AutoML - Autogluon

Mikołaj Gałkowski, Hubert Bujakowski, Kacper Kurowski

11 maja 2022

1 Wprowadzenie

W ostatnich latach mogliśmy dostrzec bardzo szybko rozwój w dziedzinie uczenia maszynowego. Zawdzięczamy to osobom, które przygotowują dane, wybierają odpowiednie zmienne i modele, następnie optymalizują modele i krytycznie analizują uzyskane rezultaty. Ponieważ złożoność tych zadań jest często poza zasięgiem osób niebędących ekspertami w dziedzinie uczenia maszynowego, szybki rozwój aplikacji uczenia maszynowego stworzył zapotrzebowanie na gotowe metody uczenia maszynowego, które można by było wykorzystać bez wiedzy eksperckiej. Powstały obszar badawczy, którego celem jest postępująca automatyzacja uczenia maszynowego nazywamy AutoML'em.

AutoML dostarcza metody, które pozwalają pokonać barierę stawianą przez uczenie maszynowe osobom niezwiązanym zawodowo/naukowo, dzięki czemu są w stanie używać ML'a przy małych wkładzie pracy. Pomaga on również przyspieszyć badania wykonywane przez badaczy za sprawą metod tj. ensembling, hyperparameter tuning, feature engineering, data preprocessing czy data splitting.

W tym artykule skupimy się na klasyfikacji binarnej w danych tabularycznych za pomocą framework'a AutoML'owego Autogluon [Erickson i in. 2020]. Sprawdzimy oferowane przez framework możliwości i stworzymy pipeline, dzięki któremu będziemy w stanie wrzucić nieprzetworzone dane i otrzymać satysfakcjonujące rezultaty.

2 Literatura powiązana

- (Gijsbers i in. 2019) — przeprowadzony przez nas benchmark opierał się o zbiory danych wykorzystane wcześniej w tym artykule. Jest zatem to przesłanka do wykorzystania akurat tych zbiorów danych
- (Erickson i in. 2020) — artykuł zawiera opis działania części pakietu autogluon poświęconej danym tabularycznym. Stanowi on zatem źródło informacji do Opisu framework'u w następnym podpunkcie.

3 Opis framework'u

Uważamy, że do głównych założeń framework'u AutoML należą

- Prostota – użytkownik może operować na danym framework'u bez wiedzy na temat uczenia maszynowego.
- Czas wykonywania – framework powinien zwrócić wyniki w podanym przez użytkownika czasie.
- Kontynuacja nauczania – trenowanie modelu może zostać przerwane i wznowione w każdej chwili.

3.1 Dane Tabularyczne

3.1.1 Interfejs `fit`

Załóżmy, że dysponujemy zestawieniem danych w pliku `train.csv`, oraz nazwą zmiennej celu — `class`. Do wytrenowania modelu przez AutoGluon potrzebne są jedynie trzy linie kodu

```
from autogluon import TabularPrediction as task
predictor = task.fit("train.csv", label="class")
predictions = predictor.predict("test.csv")
```

W wywołaniu metody `.fit()`, AutoGluon automatycznie:

- Określa typ każdej zmiennej (tj. które kolumny zawierają zmienne ciągłe w porównaniu ze zmiennymi dyskretnymi).
- Obsługuje również typowe problemy, takie jak np. braki danych oraz przeskalowywanie wartości poszczególnych zmiennych.
- Wybiera losowy podział danych odpowiednio na zbiór treningowy oraz walidacyjny. Dane używane do walidacji są oddzielone od danych ze zbioru treningowego i są potrzebne do określenia modeli jak i wartości hiperparametrów, w celu uzyskania jak najlepszych wyników.

Dodatkowo zamiast pojedynczego modelu, AutoGluon szkoli ich wiele, po czym łączy je ze sobą w całość, by zapewnić doskonałą wydajność predykcyjność. Automatyzuje również proces hiperparametryzacji poszczególnych modeli (które użytkownik musiałby wcześniej określić).

3.1.2 Ustawienia wstępne

AutoGluon oferuje 4 rodzaje ustawienia wyżej wymienionej metody `fit`:

- `best_quality`
- `high_quality`

- `good_quality`
- `medium_quality`

Ostatni przytoczony preset jest domyślnie używaną opcją. Pozostałe z nich wymagają więcej ilości pamięci oraz ich wywołanie zajmuje dużo więcej czasu. Dlatego też zalecane jest stopniowe zmienianie tych ustawień w celu poprawienia ewaluacji oraz określenia czy model jest w stanie nauczyć się w określonym przedziale czasowym.

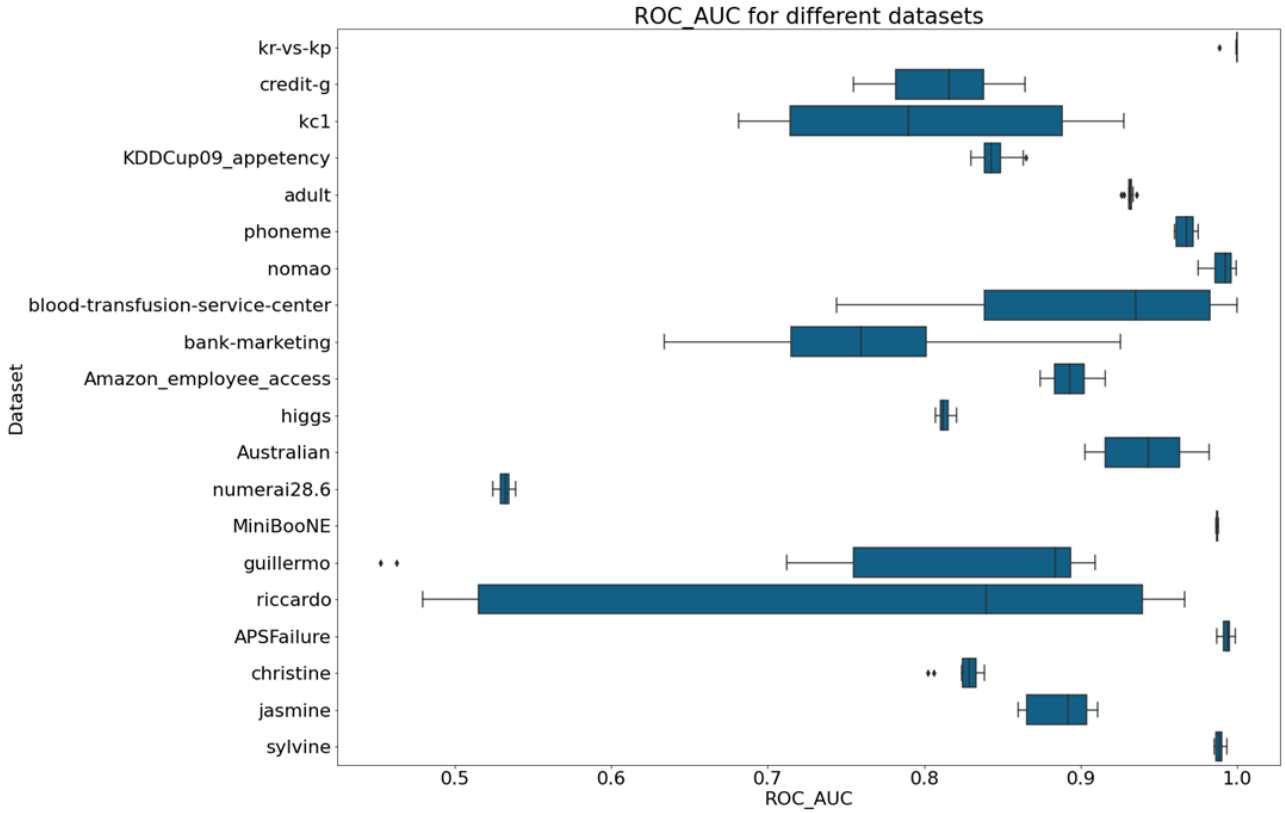
3.2 Opis preprocessingu

Autogluon wykonuje za nas potrzebny preprocessing. Obsługuje kolumny typu `boolean`, `numerical`, `categorical`, `datetime`, `object`.

4 Benchmark

4.1 Opis

Wykonaliśmy benchmark na zbiorach danych z benchmarku (Gijsbers i in. 2019). Wykorzystaliśmy 21 datasetów, w których mieliśmy do klasyfikacji problem binarny. Przygotowaliśmy pipeline, w którym przekazywaliśmy poszczególne dataset'y oraz robiliśmy preprocessing, aby pipeline zadziałał. Przetwarzaliśmy kolumny, które framework wykrywał jako kolumny zawierające bajty. Naprawiliśmy to poprzez dekodowanie **b'string'** na zwykły **string**, dzięki temu framework wykrywał kolumny poprawnie jako `object`. Aby benchmark był porównywalny do wyników uzyskanych przez autorów rzeczonego benchmarku, postanowiliśmy również zastosować cross-walidację na 10 foldach wykorzystując funkcję `StratifiedKFold` z pakietu `sklearn`.



Rysunek 1: Wykres przedstawia wyniki benchmarku na każdym zbiorze danych klasyfikacji binarnej dla 10 foldów według metryki ROC_AUC

4.2 Wyniki

Dla porównania rozważmy dataset `riccardo` oraz `sylvine`, dla pierwszego z nich mamy duże rozbieżności w wynikach, skutkiem tego, jest najprawdopodobniej duża ilość zmiennych – 1000. W drugim zaś, wyniki są zadowalające, bo w nim zawiera się tylko 21 kolumn. Jednym z głównym aspektów poprawnej predykcji danych dla AutoGluona jest rozmiar danych.

5 Podsumowanie

TO DO

Bibliografia

Erickson, Nick i in. (mar. 2020). *AutoGluon-Tabular: Robust and Accurate AutoML for Structured Data*. DOI: 10.48550/ARXIV.2003.06505. URL: https://arxiv.org/abs/2003.06505.

Gijsbers, Pieter i in. (lip. 2019). „An Open Source AutoML Benchmark". W: *arXiv e-prints*, arXiv:1907.00909, arXiv:1907.00909. arXiv: 1907.00909 [cs.LG].