# AutoML: Praca domowa 2

**Jan Cichomski 313201**
**Adam Dąbkowski 313212**

# 1 Wstęp

W sprawozdaniu skupimy się na próbie znalezienia modelu o najwyższej predykcyjnej dysponując sztucznie wygenerowanym zbiorem danych *artificial*, w którym zostały ukryte istotne zmienne w celu dokonania klasyfikacji do dwóch klas.

# 2 Podział danych

Ponieważ dysponujemy jednym zbiorem danych musieliśmy dokonać podziału na zbiór testowy na zbiór treningowy i testowy. Ustaliliśmy podział na 20% danych testowych i 80% danych treningowych wybranych w sposób losowy.

# 3 Data preprocessing

Ponieważ dostarczone dane mogą posiadać kolumny nieznaczące należy uprzednio dokonać przetwarzania danych przed rozpoczęciem procesu szukania modeli.

Rozważono różne metody eliminacji nieznaczących cech z zbioru danych. Przetestowano nasypując podejścia: usunięcie wysoce skorelowanych cech, usunięcie cech o niskiej wariancji, usunięcie cech a niskiej ważności (*importance*) z wykorzystaniem drzewa decyzyjnego. Żadne z tych rozwiązań, jak i łącznie ich z sobą, nie dało satysfakcjonujących rezultatów.

Dlatego zdecydowano się na wykorzystanie następującego rozwiązania. Do eliminacji kolejnych atrybutów nieznaczących użyto techniki RFE (Recursive Feature Elimination) oraz RFECV (Recursive Feature Elimination with Cross-Validation) z wykorzystaniem następujących modeli:

- ExtraTreesClassifier
- RandomForestClassifier

Połączone z sobą liniowe następujące modele, dzięki czemu zredukowano liczbę cech do 20, co pozwoliło na uzyskanie zadowalającej jakości modeli.

## ExtraTreesClassifier z RFE

W pierwszym kroku preprocessingu danych użyto ExtraTreesClassifier jako estymatora oraz zastosowano RFE z minimalną liczbą cech do pozostawienia wynoszącą 250.

## RandomForestClassifier z RFE

Następnie zastosowano RFE przy użyciu RandomForestClassifier jako estymatora z ustawioną minimalną liczbą cech do pozostawienia wynoszącą 125.

### ExtraTreesClassifier z RFECV

Wstępnie oczyszczone dane podano procedurze RFECV z użyciem ExtraTreesClassifier jako estymatora oraz z minimalną liczbą cech do pozostawienia wynoszącą 25 i przy użyciu trzech złożeń krzyżowej walidacji.

### RandomForestClassifier z RFECV

Ostatnim etapem przygotowywania danych jest użycie RFECV z estymatorem RandomForestClassifier z trzema złożeniami krzyżowej walidacji oraz zostawieniem co najmniej 15 kolumn.

# 4 Modele

W celu sprawdzenia czy autorzy są w stanie znaleźć lepszy model od popularnych frameworków autoML, skonstruowano model ręcznie oraz znaleziono model przy użyciu popularnego narzędzia autoML (Autogluon).

## 4.1 Model stworzony przy pomocy frameworka autoML

Rozważono trzy frameworki autoML: *MLJAR*, *auto-sklearn*, *Autogluon*.

Model stworzony przy pomocy *MLJAR* po wytrenowani się, nie był w stanie dokonywać predykcji, bo zwracał błąd

```
CatBoostError: C:/Go_Agent/pipelines/BuildMaster/catboost.git/
catboost/libs/data/model_dataset_compatibility.cpp:81:
At position 20 should be feature with name
Ensemble_prediction_0_for_-1.0_1_for_1.0 (found Ensemble_prediction).
```

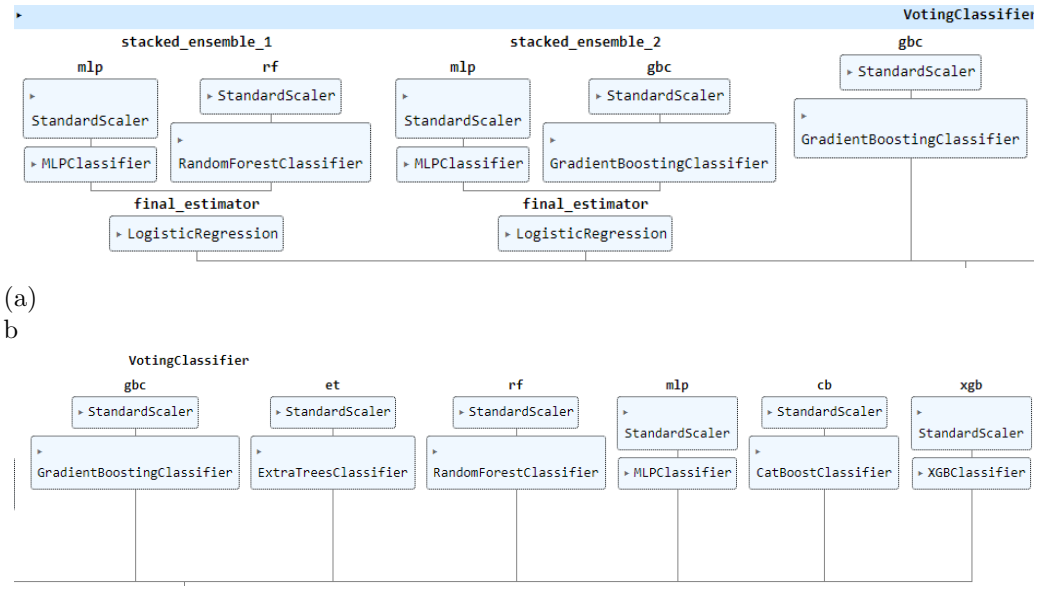
Niestety biblioteki *auto-sklearn* nie udało się zainstalować na lokalnej maszynie.

Z powyższych powodów wykorzystano bibliotekę *Autogluon*, który uzyskała satysfakcjonujące wyniki. Model trenowano przez 8h.

## 4.2 Model skonstruowany ręcznie

Do skonstruowania modelu ML ręcznie wybrano bibliotekę *sklearn*. Stworzono VotingClassifier składający się z: stacked_ensamble z Mulilayer Perceptron i Random Forest, stacked_ensamble z Mulilayer Perceptron i Gradient Boosting, Gradient Boosting, Extra Trees, Random Forest, Mulilayer Perceptron, Cat Boost i XGBoost. Schemat złożenia modeli przedstawiono na rysunku 1.

Konfiguracja poszczególnych modeli:




(a) b (b) b

**Grafika 1:** Schemat ręcznie stworzonego modelu (zdjęcie podzielone na dwie części w celu zwiększenia rozmiary zdjęcia)

- Mulilayer Perceptron: max_iter=1000, early_stopping=True, tol=1e-3, solver="lbfgs", hidden_layer_sizes=(100, 300, 200, 100), alpha=0.001,
- Extra Trees: n_estimators=500, max_depth=30, min_samples_leaf=4, min_samples_split=2
- Random Forest: n_estimators=500, max_depth=30, min_samples_leaf=4, min_samples_split=2
- Gradient Boosting: n_estimators=500, max_depth=30, min_samples_leaf=2, min_samples_split=5, max_features=None
- Cat Boost: iterations=500, learning_rate=0.03, depth=6, l2_leaf_reg=3, border_count=32, cat_features=None, loss_function="Logloss", early_stopping_rounds=50,
- XB Boost: use_label_encoder=False, eval_metric=balanced_accuracy_score, n_estimators=500, learning_rate=0.02, max_depth=6, min_child_weight=1, subsample=0.8, colsample_bytree=0.8, gamma=0, reg_alpha=0.1, reg_lambda=1.0, scale_pos_weight=1,

# 5 Wyniki

Do oceny jakości wyników wykorzystano metrykę *balanced_accuracy*.

Model stworzono z wykorzystaniem frameworka *Autogluon* uzyskał następujące wyniki:

- Na zbiorze treningowym: 0.940655
- Na zbiorze walidacyjnym: 0.889825
- Na udostępnionym zbiorze testowym: 0.9333

Model stworzony ręcznie uzyskał następujące wyniki:

- Na zbiorze treningowym: 1.0
- Na zbiorze walidacyjnym: 0.8999775
- Na udostępnionym zbiorze testowym: 0.9

# 6 Podsumowanie

Udało się stworzyć i wytrenować model ręcznie oraz z wykorzystaniem frameworka autoML.

Zredukowano liczbę cech zbioru danych z początkowych 500 do 20, co przyspieszało szybkość trenowania modeli.

Oba modele uzyskały bardzo podobne wyniki, dlatego nie da się jednoznacznie wskazać zwycięzcy. Jednak należy zauważyć, że model trenowany ręcznie uzyskał nieznacznie lepszy wynik na zbiorze walidacyjnym. Nie mniej jednak, 100% dokładność modelu stworzonego ręcznie na zbiorze treningowym, może wskazywać na problem z przetrenowaniem.

Z powodu bardzo małego rozmiaru zbioru testowego (30 wierszy), trudno stwierdzić, który model jest lepszy, gdyż taki a nie inny wynik może być przypadkiem (modele różnią się w jednym przypadku)

# 7 Bibliografia


1. AutoGluon `https://auto.gluon.ai/stable/index.html`
2. MLJAR `https://mljar.com/`
3. AutoSKLearn `https://automl.github.io/auto-sklearn/master/`
4. SKLearn `https://scikit-learn.org/stable/`