Raport badawczy: Klasyfikacja wyniku partii szachowej na podstawie cech rozgrywki

Mikołaj Kalejta, Grupa 1

25 czerwca 2025

1 Wstęp

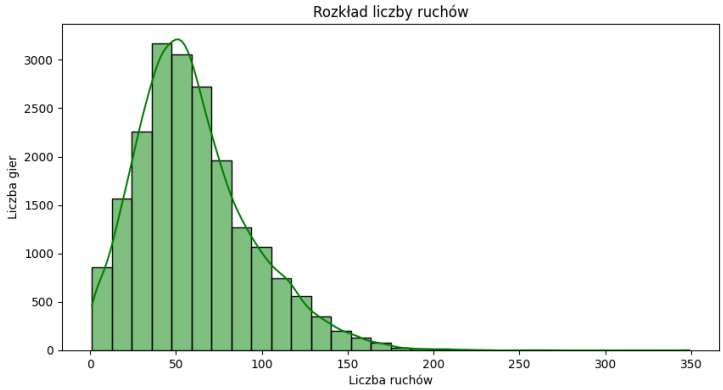
Celem projektu było zbudowanie i porównanie skuteczności różnych modeli klasyfikacyjnych przewidujących wynik partii szachowej (wygrana białych, czarnych lub remis) na podstawie cech rozgrywki dostępnych w zbiorze `games.csv`. Projekt realizuje wymagania kursu z zakresu uczenia maszynowego, klasyfikacji oraz inżynierii cech.

2 Opis danych

Zbiór danych pochodzi z serwisu Lichess i zawiera informacje o przebiegu i wyniku partii szachowych. Każdy rekord opisuje jedną partię i zawiera m.in.:

- `moves` – lista ruchów w partii,
- `opening_eco`, `opening_name`, `opening_ply` – kod i nazwa debiutu oraz liczba ruchów debiutowych,
- `turns` – liczba ruchów w partii,
- `rated` – czy partia była rankingowa,
- `increment_code` – tempo gry,
- `victory_status` – sposób zakończenia partii (mat, czas, rezygnacja, remis),
- `created_at`, `last_move_at` – znaczniki czasu,
- `winner` – zwycięzca (białe, czarne, remis).

Z danych nie korzystano z rankingów i identyfikatorów graczy, aby nie faworyzować modeli.



Rysunek 1: Rozkład liczby ruchów w partiach

3 Reguły asocjacyjne w danych szachowych

W ramach eksploracji zbioru danych przeprowadzono analizę reguł asocjacyjnych, mającą na celu wykrycie zależności pomiędzy wybranym debiutem (kod `opening_eco`) a wynikiem partii (`winner`). Do analizy wykorzystano algorytm Apriori oraz miary takie jak *support* (wsparcie), *confidence* (ufność) i *lift* (podniesienie). Poniżej przedstawiono kilka najciekawszych reguł:

- **Jeśli otwarcie to `'Van't Kruijs Opening'`, to zwycięzca to czarne.**
  support: 2,8% partii, *confidence*: 56,6%, *lift*: 1,25.

- **Jeśli otwarcie to `'Sicilian Defense: Bowdler Attack'`, to zwycięzca to czarne.**
  support: 1,6% partii, *confidence*: 56,4%, *lift*: 1,24.

- **Jeśli otwarcie to `'Nimzowitsch Defense: Kennedy Variation'`, to zwycięzca to białe.**
  support: 1,8% partii, *confidence*: 59,7%, *lift*: 1,20.

- **Jeśli otwarcie to `'King's Pawn Game: Leonardis Variation'`, to zwycięzca to czarne.**
  support: 1,8% partii, *confidence*: 52,6%, *lift*: 1,16.

- **Jeśli otwarcie to `'Philidor Defense'`, to zwycięzca to białe.**
  support: 2,0% partii, *confidence*: 57,3%, *lift*: 1,15.

Wartość *lift* powyżej 1 oznacza, że dana zależność występuje częściej niż wynikałoby to z przypadku, co wskazuje na istnienie istotnej korelacji pomiędzy wybranym debiutem a wynikiem partii. Analiza reguł asocjacyjnych pozwala lepiej zrozumieć, które otwarcia mogą sprzyjać określonym wynikom i może być punktem wyjścia do dalszych badań nad strategią gry w szachy.

4 Preprocessing

Proces przygotowania danych do uczenia maszynowego obejmował kilka kluczowych etapów. W pierwszej kolejności dokonano ekstrakcji ruchów szachowych – z kolumny `moves` wyodrębniono pierwsze $2n$ ruchów (gdzie domyślnie $n = 15$), które następnie potraktowano jako cechy kategoryczne opisujące przebieg początkowej fazy partii. Kolejnym krokiem była inżynieria cech, w ramach której do zbioru danych dodano nowe informacje: cechy czasowe, takie jak godzina rozpoczęcia partii (`created_hour`), dzień tygodnia (`created_weekday`) oraz czas trwania gry (`game_duration`), a także cechy związane z debiutem (`opening_eco`, `opening_name`, `opening_ply`), tempem gry (`increment_code`), statusem zwycięstwa (`victory_status`), liczbą ruchów (`turns`) oraz informacją, czy partia była rankingowa (`rated`).

Wszystkie cechy kategoryczne zostały zakodowane za pomocą one-hot encodingu, co pozwoliło na ich efektywne wykorzystanie przez modele klasyfikacyjne. Cecha wynikowa (`winner`) została zakodowana liczbowo. Dodatkowo, wszystkie cechy liczbowe zostały wystandaryzowane, aby zapewnić porównywalność skali i poprawić efektywność procesu uczenia. Na końcu, cały zbiór danych został podzielony na zbiór treningowy i testowy w proporcji 80/20, przy zachowaniu oryginalnych proporcji klas, co umożliwiło rzetelną ocenę skuteczności wytrenowanych modeli.

5 Modele i parametry

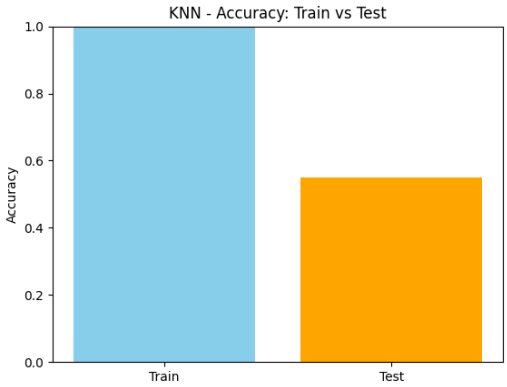
Przetestowano następujące modele klasyfikacyjne:

- **KNN** – najlepsze parametry: $k = 7$, wagi: `distance`
- **Random Forest** – 500 drzew, max_depth=20, min_samples_leaf=2, class_weight=balanced
- **XGBoost** – 500 drzew, max_depth=8, learning_rate=0.1, colsample_bytree=0.7, subsample=1.0
- **Sieć neuronowa (MLP)** – 2 warstwy ukryte (64 i 32 neurony), dropout 0.3, l2=0.001, epochs=30, batch_size=64

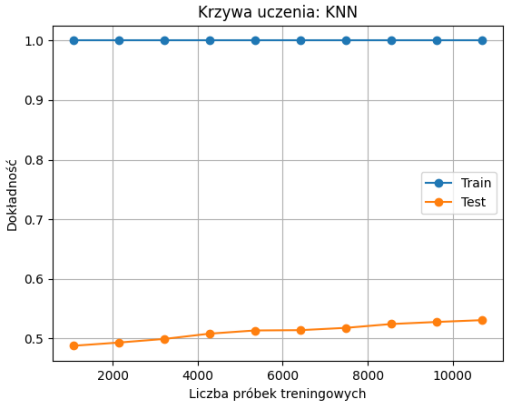
Parametry zostały wybrane na podstawie przeszukiwania siatki (GridSearchCV), a najlepsze wartości zapisano w plikach tekstowych.

6 Wyniki

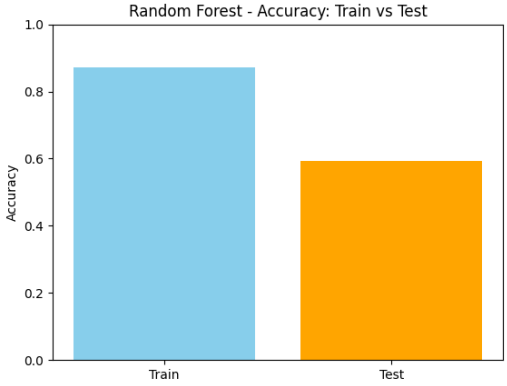
- **KNN:** dokładność testowa $\approx 52\%$
- **Random Forest:** dokładność testowa $\approx 58\%$
- **XGBoost:** dokładność testowa $\approx 86\%$
- **Sieć neuronowa:** dokładność testowa $\approx 60\%$



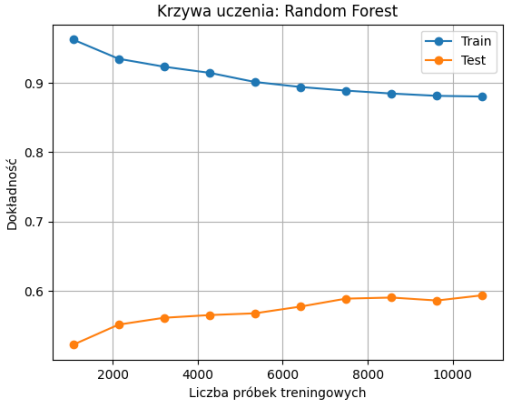
Rysunek 2: Dokładność KNN – zbiór treningowy vs testowy



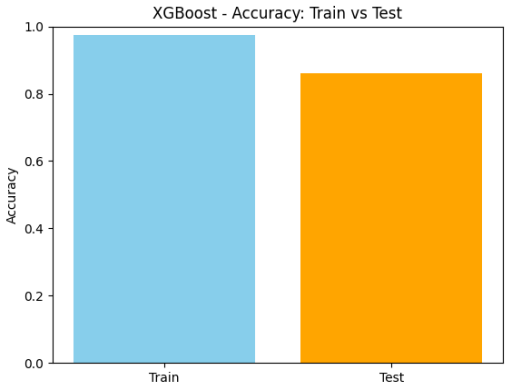
Rysunek 3: Krzywa uczenia: KNN



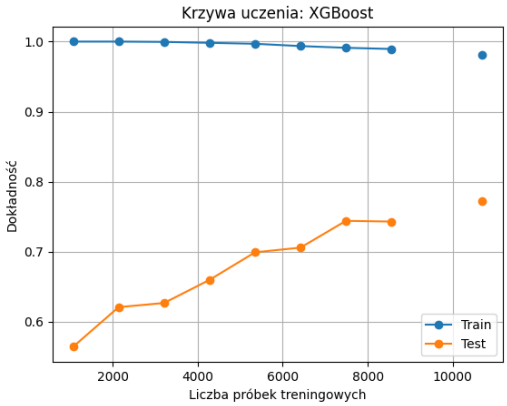
Rysunek 4: Dokładność Random Forest



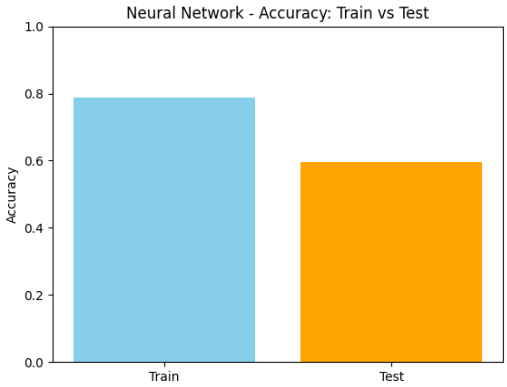
Rysunek 5: Krzywa uczenia: Random Forest



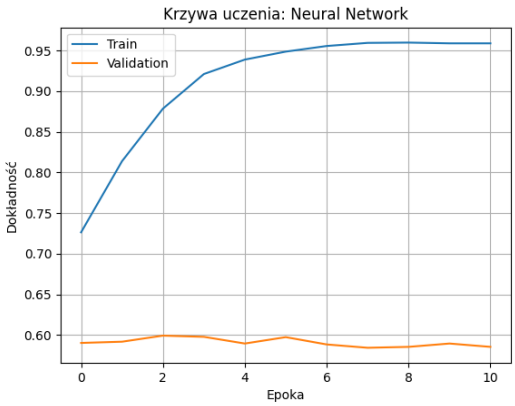
Rysunek 6: Dokładność XGBoost – zbiór treningowy vs testowy



Rysunek 7: Krzywa uczenia: XGBoost



Rysunek 8: Dokładność Neural Network – zbiór treningowy vs testowy



Rysunek 9: Krzywa uczenia: Neural Network

Na podstawie krzywych uczenia się widać, że sieć neuronowa uległa przeuczeniu – bardzo wysoka dokładność na treningu i niska na walidacji. Random Forest oraz XGBoost również wykazują przeuczenie, choć w ich przypadku dokładność testowa rośnie wraz z liczbą próbek. Model KNN natomiast osiąga niemal idealną dokładność na zbiorze treningowym, ale niską na testowym, co także wskazuje na silne przeuczenie. Żaden z modeli nie jest niedotrenowany, ale dalsza regularyzacja lub więcej danych mogłyby poprawić generalizację.

7 Podsumowanie i wnioski

Najlepsze wyniki w przeprowadzonych eksperymentach osiągnął model XGBoost, który bardzo dobrze radzi sobie z dużą liczbą cech kategorycznych oraz złożonymi zależnościami w danych. Klasyczne modele, takie jak KNN, Random Forest, a także sieć neuronowa, uzyskały niższą skuteczność, co można tłumaczyć wysoką wymiarowością oraz złożonością zbioru danych. Kluczowy wpływ na skuteczność klasyfikacji miało odpowiednie kodowanie cech (one-hot encoding) oraz inżynieria cech, w tym dodanie cech czasowych, informacji o debiutach oraz tempie gry. Warto podkreślić, że modele nie korzystały z rankingów graczy, co sprawia, że predykcja jest bardziej "uczciwa"i oparta wyłącznie na przebiegu partii. W przyszłości warto rozważyć dalszą redukcję liczby cech (feature selection), testowanie innych architektur sieci neuronowych oraz szczegółową analizę wpływu poszczególnych cech na skuteczność predykcji.