Classificazione e Identificazione degli Elementi all'interno di in un Documento

Report per l'Esame di Fondameti di Machine Learning

Luca D'Alessandro

Matricola: 146880
Ingegneria informatica
278745@studenti.unimore.com

14-06-2023

1 Introduzione

Lo scopo della nostra indagine che andremo ad affrontare in questo articolo consiste nella classificazione di elementi, come ad esempio, testo, immagini e linee presenti all'interno della maggior parte dei documenti.
Sfrutteremo gli algoritmi di machine learning, metodi matematico computazionali per apprendere informazioni direttamente dai dati.
In conclusione saremo quindi in grado di identificare, tramite il miglior modello di classificazione, gli elementi presenti all'interno di un documento.

2 Dati

I dati presi in considerazione sono stati recuperati dal data set in questione al seguente **Link** e sono stati già sottoposti ad un processo di segmentazione.
Gli attributi sono:

- height: altezza dell'elemento
- lenght: lunghezza dell'elemento
- area: area del blocco analizzato $height \cdot lenght$
- eccentricity_block: eccentricità dell'elemento $\frac{lenght}{height}$
- pixel_black: Percentuale pixel neri nel blocco $\frac{blackpix}{area}$
- pixel_and: percentuale pixel neri nel blocco dopo l'RLSA $\frac{blackpix}{area}$
- mean_tr: media delle transizioni bianco-nero $\frac{blackand}{wb\_trans}$
- blackpix: numero totale dei pixel neri nella bitmap originale
- blackand: numero totale dei pixel neri nella bitmap dopo 'RLSA
- wb_trans: numero delle transizioni bianco-nero nella bitmap originale
- class: classe di appartenenza del blocco

Le 5 classi invece sono le seguenti:

1. Testo
2. Linea orizzontale
3. Immagini
4. Linea verticale
5. Grafico

2.1 Elaborazione Dei Dati

Il numero di campioni disponibile è di 5473, ogni attributo è in versione numerica e risultano privi di valori mancanti.
Per la preparazione dei dati è bastato dunque trasformare il data set dalla sua versione originale a quella finale inserendo delle virgole tra i valori e rimuovendo gli spazi, inserendo la parte intera (0) alle cifre decimali (di tipo float) e concludendo con l'attribuzione di un nome per ciascuna colonna e il loro corrispettivo di tipo numerico, in modo da renderlo un formato .arff consentendo la conversione in .csv.

2.2 Analisi esplorativa dei dati

Elaborati i dati, notiamo il netto sbilanciamento tra i blocchi di testo e il resto delle classi (visibile nella Figura 1), a cui diamo particolare attenzione sia per la suddivisione del data set che per la scelta di una giusta metrica di valutazione a cui attenerci per le valutazioni decisionali e finali.

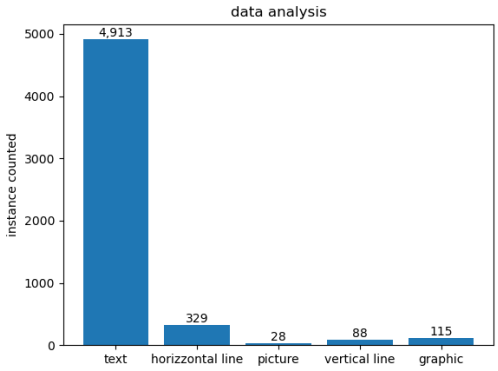

Figure 1: Istanze per classe

Inoltre, si possono notare nella Figura 2 elementi ridondanti e correlati: come l'*area*, i *blackpix* e i*pixel_black*.
A questo punto presumiamo corretto rimuovere la colonna dell'*area* dal data set, poiché elemento ridondante e linearmente dipendente dall'altezza e dalla lunghezza; dato che risulta non necessario.
Gli altri due attributi invece aggiungono informazioni al training del modello.

3 Ricerca e Sviluppo

Essendo l'oggetto di studio classificativo compareremo diversi modelli e sceglieremo il migliore tra di essi.
Tenendo in consiglio che il data set sia stato usato per provare e semplificare delle metodologie per alberi decisionali, come descritto nella documentazione dal suddetto.

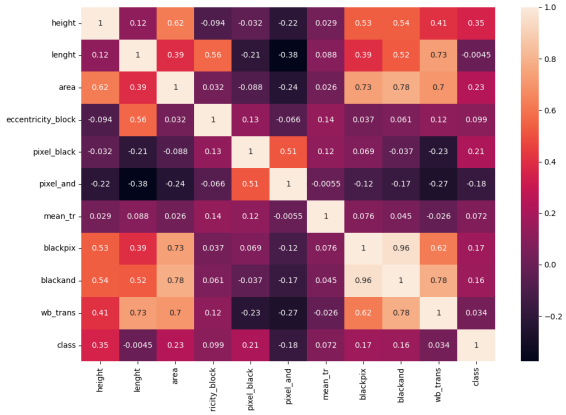

Figure 2: Correlazione tra gli attributi

3.1 Preparazione del Data Set

Iniziamo dunque separando i dati in training e testing che corrispondono rispettivamente all'80% e il 20% utilizzando la stratificazione, processo fondamentale, in quanto necessario a mantenere il giusto rapporto tra le classi all'interno dei due set.
In seguito procediamo all'adattamento e alla trasformazione utilizzando lo StandardScaler di sklearn sui dati di training.

3.2 Creazione dei modelli

Per questo problema utilizzeremo due tipi di ensemble e diversi modelli da utilizzare come stimatori per lo Stacking.
In particolare:

- Random Forest
- Decision Tree
- KNeighbords
- Softmax Regression
- Support Vector Machine

3.3 Ricerca degli iper-parametri

Procediamo con la ricerca dei migliori iper-parametri creando un dizionario per ogni modello:

```
param_grid_dt = {'criterion': ['gini', 'entropy']}
param_grid_Random_forest = {"n_estimators": [50, 60, 100], "max_features": [1, 3, 5, 7],
"max_depth": [3, 5, 10, None], "min_samples_split": [2, 3],
"min_samples_leaf": [1, 2, 3], "bootstrap": [True, False]}
param_grid_softmax = {'penalty': [None, 'l1', 'l2'], 'C': [1e-5, 5e-5, 1e-4, 5e-4, 1]}
param_grid_svc = {'C': [1e-4, 1e-2, 1, 1e1, 1e2], 'kernel': ['linear', 'rbf'],

'gamma': [0.001, 0.0001]}
param_grid_knn = {'n_neighbors': list(range(1, 10, 2))}
```

eseguiamo quindi per ogni classificatore una Grid-Search con K-fold cross validation stratificato, valutandone poi le rispettive metriche.
In python:

```
kfolds = StratifiedKFold(n_splits=4, shuffle=True, random_state=0)
grid_model = GridSearchCV(estimator=model, param_grid=hparameters, scoring=['f1_macro'],
refit='f1_macro', cv=kfolds)
```

Il riferimento di cui faremo uso sarà la macro F1-Score:

$$macroF1 - Score = \frac{\sum_{i=1}^{N-Classi} F1 - score_i}{Classi}$$

dove i è l'i-esima classe che viene presa in considerazione.
Otterremo così una stima più accurata del modello che darà importanza anche alla classe con meno semple.
Inoltre Useremo la balance accuracy come secondo riferimento, anch'essa calcolata su ogni singola classe.
Così ottenuti gli iper-parametri che nel nostro caso corrispondono ai seguenti:

```
RandomForestClassifier(bootstrap=True, max_depth=10, max_features=7, min_samples_leaf=2,
min_samples_split=3, n_estimators=60)
LogisticRegression(class_weight='balanced', multi_class='multinomial', solver='saga', C=1,
penalty=None)
DecisionTreeClassifier(class_weight='balanced', criterion='entropy')
KNeighborsClassifier(weights='distance', n_neighbors=5)
SVC(class_weight='balanced', C=100, gamma=0.001, kernel='linear')
```

3.4 Valutazione dei Modelli

Valutiamo i migliori classificatori dai loro risultati.
Di seguito l'elenco:

Modello	**balanced accuracy**	**macro F1-Score**
DecisionTree	80.33%	80.34%
RandomForest	83.58%	84.25%
Softmax	90.78%	64.83%
KNeighbors	80.47%	82.59%
SVM	89.98%	72.80%

Table 1: Risultato dei modelli

Il modello di Stacking si basa sui classificatori che hanno soddisfatto almeno l'80% della macro F1-Score

- Random Forest
- Decision Tree
- KNeighbords

Ripetere la cross validation stratificata per lo Stacking è il nostro ultimo passaggio:

	balanced accuracy	**macro F1-Score**
Stacking	82.34%	85.08%

Table 2: Risultato Stacking

alla fine della valutazione notiamo un incremento dello 0.8% rispetto al Random Forest.

3.5 Scelta Finale

Scegliamo infine il classificatore che massimizza la macro F1-score: lo Stacking.
Con il suo successivo adattamento all'intero data set di training.

4 Risultati

Arrivati alla fase del testing, dopo aver settato ogni iper-parametro e aver preso la decisione finale sul modello, possiamo procedere alla valutazione utilizzando il test set, ma non prima di averlo trasformato con lo StandardScaler precedentemente adattato.
Otteniamo il seguente risultato:

	balanced accuracy	**macro F1-Score**
Stacking	82.34%	85.08%

Con la sua relativa confusion matrix:

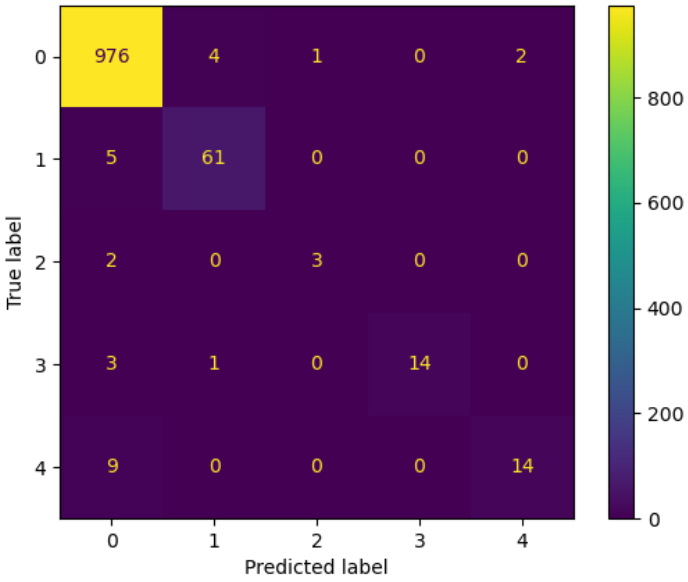

Figure 3: confusion matrix

poi applicata per ogni classe:

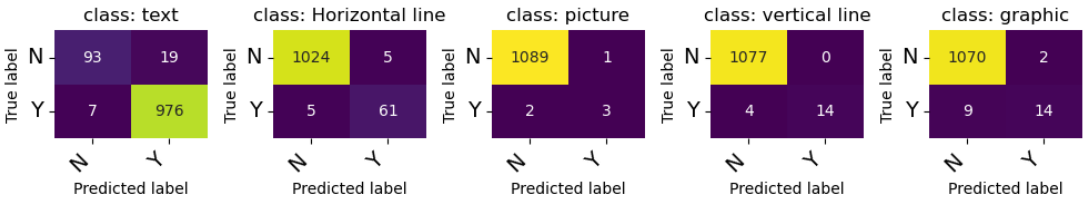

Figure 4: confiusion matrix per ciascuna classe

5 Conclusione

I dati raccolti sono sufficienti per definire un modello che riesca a prevedere il contenuto di un oggetto all'interno di un documento.
Riassumendo brevemente le fasi:
abbiamo ripulito il dataset, eliminato le colonne non necessarie, adattato lo StandardScaler e

trasformato i dati di training set e successivamente Testing set, trovando poi i migliori iper-parametri con l'utilizzo della Grid-Search cross-convalidato.
In conclusione costruiamo lo Stacking basandoci sui migliori stimatori (weak learner) e scelto il più adatto tra di essi al nostro data-set.

Una successiva analisi più approfondita sarebbe richiesta, in quanto risulta necessario bilanciare meglio il data set colmando così i valori delle classi minori con un risultato che ne aumenterebbe le prestazioni.