ANALISI DEL DATASET ‘BANK CHURNERS’ - Andrea Corongiu

INTRODUZIONE

Il dataset ‘Bank Churners’ scelto dal sito Kaggle.com offre la possibilità di svolgere un’analisi di classificazione binaria tra i soggetti che hanno abbandonato la banca, detti ‘churners’, e coloro che sono tuttora clienti. L’analisi è stata svolta servendosi di Pyspark, tramite Jupyter notebook. L’obiettivo è stato trovare un classificatore binario capace di individuare correttamente almeno il 95% dei churners, mantenendo un livello accettabile di falsi positivi, e successivamente svolgere una cluster analysis.

DESCRIZIONE DATASET

Il dataset conta 10127 osservazioni per 19 predittori, di cui la maggior parte sono variabili quantitative, sono presenti variabili qualitative sia ordinali che nominali. Per una descrizione esaustiva dei predittori si veda al link: https://www.kaggle.com/datasets/sakshigoyal7/credit-card-customers

FASE DI CARICAMENTO DATASET

Il progetto è stato svolto tramite la definizione di cinque classi:

‘Loader’ per creare il dataframe dal file csv, tramite la ridefinizione del metodo lshift, che prende in ingresso il file csv;

Solitamente un dataset non è pronto per essere analizzato tramite algoritmi di machine learning ma deve essere pre-processato. Questa fase propedeutica è svolta da una classe specifica, che utilizza una funzione per svolgere la codifica delle variabili categoriche ordinali secondo uno schema.

PREPROCESSING

La classe che si occupa del preprocessing è ‘Preprocessor’: permette di compiere le principali operazioni di pre-processing del dataset:

controllare la presenza o meno di dati mancanti quali valori na, null ecc.;

rimuovere le colonne giudicate non utili ai fini delle analisi successive;

vedere le statistiche principali per ogni variabile, sia qualitativa che quantitativa;

svolgere la codifica delle variabili categoriche ordinali secondo uno schema stabilito dall'analista. Tale operazione è svolta tramite l’utilizzo della funzione ‘ordinal_encode’ che utilizza la classe OrdinalEncoder della libreria ‘category_encoders’, quest’ultima trasforma un dataset di tipo Pandas e non Spark.

‘Preprocessor’ si occupa anche di: codificare le variabili categoriche nominali tramite ‘String indexer’ e successivamente ‘One Hot Encoder’.

Il metodo ‘assemble_and_scale’ consente di assemblare il vettore unico delle features tramite ‘VectorAssembler’ e scalare il vettore assemblato tramite ‘MinmaxScaler’. Tramite il parametro 'keep_cols' = 'sì' è possibile mantenere le colonne assemblate in maniera tale da poter controllare le distribuzioni una volta svolta la cluster analysis. Tale metodo restituisce il dataframe pronto per la fase successiva di classificazione.

ANALISI ESPLORATIVA

Come si nota dal metodo ‘summary’ della classe ‘Preprocessor’ la variabile dipendente ‘Attrition_Flag’ è fortemente sbilanciata verso la classe 'existing customer', circa l’84% delle osservazioni. Questo fatto è da tenere in considerazione nel momento in cui si svolge il train-test split del dataframe: lo split dovrà rispettare la distribuzione della variabile di risposta, dovrà essere ‘stratified’. Si potrebbe procedere anche tramite campionamento con reintroduzione per pareggiare le distribuzioni ma le performance ottenute dai classificatori sul dataframe originale sono state comunque soddisfacenti.

La classe ‘Graphics_maker’ permette di fare semplici grafici che mostrano le distribuzioni delle variabili quantitative e la loro similarità a una distribuzione normale tramite q-q plot; ‘Customer_Age’ è distribuita come una normale nei valori centrali, ma non è approssimata bene nelle code, ‘Months_on_book’ ha elevata curtosi, ‘Credit_limit’ ha la coda di destra lunga (cioè è sbilanciata a sinistra: media > mediana) ma i valori ampi sono comunque plausibili per la presenza di clienti ‘Platinum’, ‘Avg_Open_To_Buy’ sembra essere molto simile a ‘Credit_Limit’. ‘Total_Trans_Amt’ è molto sbilanciata a sinistra ma presenta tre campane.

CLASSIFICAZIONE

La classificazione si svolge tramite la classe ‘ExperimentHandler’, la quale restituisce tramite il metodo ‘start classification’ un dizionario con le statistiche dei classificatori desiderati. I classificatori vanno inseriti manualmente tramite il metodo ‘add_classifier’, al quale vanno passati il nome e la classe del classificatore desiderato. Dato che i classificatori sono presi dalla libreria ‘ml.classification’ di Pyspark presentano una struttura di base comune che viene sfruttata dal metodo ‘start classification’.

I classificatori utilizzati sono stati: Random Forest , regressione logistica, classificatore Naive Bayes e Support Vector classifier. Sono stati modificati i parametri della threshold per i classificatori che lo prevedono mentre per il Random Forest la modifica è avvenuta in maniera 'manuale', sfruttando la colonna ‘probability’: in questa maniera si stabilisce un valore soddisfacente di recall (tp / tp + fn) rispetto alla quota di falsi positivi e alla precision. Infatti, in casi di classificazione in cui l'interesse nell'individuare correttamente un'osservazione appartenente ad una classe piuttosto che all'altra è ampia, la recall è l'indice più importante, considerato insieme al false positive rate ed alla precision. Vi è sempre un trade off tra le due che è ben mostrato dalla roc curve. Per poter utilizzare la colonna ‘probability’ senza modificare il codice condiviso da tutti i classificatori nell’ ‘ExperimentHandler’ è stata creata la classe ‘MyRandomForest’ la quale, tramite il metodo ‘classify_with_threshold(threshold = float)’ permette di indicare la soglia della decision boundary.

Il miglior classificatore risulta essere il Random Forest, con una threshold al 25% di probabilità di essere churners, di seguito le statistiche rilevanti: la precision del 78%, la recall del 97% e una auc della curva roc del 0,96%.

CLUSTER ANALYSIS

Una volta svolta la classificazione è stato deciso di svolgere una cluster analysis. È stata implementata la classe ‘Clusterer’. Tramite il metodo 'perform cluster analysis' si ottengono le statistiche per 4 tipi di clusterizzazioni diverse: rispettivamente da 2 a 5 clusters. Tramite il metodo ‘take_best_clustering’ si sceglie la migliore in termini di silhouette (una misura di sintesi della distanza tra campioni di osservazioni appartenenti ad un cluster specifico dagli altri clusters che varia da 1 a –1) e si svolgono ulteriori indagini in base a tale clusterizzazione. Nel caso del dataset analizzato la migliore individua due clusters.

A questo punto bisogna capire cosa rappresentano effettivamente questi clusters: quali sono le caratteristiche delle osservazioni che appartengono a ciascun cluster? La divisione corrisponde a quella tra churners e non? Il metodo ‘take_best_clustering’ dà una risposta all’ultima domanda: la cluster analysis trova delle altre relazioni tra i dati che non coincidono con la separazione tra churners e clienti attuali; infatti, la distribuzione delle osservazioni tra i due clusters è bilanciata mentre quella churners vs clienti attuali è molto sbilanciata. Questo fatto è confermato dalla confusion matrix. Allora, tramite il metodo ‘get_clustering_insights’ è utile osservare per ogni variabile quantitativa alcune statistiche sintetiche come media, deviazione standard e percentili per avere un'idea generale del cliente tipo di ciascun cluster. La divisione è fortemente influenzata dalla variabile ‘Marital Status’: il cluster 0 è il gruppo di clienti sposati, mentre l’altro è composto da single, divorziati e condizione sconosciuta. Sorprendentemente la condizione economica del gruppo 1 è leggermente superiore a quella del cluster di sposati: la classe ‘Card category’ è leggermente più alta, di conseguenza anche il ‘Credit limit’ ma anche l’Avg_Open_To_Buy e ‘Total_Trans_Amt’. Anche se tale separazione non è utile per risolvere il problema di individuare i churners, può essere comunque utilizzata dalla banca per altri scopi commerciali.