Introduzione

Questa relazione presenta i passaggi principali svolti nel determinare il miglior classificatore da noi identificato per l'addestramento della nostra SVM.
È stato usato JupyterLab, in ambiente Python configurato mediante Anaconda.

Analisi Dataset

Il primo passo, è stato analizzare e comprendere la struttura del nostro dataset. Esso comprende 8000 dati, 20 attributi di tipo quantitativo, divisi non equamente in 4 classi differenti.
Notiamo la distribuzione non omogenea dei valori tra le classi.

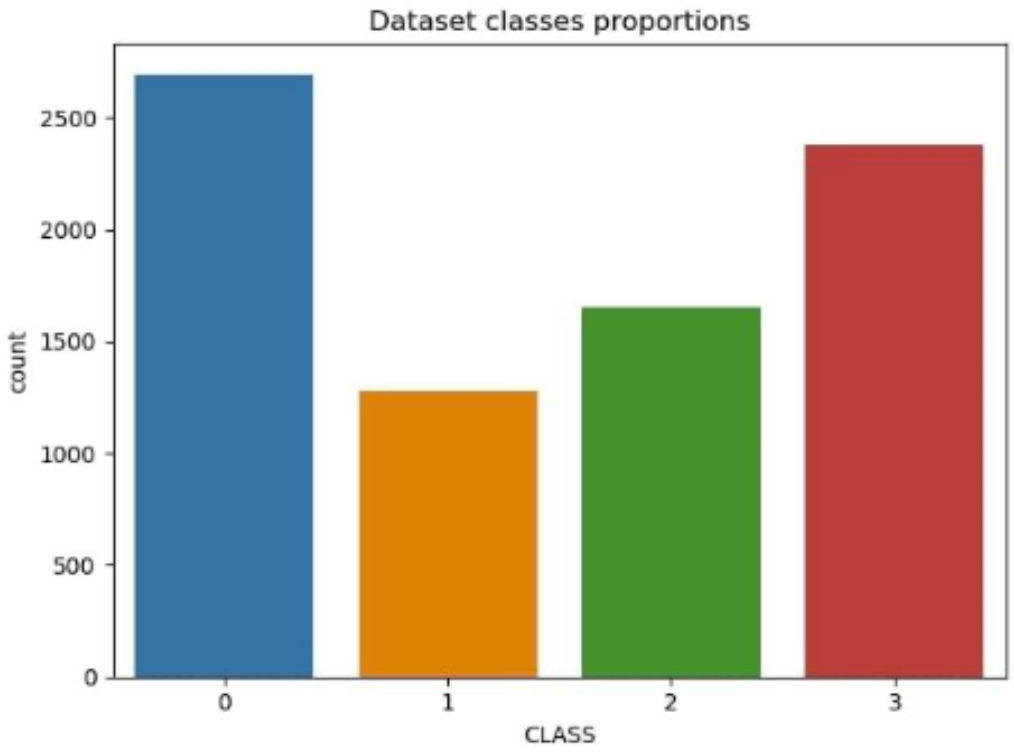
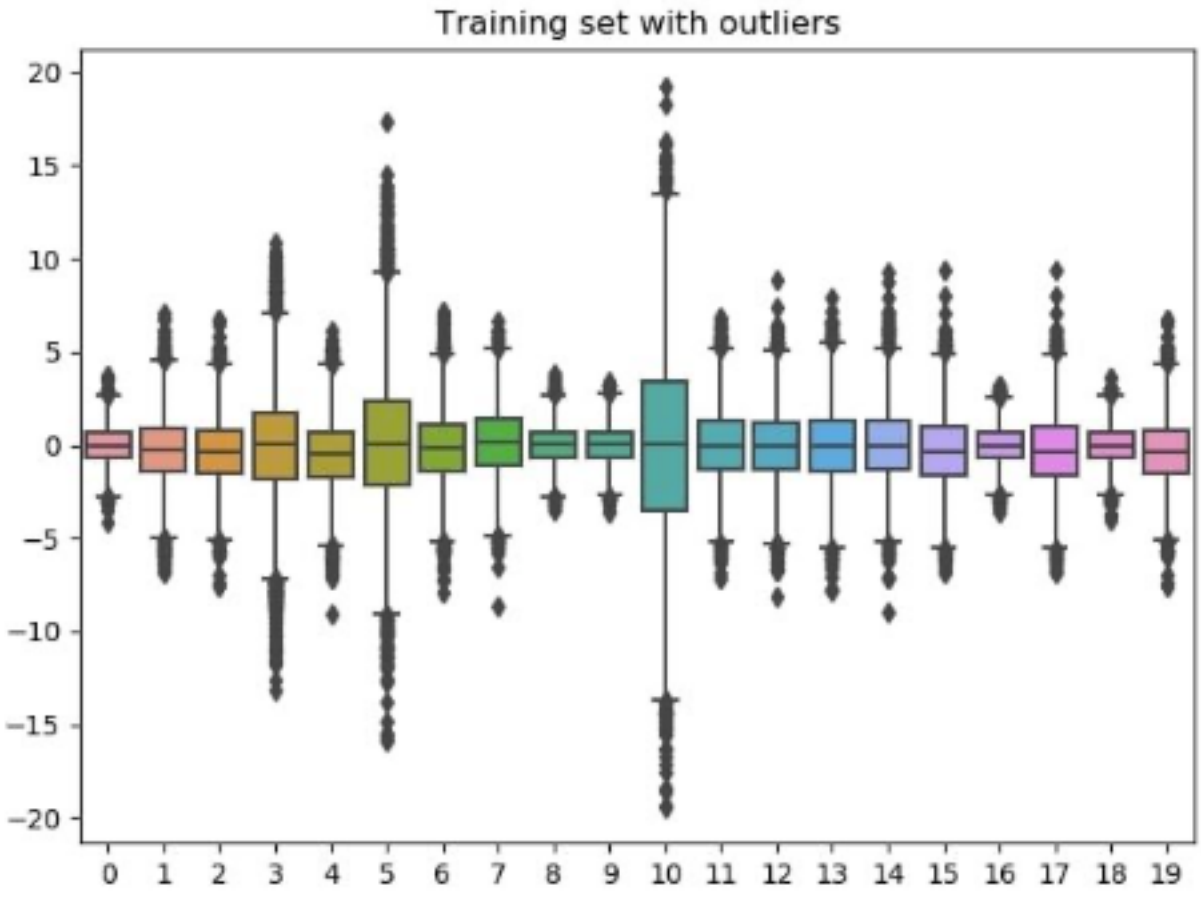


Figura 1: proporzione classi nel dataset

Per avviare una sessione di addestramento e di testing della SVM, abbiamo suddiviso il dataset in training set (80% dei dati) e test set (il restante 20%). Il metodo utilizzato, dalla libreria sklearn, ha permesso di avere questa suddivisione casualmente, rispettando le proporzioni tra le classi. La riproduzione delle La riproducibilità delle classi è stata garantita da un valore costante del parametro random_state.

Notiamo anche che nel dataset vi sono alcuni dati mancanti, mentre altri valori sono anomali (*outlier*), dunque entrambi verranno trattati adeguatamente nella fase seguente.



Pre-processing dei dati.

Il primo passaggio nella preparazione dati è stato l'identificazione e la sostituzione dei valori mancanti, mediante l'algoritmo *K-Nearest-Neighbors*, che prevede la sostituzione del valore con uno nuovo, calcolato partendo dai vicini del punto interessato. In particolare, il risultato migliore dell'algoritmo è quello che sceglie 2 vicini come riferimento per il parametro.

Per ripulire il dataset dagli *outlier*, il primo passo necessario è definire il limite, tale che un valore è definito anomalo. Solo successivamente si può decidere come sostituire tali valori.

La nostra definizione di *outlier multivariati* è stata ottenuta mediante un metodo di clustering, il DBSCAN, basato sulla densità. Esso trova un numero di cluster a partire dalla distribuzione della densità stimata dei nodi corrispondenti. È settato con i parametri *eps=6.6*, *min_samples=10, n_jobs = -1*.
Così facendo abbiamo ottenuto un ragionevole numero di *outlier* multivariati (139) nel nostro training set, rispetto al numero totale dei dati presenti.

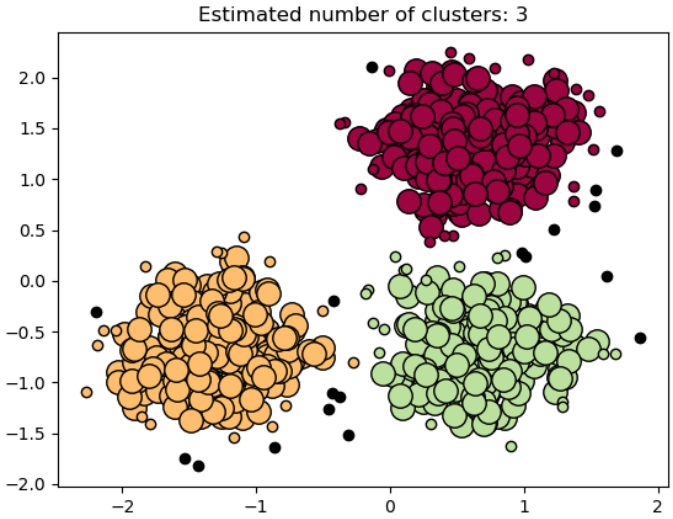


Figura 3: Esempio (dal web) di DBSCAN

Gli *outlier univariati* invece sono stati trattati nel primo step della *pipeline* attraverso *KNNReplacerIQR*.

Applichiamo uno *SMOTE()* sul *training set*: è un approccio per affrontare i set di dati squilibrati che prevede di sovracampionare la classe di minoranza. L'approccio più semplice prevede la duplicazione di esempi nella classe di minoranza, sebbene questi esempi non aggiungano nuove informazioni al modello. Invece, nuovi esempi possono essere sintetizzati dagli esempi esistenti.

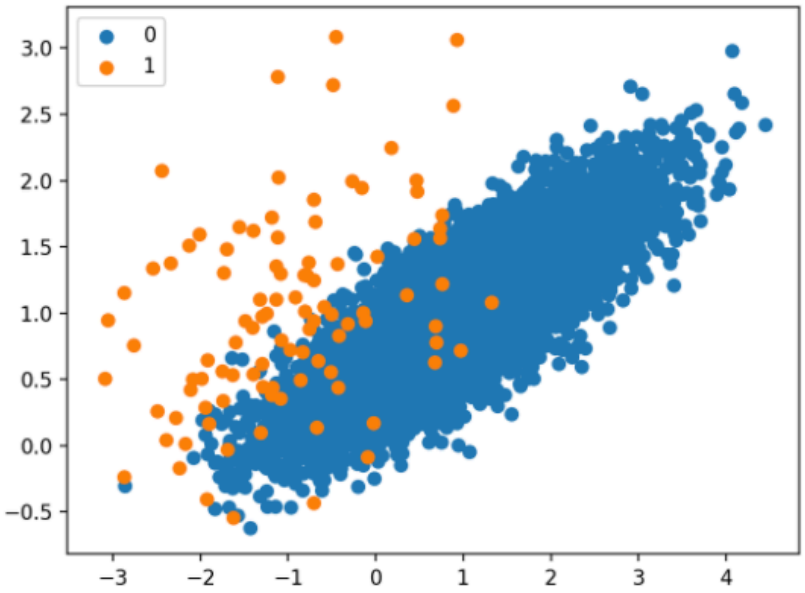


Figura 2: Dataset prima di SMOTE() (dal web).

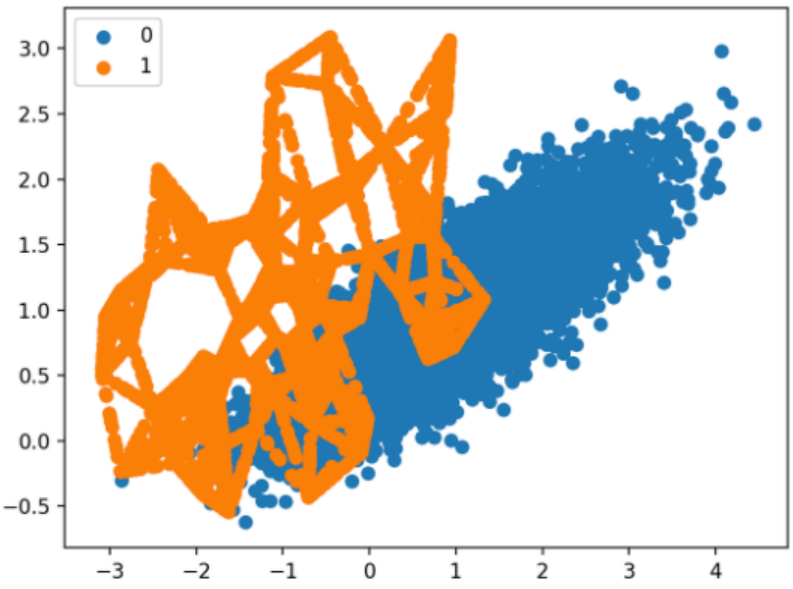


Figura 3: Dataset successivo allo SMOTE() (dal web).

Descrizione *pipeline*

1) Il rimpiazzo di valori nulli e anomali è stato effettuato dalla classe *KNNReplacerIQR*, che estende la classe *KNNImputer*, aggiungendo i metodi di *fit()* e *transform()* per poterla utilizzare come istruzione in *pipeline*.
   Essa sfrutta l'approccio dello scarto interquantile. Misurando la variabilità dei dati, definiamo gli *outlier* come punti che si trovano al di sotto di Q1 - 1,5 IQR o al di sopra di Q3 + 1,5 IQR.

2) Dal momento che abbiamo scelto un *BaggingClassifier* (descritto successivamente) con base estimator *QDA* (*Quadratic Discriminant Analysis*), abbiamo effettuato come secondo passo di *pipeline* una *PolynomialFeatures* di grado 3.

3) Il terzo passo della *pipeline* è stato lo scaling dei dati (*StandardScaler*) da *sklearn*, al fine di mappare i valori delle feature in un range omogeneo senza perdita di informazione, per evitare grandi differenze di scala tra i valori.

4) Dal momento che i dati presentano numerose feature (20), il quarto passo è una decomposizione di tipo *Principal Component Analysis* (*PCA*), per valutare l'influenza delle singole feature sulla classificazione del dato.

5) Il penultimo passo, prima della classificazione, abbiamo la *feature selection*. I nostri dati sono di tipo *labeled*, piuttosto che di una quantità da prevedere, perciò abbiamo escluso una riduzione dimensionale, un modello in regressione e un modello di clustering. Inoltre, data la mole dei dati, abbiamo optato per un modello *LinearSVC* che ha avuto la miglior perfomance rispetto a un *SDGClassifier* (con e senza kernel approximation) o un *KNeighbors Classifier*.

6) Come anticipato, in fondo alla *pipeline* abbiamo il classificatore, *Bagging Classifier*, con base estimator *QDA*.

Il *Bagging* è un algoritmo di apprendimento automatico dell'insieme che combina le previsioni di molti alberi decisionali.
Esso fornisce la base per un intero campo di algoritmi di tipo albero decisionale.
È stata la scelta più efficiente rispetto a classificatori *Random Forest, KNN,* o classificatori con diversi tipi di kernel (lineare, polinomiale o gaussiano).
La base del classificatore è lo stimatore QDA che deriva da semplici modelli probabilistici. Esso modella la distribuzione condizionale di classe dei dati P (X | y = k)per ciascuna classe K. Le previsioni possono quindi essere ottenute utilizzando la regola di Bayes:

$$P(y = k|X) = \frac{P(X|y = k)P(y = k)}{P(X)} = \frac{P(X|y = k)P(y = k)}{\sum_l P(X|y = l) \cdot P(y = l)}$$

e selezioniamo la classe K che massimizza questa probabilità condizionata.

Risultato finale

La Support Vector Machine ottenuta, è stata in grado di garantire uno score di **0.9198** sulla porzione di dati separata inizialmente. Soddisfatti del risultato ottenuto, la SVM è stata addestrata nuovamente sull'intero dataset, dove il risultato ottenuto è pari a **0.9305**. I risultati sono evidenziati dalle *Confusion Matrix* nelle figure allegate in seguito.

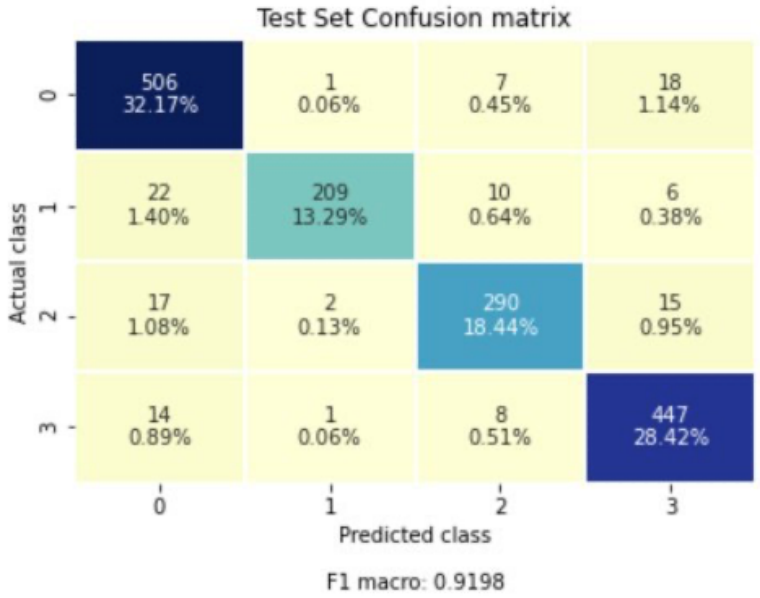


Figura 4: Matrice di confusione (test set).

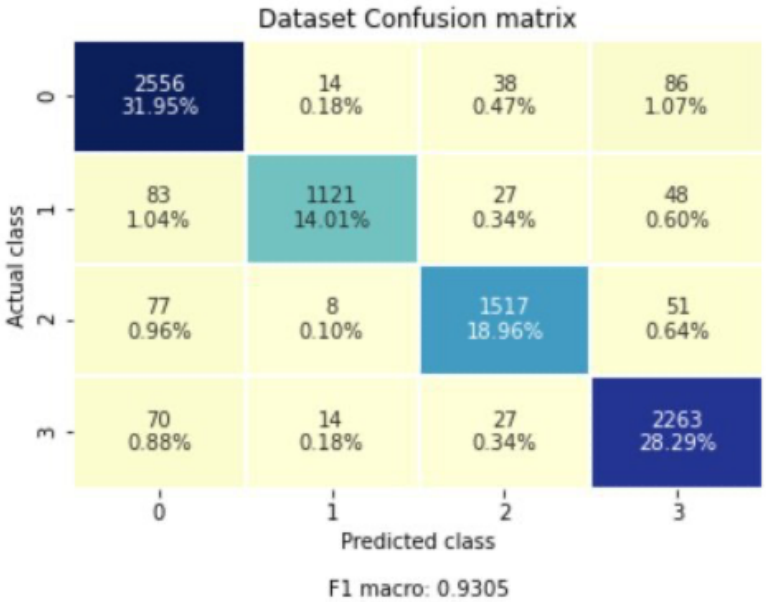


Figura 5: Matrice di confusione (dataset).