Statlog - German credit
Report per l'Esame di Fondameti di Machine Learning

Alessandro Pansera

152543
Ing. Informatica
282423@studenti.unimore.it

Abstract

Il sistema bancario moderno si basa sulla cessione di prestiti ai propri correntisti al fine di realizzare profitto attraverso interessi e fidelizzare il cliente. Alla base del modello di business descritto c'è la necessità di ponderare scrupolosamente l'affidabilità creditizia di un cliente affinché non risulti in futuro insolvente. Il Machine Learning permette di risolvere problemi di questa natura mediante la classificazione tramite la quale offrire un supporto al normale svolgimento delle mansioni di un operatore bancario.

1 Dataset

Il dataset è composto da dati bancari raccolti nel 1994 in Germania, è stato donato all'università di Amburgo e pubblicato nel 2019 all'interno dell'archivio dell'università della California. All'interno del dataset troviamo sia features categoriche che numeriche, non sono presenti sample con attributi nulli. Essendo il dataset, per la natura dei features che lo compongono, multivariato, occorre applicare delle tecniche di replacing per permettere il training di un classificatore. Non è necessario eseguire operazioni di normalizzazione o standardizzazione in quanto gli importi di natura maggiore si suppone, per la definizione del dominio delle features, non superino i 2 ordini di grandezza rispetto al valore minimo ammesso.
Le 13 variabili categoriche verranno tutte rimpiazzate mediante operazioni di replacing basate sul codice presente per ogni categoria presente:

```
# original category
cat = "A201"
# replaced category
new_cat = int(cat[-2:])
```

Per consentire la modifica dinamica delle codifiche con cui eseguire il replacing è stato predisposto un sistema basato su file .json importabili i quali indicano per ogni feature le regole di sostituzione. L'obbiettivo è quello non di variare il codice ma di cambiare il .json importato per il replace rendendo il processo più rapido e meno soggtto a potenziali errori.

1.1 Var. Target

La variabile target presente all'interno del dataset ammette 2 valori interi che sono 1 ('Good borrower') e 2 ('Bad borrower'). Considerando la natura del problema si possono eseguire le predizioni mediante un modello di classificazione binaria. Non è necessario eseguire il replace per i valori della variabile target.

1.2 Considerazioni

Prima di scegliere il tipo di algoritmo di suprvised learning da impiegare per la classificazione binaria occorre studiare con attenzione il dataset che si ha a disposizione. Il primo problema facilmente osservabile risiede nel mancato bilanciamento delle classi all'interno del dataset. La classe 'Good Borrower' riporta 700 samples mentre 'Bad Borrower' 300 samples; l'entropia rilevata per i dataset vale:

$$H(T) = -\sum_{n=1}^{i} p_n * \log_2 p_n = 0,88$$

Per evitare problemi legati a predizioni 'positive' sbagliate occorre applicare delle tecniche di over-sampling o under-sampling che verranno elencate alla voce 2.1.

Per poter creare un algoritmo di Machine Learning realmente applicabile ad un contesto lavorativo è importante valutare anche aspetti indipendenti dalla pura natura dei dati. Il classificatore che si vuole realizzare deve essere in grado predirre con precisione 'Good borrower' e 'Bad borrower'; a livello applicativo per un istituto di credito è più importante predirre con precisione i casi appartenenti alla prima classe che alla seconda. Se infatti sbagliamo la predizione legata ad un 'Finto good borrower' rischiamo di incorrere in perdite economiche al contrario di un 'Finto bad borrower' per cui la perdita economica è solo ipotetica.
Sarà necessario in fase di training andare allora a considerare le problematiche appena citate per andare a realizzare un classificatore in grado di fornire delle buone performance.

2 Modello

Come già detto il problema di Machine Learning trattato è di classficazione binaria. Per realizzare un modello di classificazione binaria sono disponibili numerose tecniche ma quelle implementate per la realizzazione del progetto sono la Logical Regression, il K-NearestNeighbour e il Random Forest che è un algoritmo di ensemble basato su Alberi Decisionali.
Se la Logical Regression ragiona in maniera simile alla Linear Regression, per cui fornito un sigmoide viene stimata la percentuale di appartenenza ad una classe e mediante una soglia viene scelta quella più adatta, il Random Forest opera in maniera differente. L'idea alla base del Random Forest è quella di creare più modelli (alberi decisionali) scorrelati da confrontare per raggiungere una buona predizione. La tecnica con cui vengono generati gli alberi decisionali è detta 'bagging' e consiste nel prendere un set di dati iniziali e crearne di nuovi tramite campionamento casuale con rimpiazzo. Sui nuovi set di dati creati verranno eseguite delle predizioni e quella statisticamente più presente verrà assunta come corretta. L'ultimo algoritmo è il KNN che, come i 'weak model' del Random Forest, non è parametrico. Si basa sulla ricerca dei k punti più vicini al punto per cui si vuole eseguire la predizione e sulla base del risultato più votato dai vicini verrà decisa la predizione.

2.1 Pre-processing

Come detto in fase di descrizione del dataset sarà necessario eseguire delle operazioni di preprocessing sui dati oltre al replacing. All'interno del progetto sono state implementate diverse soluzioni per far fronte allo sbilanciamento tra classi che costituiscono il dominio della variabile target:

- Under-sampling: tecnica basata sul ridurre i samples appartenenti alla classe dominante.
  - Near miss: è un algoritmo di under-sampling basato sulla rimozione randomica dei samples appartenenti, per il nostro problema, alla classe 'Good borrower' che è in maggioranza.
- Over-sampling: consiste nel generare fedelmente dei samples sulla base di assunzioni relative alla classe di appartenenza. Gli algoritmi implementati per il progetto appartengono alla famiglia dei 'Synthetic Minority Oversampling Technique' . Gli algoritmi SMOTE non

operano semplicemente duplicando i dati bensì selezionano i samples più simili basandosi sui features e ne generano di nuovi sulla base dell'interpolazione tra i samples selezionati nello spazio multidimensionale.

- K SMOTE: attraverso l'algoritmo K-means vengono generati dei cluster all'interno dei quali vengono selezionati i sample che verranno usati come base per l'over-sampling. Successivamente vengono scartati i cluster in cui abbiamo una sproporzione tra sample appartenenti alla classe maggioritaria e minoritaria. Per i cluster rimanenti applico l'algoritmo SMOTE base basato sul modello K-Neighbour. K-means opera creando un numero K di cluster per i quali cerca un 'centroide' in modo iterativo e in seguito collega ad ogni cluster i sample sulla base di un principio di vicinanza.
- SVM SMOTE: l'algoritmo è un successore di SMOTE e in questo caso genera dei dati sintetici sul confine tra due classi, confine rilevato grazie alle SVM.
- ADASYN: il principio di funzionamento è differente dai precedenti due in quanto vengono creati dei dati sintetici negli spazi multidimensionali in cui la classe minoritaria è poco densa.

2.2 Performance

Per visualizzare i risultati comparativi tra i vari modelli, dopo aver eseguito CrossValidation.py (working directory codice/), si può consultare il file log/master.log che riporta per ogni misura delle performance analizzata i 10 migliori risultati registrati tra tutti i modelli. La tecnica di cross-validation utilizzata, come tecnica per partizionare il dataset, la stratificazione. Per ogni tupla (¡modello¿, ¡tecnica di over/under-sampling¿) vengono testate tutte le tecniche di bilanciamento del dataset e a seguito di ciascuna fase di training (in totale avremo 12 modelli) verranno eseguiti 10 test su cui verrà eseguita una media.

2.3 Analisi performance

Come già asserito alla voce 1.2 è altamente vincolante la precisione con cui vengono predetti i 'Good borrower'. Di seguito con FP verranno indicati i 'False good borrower' e con TP i 'True good borrower'. All'interno delle Confusion Matrix plottate il risultato desiderato si traduce in un gap, il più elevato possibile, tra gli elementi all'interno della prima colonna (plot riportati su PyCharm). E' comunque importante avere una buona precisione in fase di predizione dei TN altrimenti il modello, a livello generale, perderebbe di precisione ed è un risultato non accettabile. Fissato l'obbiettivo che si vuole raggiungere si può procedere allo studio di quelli che sono gli score a cui si è maggiormente interessati per la valutazione della bontà del modello. Se l'F1 score è più un indicatore generico che è sempre buona norma valutare per stimare la bontà generale di un classificatore F.D.R., Precision e Recall sono specifici per il tipo di problema in questione. L'F.D.R., 'False discovery rate', permette di valutare il rapporto tra FP e TP; la metrica è perciò una delle principali da considerare in quanto valuta esattamente quello che è il principale vincolo del problema di classificazione in questione.

$$F1score = \frac{2 * precision * recall}{precision + recall}$$

$$F.D.R. = \frac{FP}{TP + FP}$$

Precision e Recall misurano entrambe la qualità con cui vengono rilevati i TP e FP. Se la Precision misura la precisione delle predizioni positive invece la Recall ne misura la completezza, di seguito sono riportate le formule di calcolo.

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

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

Precision e Recall sono inversamente proporzionali ed è possibile osservarlo anche dai test eseguiti sul classificatore. L'esempio più apprezzabile è legato al confronto tra i risultati ottenuti con la Logical Regression e il Random Forest applicando l'SVM SMOTE per l'over-sampling.
L'inversa proporzionalità è legata al denominatore per cui se salgono gli FP decrescono i FN per quella che è la natura della classificazione binaria. Più si è precisi a definire ciò che si cerca minore è la completezza delle predizioni positive.Per la tipologia di problema in questione, considerando soprattutto il settore in cui si opera, si predilige una buona Precision a discapito di una miglior Recall.

Le soluzioni migliori per la tipologia di problema considerato sono raggiunte applicando l'algoritmo di Logical Regression impiegando l'SVM SMOTE in fase di pre-processing per l'over-sampling oppure impiegando il Random Forest facendo uso dell'SVM SMOTE o K SMOTE. Le strade appena citate presentano differenze legate agli score che seguono:

- FDR
- Precision/Recall

Gli score citati servono per valutare il giusto tradeoff per la soluzione che si vuole applicare. Per quanto concerne l'F1-score i tre modelli indicati riportano uno score compreso tra l'85% e l'83%.
Il Random Forest presenta, con SVM/K-SMOTE, una Recall molto elevata a discapito di una Precision nettamente inferiore (circa del 10%) rispetto al medesimo valore raggiunto mediante la Linear Regression. L'upgrade legato all'applicazione della Logical Regression, combinata con l'SVM SMOTE, è riconducibile ad un incremento della Precision e ad una riduzione dell'FDR.
Applicando la Logical Regression con l'SVM SMOTE in fase di pre-processing è possibile apprezzare una Precision media superiore del 5% rispetto alla miglior performance media indicata all'interno del sito che rende fruibile il dataset.
Per ricapitolare, considerando tutti i modelli generati, quelli che hanno generato i risultati migliori fanno uso della Logical Regression impiegando una Support Vector Machine per l'over-sampling producendo risultati soddisfacenti per tutti gli score di riferimento; ciò che maggiormente influisce nella valutazione sono il massimo punteggio ottenuto mediamente per la Precision ed il minimo relativamente all'FDR.
Un'ulteriore soluzione apprezzabile, ma meno puntale, è prodotta del Random Forest applicando, per l'over-sampling, l'SVM-SMOTE oppure il K-SMOTE.
Eseguendo la cross-validation numerose volte può capitare che alcuni modelli generino risultati sporadicamente ottimi per singole metriche perciò non sono stati menzionati in quanto non stabili e globalmente non affidabili. Per esempio il Random Forest con l'under-sampling ha prodotto, in fase di sperimentale, un FDR del 12%, a discapito di F1-Score nettamente inferiore rispetto ai modelli citati in precedenza, peccando di costanza nelle rilevazioni successive. Ciò significa che i modelli proposti, a seguito di numerosi test, hanno mantenuto delle performance costanti sotto tutti i punti di vista; è preferibile optare per soluzioni costanti e bilanciate rispetto a soluzioni che presentano score sbilanciati.
I grafici che seguono illustrano le performance raggiunte riportando, per ogni metrica, le top 10 valutazioni raggiunte tra le 12 possibili combinazioni disponibili. E' inoltre presente un istogramma che, tramite una media pesata tra tutti gli score, valuta i top 5 modelli applicabili. I pesi con cui è stata realizzata la classifica globale sono riportati di seguito:

- F1: 5
- FDR: 7
- Precision: 9
- Recall: 3

Tutte le codifiche presenti all'interno degli istogrammi sono riportate alla voce 2.4. .

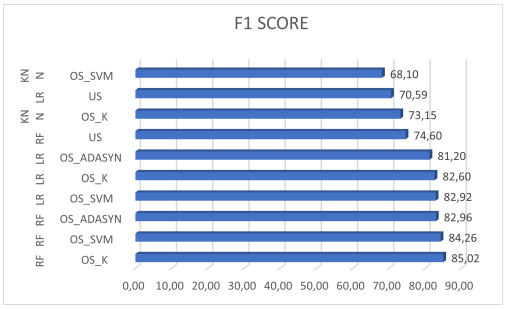

Figure 1: TOP 10 - F1

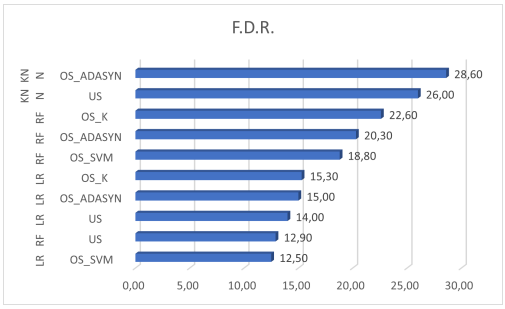

Figure 2: TOP 10 - FDR

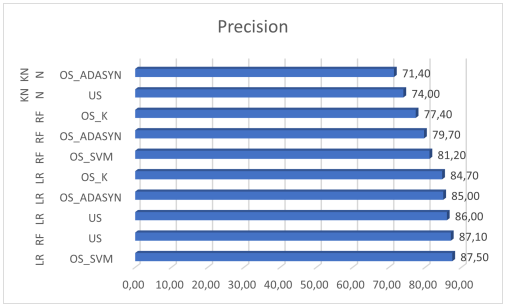

Figure 3: TOP 10 - Precision

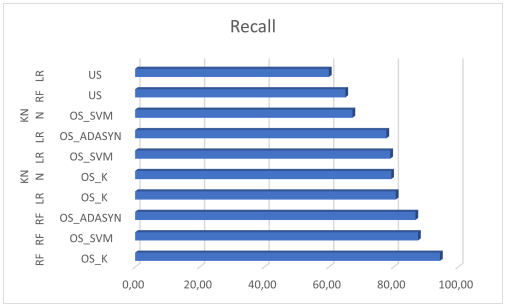

Figure 4: TOP 10 - Recall

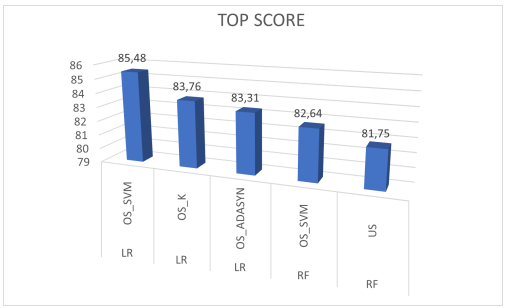

Figure 5: TOP 5 - Global

2.4 Implementazione

Il modello di classificazione può essere addestrato attraverso differenti combinazioni tra algoritmi e tecniche di under-sampling/over-sampling. Per permettere un rapido interscambio tra tecniche e algoritmi usati è stato implementato, tramite incapsulamento, il Factory Pattern che permette di pilotare tramite 2 stringhe tutti i fattori che portano all'addestramento del modello. E' stata inoltre prevista la possibilità di salvare il classificatore affinché sia utilizzabile in seguito al training.
Seguono le codifiche presenti all'interno del progetto per rappresentare modelli, tecniche di over-sampling e under-sampling:

- Modelli:
  - RF: Random Forest
  - LR: Logical Regression
  - KNN: K-Nearest Neighbour
- Tecniche di sampling:
  - US: under-sampling con Near Miss
  - OS_K: over-sampling con K-SMOTE

  - OS_SVM: over-sampling con SVM-SMOTE
  - OS_ADASYN: over-sampling con ADASYN

L'algoritmo Random Forest permette di definire dei pesi da associare alle classi; verranno applicati i pesi 1 per la classe 'Good borrower' e 2 per la classe 'Bad borrower'. A seguito di numerosi training la scelta si è dimostrata vincente permettendo di migliorare notevolmente tutte e 4 le voci con cui valutare i classificatori prodotti attraverso differenti tecniche.
Tutte le operazioni eseguite e le relative perfomance per ciascuna coppia modello/tecnica di sampling verranno registrate tramite appositi file di log dedicati presenti all'interno della cartella log/. Ogni classificatore realizzato verrà salvato in formato .joblib all'interno della cartella classifier/.
Il download e la normalizzazione dei dati sono definiti in apposite classi che favoriscono l'intercambiabilità delle operazioni senza dover ristrutturare l'intero codice.
Non è stata eseguita una ricerca degli iper-parametri ottimizzati utilizzando, dove necessario, i valori di default proposti da sklearn; di seguito, per ogni algoritmo, sono riportati i principali:

- Random Forest:
  - n. of trees: 100
  - criterio di ottimizzazione dello split: gini
  - profondità max. albero: 50
  - task eseguiti in parallelo: 10
  - peso sulle classi: 5 sui 'bad borrower' per minimizzare l'FDR
- Logical Regression
  - penalty: L2
  - algoritmo di ottimizzazione: newton-cholesky
  - iteration num. max 100
- KNN
  - n. of neighbours: 5

I plot riportano, su PyCharm, le confusion matrix per ogni modello; all'interno delle righe troviamo li valori reali mentre nelle colonne sono riportati i valori predetti. Con 0 indico i 'Good borrower' mentre con 1 indico i 'Bad borrower'. E' possibile visualizzare, tramite due file excel contenuti nella cartella dataset/ il dataset scaricato pre e post normalizzazione (data.xlsx e data_normalized.xlsx). Si noti che l'over-sampling viene fatto solo sul dataset di training, non su quello usato in fase di validazione, per avere dati non legati alle tecniche di sampling nella fase di cross-validation.
Non sono state implementate tecniche di regolarizzazione in quanto non necessarie per i modelli applicati; per la stessa ragione non è stata applicata la feature selection nei modelli indicati.
A livello pratico i modelli implementati sono stati scelti in maniera empirica cercando di raccogliere tutti i modelli affrontati a lezione: la Logical Regression è stata introdotta per fornire un esempio di modello parametrico, il KNN per portare un esempio di modelli non parametrici e il Random Forest per portare un esempio di ensembling.
La classe Training permette, tramite i parametri passabili al proprio costruttore, indicati anche in precedenza, di eseguire rapidamente vari test tra tutti i modelli e le relative tecniche di sampling implementate. La facilità con cui possono essere generati i classificatori permette di avere un codice compatto e facile da riutilizzare anche per altri progetti.