Bank Marketing
Report per l'Esame di Machine Learning

Alberto Compagnoni

Abstract

Questo progetto ha come obiettivo l'analisi del dataset "Bank Marketing", contenente dati relativi alle campagne di marketing basate su chiamate telefoniche di un istituto bancario portoghese. In particolare si vuole predire, tramite classificazione binaria, se il cliente sottoscriverà o meno un deposito bancario a termine.

1 Introduzione

Per lo svolgimento del progetto, è stato scelto il dataset "Bank Marketing". Esso contiene dati relativi alle campagne di marketing (basate su telefonate) di un istituto bancario portoghese.

Il dataset in questione ha 4521 samples e 16 feature:

- feature numeriche: `age, balance, day, duration, campaign, pdays, previous`
- feature categoriche: `job, marital, education, default, housing, loan, contact, month, poutcome`

La ground truth è

- `y`: il cliente ha sottoscritto un deposito a termine? ("yes","no")

Avremo quindi un task di classificazione binaria: a partire dalle caratteristiche del cliente e dai dati riguardanti le ultime chiamate a esso associate, vogliamo predire se il cliente sottoscriverà o meno un deposito a termine.

1.1 Criterio di splitting

Si è deciso di dedicare l'80% dei sample per il training ed il restante 20% per il testing. Per ottenere risultati riproducibili ho impostato un seed, in modo tale che lo shuffling prima dello split venga fatto nello stesso modo ad ogni esecuzione. Dopo aver fatto lo split tratterò il dataset di training e quello di testing in modo completamente distinto, quindi applicherò le tecniche di preprocessing separatamente ad entrambi.

2 Exploratory Data Analysis

Dopo aver constatato che nel dataset di training non sono presenti valori nulli, effettuo il mapping del target vector y_{train} da valori categorici ("yes", "no") a discreti.

Dopodichè individuo nella design matrix X_{train} quali sono le feature categoriche e quali sono discrete (basandomi sul `dtype` delle colonne del `DataFrame`):

- per quelle categoriche viene effettuato l'encoding ottenendo valori discreti, al fine di renderle utilizzabili dai predittori di `sklearn`
- alle feature numeriche applico lo scaling (in particolare la standardizzazione), poichè alcuni degli algoritmi utilizzati in seguito fanno uso del Gradient Descent e senza scalamento avremmo aggiornamenti disomogenei per le componenti θ_j del parameter vector θ.

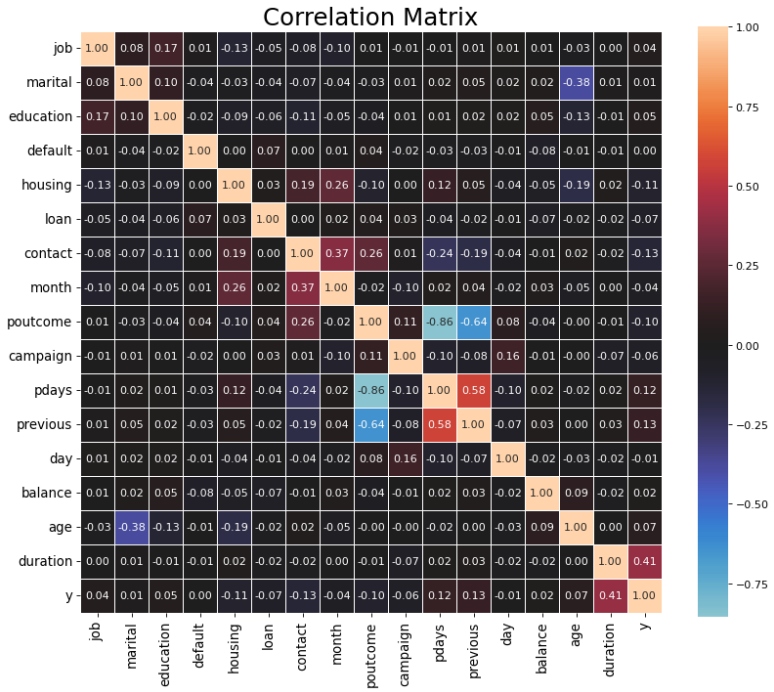

Figure 1: Correlogramma del dataset di training

Osservando la correlation matrix, notiamo che la feature maggiormente correlata con la decisione di sottoscrivere o meno un deposito è "duration". Infatti alle persone che hanno sottoscritto il deposito è tipicamente associata una durata dell'ultimo contatto telefonico più elevata.

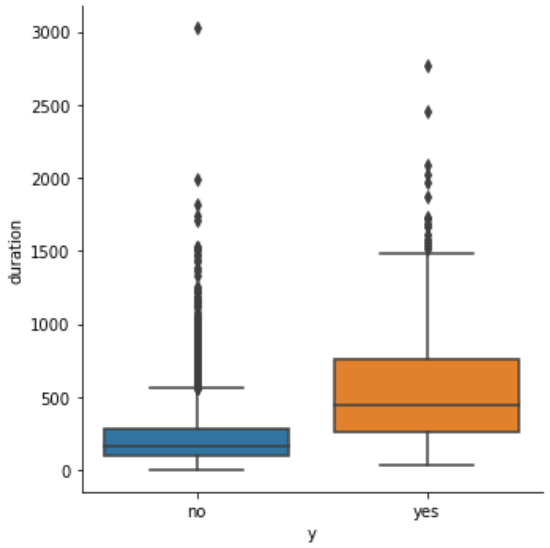

Figure 2: Durata in secondi dell'ultima chiamata in base alla sottoscrizione o meno del deposito a termine

Il codice del progetto genera inoltre altri grafici tra cui distribution plot, categorical plot e pairplot che per compattezza non ho inserito nel report.

3 Model Selection

Per il task di classificazione binaria ho scelto come candidati i seguenti modelli:

- **Logistic Regression**: classificatore adatto a trattare classi linearmente separabili e in grado di ottenere buone performance su dataset semplici. Necessita lo scalamento dei dati in quanto usa il Gradient Descent/Ascent.
- **K-Nearest Neighbors Classifier**: nonostante la necessità di avere in memoria l'intero dataset di training a inference time, ho scelto di includere questo modello per la sua semplicità e la non necessità di ottimizzare i parametri. Inoltre avendo un numero di feature e sample non eccessivo la velocità di training è risultata essere soddisfacente.
- **Decision Tree**: per la sua velocità a inference time e per le buone prestazioni in termini di accuracy su dataset semplici.
- **(Categorical) Naïve Bayes**: utilizza solo le feature categoriche, con l'assunzione che esse siano condizionalmente indipendenti tra loro. Ho utilizzato la regolarizzazione Laplace Smoothing (si veda `alpha`) per evitare di ottenere class-conditional probabilities $p(x_1, ..., x_d|y)$ nulle (causate da bin con valore 0).

3.1 Cross Validation: Grid Search

Innanzitutto uso il validation set per cercare la migliore combinazione di iperparametri in una griglia di possibilità mediante la Grid Search.

Gli iperparametri provati sono i seguenti:

- Logistic Regression: penalty $\in \{L1, L2\}$, C $\in \{10^{-5}, 5 \cdot 10^{-5}, 10^{-4}, 5 \cdot 10^{-4}, 1\}$
- K-Nearest Neighbors: k $\in \{1, 3, 5, 7, 9, 11, 13, 15\}$
- Decision Tree: criterio $\in \{gini, entropia\}$
- Naïve Bayes: fit-prior $\in \{True, False\}$

Ho ottenuto i seguenti iperparametri migliori per ogni classe di modelli:

```
###################### Grid Search ######################

Logistic Regression
Accuracy:   0.8061457402015847
    --> best value for hyperparameter "penalty":  l1
    --> best value for hyperparameter "C":  1

KNN
Accuracy:   0.8893793509242489
    --> best value for hyperparameter "n_neighbors":  9

DT
Accuracy:   0.8595107096734752
    --> best value for hyperparameter "criterion":  gini

Naive Bayes
Accuracy:   0.8827426392486799
    --> best value for hyperparameter "fit_prior":  True
```

Figure 3

3.2 Ensemble

Ho scelto inoltre di introdurre un modello ensemble, in particolare uno **StackingClassifier**, scegliendo come weak learner Logistic Regression, KNN e Decision Tree e come final estimator la Logistic Regression.

Lo scopo di questa aggiunta è quello di ottenere (sperabilmente) prestazioni migliori rispetto ai singoli base estimator.

Non ho incluso tra i weak learner il modello Naïve Bayes perchè esso necessita di essere addestrato usando solamente feature categoriche, quindi risulta difficle integrarlo nell'ensemble con gli altri modelli che vengono addestrati tenendo conto di tutte le feature.

3.3 Cross Validation: Models Performance Estimation

Uso infine il validation set per la stima delle prestazioni di ogni modello (esclusa la logistic regression, che ha registrato l'accuracy peggiore nello step precedente) con i suoi migliori iperparametri mediante la 5-fold cross validation con stratificazione.

Model	Weighted F1-Score	Accuracy
StackingClassifier	0.8691596237806577	0.8907647692625111
KNN	0.8519529170931772	0.888549857484545
Decision Tree	0.8628450473246406	0.8653171637514042
Naïve Bayes	0.8627376307783706	0.8827426392486799

Table 1: Stime di performance per i modelli candidati

3.4 Scelta del modello finale

Nonostante le stime delle prestazioni dei quattro modelli siano comparabili, la mia scelta del modello finale ricade sullo **Stacking Classifier**, in quanto ritengo possa fornire una maggiore robustezza rispetto agli altri modelli: in generale potrebbe verificarsi che uno dei tre modelli che lo compongono vada in overfitting, ma è meno probabile che questo fenomeno accada contemporaneamente in tutti e tre.

Una seconda alternativa potrebbe essere quella del K-Nearest Neighbors, ma su dataset piccoli potrei facilmente rischiare l'overfitting. Inoltre sarebbe una soluzione poco scalabile in quanto l'algoritmo diventa decisamente più lento all'aumentare del numero di sample.

3.5 Feature selection: wrapper methods

Si è deciso di evitare una feature selection con l'approcio Wrapper poichè decisamente onerosa dal punto di vista temporale. E' comunque stata implementata tramite `SequentialFeatureSelection` ed è stata commentata.

3.6 Training

Viene addestrato il modello StackingClassifier su tutto il training dataset. Questa operazione richiede circa 23.54 secondi.

E' stato fatto anche un tentativo precedente utilizzando Naïve Bayes come modello finale, in tal caso si è reso necessario fare l'addestramento utilizzando solo le feature categoriche della design matrix X_{train}.

4 Testing

4.1 Preprocessing

Dopo aver constatato che nel testing dataset non sono presenti valori nulli, si procede con la trasformazione del target vector y_{test} da valori categorici a discreti. Analogamente a quanto

fatto per il dataset di training, nella design matrix X_{test} si individuano :

- feature categoriche: a esse viene applicato il mapping per trasformarle in discrete
- feature numeriche: a esse viene applicato lo scaling, usando i coefficienti μ, σ calcolati precedentemente sul dataset di training

4.2 Prediction e testing results

Eseguendo la prediction su X_{test} sono stati ottenuti i seguenti risultati:

Modello	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Stacking Classifier	0.8895027624309392	0.8597438192536807	0.8895027624309392	0.8471134153835284

Table 2

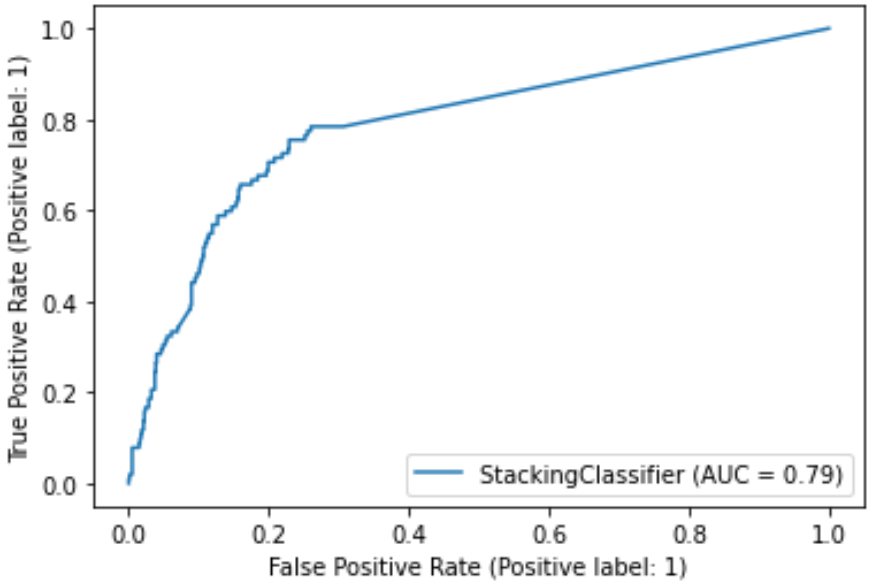

Figure 4: Curva ROC e metrica AUC per lo Stacking Classifier