

Wyddział Elektroniki i Technik Informacyjnych
Politechnika Warszawska

Przeszukiwanie i optymalizacja

Optymalizacja hiperparametrów algorytmu XGBoost za pomocą
algorytmów heurystycznych - Raport końcowy

Kamil Troszczyński

Warszawa, 2025

1 Link do repozytorium

https://gitlab-stud.elka.pw.edu.pl/ktroszcz/pop_ktroszczynski_szelazowski.git

2 Tematyka projektu

Tematem projektu jest optymalizacja hiperparametrów klasyfikatora z biblioteki `xgboost` z wykorzystaniem różnych algorytmów heurystycznych.

Ręczne strojenie parametrów tego modelu jest procesem czasochłonnym, a uzyskiwane w ten sposób wyniki często nie są w pełni satysfakcyjne. Dodatkowo, nie wszystkie hiperparametry mają istotny wpływ na jego działanie, dlatego konieczne jest ich odpowiednie odfiltrowanie w celu redukcji wymiarowości problemu optymalizacyjnego.

XGBoost (Xtreme Gradient Boosting) jest jedną z najskuteczniejszych metod klasyfikacji, dlatego warto zastosować metody optymalizacji do automatycznego doboru jego parametrów. Pozwala to uzyskiwać wysoką jakość wyników przy minimalnym nakładzie pracy i czasie potrzebnym na strojenie.

3 Tuning znaczących hiperparametrów modelu

Sposobem, aby szybciej oraz wydajniej odszukać najlepszą kombinację hiperparametrów modelu jest redukcja wymiarowości problemu. Polegać ona będzie na wybraniu jak najmniejszej ilości znaczących parametrów klasyfikatora XGBoost do wyznaczenia, by móc zmniejszyć nakład obliczeniowy i czasowy potrzebny do znalezienia optymalnego zestawu nastaw modelu.

XGBoost posiada trzy rodzaje boosterów, gdzie dla każdego istotny jest inny zestaw hiperparametrów. W tym projekcie poświęcono uwagę na optymalizację hiperparametrów w boosterze `gbtree`:

- max depth - maksymalna głębokość każdego drzewa
- min child weight - minimalna liczba instancji w węźle potomnym
- subsample - część wylosowanych danych treningowych potrzebnych do zbudowania pojedyńczego drzewa
- colsample bytree - ma znaczenie podobne do `subsample`, tylko, że losuje on cechy na podstawie, których budowane jest drzewo
- *eta* - krok uczenia
- *gamma* - minimum loss reduction
- *lambda* - czynnik regularyzacji L2
- *alpha* - czynnik regularyzacji L1
- *n_estimators* - liczba drzew

4 Zbiór danych oraz wskaźnik nastawionego modelu

Zbiór danych, który został wykorzystany, by wytrenować, zwalidować oraz przetestować model XGBoost to *Porto Seguro's Safe Driver Prediction*. Zbiór ten zawiera wiele cech oraz wskazania czy kierowca zgłosi rozszczenie z powodu zbyt dużej kwoty ubezpieczenia za samochód firmie *Porto Seguro*, co wskazuje na problem klasyfikacji binarnej. W tym konkursie *Kaggle* są zawarte dwa zbiory danych takie jak *train.csv* oraz *test.csv*. Podczas analizy tych dwóch zbiorów, można było zauważać silnie skorelowane cechy oraz niebalansowane klasy. Operacje spreparowania feature-ów w zbiorach dokonała funkcja `clean_data()`. Zostały również wykonane badania jak parametry `scale_pos_weight` i `THRESHOLD` regulują metryki, w celu wybrania wartości, dla których model najlepiej się uczy.

Funkcja celu wykorzystywana do trenowania modelu XGBoost to *Binary Cross Entropy*, która dobrze się nadaje do trenowania modeli pod klasyfikację binarną. Wskaźnikiem, który będzie wykorzystywany,

aby ewaluować jakość nastaw modelu to *znormalizowany współczynnik Giniego*. Można go wyznaczyć ze wzoru w przypadku zadania klasyfikacji binarnej:

$$NormalizedGiniScore = 2AUC - 1 \quad (1)$$

Jego wartości znajdują się w przedziale $[-1; 1]$, gdzie dla $NormalizedGiniScore = -1$ model pracuje w sposób nieprawidłowy, a dla $NormalizedGiniScore = 1$ model klasyfikuje w sposób idealny.

5 Algorytmy, które porównano w projekcie

Poniżej są przedstawione algorytmy, które zostały poddane porównaniu podczas szukania najlepszej możliwej kombinacji hiperparametrów modelu XGBoost:

- Optuna
- Algorytm cząsteczkowy z modyfikacją na liczby dyskretne i dane kategoryczne
- Ewolucja różnicowa z modyfikacją na liczby dyskretne i dane kategoryczne
- Błędzenie przypadkowe
- CMA-ES z modyfikacją na liczby dyskretne i dane kategoryczne
- TBPSA z modyfikacją na liczby dyskretne i dane kategoryczne
- One Plus One z modyfikacją na liczby dyskretne i dane kategoryczne

6 Zakresy przestrzeni przeszukiwań

W tabeli 1 znajdują się zakresy hiperparametrów, które tworzą przestrzeń przeszukiwań dla boostera *gbtree*.

Parametr	Wartość
<i>n_estimators</i>	[50; 500]
<i>max_depth</i>	[1; 5]
<i>eta</i>	[0,01; 0,1]
<i>lambda</i>	[0,1; 1,0]
<i>alpha</i>	[0,1 ; 1,0]
<i>subsample</i>	[0,0 ; 1,0]
<i>colsample_bytree</i>	[0,0 ; 1,0]
<i>gamma</i>	[0,1 ; 2,0]
<i>min_child_weight</i>	[1; 5]

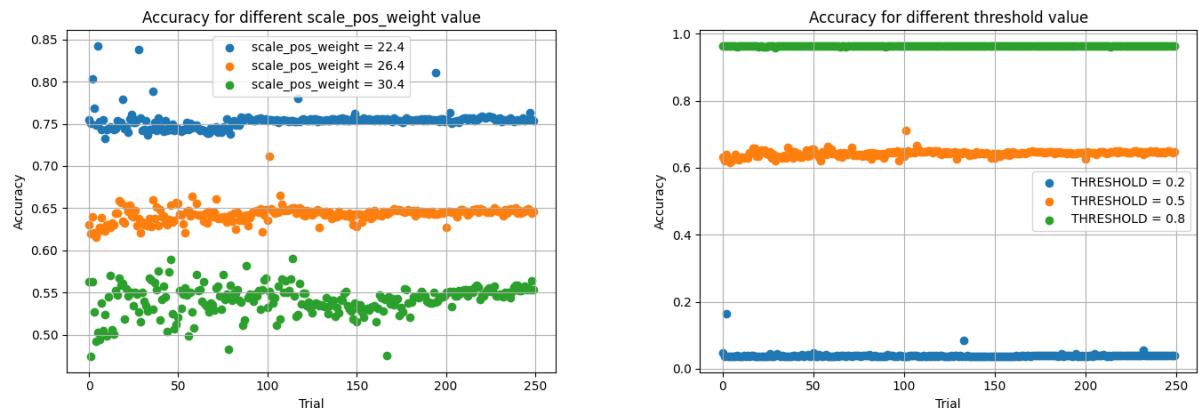
Tabela 1: Zakresy przestrzeni przeszukiwań dla boostera *gbtree*

7 Wpływ różnych wartości parametrów `scale_pos_weight` oraz `threshold` na metryki

Parametrem `scale_pos_weight` można regulować metrykę *recall*, gdy bardziej oczekujemy, żeby model był bardziej czuły na rozpoznawanie klas pozytywnych. Gdy wzrasta wartość owego parametru, rośnie wartość metryki *recall* przy jednoczesnym spadku wartości *accuracy* oraz *precision*. Jest to parametr klasyfikatora *XGBClassifier*.

Z kolei parametrem `threshold` można regulować wartość metryki *precision* i wzrost owego parametru wywołuje wzrost *precision* oraz *accuracy* i spadek metryki *recall*. Parametr ten jest progiem, od którego model przewiduje na podstawie cech klasę pozytywną.

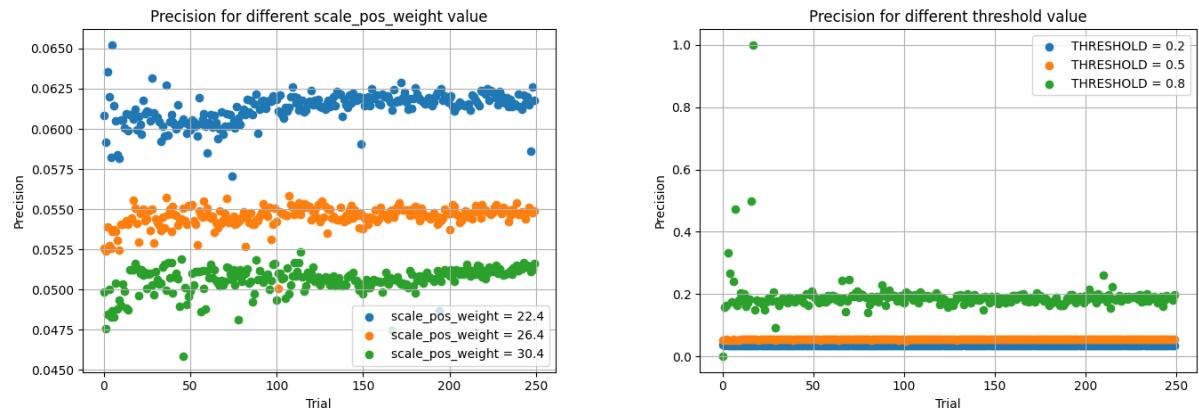
Analizując zbiór danych *Porto Seguro's Safe Driver Prediction*, można stwierdzić, iż metryką, na której zależeć powinno bardziej jest *recall*, gdyż firma ubezpieczeniowa może więcej stracić na złym kierowcy, który dostanie zbyt niską składkę do zapłaty. Wedle tego, model powinien być bardziej czuły.



(a) Accuracy dla różnych wartości `scale_pos_weight`

(b) Accuracy dla różnych wartości `THRESHOLD`

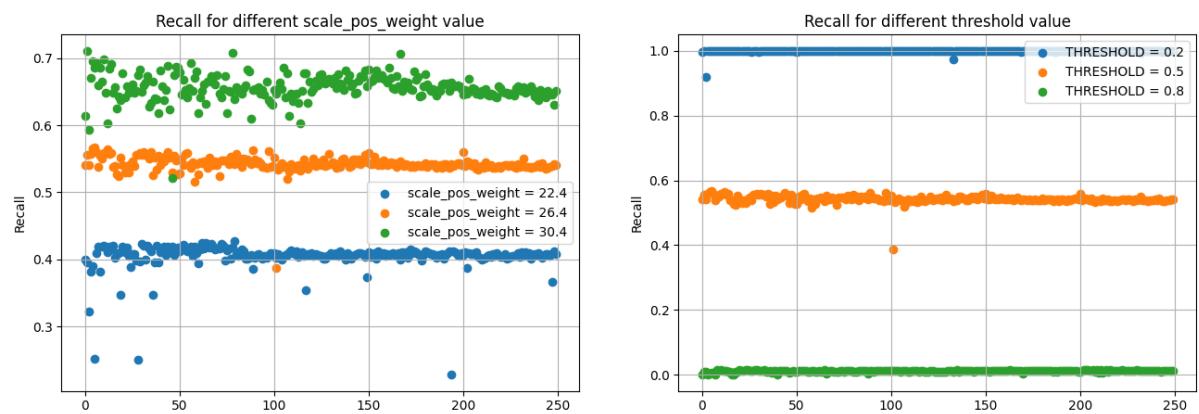
Rysunek 1: Wpływ parametrów na wartość Accuracy



(a) Precision dla różnych wartości `scale_pos_weight`

(b) Precision dla różnych wartości `THRESHOLD`

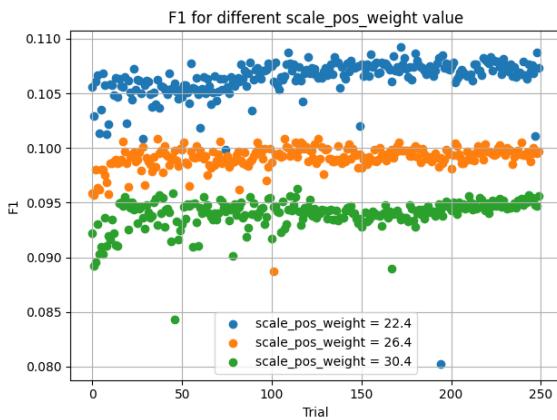
Rysunek 2: Wpływ parametrów na wartość Precision



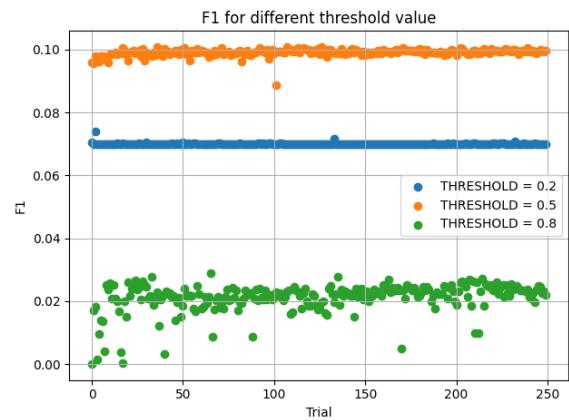
(a) Recall dla różnych wartości `scale_pos_weight`

(b) Recall dla różnych wartości `THRESHOLD`

Rysunek 3: Wpływ parametrów na wartość Recall

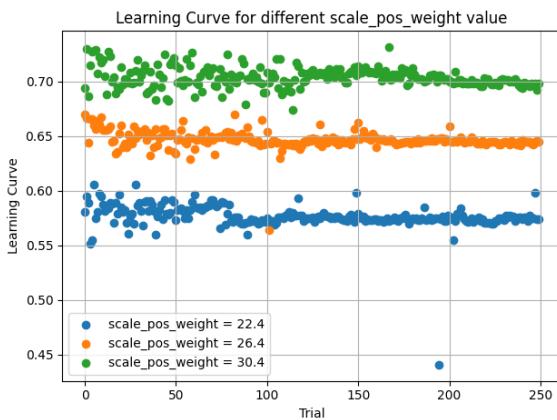


(a) F1 dla różnych wartości `scale_pos_weight`

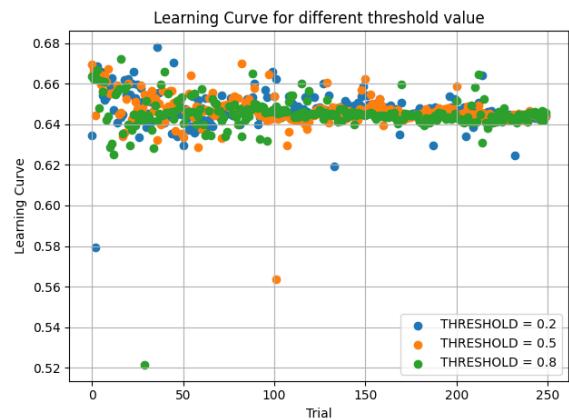


(b) F1 dla różnych wartości `THRESHOLD`

Rysunek 4: Wpływ parametrów na wartość F1

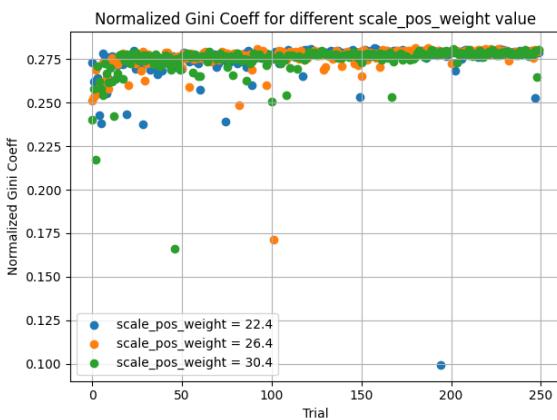


(a) L.C. dla różnych wartości `scale_pos_weight`

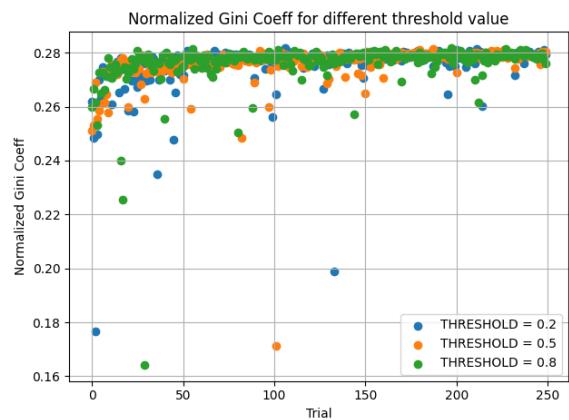


(b) L.C. dla różnych wartości `THRESHOLD`

Rysunek 5: Wpływ parametrów na Learning Curve



(a) Normalized Gini Score dla różnych wartości `scale_pos_weight`



(b) Normalized Gini Score dla różnych wartości `THRESHOLD`

Rysunek 6: Wpływ parametrów na Normalized Gini Score

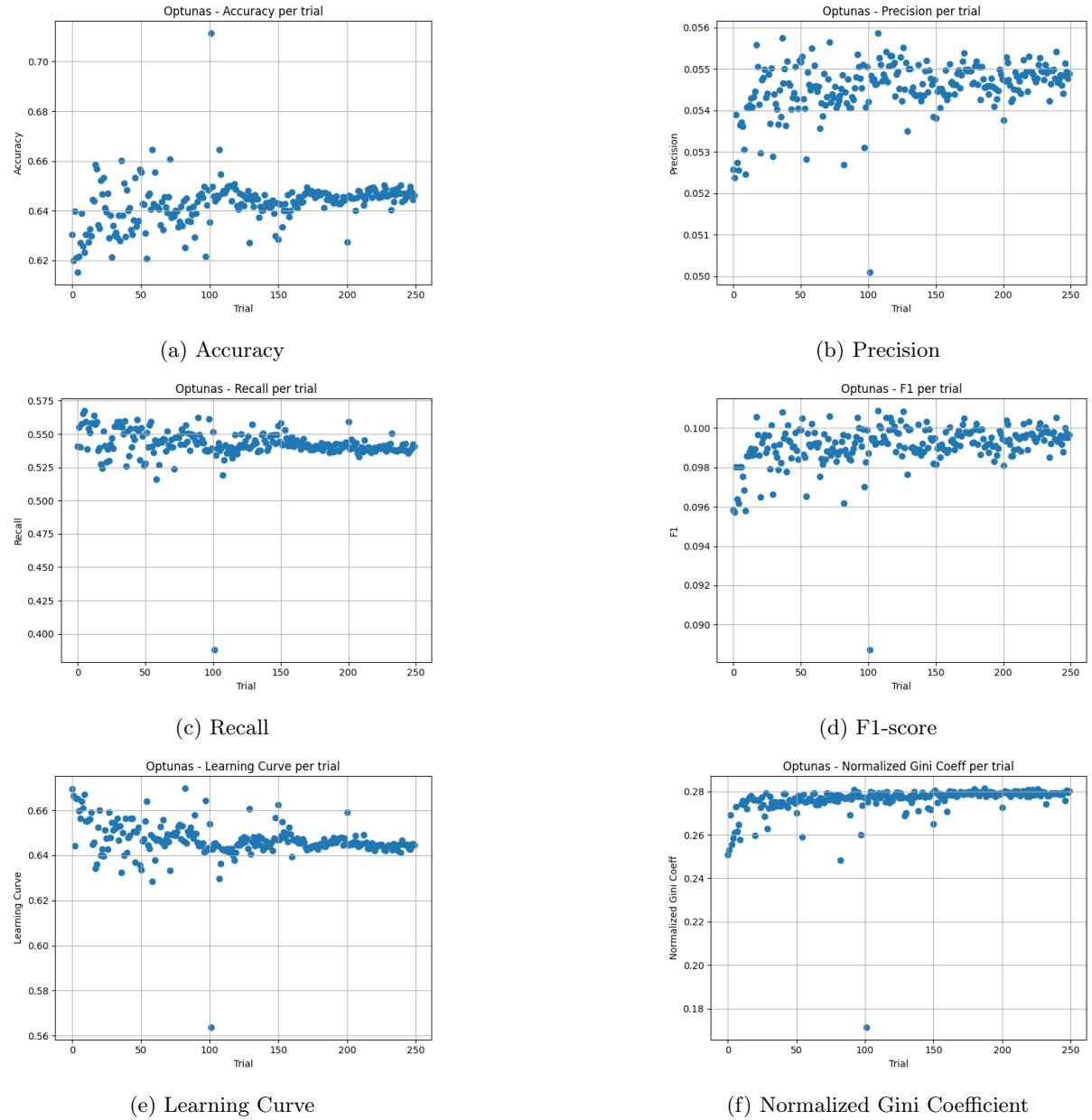
Na podstawie powyższych wykresów widać, iż dobrym kompromisem między precyzją predykcji, a czułością są wartości `scale_pos_weight` = 26,4 oraz `THRESHOLD` = 0,5.

Wartość parametru *THRESHOLD* została wybrana w pełni empirycznie, natomiast punktem wyjścia parametru *scale_pos_weight* była wartość 26,4 pochodząca ze wzoru:

$$scale_pos_weight = \frac{\sum(y_{train} == 0)}{\sum(y_{train} == 1)} \quad (2)$$

8 Porównanie metryk zgromadzonych podczas tuningu parametrów modelu XGBoost różnymi algorytmami

8.1 Optuna



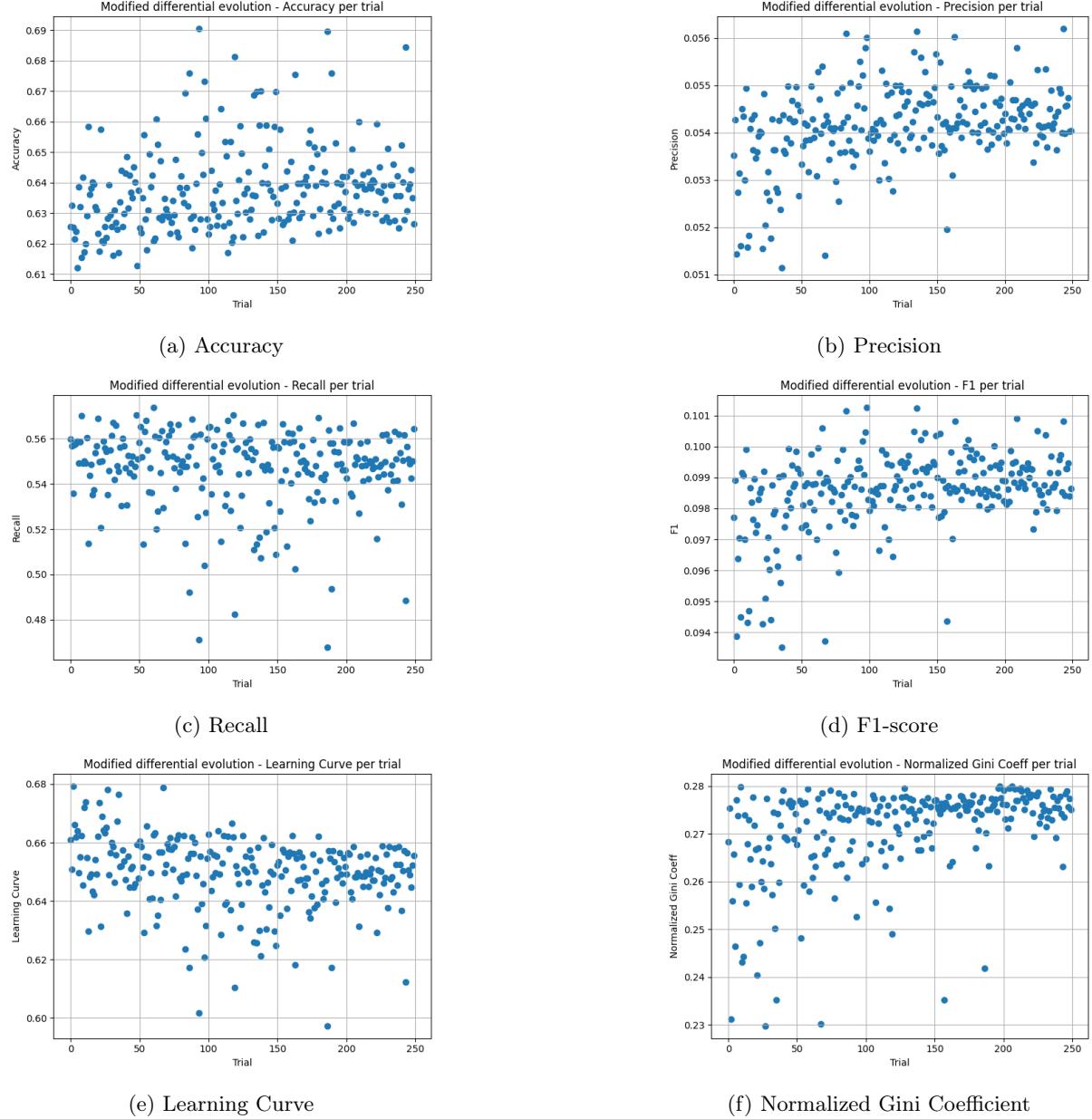
Rysunek 7: Optuna

Parametr	Wartość
<i>n_estimators</i>	488
<i>max_depth</i>	4
<i>eta</i>	0,0346076736
<i>lambda</i>	0,7878240662
<i>alpha</i>	0,7117245007
<i>subsample</i>	0,9173548637
<i>colsample_bytree</i>	0,5379229726
<i>gamma</i>	0,6105626041
<i>min_child_weight</i>	3

Tabela 2: Najlepsze hiperparametry znalezione przez optymalizator Optuna

8.2 Zmodyfikowana ewolucja różnicowa

Parametry: popsize = 40, F1 = 0,75, F2 = 0,75. crossover_rate = 0,75



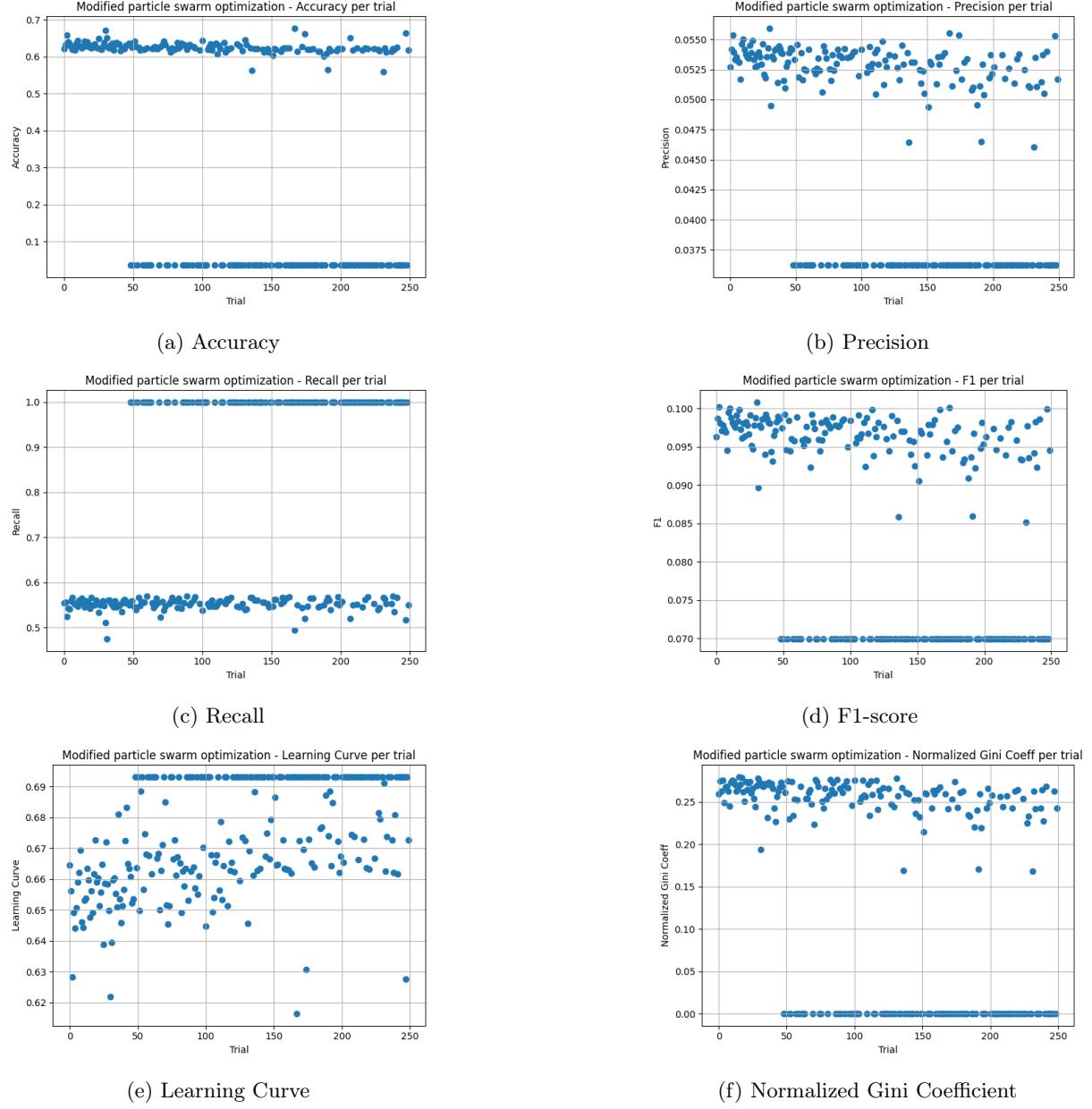
Rysunek 8: Zmodyfikowana ewolucja różnicowa

Parametr	Wartość
<i>n_estimators</i>	464
<i>max_depth</i>	3
<i>eta</i>	0,0879655135
<i>lambda</i>	0,9796755997
<i>alpha</i>	0,6441468204
<i>subsample</i>	0,8904986948
<i>colsample_bytree</i>	0,5051797857
<i>gamma</i>	1.9642965081
<i>min_child_weight</i>	5

Tabela 3: Najlepsze hiperparametry znalezione przez optymalizator Zmodyfikowana ewolucja różnicowa

8.3 Zmodyfikowany algorytm roju cząsteczkowego

Parametry: particles=30, inertia=0.2, cognitive_weight=1.25, social_weight=2.25

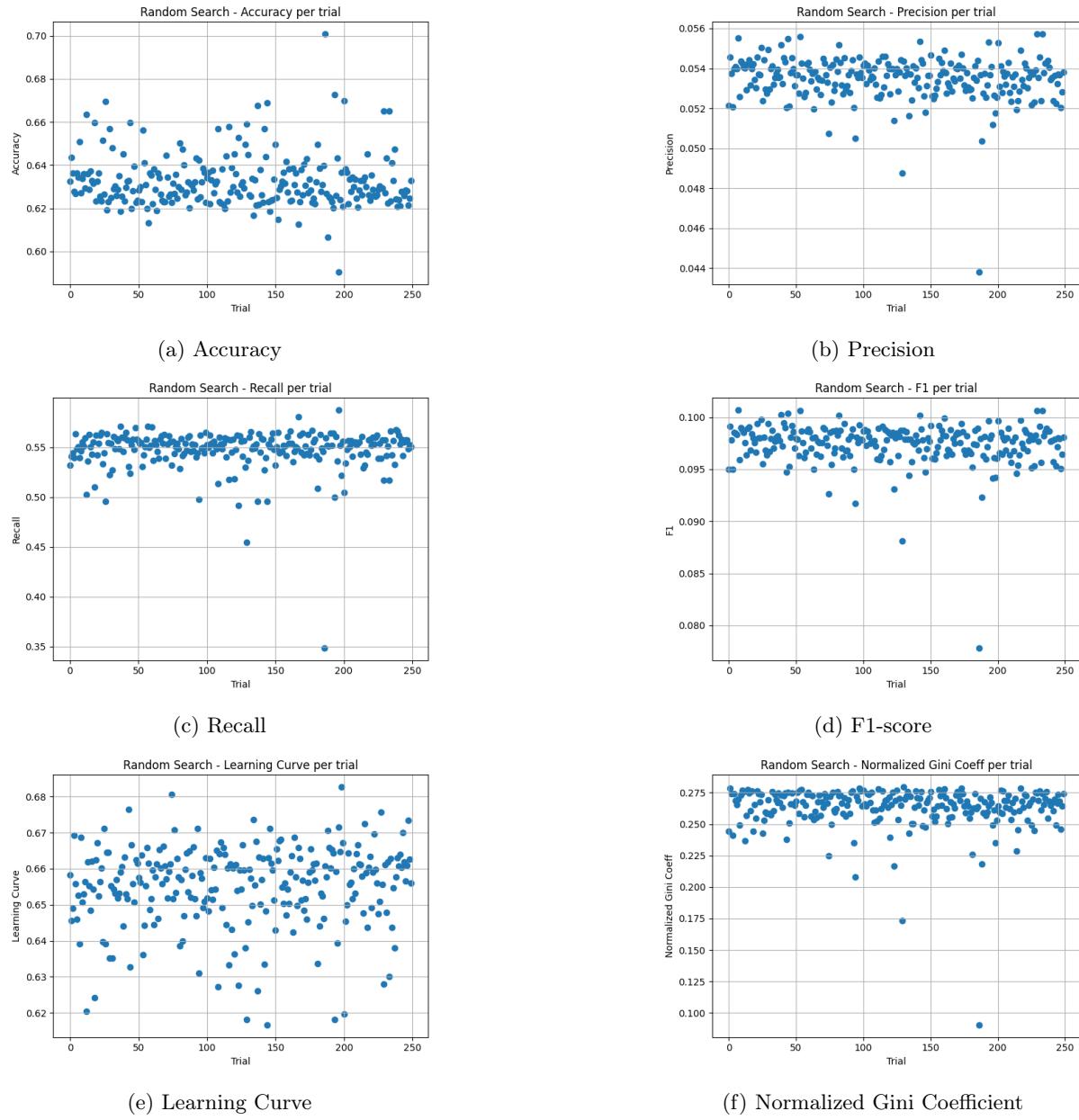


Rysunek 9: Zmodyfikowany algorytm roju cząsteczkowego

Parametr	Wartość
<i>n_estimators</i>	335
<i>max_depth</i>	4
<i>eta</i>	0,0475407694
<i>lambda</i>	0,6507705925
<i>alpha</i>	0,5824098729
<i>subsample</i>	0,9114653847
<i>colsample_bytree</i>	0,3968236034
<i>gamma</i>	1,7972630492
<i>min_child_weight</i>	2

Tabela 4: Najlepsze hiperparametry znalezione przez optymalizator Algorytm roju cząsteczkowego

8.4 Błędzenie przypadkowe



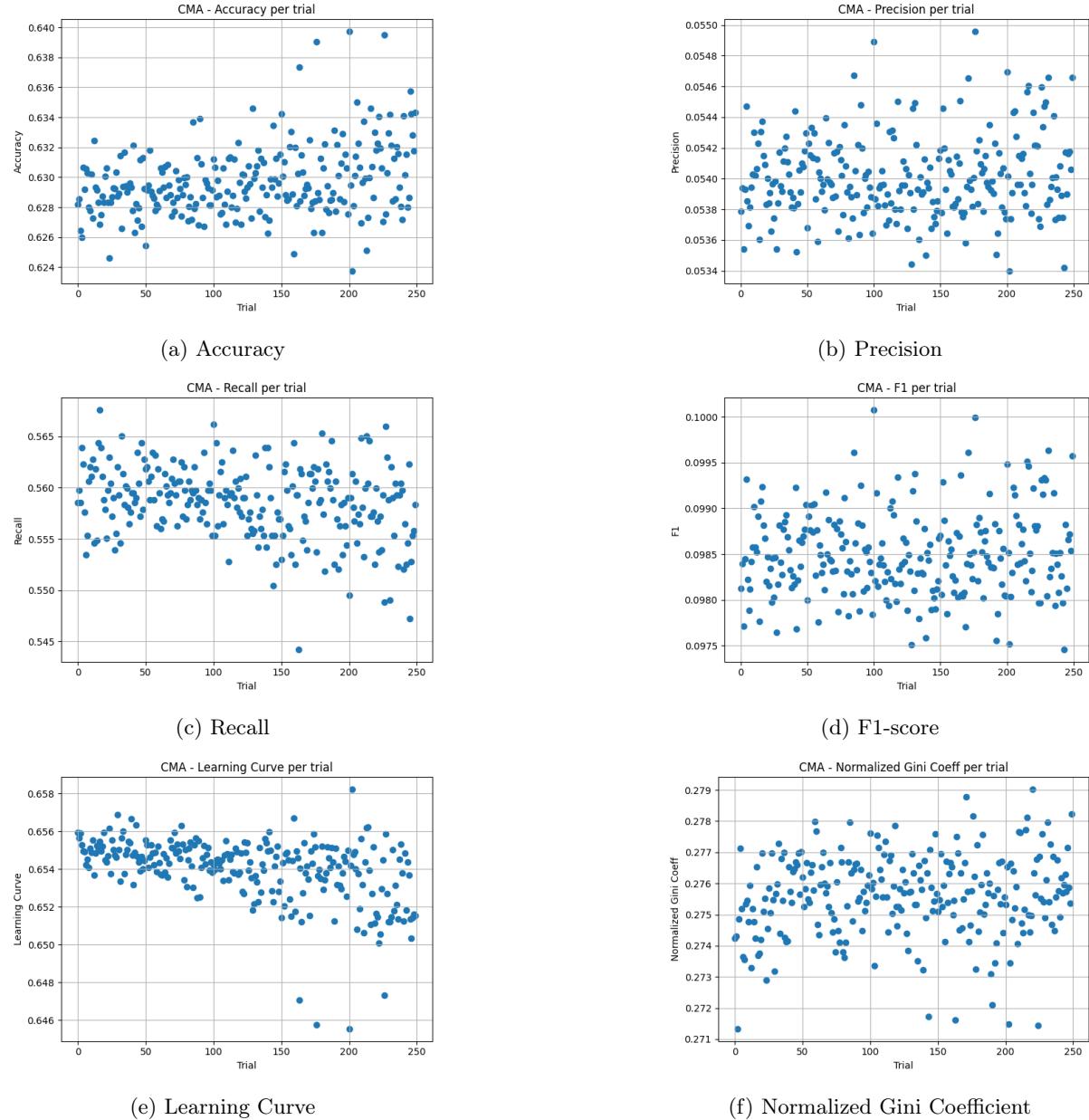
Rysunek 10: Błędzenie przypadkowe

Parametr	Wartość
<i>n_estimators</i>	429
<i>max_depth</i>	4
<i>eta</i>	0,0354786714
<i>lambda</i>	0,8004300918
<i>alpha</i>	0,6541842625
<i>subsample</i>	0,8015938667
<i>colsample_bytree</i>	0,9765098494
<i>gamma</i>	0,7835251098
<i>min_child_weight</i>	4

Tabela 5: Najlepsze hiperparametry znalezione przez optymalizator Przeszukiwanie Losowe

8.5 Zmodyfikowany CMA-ES

Parametry: popsize=40, elitist = True, diagonal=False



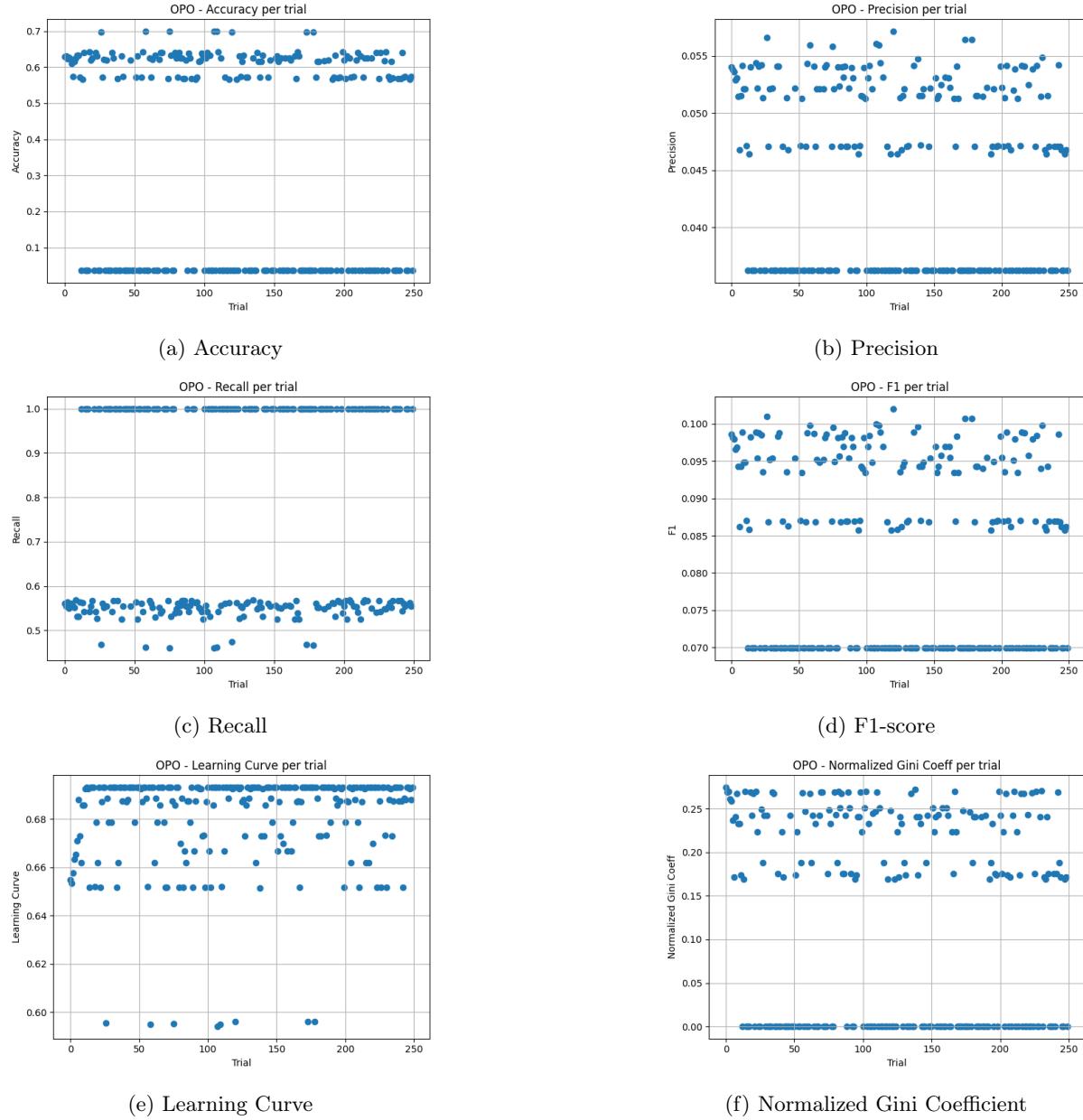
Rysunek 11: Zmodyfikowany CMA-ES

Parametr	Wartość
<i>n_estimators</i>	322
<i>max_depth</i>	16
<i>eta</i>	0,5595263674
<i>lambda</i>	0,5120556382
<i>alpha</i>	0,4870329237
<i>subsample</i>	0,127665147
<i>colsample_bytree</i>	0,4696058274
<i>gamma</i>	0,528538197
<i>min_child_weight</i>	9

Tabela 6: Najlepsze hiperparametry znalezione przez optymalizator Zmodyfikowany CMA-ES

8.6 One Point Optimization

Parametry: tabu=3



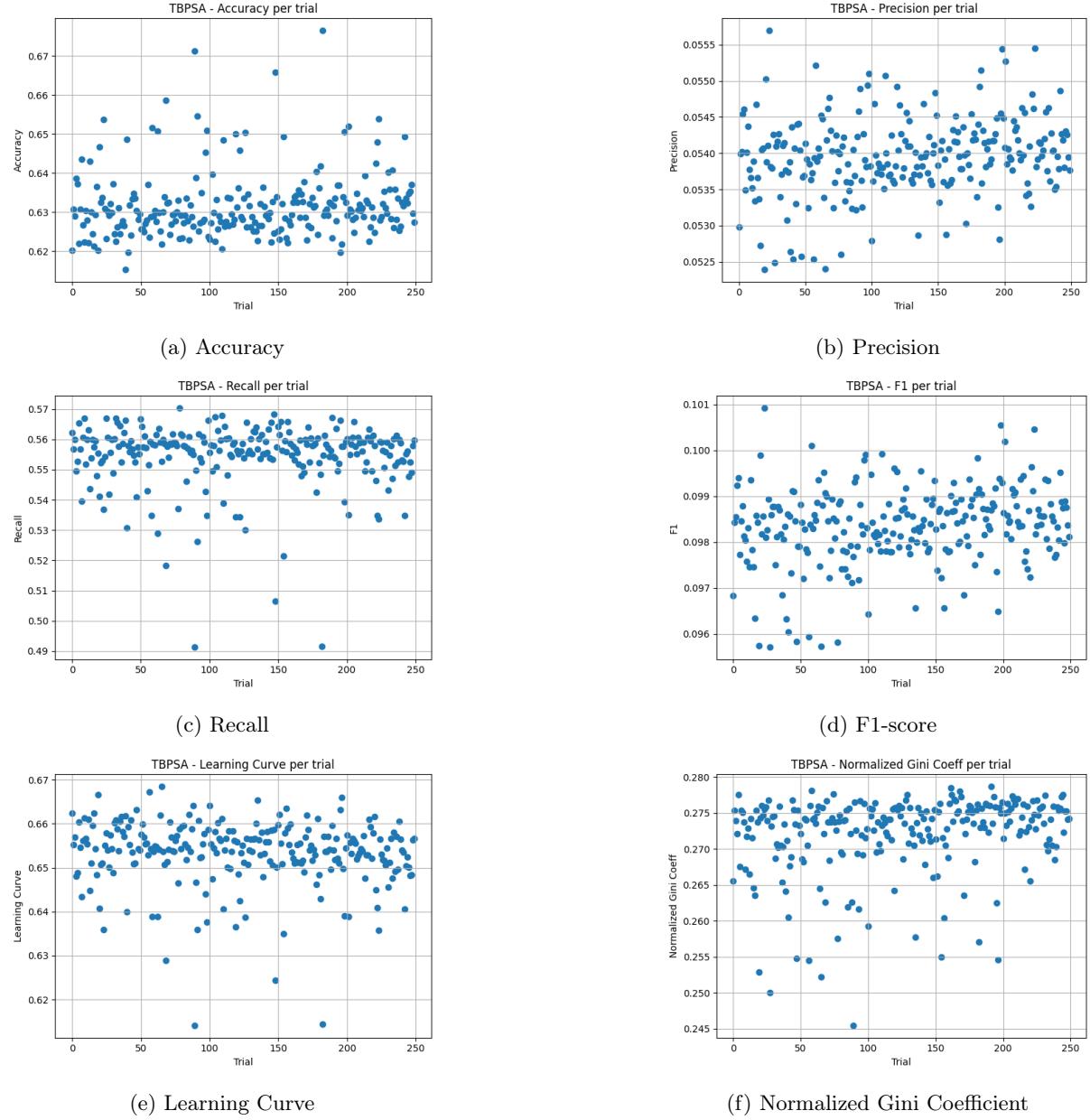
Rysunek 12: One Point Optimization

Parametr	Wartość
<i>n_estimators</i>	275
<i>max_depth</i>	3
<i>eta</i>	0,055
<i>lambda</i>	0,55
<i>alpha</i>	0,55
<i>subsample</i>	0,5
<i>colsample_bytree</i>	0,5
<i>gamma</i>	1,05
<i>min_child_weight</i>	3

Tabela 7: Najlepsze hiperparametry znalezione przez optymalizator One Point Optimization

8.7 TBPSA

Parametry: initial_popsize = 40



Rysunek 13: TBPSA

Parametr	Wartość
<i>n_estimators</i>	413
<i>max_depth</i>	3
<i>eta</i>	0,0483243593
<i>lambda</i>	0,6115157692
<i>alpha</i>	0,807171235
<i>subsample</i>	0,5712547318
<i>colsample_bytree</i>	0,5741316706
<i>gamma</i>	01,2663491476
<i>min_child_weight</i>	3

Tabela 8: Najlepsze hiperparametry znalezione przez optymalizator TBPSA

9 Znormalizowane współczynniki Giniego dla zbioru testowego

W konkursie *Porto Seguro's Safe Driver Prediction* w zbiorze testowym nie ma kolumny `target` z klasami 0, 1. Spowodowało to, iż jedyną formą porównania nastaw różnych optymalizatorów było przekazanie pliku `sample_submission.csv` z przewidzianymi prawdopodobieństwami wystąpienia klasy.

Submission and Description	Private Score	Public Score	Selected
<code>sample_submission_Optuna.csv</code> Complete (after deadline) - 2m ago	0.28350	0.27978	<input type="checkbox"/>
<code>sample_submission_PSO.csv</code> Complete (after deadline) - 1m ago	0.28321	0.27968	<input type="checkbox"/>
<code>sample_submission_RS.csv</code> Complete (after deadline) - 1m ago	0.28330	0.27925	<input type="checkbox"/>
<code>sample_submission_TBPSA.csv</code> Complete (after deadline) - 34s ago	0.28358	0.27838	<input type="checkbox"/>
<code>sample_submission_DE.csv</code> Complete (after deadline) - 3m ago	0.28230	0.27781	<input type="checkbox"/>
<code>sample_submission_CMA.csv</code> Complete (after deadline) - 3m ago	0.28083	0.27736	<input type="checkbox"/>
<code>sample_submission_OPO.csv</code> Complete (after deadline) - 3m ago	0.28207	0.27693	<input type="checkbox"/>

Rysunek 14: Wartości znormalizowanych współczynników Giniego dla każdego z optymalizatorów

Wedle zrzutu 14 widać, iż najlepszy wynik ze wszystkich skonfigurowanych modeli osiąga model, którego hiperparametry zostały wyznaczone optymalizatorem `optuna`. Jest to optymalizator, który łączy algorytmy takie jak TPE oraz CMA-ES przez co jest to jeden z lepszych optymalizatorów do wyboru, zależy nam przeszukaniu przestrzeni rozwiązań w celu znalezienia najlepszego rozwiązania.

10 Najlepsza kombinacja hiperparametrów XGBoost'a

Parametr	Wartość
<code>n_estimators</code>	488
<code>max_depth</code>	4
<code>eta</code>	0,0346076736
<code>lambda</code>	0,7878240662
<code>alpha</code>	0,7117245007
<code>subsample</code>	0,9173548637
<code>colsample_bytree</code>	0,5379229726
<code>gamma</code>	0,6105626041
<code>min_child_weight</code>	3

Tabela 9: Najlepsze hiperparametry znalezione przez optymalizator `optuna`

11 Podsumowanie

- Nastawa hiperparametrów XGBoost'a jest zadaniem wielowymiarowym, gdyż model ten posiada dużą liczbę parametrów do nastaw. Należy spośród wszystkich wybrać te, które mają wpływ na inferencję modelu w celu redukcji wymiarowości problemu co przyspieszy proces przeszukiwania przestrzeni.
- Zakresy przestrzeni przeszukiwań w sekcji 6 zostały ustalone, gdyż dla tych zakresów model osiąga największe wartości znormalizowanego współczynnika Giniego. Jeżeli zdecydowano, iż przedział wartości parametru η będzie mały to dobrze jest wskazać zakres dla większej ilości estymatorów.

Głębokość drzew nie może być zbyt duża, gdyż wtedy jest większe prawdopodobieństwo wystąpienia *overfittingu*.

- Parametr `scale_pos_weight` modelu reguluje czułość modelu na występowanie klas pozytywnych. Im większy jest to parametr, tym bardziej wzrastać będzie wartość metryki *recall*, lecz jednocześnie maleć będzie precyza modelu (*precision*) oraz jego dokładność (*accuracy*). Natomiast wzrost progu prawdopodobieństwa predykcji klasy pozytywnej (`THRESHOLD`) dla odpowiednich featurów powoduje wzrost wartości metryki *accuracy* oraz *precision*, oddziałując regresyjnie na metrykę *recall*. Najlepsze rezultaty uzyskano dla tego zadania dla `THRESHOLD = 0,5` oraz `scale_pos_weight = 26,4`.
- Optuna najlepiej wykonała zadanie optymalizacji doprowadzając do najwyższego wyniku *Normalized Gini Score* spośród wszystkich algorytmów. Jednak, mimo wysokich wartości tej metryki, uzyskano niskie wartości metryk *precision*, *accuracy* oraz *F1*. Wynikać to może z wysokiego dysbalansu zbioru danych jakim się posługiwano podczas przeszukiwania przestrzeni nastaw XGBoost'a. Jednak, w tym przypadku firma osiągnie mniejsze straty podczas przewidzenia, że dla podanych featurów klasą `target` będzie miała wartość 1, niż gdyby z większą precyją przewidywać klasy negatywne.