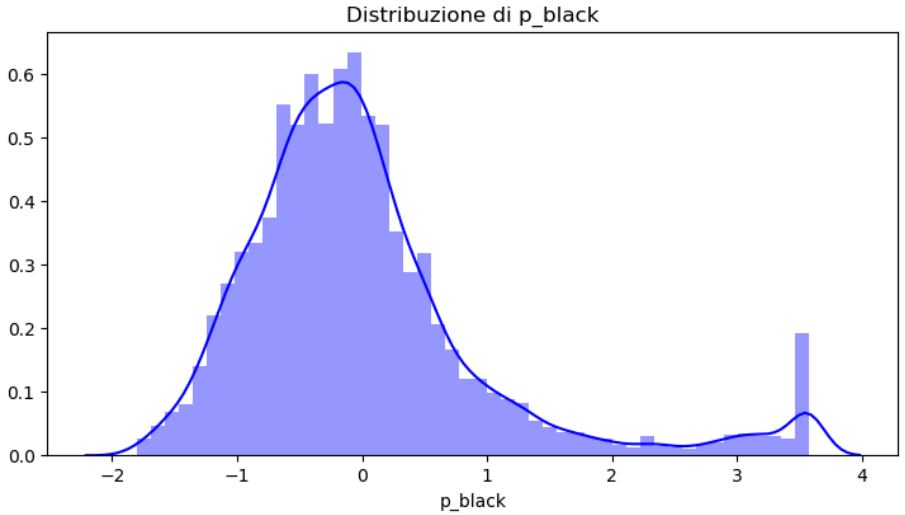

Fig9. Distribuzione della variabile 'p_black' standardizzata

Alla fine di questa fase il dataset è pronto per essere utilizzato dagli algoritmi di apprendimento. Avendo un numero sufficientemente limitato di features, possiamo mostrare, sfruttando la funzione *pairplot* offerta dalla libreria Seaborn, la relazione per ogni coppia di features del dataset, dividendo i campioni in base alla classe di appartenenza. Il grafico prodotto viene visualizzato nell'immagine seguente.

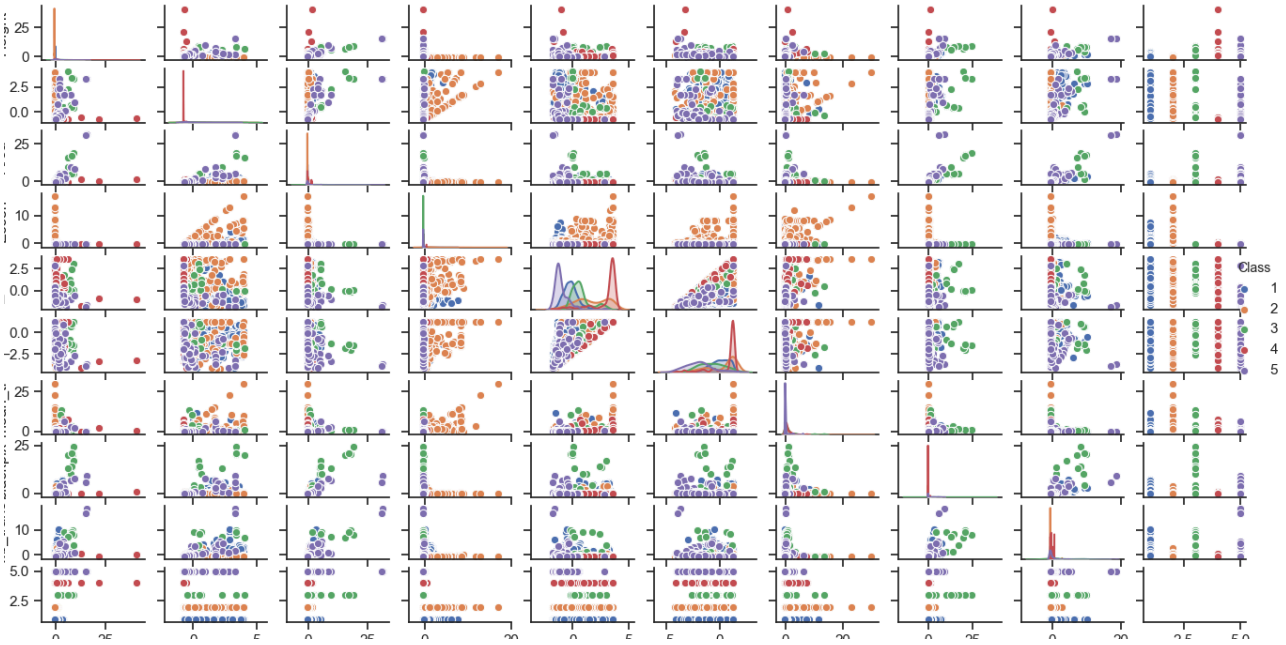

Fig10. Rappresentazione della relazione tra le features del dataset standardizzato

Precisazione: per standardizzare le features viene utilizzato l'oggetto *StandardScaler* che utilizza solo i valori del training set; sulla base di questi valori standardizza poi anche il test set. Nel capitolo seguente viene spiegata la separazione del dataset.

Capitolo 5

Separazione del Dataset

Per valutare i risultati di classificazione di un modello è necessario avere a disposizione un insieme di dati che appartengono alla stessa distribuzione dei dati utilizzati per l'addestramento, ma che sono diversi da essi. Per questo motivo il dataset iniziale viene diviso in due parti:

- **training set**, costituito dall'80% dei dati iniziali, che viene utilizzato come insieme di dati in input agli algoritmi di apprendimento;
- **test set**, costituito dal restante 20%, viene utilizzato nella fase di valutazione e serve per stimare il comportamento del classificatore quando si utilizzano nuovi dati, ovvero le sue reali prestazioni.

In base all'analisi del dataset condotta al capitolo 3 le classi non sono egualmente rappresentate dai samples, ovvero si è in presenza di classi sbilanciate. Volendo mantenere la stessa distribuzione anche nel training set e nel test set, la divisione è stata effettuata utilizzando una stratificazione rispetto al vettore delle etichette.
La distribuzione dei samples nelle diverse classi nei due insiemi di dati è mostrato nelle immagini seguenti.

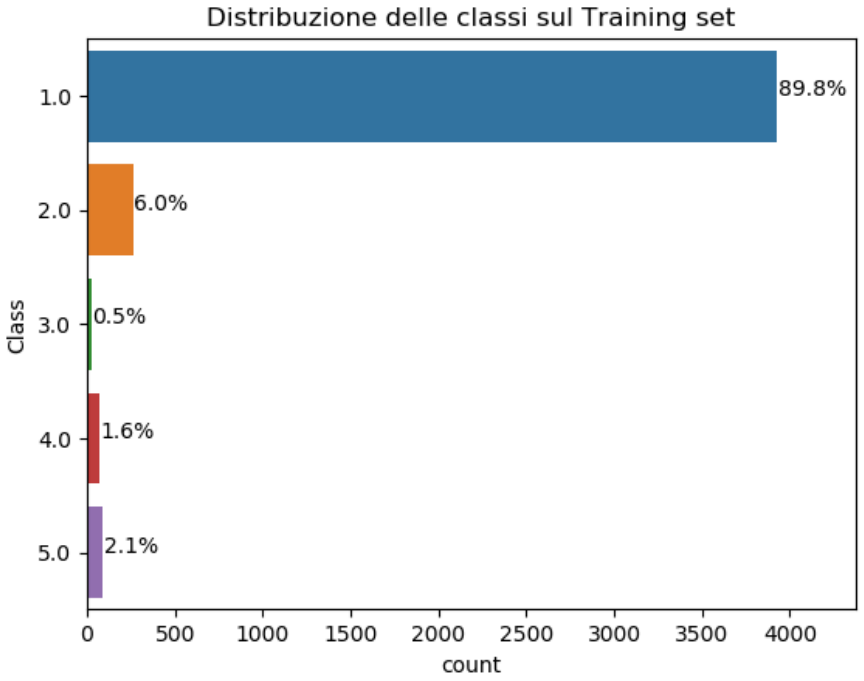

Fig11. Distribuzione delle classi nel training set

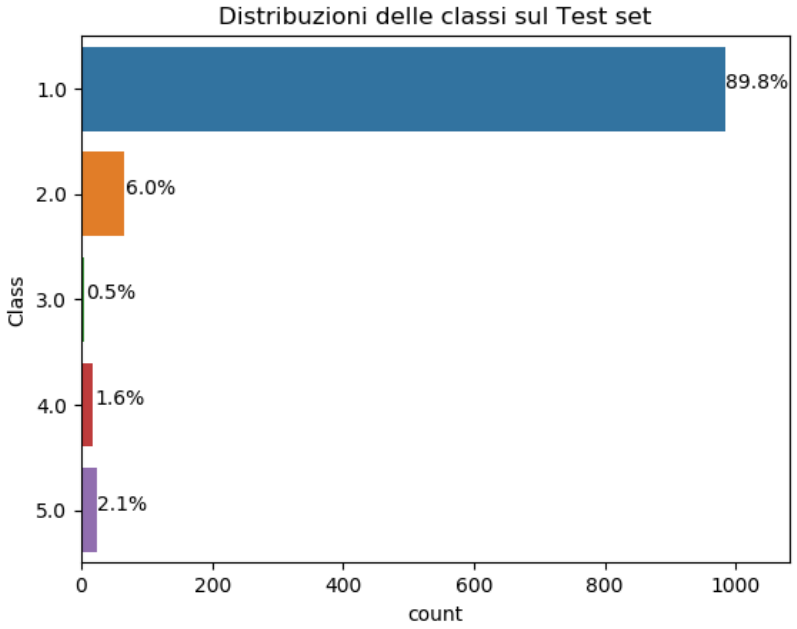

Fig12. Distribuzione delle classi nel test set

Si noti come le percentuali delle classi nei due set di dati rimangono invariate rispetto a quelle del data set iniziale (Fig1), seppur il numero di dati totali nel training set e test set sono diversi:

	Numero di campioni	Percentuale
Training set	4378	*80%*
Test set	1095	*20%*
TOTALE	5473	*100%*

Capitolo 6

Modelli

Ai fini del progetto, per la classificazione multiclasse del dataset sbilanciato vengono utilizzati diversi modelli disponibili all'interno della libreria Scikit-Learn, così da poter confrontare le performance.
I modelli scelti sono: K-Nearest Neighbors, Softmax Regression, Support Vector Machine, Multi-layer Perceptron e Naive Bayes.

6.1 k-Nearest Neighbours (k-NN)

L'algoritmo k-NN è una tecnica di apprendimento non parametrica, in cui la classificazione di un punto x dipende dall'insieme dei sui k punti più vicini. Il criterio si basa sull'assunzione che x ha una probabilità alta di appartenere alla classe maggiormente rappresentata in questo insieme di punti e per questo viene assegnato a tale classe.
Il criterio con cui valutare la distanza tra i punti può essere vario, ad esempio si può utilizzare la distanza Euclidea, ma anche quella di Manhattan, e sulla base di tale scelte la classificazione produce risultati differenti.
Nel progetto si è scelto di implementare un classificatore non parametrico di questo tipo, utilizzando la distanza Euclidea come metrica di valutazione e lasciando il parametro k come hyperparameter che viene determinato nella fase di addestramento. In realtà per k-NN è più corretto parlare di tuning dell'hyperparameter in quanto non esiste una fase di training ma tutto il costo computazionale è demandato a runtime.

6.2 Softmax Regression

L'algoritmo Softmax Regression è l'applicazione ad un problema di classificazione multiclasse dell'algoritmo Logistic Regression. Questo algoritmo di classificazione binaria utilizza la funzione sigmoid come funzione non lineare di trasformazione:

$$g(z) = \frac{1}{1+e^{-z}}$$

In casi di classificazione multiclasse è possibile adottare strategie quali OVA (One-Vs-All) o OVO (One-Vs-One).
Nel progetto si è scelto di implementare un classificatore multiclasse di questo tipo aggiungendo un parametro di regolarizzazione con norma L2 che permette di evitare l'overfitting. Gli hyperparameters che vengono determinati in fase di training sono il fattore di regolarizzazione C e il class_weight.

6.3 Support Vector Machine (SVM)

L'algoritmo SVM viene utilizzato soprattutto in problemi in cui lo spazio delle features ha dimensioni elevate.
L'obiettivo di questo metodo è massimizzare la distanza tra il piano di separazione delle classi e i dati, assumendo che in questo modo, nuovi dati che possono essere affetti da rumore vengono comunque classificati nel modo corretto.
Per problemi di classificazione multiclasse in generale si impiega l'approccio OVO.
Nel progetto si utilizza un classificatore non lineare di questo tipo, assumendo un kernel gaussiano. Gli hyperparameters di questo modello sono gamma e C, che rappresenta il fattore di regolarizzazione.

6.4 Multi-layer Perceptron (MLP)

Il MLP è una rete neurale in cui le funzioni di attivazione nel hidden layer possono essere sia la funzione sigmoidale che la funzione tanh. Nel caso di classificazione multiclasse la rete tende a minimizzare la funzione obiettivo che è chiamata Cross-entropy loss.
Nel progetto viene addestrata una rete con un solo hidden layer con 400 nodi e viene utilizzato l'algoritmo di Early Stopping per la regolarizzazione. Gli hyperparameters del modello sono activation e solver, che rappresenta l'algoritmo di ottimizzazione per la ricerca del minimo della funzione obiettivo.

6.5 Naive Bayes

Naive Bayes è un algoritmo generativo che si basa sulla regola di Bayes:

$$p(x) = \frac{p(y)P(y)}{p(x)}$$

Per modellare la $p(x|y)$ viene fatta l'assunzione di Naive Bayes che prevede l'indipendenza delle features. Per ottenere i parametri della classificazione l'algoritmo massimizza la joint likelihood.
Questo classificatore non richiede nessun tipo di parametro che deve essere determinato nell'addestramento.

Nella tabella sottostante viene riportato per ogni modello descritto la libreria di Scikit-Learn utilizzata e i relativi parametri. Una colonna separata è dedicata agli hyperparameters, individuati attraverso Model Selection e discussi nel capitolo successivo.

Modello	*Scikit-Learn*	*Parametri*	*Hyperparameters*
k-NN	KNeighborsClassifier	weights = ‘distance’ algorithm = ‘auto’ p = 2 metric = ‘minkowski’	n_neighbors
Softmax Regression	sklearn.linear_model-LogisticRegression	penalty = ‘l2’ multi_class = ‘multinomial’ solver=’newton-cg’	C class_weight
SVM	SVC	kernel = ‘rbf’ class_weight = ‘balanced’ decision_function_shape=‘ovo’	C gamma
MLP	MLPClassifier	hidden_layer_sizes = (400,) max_iter = 500 verbose= false early_stopping = True validation_fraction=0.5	activation solver
Naïve Bayes	GaussianNB	\	\

La scelta di includere alcuni parametri tra gli hyperparameters ed utilizzare altri parametri fissi è dovuta all’influenza che essi hanno sulla classificazione ed inoltre per fattori legati al costo computazione e al tempo di risoluzione.

Capitolo 7

Model Selection

In questa fase del progetto vengono determinati gli hyperparameters, descritti nel capitolo precedente, attraverso un processo di cross-validazione che viene applicato sui dati di training set.
La cross-validazione è una tecnica che permette di dividere il training set in k fold: per k volte si sceglie un fold come validation set e i restanti fold vengono utilizzati per fare addestramento del modello. Il modello addestrato viene applicato sul validation set su cui viene calcolato l'errore di classificazione per ogni iterazione, e ne viene fatto il valore medio.
Ripetendo questa tecnica con diversi valori dell'hyperparameter si ottiene il valore ottimale, ovvero quello che minimizza l'errore sul validation set.
Nel progetto viene utilizzato k=10 e la divisione del training set avviene in modo stratificato, ovvero mantenendo la proporzione tra le classi così come era nel dataset iniziale. La metrica di valutazione usata è f1-score (weighted), impiegata per la classificazione multiclasse in quanto realizza una media pesata dei valori ottenuti per ogni classe.

Per descrivere in maniera chiara il procedimento di Model Selection e come la scelta degli hyperparameters influenza le prestazioni del classificatore viene mostrata la curva di validazione per due dei modelli descritti.
Nell'immagine seguente è riportato il grafico che descrive l'errore di cross-validazione del modello k-NN al variare dell'hyperparameter k che rappresenta il numero dei vicini.

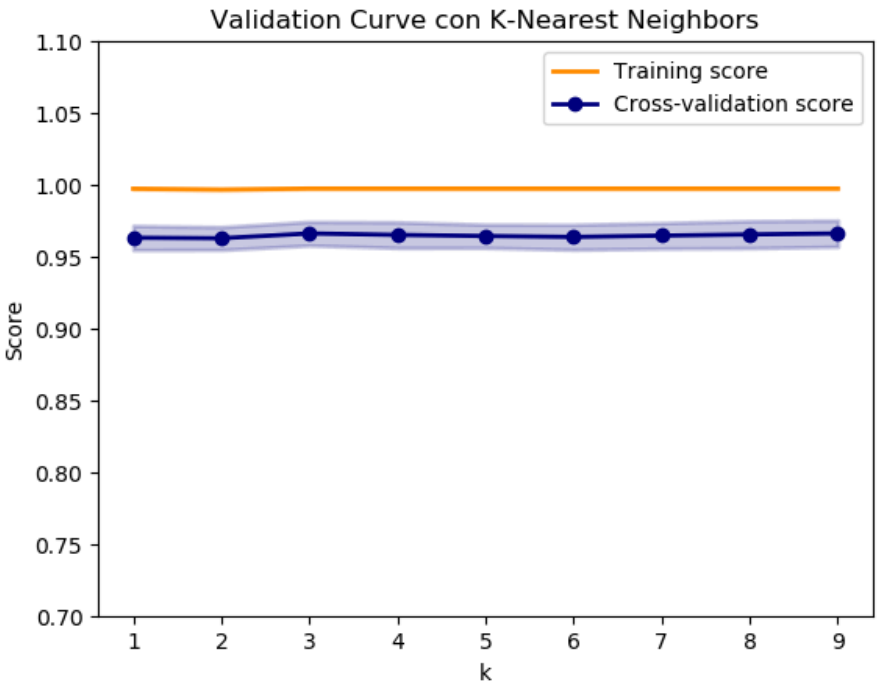

Fig13. Curva di validazione con KNN

Il valore di k scelto dall'algoritmo è 3 in quanto, come si osserva dal grafico, per questo valore la curva di validazione assume il valore massimo di f1-score e il valore più vicino alla curva di training, che come è noto, tende ad annullare l'errore.
Nella figura seguente invece è mostrata la curva di validazione del classificatore implementato con Logistic Regression ottenuta al variare dell'hyperparameter C di regolarizzazione.

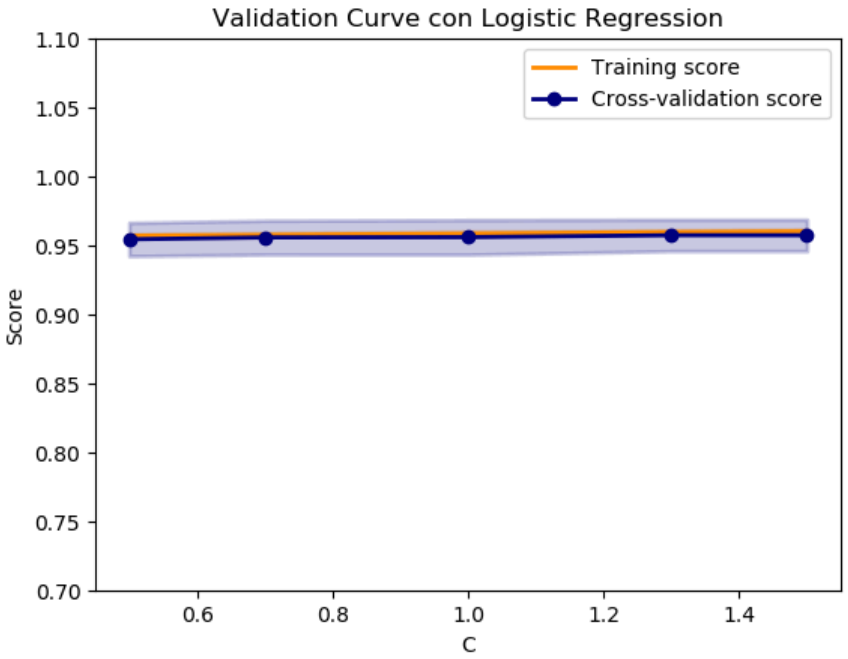

Fig14. Curva di validazione con Logistic Regression

In questo caso il valore ottimale del parametro si ottiene per C=1.3.
In questa ultima immagine si può notare come il parametro C non influisce in realtà in modo considerevole sulla classificazione, in quanto al variare di C le due curve tendono a coincidere.

La selezione degli hyperparameters dei classificatori avviene attraverso un opportuno modulo della libreria Scikit-Learn chiamato *GridSearchCV*. Vengono riportati i valori ottenuti alla fine di questa fase:

Modelli	*Hyperparameters*
KNN	n_neighbors = 3
Logistic Regression	C = 1.3 class_weights = None
SVC	C = 50 gamma = 0.005
MLP	activation = tanh solver = adam

I classificatori così ottenuti vengono utilizzati per realizzare la classificazione dei dati del test set.
Nel capitolo seguente si confrontano le prestazioni.

Capitolo 8

Metric Evaluation

Per valutare le prestazione della classificazione multiclasse vengono utilizzate delle metriche appropriate che nel caso di classificazione binaria assumono le seguenti forme:

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

$$Specificity = \frac{TN}{TN+FP}$$

$$F1 - score = \frac{2*Recall*Precision}{Recall+Precision}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$$

Nel problema in esame, in cui si è in presenza di classi sbilanciate, non ha senso definire l'*Accuracy* nel modo classico, ma viene utilizzata la seguente formula:

$$Accuracy = media_{weighted}(Recall)$$

Per confrontare i risultati dei diversi classificatori vengono mostrate le matrici di confusione di ciascun modello, da cui vengono ricavati i valori delle metriche di valutazione. Inoltre, per ciascuna classe, vengono riportati i valori di precision, recall e f1-score.

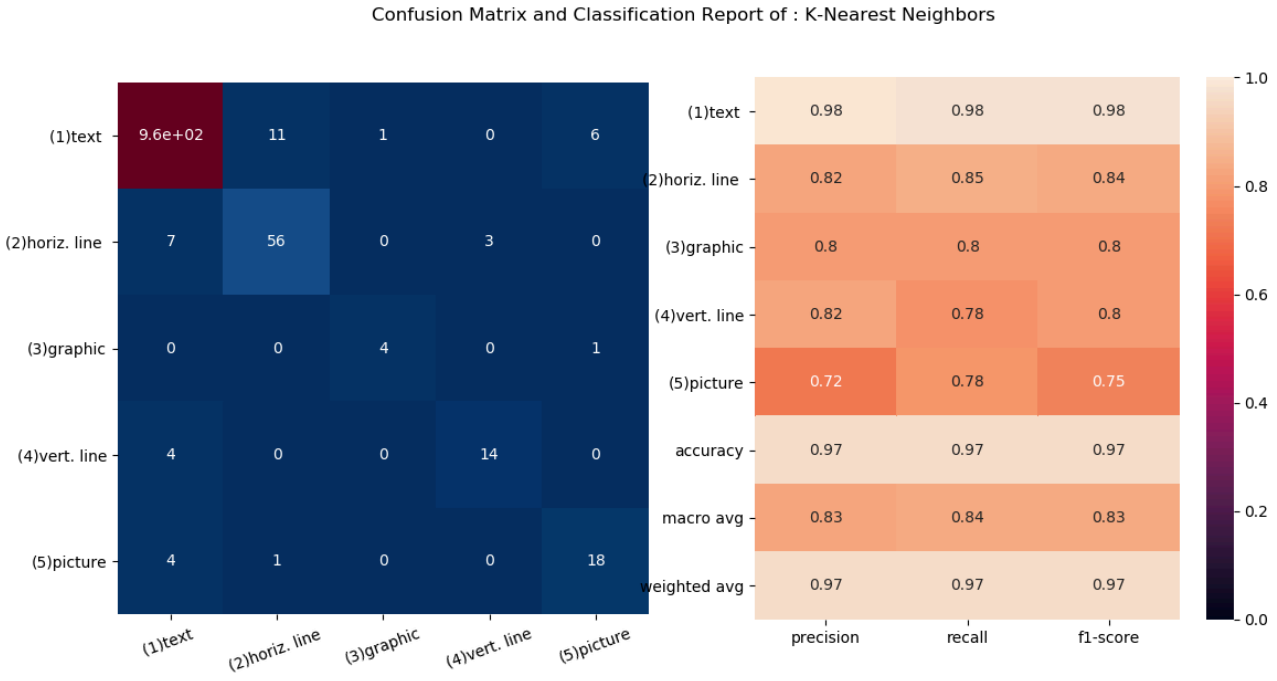

Fig15. Matrice di confusione e metriche di classificazione del modello k-NN

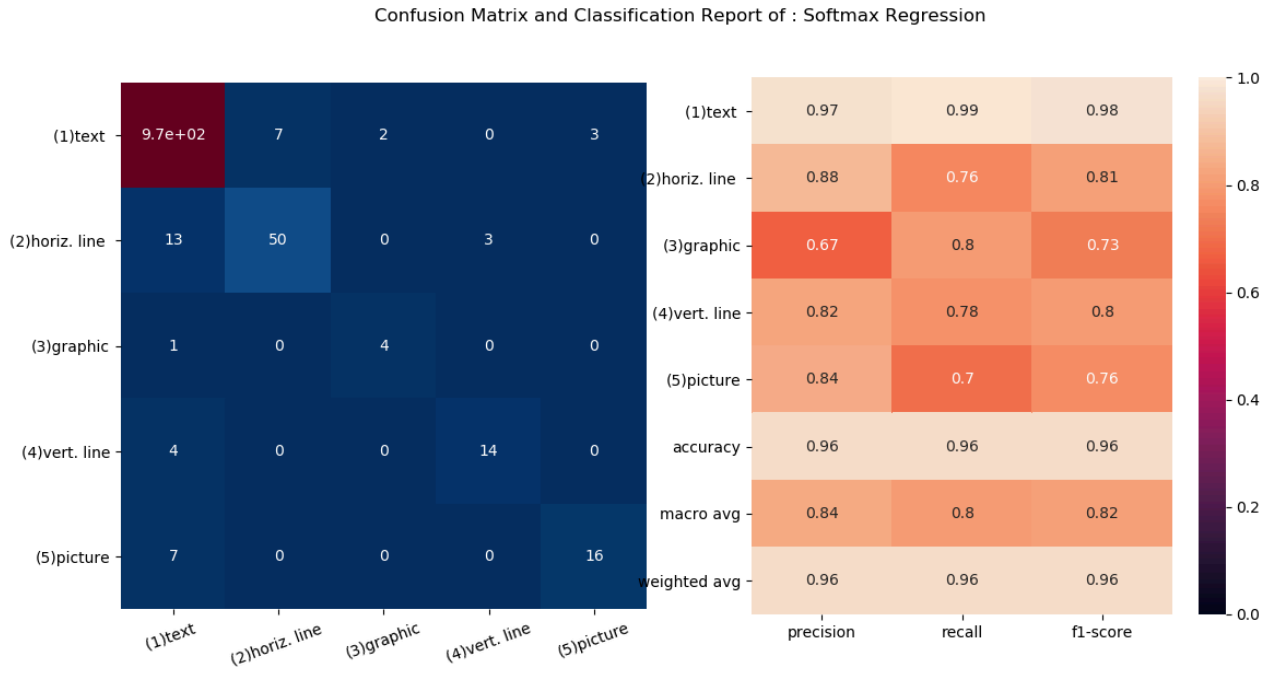

Fig16. Matrice di confusione e metriche di classificazione del modello Softmax Regression

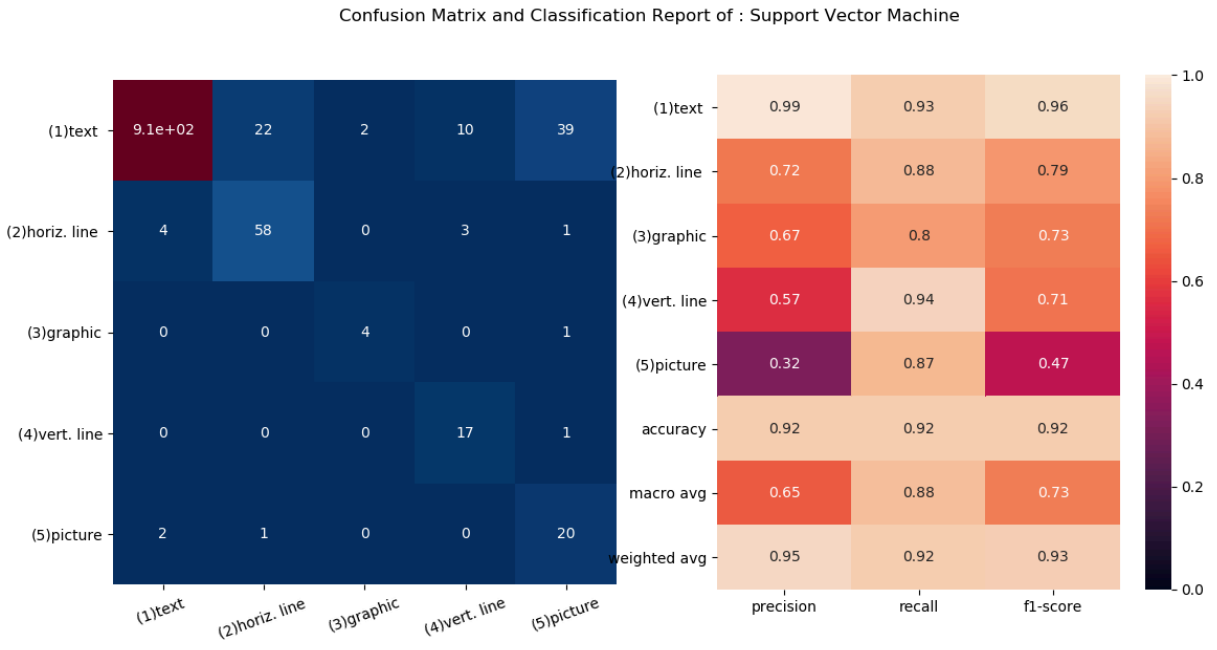

Fig17. Matrice di confusione e metriche di classificazione del modello SVM

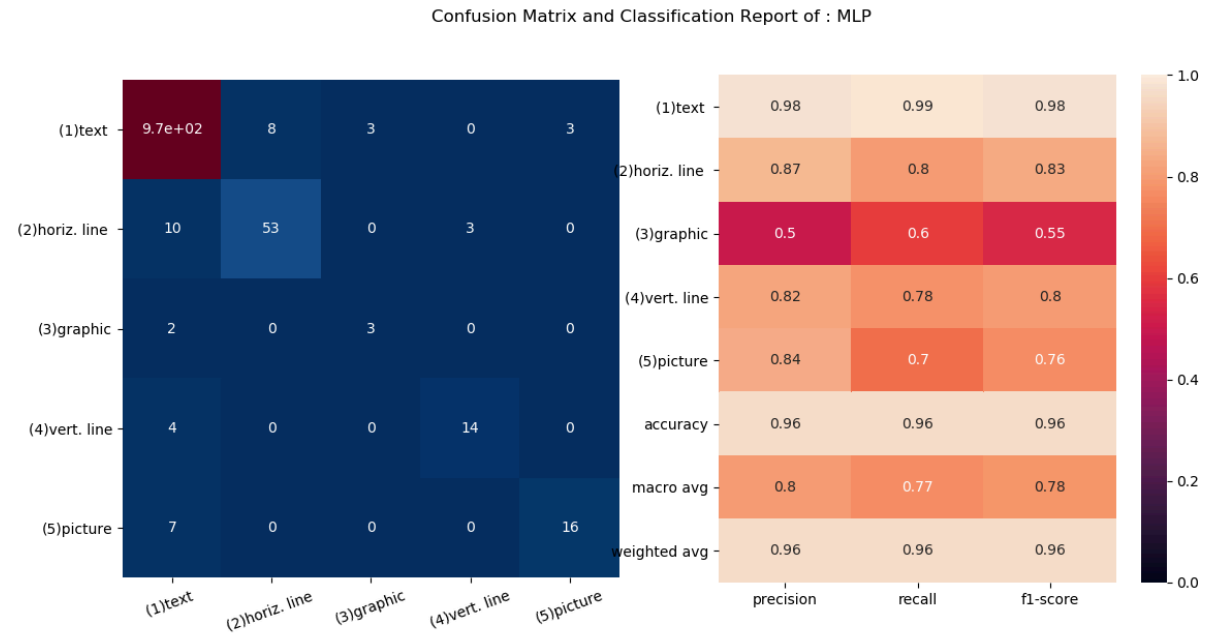

Fig18. Matrice di confusione e metriche di classificazione del modello MLP

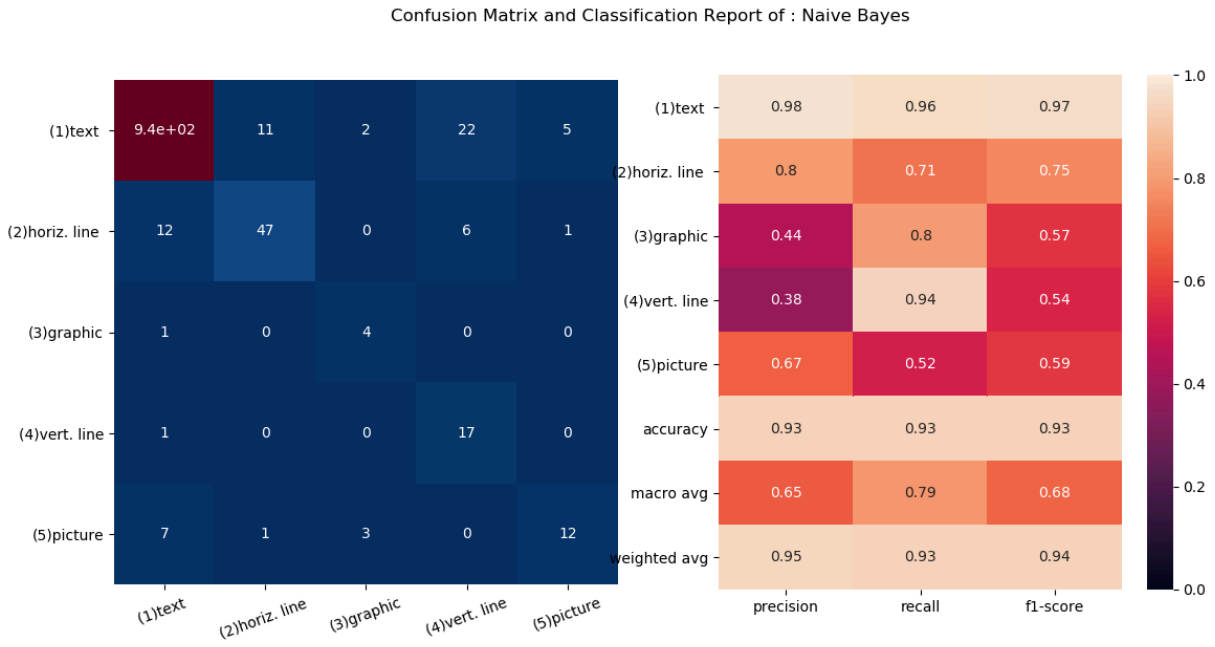

Fig19. Matrice di confusione e metriche di classificazione del modello Naive Bayes

Realizzando il confronto in base alla metrica f1-score 'weighted', il miglior risultato si ottiene con la scelta del modello KNN, che ha un valore di f1-score=0.97.

Una buona classificazione è realizzata anche dal modello Softmax Regression, che rispetto a KNN però ottiene dei risultati peggiori, in particolare per la classe *(3) grafic*, che risulta essere la classe con minor rappresentanza nel dataset.

Anche il classificatore SVC ottiene delle prestazioni molto deludenti per una particolare classe, la *(5) picture*; si noti infatti la presenza nella matrice di confusione in Fig17 di 39 campioni, definiti FP (*Falsi Positivi),* della classe *(1) text* che vengono stimati appartenenti alla classe *(5) picture*. Questo comporta un valore di Precision per questa classe molto basso, pari a 0.32.

Questo comportamento che penalizza delle classi meno rappresentate viene evidenziato nella tabella dal valore di f1-score '*macro avg'*, che risulta essere inferiore rispetto a quello '*weighted avg*'; infatti il primo viene calcolato come una media aritmetica dei valori f1-score di tutte le classi, mentre il secondo viene pesato per un fattore che rispecchia la rappresentazione delle classi nel dataset.

Anche per MLP si presenta la stessa problematica, in quanto si nota dalla Fig18 che i valori di precision, recall e f1-score sono deludenti per la classe *(3) grafic*.

Infine analizzando la Fig19, si nota come Naive Bayes si comporta bene con la classe *(1) text* che è quella maggiormente rappresentata ed invece ha comportamenti poco soddisfacenti per le altre classi.

La tabella seguente riassume i valori delle metriche di classificazione per ogni modello utilizzato (l'Accuracy è calcolata con la funzione *balanced_accuracy_score* disponibile nella libreria *Scikit-Learn*).

	k-NN	*Softmax Regression*	*SVM*	*MLP*	*Naive Bayes*
Accuracy	0.8381	0.8038	0.8837	0.7724	0.7875
Precision	0.9659	0.9623	0.9541	0.9628	0.9484
Recall	0.9653	0.9635	0.9215	0.9635	0.9342
F1-score	0.9656	0.9624	0.9328	0.9628	0.9387