

# AutoML - Projekt 2

Szymon Kiełtyka, Ryszard Czarnecki, Antoni Rakowski

26 stycznia 2026

## 1 Cel projektu

Celem było stworzenie prostego systemu automatycznego uczenia maszynowego do klasyfikacji binarnej, który miał spełniać poniższe założenia:

- osiadać portfolio zawierające maksymalnie 50 konfiguracji modeli.
- Potrafić wybrać najlepszy model (lub modele) na podstawie dostarczonych danych treningowych ( $X_{\text{train}}$ ,  $y_{\text{train}}$ ).
- Umożliwiać wykonanie ensemblingu (łączenia predykcji) z wykorzystaniem do 5 modeli.
- Być w pełni powtarzalny i możliwy do wykonania na dowolnym zbiorze danych przeznaczonym do klasyfikacji binarnej.

## 2 Portfolio modeli

### 2.1 Zewnętrzny screening

Portfolio modeli zostało stworzone na podstawie zewnętrznego screeningu, do którego użyto danych zawartych w MementoML. Ten zbiór danych zawiera wyniki trenowania modeli z różnymi konfiguracjami w języku R dla 7 algorytmów:

- catboost
- gbm
- glmnet
- kknn
- randomforest
- ranger
- xgboost

### 2.2 Wybór najlepszych modeli

W celu określenia optymalnej liczby konfiguracji dla danego modelu, przeprowadzono porównanie efektywności danych modeli, które można zobaczyć w poniższej tabeli:

Model	Średnie AUC
xgboost	0.9147
randomForest	0.9120
catboost	0.9093
ranger	0.9029
gbm	0.8726
glmnet	0.8654
kknn	0.8589

Tabela 1: Porównanie jakości predykcji modeli

Na podstawie wyników porównania efektywności modeli, do naszego portfolio zostanie dodanych po 10 konfiguracji modeli agtboost, xgboost i ranger, oraz po 5 konfiguracji modeli randomForest, gbm, kknn i svm. Konfiguracje te zostaną wybrane na podstawie najlepszego średniego AUC.

## 2.3 Zmiana nazw parametrów

Gdyż wyliczenia w MementoML zostały przeprowadzone w języku R, należy przekonwertować parametry danych modeli na ich odpowiedniki w języku Python. Dokładne zmiany przedstawione są w poniższej tabeli:

Tabela 2: Porównanie parametrów między pakietami R a implementacjami w Python

Algorytm	R	Parametr R	Python	Parametr Python
XGBoost	xgboost	nrounds eta	xgboost	n_estimators learning_rate
Ranger	ranger	num.trees min.node.size	RandomForest	n_estimators min_samples_leaf
Random Forest	randomForest	ntree nodesize replace	RandomForest	n_estimators min_samples_leaf bootstrap
GBM	gbm	n.trees shrinkage interaction.depth n.minobsinnode bag.fraction	LGBMClassifier	n_estimators learning_rate max_depth min_samples_leaf subsample
k-NN	kknn	k distance	KNeighbors	n_neighbors p
ElasticNet	glmnet	alpha lambda	LogisticRegression z <code>penalty='elasticnet'</code>	l1_ratio C (jako $1/\lambda$ )

Tabela 3: Mapowanie nazw hiperparametrów między pakietami R a bibliotekami Python

Do modeli które mają elementy losowe dodano również parametr `random_state=42`

## 3 Działanie modelu

### 3.1 Wczytywanie portfolio

Podczas inicjacji klasy naszego systemu, wczytywane są konfiguracje modeli z pliku `selected_models.json`

### 3.2 Preprocessing

W początkowej części działania metody `fit()`, nasz system wykonuje podstawowy preprocessing. Usuwane są wtedy braki danych za pomocą `SimpleImputer()`, dla zmiennych liczbowych wykonywana jest standaryzacja za pomocą `MinMaxScaler()`, a dla zmiennych kategoriycznych one-hot-encoding

### 3.3 Wybór najlepszych modeli

W kolejnej części działania metody `fit()`, nasz system wykonuje 5-krotną walidację krzyżową dla każdej konfiguracji modelu z portfolio przy użyciu funkcji `cross_val_score()` i oblicza dla każdej konfiguracji średnią wartość AUC.

### 3.4 Ensembling

Po obliczeniu średniego AUC dla każdej konfiguracji, nasz system wykonuje ensembling 5 najlepszych konfiguracji, za pomocą `VotingClassifier()` z głosowaniem miękkim. Finalny model trenowany jest na całym zbiorze danych treningowych.

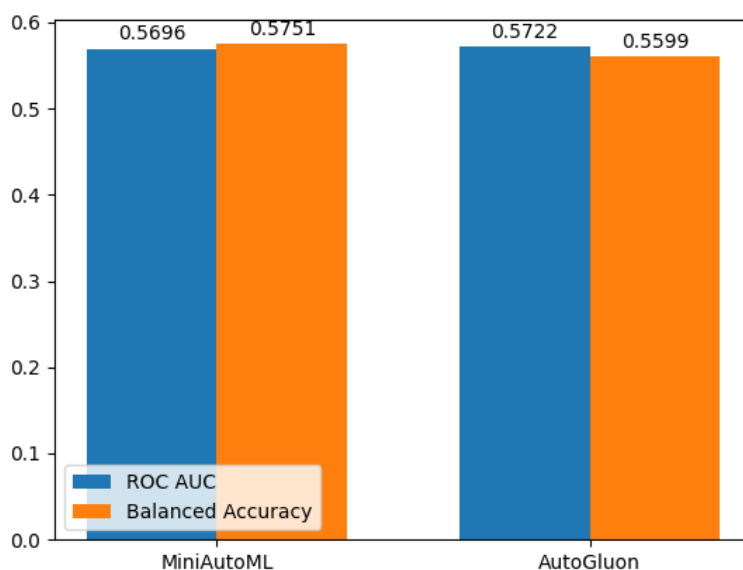
## 4 Wnioski

### 4.1 Czas działania

Ponieważ do portfolio dodawane były konfiguracje modeli, które miały najlepszy średni wynik AUC, są to w większości bardzo złożone modele, które wymagają dużej mocy obliczeniowej, zatem czas działania naszego systemu jest znacznie dłuższy, niż np. systemu AutoGluon.

### 4.2 Jakość predykcji

Dla przykładowego udostępnionego zbioru danych, jakość predykcji naszego systemu jest zbliżona do jakości predykcji systemu AutoGluon, co można zobaczyć na poniższym wykresie:



Rysunek 1: Porównanie jakości predykcji

## 5 Literatura

[Kretowicz and Biecek, 2020] Kretowicz, W. and Biecek, P. (2020). Mementoml: Performance of selected machine learning algorithm configurations on openml100 datasets.