Metodi di Ottimizzazione per Big Data

Progetto A.A 2019/2020

La Delfa Gabriele

Indice:

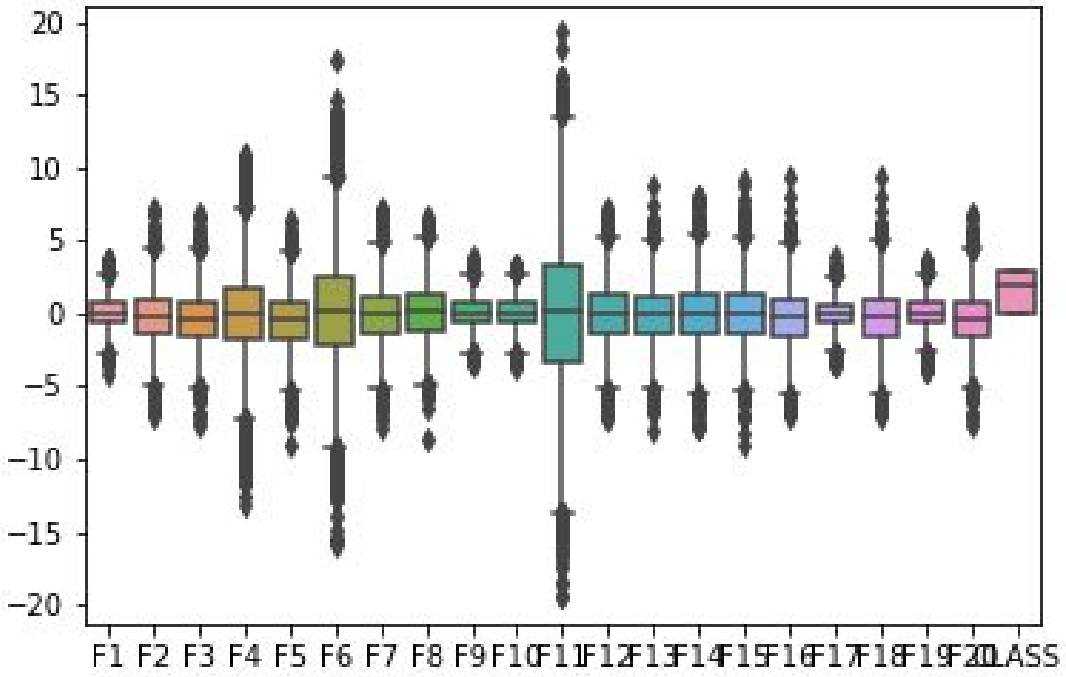
Obiettivo:

L'obiettivo del progetto è risolvere un problema di classificazione multi-classe con 4 classi, utilizzando algoritmi di machine Learning. Il classificatore deve prevedere la classe corretta sulla base delle 20 caratteristiche di ciascun'istanza. La metrica di valutazione del progetto è la f1-macro.

Progettazione:

Come prima cosa si sono importate tutte le librerie necessarie per modellare i dati e per le fasi di pre-processing e feature-selection.

Pre-processing



Per una prima esplorazione dei dati è stato analizzato un grafico che riporta i box-plot di tutte le features, e, attraverso l'osservazione delle distribuzioni delle variabili, si nota che vi è una numerosa presenza di **outlier**. Proprio per questo motivo, si e' deciso di non intervenire su essi al fine di evitare un'importante riduzione dei dati.

Dopo aver letto il dataset, controllando i **NaN** presenti nel dataset, notiamo che la proporzione di questi, sia per features che per osservazioni, è molto bassa, quindi per gestirli non sono stati eliminate né righe né colonne. I valori mancanti sono quindi stati riempiti nel dataset con il valore della mediana, per cercare di "bilanciare" questa forte presenza di outlier: la media, essendo uno stimatore non robusto, in questo caso avrebbe potuto danneggiare le performance del modello. Successivamente, il dataset è stato separato in training e test set con una proporzione 80:20.

Osservando il training set è stato notato una frequenza molto alta di una classe ed una forte sottorappresentazione delle altre. Per evitare un bias predittivo del modello verso le classi più frequenti nel dataset, è stato effettuato un bilanciamento del dataset. Il **bilanciamento** e' stato fatto attraverso **SMOTE**, che non va ad effettuare un semplice over-sampling duplicando le osservazioni appartenenti alle classi meno rappresentate, ma va invece a sintetizzare sulla base delle osservazioni già viste nei nuovi esempi relativi alle classi sottorappresentate.

Sul training set sono stati costruiti, “fittati”, gli ***Scaler***. I tipi di scaler utilizzati sui dati sono stati diversi: Normalizer - RobustScaler - StandardScaler - MinMaxScaler.

Sotto possiamo vedere le performance dei vari modelli per i diversi scaler presi in considerazione:

Scaler: **StandardScaler**
F1 per **RF** = 0.7430621459246365
F1 per **DT** = 0.5958723695668293
F1 per **SVM** = 0.8250540574393442
F1 per **MLP** = 0.8517646645631137
F1 per **XGboost** = 0.7489474583104999
F1 per **KNeighbors** = 0.7305366435506961

Scaler: **MinMaxScaler**
F1 per **RF** = 0.7495872155444695
F1 per **DT** = 0.6019694037560086
F1 per **SVM** = 0.8206463280410935
F1 per **MLP** = 0.8744393500981349
F1 per **XGboost** = 0.7510932504526752
F1 per **KNeighbors** = 0.7640522360099877

Scaler: **RobustScaler**
F1 per **RF** = 0.7469205474787752
F1 per **DT** = 0.6111220529843471
F1 per **SVM** = 0.8268447622043209
F1 per **MLP** = 0.8670399589308578
F1 per **XGboost** = 0.7245429450849343
F1 per **KNeighbors** = 0.7262041968737603

Scaler: **Normalizer**

F1 per **RF** = 0.7534126384970741
F1 per **DT** = 0.582137453703225
F1 per **SVM** = 0.831612493669875
F1 per **MLP** = 0.8560655751999657
F1 per **XGboost** = 0.7397809740579734
F1 per **KNeighbors** = 0.7649222889763431

È stato, quindi, scelto il MinMaxScaler, poiché è quello che ha fornito i risultati migliori.

Feature Selection

Sono state utilizzate le tecniche del RFE e del KBest, utilizzando per quest'ultimo le funzioni chi2 e f_classif. I risultati migliori sono stati raggiunti utilizzando RFE.

I classificatori che sono stati utilizzati sono:

- **MLP**
- **SVM**
- **RF**
- **XGboost**

MLP:

Sono stati considerati i seguenti parametri:

max_iter – activation – hidden_layer_sizes – shuffle – learning_rate – learning_rate_init – solver – warm_start –early_stopping – momentum – n_iter_no_change.

Il modello che ha dato il risultato migliore è stato il seguente:

MLP: param max_iter = 3500, activation = 'relu', hidden_layer_sizes=(548, 250, 125), shuffle = True, learning_rate = 'adaptive', learning_rate_init = 0.008, solver = 'sgd', warm_start = True, early_stopping = True, momentum = 0.85, n_iter_no_change = 45
F1 per **MLP** = **0.8844886779375569**

SVM

Sono stati considerati i seguenti parametri:

Kernel – C – Gamma – Decision_function_shape.

Il modello che ha dato il risultato migliore è stato il seguente:

SVM: param C = 200, kernel = 'rbf', decision_function_shape = 'ovo', gamma = 0.5
F1 per **SVM** = **0.8448632047678415**

Decision Tree

Sono stati considerati i seguenti parametri

Criterion – Splitter – Max_features – Min_samples_leaf – Max_depth – Min_samples_split.

Il modello che ha dato il risultato migliore è stato il seguente:

DT: param criterion = 'entropy', max_depth = 50, splitter = 'best', max_features = None, min_samples_leaf = 1, min_samples_split = 2
F1 per **DT** = **0.6196128786751702**

RF

Sono stati considerati gli stessi parametri del DT, con l'aggiunta di un ulteriore parametro

N_estimators.

Il modello che ha dato il risultato migliore è stato il seguente:

RF: param criterion = 'entropy', max_depth = 125, max_features = None, min_samples_leaf = 1, min_samples_split = 2, n_estimators = 400
F1 per **RF** = **0.7983518844941928**

XGboost

Sono stati considerati i seguenti parametri:

learning_rate – n_estimators – max_depth – booster.

Il modello che ha dato il risultato migliore è stato il seguente:

XGboost: param earning_rate = 0.01, n_estimators =500, max_depth = 100, booster = 'gbtree'
F1 per **XGboost** = **0.7787323199317618**

KNeighbors

Sono stati considerati i seguenti parametri

N_neighbors – Weights – Algorithm – Leaf_size – P.

Il modello che ha dato il risultato migliore è stato il seguente:

KNeighbors: param n_neighbors = 5, weights = 'uniform', algorithm = 'auto', leaf_size = 30, p = 5
F1 per **KNeighbors** = **0.784862408546619**

Conclusione

Si può notare come il migliore classificatore trovato sia MLP, che ha permesso di raggiungere dei risultati soddisfacenti.