

Progetto MOBD

Corso di Laurea Magistrale in Ingegneria Informatica

Data Science & Engineering

Anno accademico 2021-2022

Giulio Appetito 0321669

Anastasia Brinati 0321654

Indice

1. Scopo di questo elaborato
2. Analisi
   1. Training e Test
   2. Preprocessing
      1. One Hot Encoding
      2. Sostituzione Valori Mancanti
      3. Scaling
      4. Anomaly Detection
      5. Balancing
3. Considerazioni conclusive
4. **Scopo di questo elaborato**

La seguente relazione ha l’obiettivo di illustrare il progetto per l’esame di Metodi di Ottimizzazione per Big Data, entrando nel dettaglio dei passi seguiti per l’addestramento di un classificatore al fine di risolvere un problema di classificazione binaria ed ottenere una buona soglia di accuratezza.

Abbiamo utilizzato Google Colab per la stesura del codice, utilizzando librerie di Data Analysis e Machine Learning quali Numpy, Pandas ed Sklearn, già presenti nell’ambiente.

1. **Analisi**

L’analisi ha seguito la pipeline presentata a lezione basata sul CRISP-DM, a partire dalla fase di Data Preparation.

* 1. **Training e Test Set**

In primo luogo, abbiamo suddiviso il dataset ‘train.csv’ in due sottoinsiemi, uno di training ed uno di test, per poter procedere all’addestramento di un classificatore tramite il primo insieme, e poi testarne le performance con il secondo set ottenuto. La suddivisione è avvenuta utilizzando la funzione train\_test\_split(x, y, test\_size = 0.3), dalla quale abbiamo ottenuto, con un rapporto 70-30, sia il set di training con le relative etichette in un vettore separato, che il test set allo stesso modo. Inoltre abbiamo utilizzato il parametro stratify=y per ottenere y di test e di training distribuite con la stessa media.

* 1. **Preprocessing**

A seguito della separazione ci siamo occupati di controllare quale fosse il grado di sbilanciamento delle due classi, che è risultato essere circa 0,7591 a 0,2408.

Il training set fornitoci consiste di 14 features, indicate come F0 - F13, sia di tipo continuo che categorico, per cui la prima questione che abbiamo affrontato sono stati i valori di queste ultime.

* + 1. **One Hot Encoding**

Dapprima abbiamo ragionato su come convertire queste features, che corrispondevano a ben 8 attributi su 14 totali, in valori numerici. Per questo scopo è stata utilizzata la tecnica one hot encoding, ed abbiamo applicato questa conversione da categorical data a features numeriche sulle colonne F1, F3, F5, F6, F7, F8, F9, F13.

Non potendo risalire al significato delle features, tranne alcune come il genere, la nazionalità e (crediamo) l’età, abbiamo fatto alcune assunzioni ‘ad occhio’ per valutare se impattassero le performance una volta completato il training; ad esempio per quanto riguarda il genere, abbiamo optato per lasciare la conversione in due colonne ‘Female’ e ‘Male’ binarie e mutuamente esclusive, piuttosto che una sola feature binaria; altro esempio è stato sulla colonna F3 (categorica), che abbiamo notato essere strettamente correlata alla feature F4 (continua), infatti i valori Ki della prima corrispondevano 1 ad 1 con i valori numerici della seconda, per cui il tentativo è stato cancellare completamente la feature F3, che però non è risultato migliore di applicare anche su di essa il processo di encoding, trasformandola dunque in altre 6 colonne binarie (e anch’esse mutuamente esclusive); ragionamento simile al precedente è stato fatto anche sugli attributi F10 ed F11 che risultano assumere entrambe il valore ‘0’ per il 91 e 95% delle istanze.

Ovviamente le scelte fatte per la trasformazione del set di colonne sono state ripetute anche sul test set.

* + 1. **Sostituzione Valori Mancanti**

A seguito di questi ragionamenti, abbiamo guardato ai valori mancati nel dataset **iniziale**, rispettivamente:

F1: 1836, F6: 1843, F13: 583, tutti e tre relativi a feature categoriche. Tuttavia l’encoding ha portato a trasformare i valori non disponibili come ad essere un’opzione aggiuntiva nel dominio di un determinato attributo, quindi una nuova colonna, per cui le opzioni che abbiamo considerato a questo punto sono state: scartare le tuple con l’insieme di valori mancanti, prima dell’encoding; oppure andare ad imputare dei valori validi sostituibili, magari con algoritmi che non guardano alla media di ogni singola feature ma all’istanza nel complesso, tipo KNN, tuttavia andando ad introdurre parecchio rumore, dato che ad esempio nel caso della colonna F6, dopo l’encoding un F6 nan ci saremmo ritrovati con una tupla con 16 ‘nan’ (P1,..,P16); o ancora eliminare le colonne ‘nan’ generate dall’encoding, lasciando tuple con tutti ‘zeri’ in corrispondenza di ogni possibile valore assumibile dalla feature;

La strategia da noi adottata è stata utilizzare un SimpleImputer che andasse a sostituire i valori nan con il ‘most frequent’, non appena suddiviso il dataset in test e training, e dunque prima di effettuare l’encoding. Per quanto riguarda il test set i valori mancanti sono stati sostituiti con il most frequent calcolato sul training set.

* + 1. **Scaling**

Lo step successivo di cui ci siamo occupati è stato lo scaling delle features, in quanto anche le stesse feature continue erano molto differenti per unità di grandezza, ad esempio F0 (età) e F2 (forse l’annual income?).

* + 1. **Anomaly Detection**

Utilizzando Isolation Forest come visto a lezione andiamo a selezionare solo quei valori inlier, anche se è il risultato ha escluso un numero irrisorio di valori (<20).

* + 1. **Balancing**

Tenendo presente la scelta dell’Isolation Forest, abbiamo optato per l’undersampling, che viene effettuato appunto dopo l’anomaly detection, di modo da addestrare la foresta con più dati possibile e senza escludere gli outliers. Il sampler adottato è ClusterCentroids(random\_state=42), dato che abbiamo constatato ci garantisse le migliori performance. Durante lo svolgimento abbiamo anche testato la possibilità dell’oversampling, effettuato prima dell’anomaly detection, ma questo ha prodotto risultati peggiori rispetto all’undersampling.

* + 1. **Scelta Classificatore**

La scelta del classificatore è avvenuta attraverso una serie di prove, con differenti training set, che hanno evidenziato una maggiore accuracy nel GradientBoostingClassifier.

Il livello di accuratezza da noi raggiunto è compreso fra 87,54 e 87,71.

1. **Considerazioni Conclusive**

Dato che il progetto prevedeva un benchmark sull’accuratezza, riteniamo che le due classi del problema avessero la stessa ‘importanza’, per cui una stima basata su questa metrica è ovviamente sensata.

D’altra parte, non avendo questa certezza assoluta, abbiamo confrontato varie metriche, quali l’F1 score e la matrice di confusione, sia per valutare quale classificatore fosse di volta in volta ‘migliore’, sia addestrando lo stesso modello su diversi training set.