Politechnika Warszawska

WYDZIAŁ MATEMATYKI
I NAUK INFORMACYJNYCH

Praca dyplomowa inżynierska

na kierunku Inżynieria i Analiza Danych

Przewidywanie rezultatów meczów tenisowych z wykorzystaniem różnych modeli uczenia maszynowego

Paweł Świderski

Numer albumu 319116

Kacper Wnęk

Numer albumu 320669

promotor
dr hab. inż. Marek Gągolewski, prof. uczelni

WARSZAWA 2025

Streszczenie

Przewidywanie rezultatów meczów tenisowych z wykorzystaniem różnych modeli uczenia maszynowego

Celem projektu było opracowanie wysokiej jakości modelu predykcyjnego do prognozowania wyników meczów tenisowych oraz stworzenie strony internetowej prezentującej predykcje modelu w sposób przystępny dla użytkownika.

W ramach badań przetestowano trzy modele uczenia maszynowego: regresję logistyczną, XGBoost oraz sieć neuronową. Dane wejściowe obejmowały cechy związane z zawodnikami oraz ze środowiskiem. Wyniki eksperymentów wykazały, że sieć neuronowa przewyższyła pozostałe metody oraz osiągnęła jedynie o 0,0032 niższy wskaźnik Briera od predykcji bukmacherów. Dodatkowo model osiągnął zwrot z inwestycji (ROI) na poziomie **5,62%** oraz procentowy zysk wynoszący **29,57%**, co potwierdza jego skuteczność w praktycznych zastosowaniach w obszarze zakładów bukmacherskich.

Integralną częścią projektu było stworzenie strony internetowej przy użyciu Django, na której dla każdego wybranego meczu prezentowane są prognozy modelu oraz różnorodne statystyki — zarówno dla danej rozgrywki, jak i zawodników — wykorzystywane w procesie predykcji. System uwzględniał złożony przepływ danych, który obejmował pozyskiwanie danych w czasie rzeczywistym, iteracyjne przetwarzanie oraz wgrywanie do bazy danych używanej przez stronę.

Zaprojektowany system łączy skuteczny model predykcyjny z wygodną platformą raportowania wyników, co potwierdza jego potencjał w zastosowaniach praktycznych, szczególnie w analizie danych tenisowych.

Słowa kluczowe: uczenie maszynowe, sieć neuronowa, XGBoost, regresja logistyczna, model predykcyjny, analiza danych sportowych, tenis ziemny, predykcje bukmacherskie

Abstract

Predicting results of the tennis matches using various types of machine learning models

The project's goal was to develop a predictive model for high-quality forecasting of tennis match results and create a user-friendly website presenting them.

We tested three machine learning models: logistic regression, XGBoost and a neural network. Input data included features related to the players and the environment. Experiments showed that the neural network outperformed the other methods and achieved only a 0,0032 lower Brier score than bookmakers' predictions. Additionally, the model achieved a return on investment (ROI) of **5.62%** and a percentage profit of **29.57%**, confirming its effectiveness in practical applications within the domain of sports betting.

An integral part of the project was the creation of a website based on the Django framework, which, for each selected match, presents the model's predictions and various game and player statistics that were used in the prediction process. The system included a complex data flow that included real-time data acquisition, iterative processing, and uploading to the database used by the site.

The designed system combines an effective predictive model with a convenient platform for visualization and reporting results.

Keywords: machine learning, neural network, XGBoost, logistic regression, predictive model, sports data analysis, tennis, bookmaker predictions

Spis treści

1. Wstęp

Prognozowanie wyników sportowych to jedno z najciekawszych i jednocześnie najtrudniejszych wyzwań w dziedzinie analizy danych. Wyniki meczów są funkcją wielu czynników, takich jak aktualna forma zawodników, warunki środowiskowe czy historyczne rezultaty. Dzięki rosnącej dostępności danych oraz rozwojowi zaawansowanych metod uczenia maszynowego możliwe stało się budowanie coraz dokładniejszych modeli predykcyjnych. W pracy tej opracowano system predykcji wyników meczów tenisowych, który łączy wysoką jakość prognoz z praktycznym zastosowaniem, dzięki integracji z dynamiczną i interaktywną stroną internetową.

Celem projektu było wytrenowanie modelu predykcyjnego opartego na uczeniu maszynowym, który osiąga wysoką dokładność w prognozowaniu wyników meczów tenisowych. W ramach analizy przetestowano trzy różne podejścia: regresję logistyczną, XGBoost oraz sieć neuronową. Dane wejściowe do modelu zostały podzielone na trzy kategorie: cechy zawodników, cechy środowiskowe oraz cechy analityczne, takie jak miara niepewności. Wyniki eksperymentów wykazały, że sieć neuronowa przewyższała pozostałe modele pod względem jakości prognoz oraz najbardziej zbliżyła się do rezultatów predykcji opartych na kursach bukmacherskich.

Integralnym elementem projektu było stworzenie strony internetowej opartej na bibliotece Django. Strona ta umożliwia użytkownikom przeglądanie prognoz modelu dla wybranych meczów oraz analizę szczegółowych statystyk, które były wykorzystywane podczas generowania przewidywań. Kluczowym aspektem systemu było opanowanie złożonego przepływu danych, obejmującego pozyskiwanie ich w czasie rzeczywistym, iteracyjne przetwarzanie oraz ładowanie do bazy danych obsługiwanej przez stronę internetową.

Niniejsza praca ma na celu nie tylko zaprezentowanie skuteczności modelu predykcyjnego, ale także przedstawienie kompletnego rozwiązania integrującego techniki uczenia maszynowego z platformą wizualizacyjną i raportującą, co pozwala na praktyczne zastosowanie opracowanego systemu w analizie sportowej.

Tab. 1.1. Wkład autorów prac dyplomowych

Rola	**Paweł Świderski 319116**	**Kacper Wnęk 320669**
Przygotowanie tekstu pracy – wersja początkowa	Wstęp, Inżynieria cech, Strona internetowa projektu, Sieć neuronowa, Podsumowanie i Wnioski	Wstęp, Narzędzia i technologie, Zbiory danych, Modele uczenia, Podsumowanie i Wnioski
Przygotowanie tekstu pracy – przegląd i redakcja	Całość tekstu pracy	Całość tekstu pracy
Oprogramowanie	Moduł przetwarzania danych, moduł modelowania, moduł ewaluacji modeli, moduł zbierania danych, strona internetowa	Moduł przetwarzania danych, moduł modelowania, moduł ewaluacji modeli, moduł zbierania danych
Kluczowe rozwiązania pracy • Konceptualizacja • Badania • Metodologia	Ogólny pomysł na rozwiązanie, szukanie istniejących rozwiązań, pozyskiwanie danych, stworzenie modelu sieci neuronowej, stworzenie strony internetowej	Ogólny pomysł na rozwiązanie, szukanie istniejących rozwiązań, pozyskiwanie danych, stworzenie modelu regresji logistycznej oraz XGBoost
Analiza formalna w tym analiza EDA i walidacja	Analiza jakości danych, przetworzenie danych, walidacja modeli	Analiza jakości danych, przetworzenie danych, walidacja modeli
Przygotowanie danych	TAK	TAK
Zarządzanie projektem	TAK	TAK
Zasoby	NIE	NIE
Wizualizacja	TAK	TAK
Pozyskanie finansowania	-	-

Niniejszym oświadczamy, że:

1. Nie używaliśmy narzędzi informatycznych do generowania treści niniejszej pracy.

2. Nie używaliśmy narzędzi informatycznych do generowania kodu programów stworzonych na potrzeby niniejszej pracy.

3. Bierzemy pełną odpowiedzialność za zawartość niniejszej pracy, która składa się zarówno z jej tekstu, jak i stworzonego na jej potrzeby oprogramowania.

1.1. Przegląd literatury

Predykcja wyników meczów tenisowych przy użyciu metod uczenia maszynowego jest obszarem, który zyskał na znaczeniu w ostatnich latach dzięki rosnącej dostępności danych sportowych oraz postępom w dziedzinie sztucznej inteligencji. W literaturze wyróżniono dwie kluczowe prace, które w znacznym stopniu pokrywają się z tematem niniejszej pracy: badania Michala Sipko oraz Alexandra De Seranno. Obie te prace stanowią istotny punkt odniesienia dla naszego podejścia, a ich analiza pozwala wskazać luki, które niniejsza praca stara się wypełnić.

1.1.1. Praca Michala Sipko

W pracy *„Machine Learning for the Prediction of Professional Tennis Matches"* [10] Michal Sipko zaproponował wykorzystanie dwóch modeli predykcyjnych: regresji logistycznej oraz sieci neuronowych. Cechy użyte w modelach obejmowały szeroki zestaw zmiennych związanych z wynikami meczów i statystykami zawodników, takie jak ranking ATP, szczegółowe statystyki serwisowe i odbioru serwisów (np. procent wygranych punktów przy pierwszym i drugim serwisie), a także cechy opisujące skuteczność zawodników w kluczowych momentach meczu, takich jak break pointy. Dodatkowo wprowadzono zmienne odnoszące się do specyficznych warunków gry, takie jak zmęczenie zawodnika wynikające z ostatnich dni oraz historia bezpośrednich spotkań (*head-to-head balance*).

W pracy szczególną uwagę poświęcono ocenie wyników modeli przy użyciu zwrotu z inwestycji (ROI), który w przypadku sieci neuronowej osiągnął poziom 4,35% na przestrzeni lat 2013–2014.

1.1.2. Praca Alexandra De Seranno

W pracy *„Predicting Tennis Matches Using Machine Learning"* [3] Alexander De Seranno skoncentrował się na przewidywaniu wyników meczów singlowych na poziomie ATP z wykorzystaniem zestawu 84 cech, które zostały podzielone na kategorie związane z zawodnikami (*Player*) oraz środowiskiem meczu (*Environment*). Wśród cech zawodników uwzględniono między innymi statystyki dotyczące wygranych i przegranych meczów, systemy rankingowe, wyniki na różnych nawierzchniach, a także wyniki ostatnich meczów. Natomiast cechy środowiskowe obejmowały typ nawierzchni, format meczu (*Best-of-X*) oraz zmienne czasowe związane z meczem.

Autor, podobnie jak Sipko, zastosował regresję logistyczną oraz jednowarstwową sieć neu-

ronową, testując modele pod kątem ich zastosowania na rynku zakładów bukmacherskich. De Seranno odnotował procentowy zysk na poziomie 19% w ciągu trzech sezonów, przy czym wynik ten osiągnięto dla jednowarstwowej sieci neuronowej, co również wskazuje na praktyczną użyteczność tego rodzaju modeli.

W analizowanych pracach procentowy zysk oraz zwrot z inwestycji odgrywają kluczową rolę jako wskaźniki oceny modeli w kontekście zakładów sportowych, ponieważ pozwalają na bezpośrednie porównanie wyników modeli z prognozami bukmacherów. Umożliwiają one również ocenę jakości modeli w sposób uniwersalny i niezależny od specyfiki danych testowych, takich jak rok, rodzaj turnieju czy poziom zawodników. Dzięki temu te metryki pozwalają porównywać wyniki między różnymi badaniami, nawet jeśli opierają się one na różnych zbiorach danych.

Metryki takie jak log-loss, wskaźnik Briera czy dokładność, mają ograniczoną przydatność w porównaniach między badaniami, ponieważ ich wartość zależy od charakterystyki danych testowych. Wartości te mogą się różnić dla tego samego modelu zarówno w zależności od okresu, z którego pochodzą dane, co wpływa na liczbę niespodziewanych zwycięstw, jak i od poziomu analizowanych turniejów, takich jak ATP 250 w porównaniu do Wielkiego Szlema. W efekcie porównywanie wyników między pracami staje się trudne, jeśli każda z nich korzysta z innego zestawu danych testowych, co jest częstą praktyką w badaniach nad przewidywaniem wyników meczów sportowych.

1.1.3. Wkład niniejszej pracy

Inżynieria cech

Niniejsza praca łączy podejścia zastosowane w dwóch kluczowych badaniach dotyczących przewidywania wyników meczów tenisowych, jednocześnie rozszerzając i modyfikując zestaw cech w celu zwiększenia ich użyteczności predykcyjnej.

W pracy Michala Sipko uwzględniono wyłącznie cechy związane z zawodnikami, pomijając zmienne środowiskowe, takie jak typ nawierzchni czy format meczu (*Best-of-X*). Ponadto, nie zastosowano żadnego innego systemu rankingowego poza rankingiem ATP.

Z kolei w badaniach Alexandra De Seranno, mimo uwzględnienia zarówno rankingu Elo, jak i zmiennych środowiskowych, brakowało szczegółowych statystyk meczowych, takich jak liczba asów, błędów serwisowych czy procent wygranych punktów przy serwisie.

Niniejsza praca uzupełnia te luki, integrując szczegółowe statystyki meczowe z innymi cechami opisującymi kontekst meczu. Dodatkowo zmodyfikowano niektóre cechy wprowadzone w poprzednich badaniach. Przykładowo, ranking Elo został wzbogacony o dodatkowe modyfikacje związane

z nawierzchnią oraz innymi czynnikami środowiskowymi, co uczyniło go bardziej dynamicznym i lepiej dostosowanym do specyfiki tenisa. Wprowadzono również zmiany w sposobie obliczania niektórych cech, takich jak zmęczenie zawodnika. Ponadto, dodano nowe cechy, które nie były obecne w żadnej z analizowanych prac, takie jak forma zawodnika w bieżącej edycji turnieju czy warunki pogodowe.

Modele uczenia

W niniejszej pracy zastosowano model XGBoost, który nie był wykorzystywany w badaniach Michala Sipko i Alexandra De Seranno. Jest to popularny algorytm oparty na zespołach drzew decyzyjnych. Celem jego zastosowania było sprawdzenie, czy dzięki swoim wbudowanym mechanizmom, takim jak ważenie cech czy elastyczne regularyzacje, XGBoost może przewyższyć inne metody uczenia maszynowego (regresję logistyczną oraz sieć neuronową) pod względem jakości predykcji.

W pracy zastosowano również bardziej złożone architektury sieci neuronowych, które w porównaniu do dotychczas wykorzystywanych jednowarstwowych sieci neuronowych mogą lepiej modelować złożone nieliniowe zależności w danych. Dodanie kolejnych warstw ukrytych umożliwia bardziej precyzyjne uchwycenie struktury danych, co jest szczególnie przydatne w analizie skomplikowanych cech.

Strona internetowa

Kolejnym istotnym wkładem niniejszej pracy jest stworzenie strony internetowej prezentującej wyniki modelu predykcyjnego oraz powiązane statystyki meczowe. Dotychczasowe badania, takie jak te przeprowadzone przez Michala Sipko i Alexandra De Seranno, koncentrowały się na tworzeniu i ocenie modeli predykcyjnych, jednak nie obejmowały opracowania narzędzia pozwalającego na ich praktyczne zastosowanie.

W niniejszej pracy opracowano platformę, która umożliwia użytkownikom łatwy dostęp do wyników modelu w przejrzystej formie, co znacząco zwiększa użyteczność wyników badania. Taka strona internetowa stanowi nowość w tej dziedzinie, łącząc elementy analizy danych z ich wizualizacją i praktycznym zastosowaniem, co nie było dotychczas poruszane w literaturze.

1.2. Obstawianie meczów tenisowych

Obstawianie meczów tenisowych łączy w sobie elementy przewidywalności i nieprzewidywalności, co czyni je szczególnie atrakcyjnym tematem zarówno dla graczy, jak i badaczy. Tenis, jako sport indywidualny, wyróżnia się specyficzną dynamiką, która odróżnia go od sportów drużynowych.

Mimo że system punktowy oraz indywidualny charakter zawodników mogą sugerować stosunkowo dużą przewidywalność wyników, w praktyce cechuje się on znaczną zmiennością, a każdy mecz niesie ze sobą potencjał niespodzianki.

W niniejszej pracy skupiono się na prognozowaniu zwycięzcy meczu przed jego rozpoczęciem. Takie podejście pozwala na uniknięcie konieczności pobierania danych na żywo, co upraszcza proces analizy i zmniejsza koszty operacyjne. Dodatkowo dane, które zostały wykorzystane, są oparte wyłącznie na informacjach dostępnych przed rozpoczęciem meczu, co oznacza, że nie pozwalają one na uczenie modeli do prognozowania wyników w trakcie meczu.

1.2.1. Kursy bukmacherskie

Kursy bukmacherskie to wartości określające potencjalny zwrot z poprawnie postawionego zakładu. Na przykład w meczu pomiędzy Alcarazem a Djokoviciem kurs na zwycięstwo Alcaraza wynoszący 2,0 oznacza, że za każdą postawioną złotówkę na jego wygraną można otrzymać dwa złote. Wielkość kursów jest determinowana przez ocenę prawdopodobieństwa danego wyniku przez bukmacherów. Kursy na bardziej spodziewane wyniki są niższe, natomiast na mniej prawdopodobne scenariusze – wyższe.

1.2.2. Przekształcenie kursów bukmacherskich na prawdopodobieństwa

Kursy bukmacherskie o_1 i o_2, przypisane dwóm zawodnikom, można przekształcić na implikowane prawdopodobieństwa p_1 i p_2. Proces ten odbywa się przy użyciu następujących wzorów:

$$p_1 = \frac{\frac{1}{o_1}}{\frac{1}{o_1} + \frac{1}{o_2}}, \tag{1.1}$$

$$p_2 = \frac{\frac{1}{o_2}}{\frac{1}{o_1} + \frac{1}{o_2}}, \tag{1.2}$$

gdzie:

- o_1, o_2 to kursy bukmacherskie na obu przeciwników,
- p_1, p_2 to znormalizowane prawdopodobieństwa wyniku meczu, przy czym $p_1 + p_2 = 1$.

Warto zauważyć, że kursy bukmacherskie zawierają tzw. margines (ang. *overround*), który powoduje, że surowe prawdopodobieństwa (bez normalizacji) sumują się do wartości większej niż 1. Normalizacja eliminuje ten efekt, umożliwiając porównanie wyników z innymi modelami.

1.2.3. Strategia obstawiania meczów

Kryterium Kelly'ego (ang. Kelly criterion) to metoda matematyczna stosowana w zarządzaniu kapitałem, która określa optymalną wielkość zakładu w celu maksymalizacji długoterminowego wzrostu kapitału. Opiera się na przewadze statystycznej gracza i uwzględnia prawdopodobieństwo sukcesu oraz stosunek zysku do ryzyka, minimalizując ryzyko bankructwa. Metoda ta gwarantuje lepsze wyniki w długim okresie niż jakakolwiek inna istotnie różna strategia [6]. Gracz wykorzystujący tę technikę obstawia ułamek wcześniej ustalonej maksymalnej wartości zakładu zgodnie ze wzorem:

$$Z_i = \begin{cases} s \cdot \frac{p_i^{\text{bettor}}(o_i+1)-1}{o_i}, & p_i^{\text{bettor}} > p_i^{\text{implied}} \\ 0 & \text{w p.p.} \end{cases} \tag{1.3}$$

gdzie:

- Z_i – wielkość zakładu,
- s – maksymalna wartość zakładu,
- p_i^{bettor} – ocena prawdopodobieństwa wygranej według gracza,
- p_i^{implied} – prawdopodobieństwo wynikające z kursów bukmacherskich,
- o_i – kurs bukmacherski na danego zawodnika.

1.3. Metryki

Ocena skuteczności modeli uczenia maszynowego wymaga odpowiednich metryk, które pozwalają mierzyć ich użyteczność w kontekście analizowanego problemu. Wybór właściwych wskaźników zależy od rodzaju modelu oraz celu analizy. W tej sekcji przedstawiono metryki, które zostały wykorzystane do oceny skuteczności modeli.

1.3.1. Dokładność

Dokładność (ang. accuracy) jest jednym z najprostszych wskaźników oceny modelu, który mierzy, jak często predykcje modelu przewidują większą szansę wygranej na faktycznego zwycięzcę meczu. Metryka nie jest wykorzystywana podczas optymalizacji modeli, ze względu na nastawienie autorów na dobrze skalibrowane wyniki probabilistyczne, a nie jedynie na klasyfikację wyników. Jednak ze względu na swoją łatwą interpretowalność służy ona jako jedna z metryk do oceny modelu na zbiorze testowym.

1.3.2. Logarytmiczna funkcja straty

Logarytmiczna funkcja straty (ang. log-loss) jest miarą, która ocenia, jak dobrze model przewiduje prawdopodobieństwa wyników. Jest to funkcja, która penalizuje błędy przewidywań w zależności od odległości pomiędzy przewidywanymi prawdopodobieństwami a rzeczywistymi wynikami. Funkcja ta jest szczególnie przydatna w przypadku, gdy celem jest uzyskanie jak najbardziej precyzyjnych prognoz [9], a zatem jest logicznym wyborem w przypadku rozważanego problemu.

Wartość logarytmicznej funkcji straty jest obliczana na podstawie różnicy pomiędzy przewidywanym prawdopodobieństwem a rzeczywistym wynikiem, przy czym większe błędy są silniej karane. Zatem im mniejsze straty, tym lepsze dopasowanie modelu do danych.

1.3.3. Wskaźnik Briera

Wskaźnik Briera (ang. Brier score) to inna metryka stosowana do oceny jakości prognoz probabilistycznych. Jest to średnia kwadratów różnic pomiędzy przewidywanymi prawdopodobieństwami a rzeczywistymi wynikami (0 lub 1 w przypadku zwycięzcy) opisana wzorem:

$$BS = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{p}_i)^2, \tag{1.4}$$

gdzie:

- N – liczba obserwacji,
- y_i – rzeczywista wartość (0 lub 1) dla i-tej obserwacji,
- $\hat{p}_i$ – przewidywane prawdopodobieństwo dla i-tej obserwacji.

Podobnie jak logarytmiczna funkcja straty, wskaźnik Briera penalizuje większe różnice pomiędzy przewidywaniami a rzeczywistością, co pozwala na dokładniejszą ocenę skuteczności modelu.
Jest to miara, która wskazuje, jak dobrze przewidywane prawdopodobieństwa są zgodne z rzeczywistymi wynikami, przy czym mniejsze wartości wskaźniku Briera oznaczają lepszą jakość prognoz.

1.3.4. Zwrot z inwestycji

Zwrot z inwestycji (ang. return on investment, ROI) to jedna z najczęściej stosowanych metryk w ocenie skuteczności inwestycji, w tym również w analizie zakładów bukmacherskich. ROI pozwala oszacować procentowy zwrot zainwestowanego kapitału i jest definiowane jako:

$$ROI = \frac{\text{zysk netto}}{\text{całkowita suma stawek}} \cdot 100\%, \tag{1.5}$$

gdzie:

- zysk netto – różnica między uzyskanym zyskiem a zainwestowanym kapitałem,
- całkowita suma stawek – suma wszystkich środków przeznaczonych na zakłady.

Wartość ROI wyrażana jest w procentach i pozwala na porównanie efektywności prognoz różnych modeli predykcyjnych. Przykładowo, ROI wynoszące 5% oznacza, że każda zainwestowana jednostka kapitału przyniosła 5% zysku netto.

1.3.5. Procentowy zysk

W przedstawianym rozwiązaniu procentowy zysk zdefiniowano jako:

$$PZ = \frac{S_k - S_p}{S_p} \cdot 100\%, \tag{1.6}$$

gdzie:

- S_k – ilość dostępnych środków po zakończonych inwestycjach,
- S_p – ilość dostępnych środków na początku.

Metryka ta została wykorzystana obok popularnego zwrotu z inwestycji, ponieważ sam ROI może mieć pewne ograniczenia przy porównywaniu modeli. W sytuacji, gdy modele osiągają jednakową wartość ROI, ale różnią się liczbą obstawionych meczów, trudno uznać je za równorzędne pod względem efektywności. Na przykład, model, który obstawi tylko jeden zwycięski zakład za jedną jednostkę przy kursie równym dwa, osiągnie ROI na poziomie 100%, generując jednak bardzo niewielki realny zysk. Z kolei model z ROI wynoszącym 5%, ale obstawiający setki zakładów, może znacząco zwiększyć ogólny kapitał. Dlatego liczba obstawionych zakładów i rzeczywista zmiana stanu środków są równie ważne dla pełniejszej oceny skuteczności modelu.

2. Narzędzia i technologie

2.1. Język programowania

Projekt został zrealizowany w języku `Python`, który zapewnia szeroką gamę bibliotek i narzędzi, które wspierają zarówno przetwarzanie danych, jak i implementację zaawansowanych modeli uczenia maszynowego.

2.2. Narzędzia do analizy danych

- **Pandas**: Pandas jest jedną z kluczowych bibliotek w Pythonie w dziedzinie analizy i przetwarzania danych, cechującą się swoją wydajnością i łatwością obsługi. Został wykorzystany do przetworzenia danych o meczach tenisowych pochodzących z różnych źródeł oraz przygotowania danych do postaci, która została wykorzystana przez modele.
- **Scikit-learn**: Scikit-learn to biblioteka uczenia maszynowego oferująca szeroką gamę modeli i narzędzi do ich oceny. W projekcie wykorzystano ją do implementacji modelu regresji logistycznej, a także do obliczania różnych metryk oceny modeli, takich jak dokładność, logarytmiczna funkcja straty (log-loss) oraz wskaźnik Briera.
- **PyTorch**: Jedna z najpopularniejszych bibliotek do uczenia maszynowego, opracowana przez Facebook AI Research (FAIR). Pozwala na budowanie, trenowanie i wdrażanie modeli uczenia maszynowego, w tym sieci neuronowej użytej w projekcie.
- **GeoPy**: GeoPy to biblioteka Python umożliwiająca wykonywanie geokodowania, odwróconego geokodowania oraz obliczania odległości geograficznych. W kontekście przedstawianego projektu GeoPy został użyty do przetwarzania danych geograficznych dotyczących lokalizacji turniejów.

2.3. Open-Meteo Historical Weather API

Dla uzyskania danych pogodowych podczas turniejów tenisowych skorzystano z Open-Meteo Historical Weather API. To API pozwala na dostęp do historycznych danych pogodowych, takich jak temperatura, opady, prędkość wiatru i inne parametry atmosferyczne. Główne zalety API to:

- **Szczegółowe dane historyczne:** Możliwość uzyskania szczegółowych danych pogodowych z przeszłości, które mogą wpływać na wyniki meczów tenisowych.
- **Łatwość integracji:** API jest łatwe do integracji z aplikacjami Python dzięki prostemu formatowi JSON, co umożliwia szybkie pobieranie i przetwarzanie danych.

2.4. Optuna

Do optymalizacji modeli wykorzystano oprogramowanie Optuna. Technologia ta umożliwia optymalizację hiperparametrów przy użyciu funkcji celu oraz optymalizacji bayesowskiej. Posiada również panel internetowy działający w czasie rzeczywistym, który pozwala na przegląd kluczowych właściwości, takich jak historia optymalizacji oraz ważność hiperparametrów, prezentowane za pomocą wykresów i tabel.

2.5. Technologie internetowe

- **Django**: Django to popularny framework internetowy napisany w Pythonie, umożliwiający szybkie tworzenie skalowalnych aplikacji internetowych. W naszej pracy Django pełni rolę narzędzia, które udostępnia prognozy meczów użytkownikom oraz zapewnia integrację między stroną a modelami. Django oferuje wbudowane mechanizmy do tworzenia API oraz zarządzania bazami danych, co czyni go idealnym narzędziem do szybkiego prototypowania i budowy skalowalnych aplikacji. Dzięki Django REST Framework (DRF) można w prosty sposób tworzyć i wystawiać API w stylu RESTful. Django zawiera również wbudowaną obsługę bazy danych SQLite, co pozwala na szybkie rozpoczęcie pracy bez potrzeby konfiguracji zewnętrznego systemu baz danych. Dzięki tym funkcjom Django pozwala skupić się na logice biznesowej, minimalizując nakład pracy związany z konfiguracją infrastruktury backendowej.

3. Zbiory danych

W projekcie zostały wykorzystywane ogólnodostępne zbiory danych:

- Zbiór danych 1 – baza danych zawierająca wyniki meczów i kursy bukmacherskie

- Zbiór danych 2 – obszerna baza danych o meczach ATP, zawierająca szczegółowe informacje na temat zawodników, wyników, lokalizacji meczów oraz rodzajów nawierzchni.

3.1. Zbiór danych 1

Pierwszy zbiór zawiera pogrupowane po roku pełne informacje o rezultatach meczów w poszczególnych sezonach.

Tab. 3.1. Rodzaj danych w zbiorze nr 1

Dostępność	Ogólnodostępne
Format danych	CSV
Częstotliwość odświeżania	miesięczna

Każdy plik CSV składa się z następujących kolumn:

Informacje ogólne

ATP – ID turnieju,
Location – lokalizacja,
Tournament – nazwa turnieju,
Date – data meczu,
Series – ranga turnieju,
Court – rodzaj kortu,

Surface – rodzaj nawierzchni,

Round – runda turnieju,

Best of – informacja, do ilu wygranych setów jest rozgrywany mecz,

Winner – wygrany meczu,

Loser – przegrany meczu,

Wrank – pozycja wygranego w rankingu przed meczem,

Lrank – pozycja przegranego w rankingu przed meczem,

WPts – liczba punktów wygranego w rankingu przed meczem,

LPts – liczba punktów przegranego w rankingu przed meczem,

W1, W2, W3, W4, W5 – liczba wygranych gemów w poszczególnych setach przez zwycięzcę spotkania,

L1, L2, L3, L4, L5 – liczba wygranych gemów w poszczególnych setach przez pokonanego zawodnika,

Wsets|Lsets – liczba wygranych setów przez wygranego|przegranego,

Comment – komentarz do meczu (zakończony, poddany lub walkower).

Informacje o kursach

(Nie wszystkie pliki zawierają wszystkie poniższe kolumny)

B365W – kursy Bet365 na zwycięzcę meczu,

B365L – kursy Bet365 na przegranego meczu,

B&WW – kursy zakładów Bet&Win na zwycięzcę meczu,

B&WL – kursy zakładów Bet&Win na przegranego meczu,

CBW – kursy zakładów Centrebet na zwycięzcę meczu,

CBL – kursy zakładów Centrebet na przegranego meczu,

EXW – kursy zakładów Expekt na zwycięzcę meczu,

EXL – kursy zakładów Expekt na przegranego meczu,

LBW – kursy zakładów Ladbrokes na zwycięzcę meczu,

LBL – kursy zakładów Ladbrokes na przegranego meczu,

GBW – kursy zakładów Gamebookers na zwycięzcę meczu,

GBL – kursy zakładów Gamebookers na przegranego meczu,

IWW – kursy zakładów Interwetten na zwycięzcę meczu,

IWL – kursy zakładów Interwetten na przegranego meczu,

PSW – kursy zakładów Pinnacles Sports na zwycięzcę meczu,

PSL – kursy zakładów Pinnacles Sports na przegranego meczu,

SBW – kursy zakładów Sportingbet na zwycięzcę meczu,

SBL – kursy zakładów Sportingbet na przegranego meczu,

SJW – kursy zakładów Stan James na zwycięzcę meczu,

SJL – kursy zakładów Stan James na przegranego meczu,

UBW – kursy zakładów Unibet na zwycięzcę meczu,

UBL – kursy zakładów Unibet na przegranego meczu,

MaxW – maksymalne kursy na zwycięzcę meczu (według portalu Oddsportal.com),

MaxL – Maksymalne kursy na przegranego meczu (według portalu Oddsportal.com),

AvgW – średnie kursy na zwycięzcę meczu (według portalu Oddsportal.com),

AvgL – średnie kursy na przegranego meczu (według portalu Oddsportal.com).

3.2. Zbiór danych 2

Ten plik zawiera historyczne rankingi, wyniki i statystyki meczowe. Większość kolumn w plikach wyników jest samowyjaśniająca.

Tab. 3.2. Rodzaj danych w zbiorze nr 2

Dostępność	Ogólnodostępne
Format danych	CSV
Częstotliwość odświeżania	roczna

Spis kolumn

tourney_id – unikalny identyfikator dla każdego turnieju, na przykład 2020-888. Dokładne formaty są zapożyczone z różnych źródeł, więc chociaż pierwsze cztery znaki zawsze oznaczają rok, reszta identyfikatora nie podąża za przewidywalną strukturą,

tourney_name – nazwa turnieju,

surface – rodzaj nawierzchni,

draw_size – liczba zawodników w drabince, często zaokrąglana do najbliższej potęgi liczby 2 (Na przykład turniej z 28 zawodnikami może być podany jako 32),

tourney_level – Dla mężczyzn: 'G' = wielkie szlemy, 'M' = Masters 1000, 'A' = inne turnieje rangi tour, 'C' = Challengers, 'S' = Satelity/ITF-y, 'F' = finały sezonowe i inne wydarzenia kończące sezon, 'D' = Puchar Davisa,

tourney_date – osiem cyfr, YYYYMMDD, zazwyczaj poniedziałek tygodnia turnieju,

match_num – identyfikator meczu, często zaczynający się od 1, czasami liczący się od 300 w dół, a czasami arbitralny,

winner_id – identyfikator gracza używany w tym repozytorium dla zwycięzcy meczu,

winner_seed – numer rozstawienia zwycięzcy,

winner_entry – 'WC' = dzika karta, 'Q' = kwalifikant, 'LL' = szczęśliwy przegrany, 'PR' = chronione miejsce w rankingu, 'ITF' = zgłoszenie ITF, oraz kilka innych, które są czasami używane,

winner_name – imię i nazwisko zwycięzcy,

winner_hand – P = praworęczny, L = leworęczny, U = nieznane. Dla graczy oburęcznych jest to ich ręka serwisowa,

winner_ht – wzrost w centymetrach, jeśli dostępny,

winner_ioc – trzyznakowy kod kraju,

winner_age – wiek, w latach, na dzień turnieju,

loser_id – identyfikator przegranego gracza,

loser_seed – jw.,

loser_entry – jw.,

loser_name – jw.,

loser_hand – jw.,

loser_ht – jw.,

loser_ioc – jw.,

loser_age – jw.,

score – wynik meczu,

best_of – '3' lub '5', wskazujące liczbę setów w meczu,

round – runda turnieju,

minutes – czas trwania meczu, jeśli dostępny,

w_ace – liczba asów zwycięzcy,

w_df – liczba podwójnych błędów zwycięzcy,

w_svpt – liczba serwisów zwycięzcy,

w_1stIn – liczba pierwszych serwisów zwycięzcy,

w_1stWon – liczba wygranych punktów po pierwszym serwisie zwycięzcy,

w_2ndWon – liczba wygranych punktów po drugim serwisie zwycięzcy,

w_SvGms – liczba gier serwisowych zwycięzcy,

w_bpSaved – liczba obronionych breakpointów przez zwycięzcę,

w_bpFaced – liczba breakpointów, które bronił zwycięzca,

l_ace – liczba asów przegranego,

l_df – liczba podwójnych błędów przegranego,

l_svpt – liczba serwisów przegranego,

l_1stIn – liczba pierwszych serwisów przegranego,

l_1stWon – liczba wygranych punktów po pierwszym serwisie przegranego,

l_2ndWon – liczba wygranych punktów po drugim serwisie przegranego,

l_SvGms – liczba gier serwisowych przegranego,

l_bpSaved – liczba obronionych breakpointów przez przegranego,

l_bpFaced – liczba breakpointów, których bronił przegrany,

winner_rank – ranking ATP lub WTA zwycięzcy, na dzień turnieju lub najnowsza data rankingu przed datą turnieju,

winner_rank_points – liczba punktów rankingowych, jeśli dostępna,

loser_rank – jw.,

loser_rank_points – jw..

3.3. Ocena jakości danych

Tab. 3.3. Raport braków danych dla zbioru danych 1

ZBIÓR DANYCH 1		
KATEGORIA	**LICZBA**	**KOMENTARZ**
Wymiary ramki danych	14111 rzędów x 39 kolumn	
	LICZBA BRAKUJĄCYCH WARTOŚCI	
Best of	15	wymaga obsługi
WRank	2	zawodnik bez rankingu
LRank	15	zawodnik bez rankingu
WPts	2	zawodnik bez rankingu
LPts	15	zawodnik bez rankingu
W1	57	wymaga obsługi
L1	56	wymaga obsługi
W2	115	wymaga obsługi/krótszy mecz
L2	115	wymaga obsługi/krótszy mecz
W3	4,078	wymaga obsługi/krótszy mecz
L3	4,078	wymaga obsługi/krótszy mecz
W4	7,012	wymaga obsługi/krótszy mecz
L4	7,012	wymaga obsługi/krótszy mecz
W5	7,525	wymaga obsługi/krótszy mecz
L5	7,525	wymaga obsługi/krótszy mecz
Wsets	56	wymaga obsługi
Lsets	57	wymaga obsługi
B365W	36	nie do modelu
B365L	36	nie do modelu
PSW	34	nie do modelu
PSL	34	nie do modelu
MaxW	10	nie do modelu
MaxL	10	nie do modelu
AvgW	10	nie do modelu
AvgL	10	nie do modelu

Tab. 3.4. Raport braków danych dla zbioru danych 2

ZBIÓR DANYCH 2		
KATEGORIA		**KOMENTARZ**
Wymiary ramki danych	14111 rzędów x 52 kolumn	
	LICZBA BRAKUJĄCYCH WARTOŚCI	
surface	53	wymaga obsługi
winner_seed	5,001	nierozstawiony zawodnik
winner_entry	7,318	zwykłe wejście do turnieju
winner_ht	282	wymaga obsługi
winner_age	3	wymaga obsługi
loser_seed	6,469	nierozstawiony zawodnik
loser_entry	6,652	zwykłe wejście do turnieju
loser_hand	1	wymaga obsługi
loser_ht	519	wymaga obsługi
loser_age	9	wymaga obsługi
minutes	595	wymaga obsługi
w_ace	440	wymaga obsługi
w_df	440	wymaga obsługi
w_svpt	440	wymaga obsługi
w_1stIn	440	wymaga obsługi
w_1stWon	440	wymaga obsługi
w_2ndWon	440	wymaga obsługi
w_SvGms	439	wymaga obsługi
w_bpSaved	440	wymaga obsługi
w_bpFaced	440	wymaga obsługi
l_ace	440	wymaga obsługi
l_df	440	wymaga obsługi
l_svpt	440	wymaga obsługi
l_1stIn	440	wymaga obsługi
l_1stWon	440	wymaga obsługi
l_2ndWon	440	wymaga obsługi
l_SvGms	439	wymaga obsługi
l_bpSaved	440	wymaga obsługi

KATEGORIA		KOMENTARZ
l_bpFaced	440	wymaga obsługi
winner_rank	33	zawodnik bez rankingu
winner_rank_points	33	zawodnik bez rankingu
loser_rank	75	zawodnik bez rankingu
loser_rank_points	75	zawodnik bez rankingu

Kolumny nieuwzględnione w tabelach nie mają żadnych braków danych.

3.4. Przetworzenie danych

W ramach łączenia zbiorów dane źródłowe zostały przetworzone w języku `Python`. Celem przetworzenia było stworzenie w obu tabelach klucza meczu, który by to umożliwił. Stworzone zostały kolumny zawierające informacje potrzebne do unikalnej identyfikacji. Głównymi krokami były:

1. W zbiorze nr 1 stworzono nową kolumnę zawierającą identyfikator zawodnika (winner_id, loser_id), pobierając wartości tych kolumn ze zbioru nr 1 na podstawie tego samego nazwiska i pierwszej litery imienia zawodnika. Ten krok wymagał dla niektórych zawodników ręcznych poprawek.
2. W zbiorze nr 2 zmieniono rangę turnieju na format znajdujący się w zbiorze nr 1.
3. Dodano kolumnę z lokalizacją turnieju dla zbioru danych nr 2, bazując na nazwie turnieju, z ręcznymi poprawkami, gdy to było konieczne.
4. Stworzono nową kolumnę zawierającą identyfikator turnieju (tournament_id) na podstawie lokalizacji i rangi turnieju.
5. Stworzono nową kolumnę, która będzie kluczem głównym, łącząc identyfikator meczu (match_id) na podstawie ID turnieju, ID obu zawodników i roku.

W wyniku połączenia zbiorów uzupełnione zostały niektóre z występujących braków danych.

4. Inżynieria cech

Jakość danych wejściowych ma kluczowe znaczenie dla efektywności modelu predykcyjnego. Proces inżynierii cech polega na przekształcaniu surowych danych w sposób, który umożliwia modelowi skuteczniejsze rozpoznawanie wzorców i zależności. W niniejszym projekcie dane zostały podzielone na trzy kategorie: cechy zawodników, cechy środowiskowe oraz cechy analityczne. W tym rozdziale omówiono proces pozyskiwania, przetwarzania oraz tworzenia cech, które zostały wykorzystane w trenowaniu modeli.

4.1. Wiek zawodnika

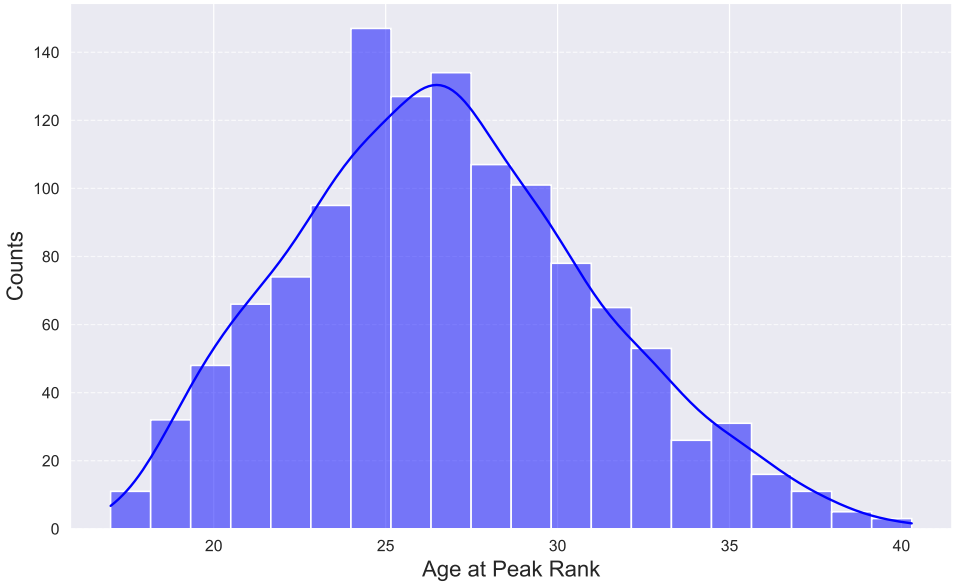
Mlakar i Tusar [8] ustalili, że najbardziej prawdopodobnym wiekiem szczytu formy zawodnika jest 25 lat. Może to wynikać z faktu, że starsi zawodnicy mają większe doświadczenie w kluczowych momentach meczu, co pozwala im podejmować lepsze decyzje taktyczne, natomiast młodsi zawodnicy zazwyczaj dysponują większą siłą fizyczną oraz wytrzymałością. Wiek w połowie kariery zawodniczej, około 25 lat, wydaje się optymalną kombinacją tych dwóch czynników. Potwierdza to również wykres 4.1 przedstawiający rozkład wieku zawodników, w którym osiągnęli swoją najlepszą pozycję rankingową. Uwzględniono zatem wiek zawodnika jako oddzielną cechę:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{age}\}. \tag{4.1}$$

Dzięki temu model może lepiej odzwierciedlać różnice między zawodnikami znajdującymi się w różnych fazach kariery.

4.2. Dominująca ręka zawodnika

Dominująca ręka zawodnika (lewa lub prawa) również może mieć znaczenie w kontekście przewidywania wyników. Leworęczni zawodnicy często mają przewagę taktyczną, wynikającą



Rys. 4.1. Rozkład wieku w momencie osiągnięcia najwyższego rankingu

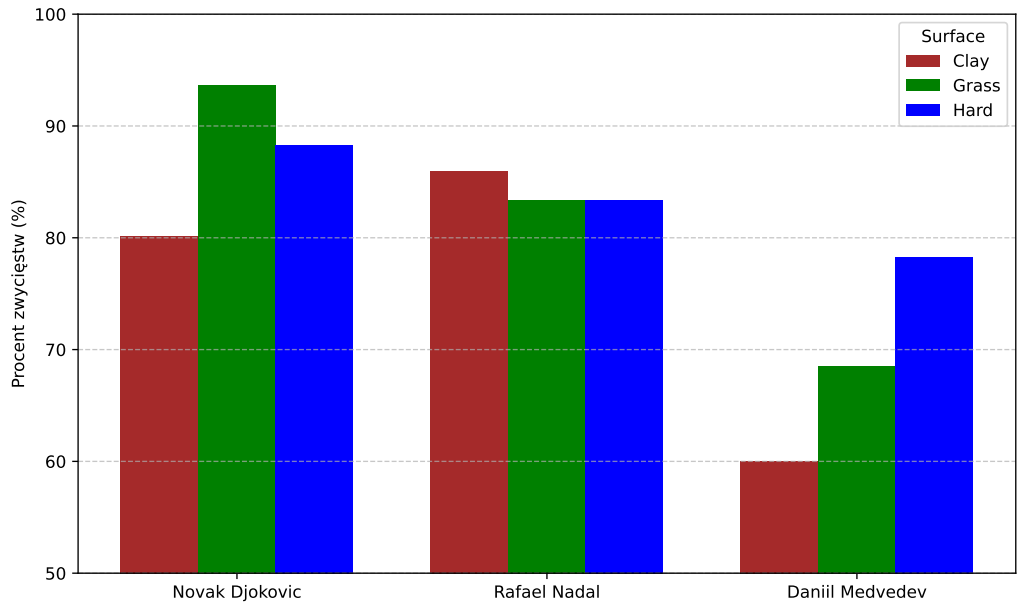
z nietypowego kąta serwisu i trajektorii uderzeń, co może wpływać na skuteczność ich gry przeciwko praworęcznym przeciwnikom. W modelu uwzględniono tę cechę jako zmienną binarną:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{is right handed}\}. \tag{4.2}$$

4.3. Nawierzchnia

Mecze tenisowe rozgrywane są na trzech różnych nawierzchniach: twardej, trawiastej i ceglanej. Każda z nich cechuje się inną charakterystyką takich własności jak m.in. szybkość odbicia piłki, wysokość odbicia piłki, czy łatwość poślizgu butów. Rodzaj nawierzchni zatem może mieć duży wpływ na wyniki zawodnika, gdyż zawodnik lepiej radzi sobie na nawierzchniach, które wynagradzają jego styl gry i gorzej na innych. Potwierdza to wykres 4.2 oraz analiza jakości gry topowych zawodników wykonana przez Barnetta [1]. Przypomnijmy, że oryginalna ramka danych zawiera informację o tym, na jakiej nawierzchni rozgrywany jest mecz, w postaci kolumny zawierającej trzy możliwe wartości: hard, clay, grass. Wystarczy zatem jedynie zastosować one-hot encoding, aby otrzymać reprezentację:

$$\{\text{is hard}, \text{is clay}, \text{is grass}\}. \tag{4.3}$$



Rys. 4.2. Procent zwycięstw czołowych zawodników ze względu na nawierzchnię

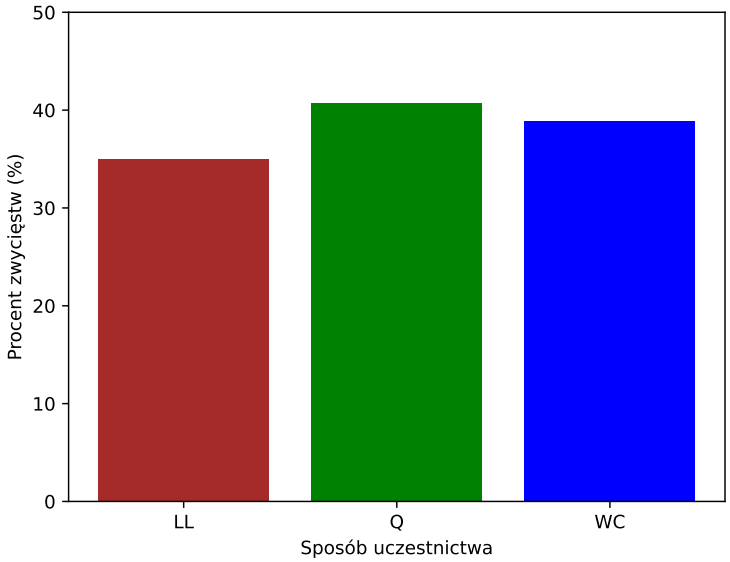
4.4. Sposób uczestnictwa zawodnika w turnieju

W zbiorze danych znajduje się kolumna `winner_entry` oraz `loser_entry`, która opisuje sposób, w jaki zawodnik zakwalifikował się do turnieju. Może ona przyjmować następujące wartości:

- **LL (Lucky Loser)** – zawodnik, który przegrał w kwalifikacjach, ale dostał się do turnieju głównego w wyniku wycofania się innego zawodnika,
- **Q (Qualifier)** – zawodnik, który przeszedł przez kwalifikacje,
- **WC (Wild Card)** – zawodnik zaproszony do turnieju głównego przez organizatorów,
- wartość pusta – oznaczająca standardowe zakwalifikowanie się na podstawie rankingu.

Sposób uczestnictwa zawodnika w turnieju może mieć istotny wpływ na wyniki meczów. Oprócz tego, że niestandardowe uczestnictwo oznacza ranking zbyt niski na standardowy start w turnieju, to przykładowo:

- **Szczęśliwi przegrani** często mają mniej czasu na przygotowanie się do turnieju, co może negatywnie wpływać na ich wyniki.
- **Kwalifikanci** muszą rozegrać więcej meczów przed turniejem głównym, co może powodować zmęczenie, ale jednocześnie pozwala im lepiej dostosować się do warunków.



Rys. 4.3. Procent zwycięstw dla sposobów uczestnictwa w turnieju

- **Zawodnicy z dziką kartą** są zapraszani przez organizatorów i często reprezentują młodych zawodników lub lokalnych faworytów.

Negatywny wpływ na procent zwycięstw zawodników, którzy występują w turnieju poprzez jeden z tych trzech niestandardowych sposobów, potwierdza wykres 4.3. Aby uwzględnić tę informację w modelu, wykonano one-hot encoding dla wartości niepustych w kolumnach `winner_entry` oraz `winner_entry`. Powstały następujące cechy binarne:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{is LL}, \text{is Q}, \text{is WC}\}. \tag{4.4}$$

4.5. Rozstawienie zawodnika

W zbiorze danych znajdują się kolumny `winner_seed` oraz `loser_seed`, które zawierają informacje o rozstawieniu zawodnika w turnieju. Wartość w tej kolumnie odpowiada pozycji zawodnika na liście rozstawionych graczy, określanej na podstawie rankingu światowego. Jeśli zawodnik nie był rozstawiony, kolumna przyjmuje wartość pustą. Chociaż mogłoby się wydawać, że uwzględnianie rozstawienia zawodnika jest niepotrzebne, gdy w zbiorze cech znajduje się informacja o rankingu, to nie jest to prawda. Wynika to z faktu, że rozstawienie wpływa bezpośrednio na przebieg drabinki turniejowej, co może zmieniać dynamikę gry i potencjalne wyniki. Aby uniknąć braków danych i zawrzeć kluczową informację znajdującą się w tych kolumnach `winner_seed`

oraz `loser_seed`, zostały one przekształcone na cechy binarne postaci:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{is seeded}\}. \tag{4.5}$$

4.6. Ranking

Ranking ATP jest obecnie oficjalnym systemem klasyfikacji zawodników tenisowych. Bazuje on na punktach zdobywanych przez zawodników w turniejach w ciągu ostatnich 52 tygodni. Chociaż ranking ATP jest szeroko uznawanym standardem, jego ograniczenia mogą wpływać na dokładność analizy formy zawodników oraz przewidywania wyników przyszłych spotkań.

4.6.1. Wady rankingu ATP

1. **Uwzględnia tylko punkty z ostatniego roku:** Na ranking składają się jedynie punkty zdobyte przez ostatni rok, po czym są one usuwane. Oznacza to, że ranking nie uwzględnia żadnych wyników zawodnika sprzed więcej niż roku.

2. **Brak uwzględnienia jakości przeciwników:** W rankingu ATP liczba punktów przyznawanych za zwycięstwo jest niezależna od jakości rywala. Dla takiej samej rundy w turnieju tej samej kategorii, zwycięstwo nad zawodnikiem spoza pierwszej setki daje taką samą liczbę punktów jak wygrana z zawodnikiem z czołówki rankingu.

3. **Brak utraty punktów** Nie da się stracić punktów ATP po rozegranym meczu, co oznacza, że większa liczba rozegranych turniejów będzie skutkowała większą ilością punktów, co niekoniecznie świadczy o większych umiejętnościach zawodnika.

4.6.2. Zalety systemu Elo w porównaniu do rankingu ATP

System rankingu Elo [5], zaprojektowany pierwotnie dla szachów, oferuje bardziej dynamiczne i precyzyjne podejście do oceny poziomu zawodników. W porównaniu do rankingu ATP, system Elo wyróżnia się następującymi zaletami:

1. **Brak wygasania punktów:** Ponieważ w rankingu Elo można tracić i zyskiwać punkty to nie ma potrzeby na wygasanie starszych punktów.

2. **Uwzględnienie jakości przeciwników:** Prawdopodobieństwo zwycięstwa zawodnika jest obliczane na podstawie różnicy w rankingach obu graczy. Zwycięstwo nad wyżej notowanym rywalem skutkuje większym wzrostem rankingu, co odzwierciedla trudność meczu.

3. **Dynamiczne współczynniki:** Współczynnik K (kontrolujący wagę zmian w rankingu) może być dostosowywany do różnych czynników jak np. ostatniej passy zwycięstw/porażek (momentum) zawodnika czy liczby rozegranych meczów, co pozwala na bardziej precyzyjne odwzorowanie aktualnej formy zawodnika.

Dzięki tym cechom ranking Elo stanowi potencjalnie lepsze narzędzie zarówno dla analizy wyników historycznych, jak i przewidywania rezultatów przyszłych spotkań tenisowych.
Zaimplementowano zatem ranking Elo, uwzględniając jednak parę modyfikacji specjalnie pod realia tenisa. Wybór modyfikacji oraz konkretnych wartości hiperparametrów implementowanego rankingu Elo dokonany został po szeregu eksperymentów i optymalizacji.

4.6.3. Opis zaimplementowanego rankingu Elo

System opiera się na kilku kluczowych elementach:

1. **Początkowe wartości rankingu.**
   W odróżnieniu od klasycznego rankingu Elo, gdzie wszyscy zawodnicy rozpoczynają z równym Elo (zazwyczaj 1500), zawodnicy zaczynają z wartościami rankingu zależnymi od ich miejsca w rankingu ATP, aby uniknąć negatywnych efektów początkowego okresu, w którym ranking się stabilizuje. Początkowe rankingi zostały przypisane następująco:
   - Zawodnicy z rankingu ATP w **pierwszej dziesiątce** otrzymują początkowy ranking Elo równy **2000 punktów**.
   - Zawodnicy zajmujący miejsca od **11 do 50** rozpoczynają z wartością **1900 punktów**.
   - Zawodnicy zajmujący miejsca od **51 do 100** rozpoczynają z wartością **1800 punktów**.
   - Zawodnicy spoza pierwszej setki, ale w **pierwszej pięćsetce**, otrzymują początkowy ranking równy **1600 punktów**.
   - Pozostali zawodnicy (spoza pierwszej pięćsetki) zaczynają z wartością **1500 punktów**.

2. **Dynamiczne współczynniki K.**
   (a) **Podstawowy współczynnik K:** Liczba rozegranych meczów przez zawodnika przez ostatni rok wpływa na wartość bazową współczynnika K. Wartość ta maleje wraz ze wzrostem liczby meczów, co odzwierciedla stabilizację rankingu bardziej doświadczonych zawodników:

$$K_{\text{base}} = \frac{210}{\sqrt{\text{total\_matches} + 5}}, \tag{4.6}$$

gdzie:

- total_matches to liczba wszystkich meczów rozegranych przez zawodnika.
- Dodanie stałej 5 w mianowniku zapobiega nieproporcjonalnie dużym wartościom K na początku kariery zawodnika.

Wartości 210 oraz 5 we wzorze zostały dobrane w sposób eksperymentalny.

(b) **Uwzględnienie momentum zawodnika:** Wartość K jest dodatkowo modyfikowana przez czynnik *momentum*, który uwzględnia bieżącą formę zawodnika. Wartość *momentum*:

- Rośnie o 0,06 wraz z serią zwycięstw zawodnika (maksymalnie do 1,3).
- Maleje o 0,06 w przypadku serii porażek (minimalnie do 0,7).

Współczynnik K po uwzględnieniu momentum jest obliczany jako:

$$K = K_{\text{base}} \cdot \text{momentum}. \tag{4.7}$$

Podobnie jak dla podstawowego współczynnika K wartości 0,06, 1,3 oraz 0,7 we wzorze zostały dobrane w sposób eksperymentalny.

(c) **Uwzględnienie nawierzchni:** System oddzielnie przechowuje rankingi dla każdej nawierzchni i oblicza wartość *blended Elo*, która jest średnią ważoną rankingu ogólnego i rankingu dla konkretnej nawierzchni. Wagi zależą od liczby meczów rozegranych na danej nawierzchni:

$$\text{blended Elo} = w \cdot \text{surface Elo} + (1 - w) \cdot \text{general Elo}, \tag{4.8}$$

gdzie w zależy od liczby meczów rozegranych na nawierzchni (większa liczba meczów zwiększa w).

4.6.4. Proces aktualizacji rankingu

Wzory zastosowane do aktualizacji rankingu są standardowymi formułami używanymi w konstrukcji rankingu Elo i nie zawierają żadnych modyfikacji wprowadzonych przez autorów pracy. Po każdym meczu system aktualizuje rankingi zawodników na podstawie prawdopodobieństwa zwycięstwa obliczanego według wzoru:

$$P(A) = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}}, \tag{4.9}$$

gdzie:

- $P(A)$ to oczekiwane prawdopodobieństwo zwycięstwa zawodnika A,

- R_A i R_B to rankingi zawodników A i B.

Nowe wartości rankingu są obliczane według wzoru:

$$R_{\text{nowy}} = R_{\text{stary}} + K \cdot (\text{wynik} - P), \tag{4.10}$$

gdzie wynik wynosi 1 dla zwycięzcy i 0 dla przegranego.

Zaimplementowany ranking Elo wykazuje lepsze od rankingu ATP wartości predykcyjne, osiągając o prawie 2,5 punktu procentowego lepszą dokładność przewidywań i około 0,012 mniejszy wskaźnik Briera.

Tab. 4.1. Porównanie skuteczności rankingu ATP i rankingu Elo

Metryka	**Ranking ATP**	**Ranking Elo**
Dokładność przewidywań (%)	63,05	64,37
Wskaźnik Briera	0,2324	0,2203

Pomimo wyznaczenia *blended Elo*, zdecydowano do zbioru cech dołączyć również *surface Elo* i *general Elo*, gdyż model może uznać inną ich kombinację za bardziej wartościową oraz będzie miał dostęp do większej ilości informacji. Reprezentujemy więc ranking Elo poprzez cechy:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{blended Elo}, \text{general Elo}, \text{surface Elo}\}. \tag{4.11}$$

Mimo lepszych wyników rankingu Elo zdecydowano pozostawić w zbiorze cech ranking ATP, gdyż może on być przydatny do modelu dla meczów z początku zbioru danych, gdy ranking Elo nie miał jeszcze wystarczająco czasu, aby się ustabilizować. Zawiera on też w pewien sposób dodatkowe informacje, które potencjalnie model będzie w stanie wykorzystać, nawet jeżeli dla wybranej metryki prezentuje się on gorzej. Cechy związane z rankingiem ATP zawodników mają zatem następującą reprezentację:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{ATP Rank}\}. \tag{4.12}$$

4.7. H2H

Podczas analizy danych dotyczących wyników meczów tenisowych istotnym elementem było uwzględnienie relacji pomiędzy zawodnikami, którzy wcześniej rywalizowali ze sobą. Wiadome jest, że niektórzy zawodnicy grają dobrze/słabo na konkretnych przeciwników, co może wynikać ze specjalnego zestawienia stylów gry lub innych czynników. Aby zawrzeć tę informację, wprowadzone zostały dwie cechy, które zawierają liczbę wygranych we wspólnych starciach dla obu zawodników. Z podobnego powodu, dla którego tworzymy cechę informującą o nawierzchni przewidywanego meczu, tj. faktu, że wpływa ona znacząco na rozgrywkę w zależności od stylu gry zawodnika, tworzymy dodatkowe dwie cechy, które zawierają liczby zwycięstw we wspólnych pojedynkach ograniczone do tej samej nawierzchni, co przewidywany mecz. Razem otrzymujemy cztery nowe cechy:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{wins}\} \times \{\text{career}, \text{surface}\}. \tag{4.13}$$

4.8. Do x wygranych setów

W zbiorze danych znajduje się kolumna `best_of`, która określa liczbę setów potrzebnych do zwycięstwa w meczu. Może ona przyjmować dwie wartości:

- 5 – mecz rozgrywany jest do trzech wygranych setów (w turniejach wielkoszlemowych),
- 3 – mecz rozgrywany jest do dwóch wygranych setów (w pozostałych turniejach).

Mecze rozgrywane do pięciu setów różnią się pod względem intensywności i długości, co może wpływać na wyniki zawodników. W takich meczach większe znaczenie mają czynniki takie jak kondycja fizyczna, wytrzymałość czy doświadczenie, ponieważ zawodnicy muszą grać przez dłuższy czas. Skutkuje to również mniejszą liczbą „niespodziewanych" zwycięstw, gdyż zawodnikom o niższych miejscach rankingowych ciężko jest utrzymać wysoki poziom gry przez dłuższy czas, a wyżej notowani zawodnicy mają więcej czasu na wrócenie na swój nominalny wysoki poziom, jeżeli wystąpił u nich okres słabszej gry. Informacja o pięciosetowym meczu uwzględniona została zatem jako binarna cecha środowiskowa:

$$\{\text{is best of 5}\}. \tag{4.14}$$

4.9. Forma zawodnika

Profesjonalni zawodnicy często przechodzą swoje lepsze i gorsze okresy, mówi się wtedy, że zawodnik ma „formę" albo nie ma „formy". Często nie są one wytłumaczalne przez dostępne dla nas informacje, dlatego ważne jest, aby zidentyfikować te okresy, ponieważ dostarczają one dodatkowych informacji dla modelu, których nie może dowiedzieć się z innych cech. Objawiają się one wysoką liczbą zwycięstw lub przegranych w ostatnich meczach zawodnika. Zdecydowano zatem śledzić procent zwycięstw zawodnika w jego 10 ostatnich meczach, zarówno we wszystkich meczach, jak i na konkretnej nawierzchni, gdyż gorsza lub lepsza „forma" może być ograniczona do nawierzchni:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{last 10 games win \%}\} \times \{\text{general}, \text{surface}\}. \tag{4.15}$$

Nie trzeba dodatkowo liczyć ilości przegranych zawodnika, gdyż liczba zwycięstw jednoznacznie je wyznacza.

4.10. Historia zwycięstw i aktualna forma w turnieju

Turnieje tenisowe różnią się od siebie wieloma rzeczami, nie tylko nawierzchnią. Lokalizacja turnieju wpływa na warunki atmosferyczne, takie jak temperatura czy wilgotność, które mogą znacząco wpłynąć na styl gry i wyniki zawodników. Ponadto niektóre turnieje mają specyficzne tradycje, charakterystyki lub typ organizacji, które mogą faworyzować pewne typy zawodników. Niektórzy zawodnicy mają swoje ulubione lub wręcz przeciwnie, znienawidzone turnieje, a powody tych preferencji nie zawsze dają się wytłumaczyć na podstawie danych. Doskonałym przykładem jest niezwykła historia sukcesów Rafaela Nadala na turnieju Roland Garros, który stał się jego „domem" w świecie tenisa, dzięki niespotykanej liczbie zwycięstw na kortach ziemnych w Paryżu.

W związku z tym uwzględnienie historycznych wyników zawodnika w konkretnym turnieju jest kluczowe dla lepszego modelowania jego szans w bieżącej edycji tego turnieju. Wprowadzono zatem następujący zestaw cech, zawierających sumaryczną liczbę zwycięstw i porażek we wszystkich edycjach turnieju.

Tab. 4.2. Statystyki Rafaela Nadala w turnieju Roland Garros
Źródło: Powszechnie dostępne dane na temat kariery Rafaela Nadala.

Statystyka	**Wartość**
Liczba występów	16
Tytuły	13
Bilans wygrane-przegrane	100-2
Procent wygranych	98%
Najwięcej tytułów z rzędu	5
Najdłuższa seria zwycięstw	39 meczów

$$\{\text{Player1}, \text{Player2}\} \times \{\text{all time tournament wins}, \text{all time tournament losses}\}. \tag{4.16}$$

Oprócz długoterminowej historii wyników w turnieju istotne jest również uchwycenie bieżącej formy zawodnika w aktualnej edycji. Zawodnik, który wygrywa swoje mecze w sposób dominujący (np. z dużą liczbą wygranych gemów i setów), jest bardziej prawdopodobnym kandydatem do dalszych zwycięstw, nawet jeśli jego ogólna historia w turnieju jest przeciętna. Podobnie zawodnik, który wygrywa mecze w wyrównanych pojedynkach (np. minimalnym stosunkiem gemów), może być bardziej podatny na porażki w kolejnych rundach. Cechy stworzone do uwzględnienia tego faktu zliczają bilans setów i gemów do tej pory w turnieju:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{game difference current tournament}, \text{set difference current tournament}\}. \tag{4.17}$$

Zliczanie zwycięstw i porażek w tym przypadku byłoby oczywiście bez sensu, gdyż w bieżącej edycji każdy mecz odbywa się między zawodnikami z taką samą liczbą zwycięstw (nr rundy) i porażek (0).

4.11. Zmęczenie

Tenis jest bardzo wymagającym fizycznie sportem. Mecze często trwają ponad 1,5 godziny, a dla turniejów wielkoszlemowych regularnie potrafią trwać nawet ponad 4 godziny. Dlatego

dla prawie każdego zawodnika przygotowanie fizyczne oraz regeneracja są kluczowe. Mimo to nierzadko podczas meczu problemy fizyczne, które są w większości spowodowane zmęczeniem nagromadzonym z poprzednich meczów, mają negatywny wpływ na grę tenisisty. Aby zamodelować to zmęczenie, stworzono współczynnik zmęczenia *fatigue score*, zdefiniowany jako ważona suma minut z ostatnich siedmiu dni, w taki sposób, aby starsze mecze miały mniejszy wpływ na wynik. Konkretnie wyznaczony jest za pomocą poniższego wzoru:

$$\text{fatigue score} = \sum_{m \in \text{last\_7\_days}} 0{,}85^{(\text{days}(m)-1)} \cdot \text{duration}(m), \tag{4.18}$$

gdzie:

- last_7_days – zbiór wszystkich meczów rozegranych przez zawodnika w ciągu ostatnich 7 dni,
- days(m) – ilość dni upłyniętych od rozegrania meczu m,
- duration(m) – liczba minut spędzonych przez zawodnika na korcie w meczu m.

Wartość czynnika ważenia (0,85) oraz rozpatrywanie siedmiu poprzednich dni wybrane zostały drogą eksperymentalną. Chciano tak dobrać wartości, aby *fatigue score* rzeczywiście odzwierciedlał zmęczenie, a nie to, jak dobry jest dany zawodnik. Problem polega na tym, że mniejsza liczba minut na korcie oznacza również łatwiejsze radzenie sobie z rywalami, co za tym idzie wyższy poziom gry danego zawodnika. Gdy spotka się ze sobą dwóch zawodników, z czego jeden spędził w ostatnich trzech dniach dwa razy mniej minut na korcie niż jego przeciwnik, to jego potencjalne zwycięstwo będzie w małym stopniu spowodowane tym, że był mniej zmęczony, a raczej tym, że jest ogólnie dużo lepszy. Aby ocenić jakość współczynnika zmęczenia, analizowano zatem mecze pomiędzy zawodnikami o zbliżonych rankingach, czyli takie, w których wynik nie jest w znaczącym stopniu determinowany różnicą w rankingu ATP lub Elo obu zawodników. Dla wybranych wartości czynnika ważenia oraz liczby dni wstecz, w 54,7% przypadków, gdy różnica miejsc w rankingu ATP wynosiła mniej niż pięć, zwycięstwo odnosił zawodnik o niższym współczynniku zmęczenia. Wynik ten wskazuje na predykcyjną wartość dodaną tej cechy. Dodano więc współczynnik zmęczenia do zbioru cech:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{fatigue score}\}. \tag{4.19}$$

4.12. Kontuzje

Trudy meczów tenisowych oprócz zmęczenia nierzadko powodują u grającego tenisisty kontuzje. Konsekwencją tego jest krótszy lub dłuższy okres, w którym zawodnik wraca do zdrowia i nie rozgrywa meczów w turniejach. Pierwszy mecz po powrocie z takiej wymuszonej przerwy jest często bardzo niepewny, długi czas bez grania w turniejach powoduje u zawodnika brak ograna na najwyższym poziomie, które jest kluczowe w rozgrywkach ATP. Możliwe również, że kontuzja nie została w pełni wyleczona i znowu zacznie się nasilać, powodując znacznie gorszą jakość gry tenisisty, albo nawet wczesne zejście z kortu. Dodano zatem binarną cechę informującą, czy dany mecz jest pierwszym po kontuzji:

$$\{\text{Player1}, \text{Player2}\} \times \{\text{is first match after injury}\}. \tag{4.20}$$

Przyjęto, iż zawodnik doznał kontuzji, jeżeli poddał on mecz („Retired" w kolumnie „Comment"). W rzeczywistości zawodnik mógł się skontuzjować poza meczem albo dograć mecz z bólem, czy też poddać mecz z innego powodu, jednak nie było lepszego sposobu na zidentyfikowanie kontuzji z dostępnych danych. Procent zwycięstw dla zawodników grających pierwszy mecz po kontuzji wynosi 48,8%, co potwierdza zwiększone (chociaż nie dużo) szanse na porażkę w pierwszym meczu po powrocie z kontuzji.

4.13. Agregacja statystyk meczowych

Statystyki meczowe dostępne w źródłowych ramkach danych to:

- **w_ace** – liczba asów zwycięzcy,
- **w_df** – liczba podwójnych błędów zwycięzcy,
- **w_svpt** – liczba serwisów zwycięzcy,
- **w_1stIn** – liczba trafionych pierwszych serwisów zwycięzcy,
- **w_1stWon** – liczba wygranych punktów po pierwszym serwisie zwycięzcy,
- **w_2ndWon** – liczba wygranych punktów po drugim serwisie zwycięzcy,
- **w_SvGms** – liczba gemów serwisowych zwycięzcy,

- **w_bpSaved** – liczba obronionych breakpointów przez zwycięzcę,
- **w_bpFaced** – liczba breakpointów, których bronił zwycięzca,

Analogiczne kolumny istnieją dla przegranego.

Na podstawie tych statystyk obliczono następujące nowe wartości:

- $\mathbf{w\_2ndIn} = \mathbf{w\_svpt} - \mathbf{w\_1stIn} - \mathbf{w\_df}$
  Liczba trafionych drugich serwisów zwycięzcy.
- $\mathbf{winner\_1st\_serve\_in\_pct} = \frac{\mathbf{w\_1stIn}}{\mathbf{w\_svpt}}$
  Procent trafionych pierwszych serwisów zwycięzcy.
- $\mathbf{winner\_1st\_serve\_win\_pct} = \frac{\mathbf{w\_1stWon}}{\mathbf{w\_1stIn}}$
  Procent wygranych punktów po trafionym pierwszym serwisie zwycięzcy.
- $\mathbf{winner\_2nd\_serve\_in\_pct} = \frac{\mathbf{w\_2ndIn}}{\mathbf{w\_svpt} - \mathbf{w\_1stIn}}$
  Procent trafionych drugich serwisów zwycięzcy.
- $\mathbf{winner\_2nd\_serve\_win\_pct} = \frac{\mathbf{w\_2ndWon}}{\mathbf{w\_svpt} - \mathbf{w\_1stIn}}$
  Procent wygranych punktów po trafionym drugim serwisie zwycięzcy.
- $\mathbf{winner\_service\_games\_won\_pct} = \frac{\mathbf{w\_SvGms} - (\mathbf{w\_bpFaced} - \mathbf{w\_bpSaved})}{\mathbf{w\_SvGms}}$
  Procent wygranych gemów serwisowych zwycięzcy.
- $\mathbf{winner\_1st\_serve\_return\_win\_pct} = 1 - \mathbf{loser\_1st\_serve\_win\_pct}$
  Procent wygranych punktów z returnu zwycięzcy po pierwszym serwisie przeciwnika.
- $\mathbf{winner\_2nd\_serve\_return\_win\_pct} = 1 - \mathbf{loser\_2nd\_serve\_win\_pct}$
  Procent wygranych punktów z returnu zwycięzcy po drugim serwisie przeciwnika.
- $\mathbf{winner\_return\_games\_win\_pct} = 1 - \mathbf{loser\_service\_games\_won\_pct}$
  Procent wygranych gemów przy returnie zwycięzcy.
- $\mathbf{winner\_bp\_won\_pct} = \frac{\mathbf{l\_bpFaced} - \mathbf{l\_bpSaved}}{\mathbf{l\_bpFaced}}$
  Procent wygranych breakpointów przez zwycięzcę przy serwisie przeciwnika.
- $\mathbf{winner\_bp\_saved\_pct} = \frac{\mathbf{w\_bpSaved}}{\mathbf{w\_bpFaced}}$
  Procent wygranych (obronionych) breakpointów przez zwycięzcę przy jego serwisie.

Aby zapewnić, że każdy mecz (rząd ramki) zawiera jedynie dane dostępne przed jego rozpoczęciem, konieczne jest zamienienie statystyk meczowych na wartości zagregowane na stan sprzed meczu. Najprostszym sposobem byłoby policzenie zwykłej średniej arytmetycznej z wartości danej statystyki dla danego zawodnika we wszystkich jego poprzednich meczach. Czyli:

$$\text{winner\_stat\_avg} = \frac{1}{n}\sum_{i=1}^{n}\text{winner\_stat}_i,$$

gdzie:

- winner_stat_i – wartość statystki dla i-tego poprzedniego meczu.

Takie podejście ma jednak kluczowy problem, nie bierze ono pod uwagę, jak relewantny dany przeszły mecz jest do meczu, dla którego wykonywana jest agregacja, innymi słowy, wszystkie przeszłe mecze traktowane są z taką samą wagą.

Wygaszanie czasowe

Berthelot [2] stwierdził, iż zdolności ludzkie, m.in. lekkoatletów, zmieniają się w czasie, podążając wykładniczym wzrostem lub spadkiem. Nie jesteśmy w stanie jasno określić oraz zamodelować wszystkich czynników, które wpływają na zmianę jakości gry zawodnika w czasie, ale możemy założyć, że mecz, który odbył się tydzień temu, w większym stopniu odzwierciedla aktualny stan gry zawodnika niż mecz rozegrany 10 lat temu. Wprowadzone zostało zatem czasowe wygaszanie wag meczów, czyli im dawniej rozegrany został mecz, tym mniejszą wagę będzie on miał przy wyliczaniu średniej. Do określenia konkretnych wag użyto funkcji wykładniczego zaniku:

$$W_{\text{time}}(t) = e^{-\alpha t} \tag{4.21}$$

gdzie:

- t – liczba lat upłyniętych od rozegrania meczu.

Parametr λ jest hiperparametrem modelu, ale ze względu na długi czas liczenia agregacji dla statystyk meczowych (ok. 45 min) uwzględnienie go podczas optymalizacji wydłużyłoby jej czas, na więcej niż można było sobie pozwolić. Wybrana została zatem wartość $\lambda = 0{,}3$, która według autorów miała najwięcej sensu, analizując wartości $W(t)$ dla różnych wartości t.

Nawierzchnia

Jak ustalono w podrozdz. 4.3, nawierzchnia jest ważnym elementem każdego meczu, dlatego przy agregacji również powinniśmy brać ją pod uwagę. Zdecydowano zatem ważyć przeszłe mecze

według tego, jak „podobne” są dwie nawierzchnie. Oczywiście dla tych samych nawierzchni waga będzie wynosić 1, natomiast dla określenia wagi dla różnych nawierzchni postanowiono skorzystać z Dywergencji Kullbacka–Leiblera (zwanej też entropią względną lub relatywną entropią), w celu określenia rozbieżności między rozkładem danej statystyki na nawierzchni `A`, a rozkładem tej samej statystyki na nawierzchni `B`:

$$D_{KL}(P \parallel Q) = \sum_{x \in X} P(x) \log \left(\frac{P(x)}{Q(x)} \right), \tag{4.22}$$

gdzie:

- X – zbiór przedziałów (binów) wartości statystyki,
- $P(x) = \frac{n(x)}{\sum_{x \in X} n(x)}$ – prawdopodobieństwo przypisane do przedziału x w rozkładzie P (dla nawierzchni s_1),
- $Q(x) = \frac{m(x)}{\sum_{x \in X} m(x)}$ – prawdopodobieństwo przypisane do przedziału x w rozkładzie Q (dla nawierzchni s_2),
- $n(x)$, $m(x)$ – liczba obserwacji przypadająca na przedział x odpowiednio w rozkładzie P i Q.

Aby uniknąć zerowych prawdopodobieństw, do każdego $P(x)$ i $Q(x)$ dodano małą stałą $\epsilon = 10^{-9}$. Po uwzględnieniu tej stałej rozkłady są ponownie normalizowane, aby spełniały warunek:

$$\sum_{x \in X} P(x) = 1 \quad \text{oraz} \quad \sum_{x \in X} Q(x) = 1.$$

Wagi podobieństwa między nawierzchniami są obliczane jako:

$$W_{\text{surface}}(s_1, s_2) = \frac{1}{1 + D_{KL}(P \parallel Q)}. \tag{4.23}$$

W rezultacie dla każdej statystyki otrzymano macierz wag, tabela 4.3 przedstawia taką macierz dla statystyki asów.

Wspólni Przeciwnicy

Turnieje tenisowe rozgrywane są w systemie pucharowym, czyli uczestnicy rozmieszczani są w drabince, gdzie każdy mecz jest eliminacyjny. Co za tym idzie, każdy zawodnik w kolejnych meczach trafia na zawodnika, który też wygrał wszystkie swoje poprzednie mecze, czyli na teoretycznie cięższych zawodników. Powoduje to błąd systematyczny przy agregacji statystyk, gdyż słabsi zawodnicy, którzy szybciej odpadają z turniejów, będą średnio rzadziej grać z mocnymi przeciwnikami niż zawodnicy, którzy regularnie daleko zachodzą w turniejach. W celu

Tab. 4.3. Wagi podobieństwa między różnymi nawierzchniami na podstawie dywergencji KL dla statystki asów.

	Hard	**Clay**	**Grass**
Hard	1,000000	0,833489	0,964563
Clay	0,851985	1,000000	0,734754
Grass	0,958417	0,703039	1,000000

zapobiegnięcia temu błędowi Knottenbelt [7] zastosował metodę Wspólnych Przeciwników (Common Opponents), zwizualizowaną na rysunku 4.4, według której agregacja statystyk liczona jest jedynie na podstawie meczów przeciwko wspólnym przeciwnikom danych dwóch zawodników. Konkretnie:

$$\text{PlayerA\_CO\_Stat} = \frac{1}{n}\sum_{i=1}^{n}\text{PlayerA\_Stat}(C_i), \tag{4.24}$$

gdzie:

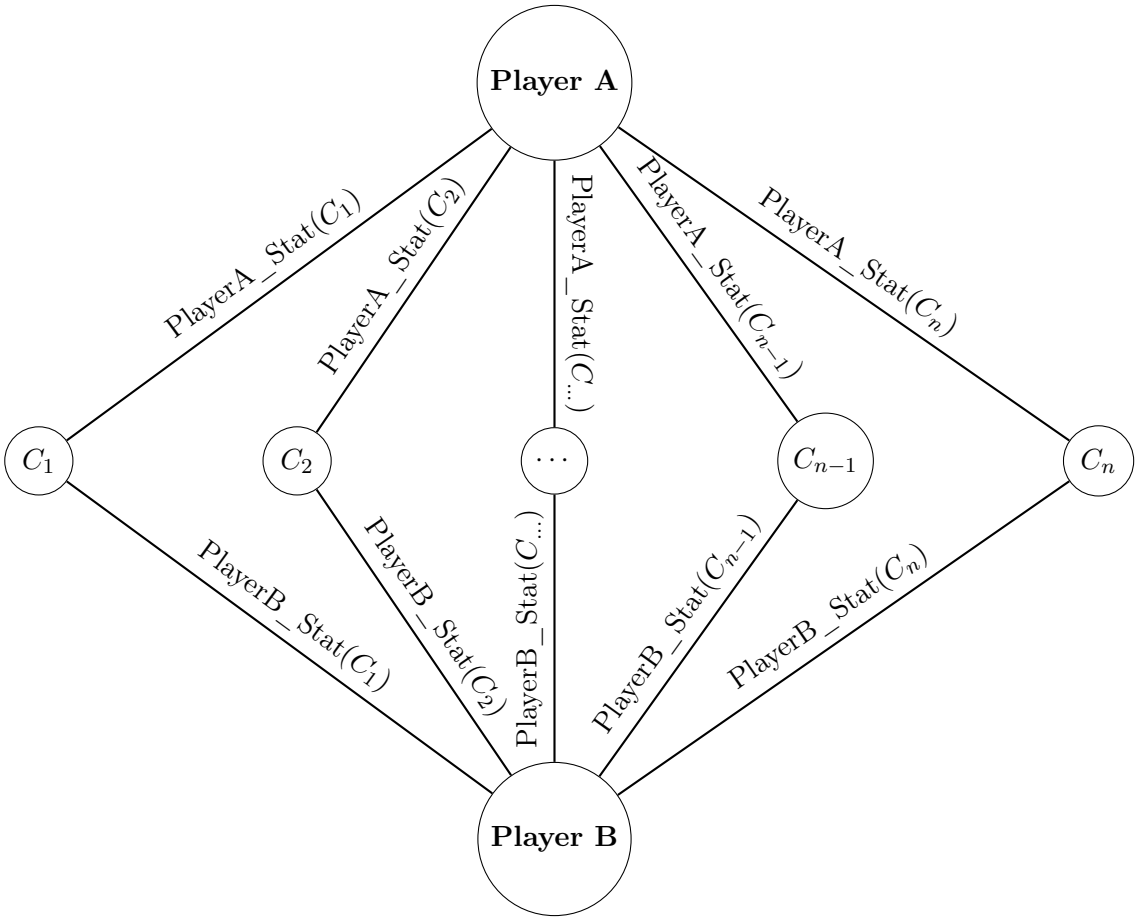
- n – liczba wspólnych przeciwników zawodnika A i zawodnika B,
- C_i – i-ty wspólny przeciwnik zawodnika A i zawodnika B,
- $\text{PlayerA\_Stat}(C_i) = \frac{1}{m}\sum_{j=1}^{m}\text{PlayerA\_Stat}_j(C_i)$,
- m – liczba meczów zawodnika A z wspólnym przeciwnikiem C_i,
- $\text{PlayerA\_Stat}_j(C_i)$ – wartość statystyki zawodnika A w j-tym meczu przeciwko zawodnikowi C_i.

Podsumowując, w ramach agregacji statystyk stworzono następujące cechy:

$$\{\text{Player1}, \text{Player2}\} \times \{(\text{stat avg}, \text{CO stat avg}) \text{ for all stats}\}, \tag{4.25}$$

gdzie:

- stat avg – zagregowana statystyka z użyciem wygaszania czasowego i ważenia ze względu na nawierzchnię,
- CO stat avg – zagregowana statystyka z użyciem wygaszania czasowego, ważenia ze względu na nawierzchnię oraz metody Wspólnych Przeciwników.



Rys. 4.4. Wizualizacja metody Wspólnych Przeciwników do agregacji staystyki

Zagregowane statystyki, które nie korzystają z metody Wspólnych Przeciwników, również uwzględniono w finalnym zbiorze cech. Wynika to z faktu, że w wielu przypadkach zbiór wspólnych przeciwników zawiera bardzo mało elementów, co ogranicza jego przydatność. W takich sytuacjach te statystyki mogą okazać się wartościowe dla modelu.

4.14. Niepewność

Przedstawiony sposób obliczania średnich wartości statystyk prowadzi do pewnych problemów związanych z niepewnością wyników. Jednym z głównych wyzwań jest sytuacja, w której liczba wspólnych przeciwników jest niewielka. W takich przypadkach analiza opiera się na małej próbie meczów, co znacząco zwiększa ryzyko błędów i sprawia, że wyniki mogą być mało reprezentatywne. Z tego powodu, oprócz metody Wspólnych Przeciwników, postanowiono uwzględnić podejście, które bierze pod uwagę wszystkie wcześniejsze mecze zawodników.

Dodatkowym problemem jest brak pełnej historii danych meczowych sprzed 2018 roku. W efekcie, dla spotkań z początku analizowanego okresu (np. z początku 2018 roku), obliczenia mogą znacząco odbiegać od rzeczywistości. Gdyby dostępne były pełne dane historyczne, uzyskane wartości byłyby dokładniejsze i bardziej wiarygodne. Taka niekompletność danych prowadzi do potencjalnych zniekształceń wniosków wyciąganych na podstawie analizowanych statystyk.

Aby zminimalizować te ograniczenia, wprowadzono współczynnik niepewności dla wyliczanych wartości, który informuje, jak „pewne" są one w skali od 0 do 1. Do konstrukcji tego współczynnika wykorzystano wagi przypisywane meczom, uwzględniające wygaszanie czasowe oraz ważenie względem nawierzchni. Im większa suma wag, tym bardziej wiarygodne są uzyskane wyniki, co przekłada się na wyższą pewność końcowych oszacowań. Opisany sposób obliczania niepewności oparty jest na rozwiązaniu zaproponowanym przez Sipko [10].

Niepewność dla prostej agregacji

Dla wyznaczenia niepewności najpierw liczymy sumę wag z meczów uwzględnianych w agregacji:

$$S = \sum_{j=1}^{n} W_{\text{surface}}(s_j, s_{\text{current}}) + W_{\text{time}}(t_j),$$

gdzie:

- s_j – nawierzchnia w j-tym meczu,
- $s_{current}$ – nawierzchnia w meczu, dla którego liczymy agregację,
- t_j – czas upłynięty od j-tego meczu w latach.

a następnie wyznaczamy współczynnik niepewności jako:

$$\text{Uncertainty} = \frac{1}{S_1 * S_2}, \tag{4.26}$$

gdzie:

- S_1, S_2 – sumy wag S odpowiednio dla pierwszego i drugiego zawodnika rozpatrywanego meczu.

Niepewność dla metody Wspólnych Przeciwników

Aby obliczyć niepewność dla metody Wspólnych Przeciwników, można by przypuszczać, że wystarczy ograniczyć zbiór meczów do tych rozgrywanych jedynie ze wspólnymi przeciwnikami

i zastosować ten sam wzór co w przypadku „prostej agregacji". Takie rozwiązanie jednak nie uwzględnia niepewności na poziomie pojedynczego wspólnego przeciwnika.

Rozważmy dwa przypadki z dwoma graczami P_1 i P_2 oraz dwoma wspólnymi przeciwnikami C_1 i C_2. Dla każdego przypadku zliczane są sumy wag dla meczów graczy P_1 i P_2 względem przeciwników.

Przypadek 1: Równe wagi dla obu graczy

$$\text{Dla przeciwnika } C_1 : S_1(C_1) = 5, \quad S_2(C_1) = 5,$$
$$\text{Dla przeciwnika } C_2 : S_1(C_2) = 5, \quad S_2(C_2) = 5.$$

Przypadek 2: Nierówne wagi dla graczy

$$\text{Dla przeciwnika } C_1 : S_1(C_1) = 1, \quad S_2(C_1) = 9,$$
$$\text{Dla przeciwnika } C_2 : S_1(C_2) = 9, \quad S_2(C_2) = 1.$$

gdzie:

- $S_i(C_j)$ – suma wag dla meczów zawodnika i przeciwko wspólnemu przeciwnikowi j.

Chcemy, żeby przypadek pierwszy miał mniejszą niepewność, ponieważ dane są bardziej spójne i nie ma dużych dysproporcji w danych historycznych, co oznacza, że agregacja wyników jest bardziej reprezentatywna dla obu graczy. Obliczmy teraz niepewność dla obu podejść.

Podejście 1: Mnożenie na poziomie przeciwnika – niepewność (*Uncertainty*) obliczana jest jako:

$$\text{Uncertainty} = \frac{1}{\sum_j S_1(C_j) \cdot S_2(C_j)}. \tag{4.27}$$

Niepewność dla przypadku 1:

$$\text{Uncertainty} = \frac{1}{S_1(C_1) \cdot S_2(C_1) + S_1(C_2) \cdot S_2(C_2)} = \frac{1}{5 \cdot 5 + 5 \cdot 5} = \frac{1}{25 + 25} = \frac{1}{50}.$$

Niepewność dla przypadku 2:

$$\text{Uncertainty} = \frac{1}{S_1(C_1) \cdot S_2(C_1) + S_1(C_2) \cdot S_2(C_2)} = \frac{1}{1 \cdot 9 + 9 \cdot 1} = \frac{1}{9 + 9} = \frac{1}{18}.$$

Podejście 2: Sumowanie wag przed mnożeniem – Niepewność obliczana jest jako:

$$\text{Uncertainty} = \frac{1}{\left(\sum_j S_1(C_j)\right) \cdot \left(\sum_j S_2(C_j)\right)}. \tag{4.28}$$

Niepewność dla przypadku 1:

$$\text{Uncertainty} = \frac{1}{(S_1(C_1) + S_1(C_2)) \cdot (S_2(C_1) + S_2(C_2))} = \frac{1}{(5+5) \cdot (5+5)} = \frac{1}{10 \cdot 10} = \frac{1}{100}.$$

Niepewność dla przypadku 2:

$$\text{Uncertainty} = \frac{1}{(S_1(C_1) + S_1(C_2)) \cdot (S_2(C_1) + S_2(C_2))} = \frac{1}{(1+9) \cdot (9+1)} = \frac{1}{10 \cdot 10} = \frac{1}{100}.$$

Na podstawie tego przykładu widać, że podejście 1 (mnożenie na poziomie przeciwnika) daje różne wyniki dla obu przypadków ($\frac{1}{50}$ i $\frac{1}{18}$), odzwierciedlając proporcje wag, natomiast podejście 2 (sumowanie wag przed mnożeniem) ignoruje te różnice, dając identyczne wyniki ($\frac{1}{100}$) w obu przypadkach. Ponieważ korzystając z podejścia 1, niepewność dla przypadku 2 jest większa, zdecydowano wybrać podejście 1.

Podsumowując, dodano następujące cechy reprezentujące niepewność:

$$\{\text{CO uncertainty}, \text{uncertainty}\} \tag{4.29}$$

4.15. Pogoda

Zarówno De Seranno [3] jak i Sipko [10] zasugerowali, że uwzględnienie danych pogodowych może pozytywnie wpłynąć na jakość przewidywań modelu. Zdecydowano zatem wziąć pod uwagę następujące parametry:

- **Temperatura** – Wartość w stopniach Celsjusza w momencie rozpoczęcia meczu. Wyższe temperatury mogą prowadzić do szybszego zmęczenia zawodników i zmieniać dynamikę gry.
- **Wilgotność powietrza** – Wartość procentowa określająca poziom wilgotności w trakcie meczu. Wyższa wilgotność może wpływać na komfort zawodników oraz ich zdolność do utrzymania intensywności gry.
- **Wiatr** – Prędkość wiatru mierzona w km/h. Silny wiatr może zakłócać serwisy i uderzenia piłki, zmuszając zawodników do dostosowania stylu gry.

Oryginalne dwa źródła danych nie zawierają żadnych informacji o pogodzie. Pozyskanie ich wymaga posiadania dla każdego meczu koordynat lokalizacji turnieju oraz godziny rozgrywania

meczu, obydwu z tych rzeczy nie ma w zbiorze danych 1 i 2, zatem konieczne było ich dodanie lub wytworzenie.

Koordynaty lokalizacji turnieju:
Za pomocą biblioteki **geopy** i kolumny `tournament_location` stworzone zostały kolumny `latitude` oraz `longitude`.

Godzina rozgrywania meczu:
Ponieważ nie został znaleziony żaden publicznie dostępny zbiór danych, który zawierałby tę informację, zdecydowano się na zdobycie jej poprzez skrobanie strony https://www.tennisexplorer.com, zawierającej historię wszystkich interesujących nas meczów tenisowych wraz z godziną ich rozegrania. Co ważne, godzina rozegrania widniejąca na stronie jest UTC+1. Napisany został skrypt w języku `Python`, który pobiera wyżej wymienioną stronę i zwraca plik w formacie JSON. Następnie informacje o godzinie rozgrywania meczów przepisane zostały do głównej ramki poprzez podobne działania, co przy łączeniu dwóch oryginalnych zbiorów danych w jeden, czyli ujednolicenie nazw zawodników i turniejów, a następnie wytworzenie `match_id` odpowiadające temu w głównej ramce danych.

Posiadając już wszystkie konieczne dane, wykonano serię zapytań do API Open-Meteo, aby otrzymać wartości temperatury, wilgotności, prędkość wiatru oraz odczuwalną temperaturę. Finalnie do zbioru cech dołączono następujące cechy środowiskowe:

$$\{\text{temperature}, \text{apparent temperature}, \text{humidity}, \text{windspeed}\} \tag{4.30}$$

4.16. Podsumowanie

Podsumowując, opracowano 101 cech, które będą stanowiły wektor wejściowy dla modeli. Dokładne przedstawienie utworzonych cech prezentuje tabela 4.4.

Tab. 4.4. Podsumowanie stworzonych cech

Kategoria	**Typ**	**Liczba stworzonych cech**	**Reprezentacja**
Wiek	Zawodnik	2	4.1
Dominująca ręka	Zawodnik	2	4.2
Nawierzchnia	Środowisko	3	4.3
Sposób uczestnictwa	Zawodnik	6	4.4
Rozstawienie	Zawodnik	2	4.5
Ranking	Zawodnik	8	4.11, 4.12
H2H	Zawodnik	4	4.13
Do x wygranych	Środowisko	1	4.14
Forma	Zawodnik	4	4.15
Wyniki w turnieju	Zawodnik	8	4.16, 4.17
Zmęczenie	Zawodnik	2	4.19
Kontuzje	Zawodnik	2	4.20
Agregacja statystyk	Zawodnik	52	4.25
Niepewność	Analityczna	2	4.29
Pogoda	Środowisko	3	4.30

5. Modele uczenia

W tym rozdziale przedstawiono proces projektowania, trenowania i ewaluacji modeli predykcyjnych dla wyników meczów tenisowych. Omówiono m.in. założenia teoretyczne, przygotowanie danych, metody walidacji oraz proces uczenia. Szczególną uwagę poświęcono specyfice danych w postaci szeregów czasowych oraz ich podziałowi na zbiory treningowe, walidacyjne i testowe.

5.1. Założenia modelu

Projektując model predykcyjny dla wyników meczów tenisowych, przyjęto następujące założenia:

1. **Przewidywanie w formie prawdopodobieństwa:** Model przewiduje wartość zmiennej objaśnianej, zwracając prawdopodobieństwo z przedziału od 0 do 1, które reprezentuje oszacowaną szansę na zwycięstwo zawodnika `Player1`. Wartość bliższa 1 oznacza większe prawdopodobieństwo wygranej `Player1`, natomiast wartość bliższa 0 wskazuje na większą szansę na zwycięstwo `Player2`.

   Zmienna objaśniana została zdefiniowana jako zmienna binarna w następujący sposób:

   $$\text{Target} = \begin{cases} 1, & \text{gdy } \texttt{Player1} \text{ wygra}, \\ 0, & \text{gdy } \texttt{Player1} \text{ przegra}. \end{cases} \tag{5.1}$$

2. **Wykorzystanie danych dostępnych przed meczem:** Model opiera swoje prognozy wyłącznie na danych, które są dostępne przed rozpoczęciem meczu. Takie podejście eliminuje ryzyko korzystania z informacji związanych z przebiegiem meczu, zapewniając realistyczne i praktyczne zastosowanie modelu.

3. **Symetryczność modelu:** Model jest symetryczny względem zawodników, co oznacza, że zamiana miejscami danych wejściowych dla `Player1` i `Player2` zmienia wynik predykcji w sposób komplementarny. Formalnie, dla reprezentacji danych D, gdzie zamiana zawodników jest oznaczona jako D', symetryczność można wyrazić wzorem:

   $$P(\texttt{Player1 wins} \mid D) = 1 - P(\texttt{Player1 wins} \mid D'), \tag{5.2}$$

gdzie:

- D – dane wejściowe z zawodnikiem `Player1` jako graczem pierwszym,
- D' – dane wejściowe z zamienionymi miejscami zawodników (`Player2` jako gracz pierwszy),
- $P(\texttt{Player1 wins} \mid D)$ – prawdopodobieństwo zwycięstwa `Player1` przy danych D.

Przyjęte założenia zapewniają spójność i użyteczność modelu, umożliwiając jego zastosowanie w różnych scenariuszach, zarówno w analizie danych historycznych, jak i w rzeczywistych prognozach przedmeczowych.

5.2. Przygotowanie danych

Aby spełnić założenia modelu i zadbać o dostarczenie danych w odpowiedniej formie, wykonano następujące kroki przetwarzania na wektorze cech:

1. **Standaryzacja nazw cech zawodników:** Cechy związane z zawodnikami były oznaczane jako `winner_...` dla zwycięzcy meczu oraz `loser_...` dla przegranego. Aby ujednolicić reprezentację i odzwierciedlić brak znajomości zwycięzcy przed rozegraniem spotkania, nazwy te zostały zamienione na `player1_...` oraz `player2_...`.

2. **Dodanie zmiennej target:** Zgodnie z założeniami modelu dodano zmienną target, która na początku dla każdego wiersza ma wartość 1, gdyż `player_1` jest zwycięzcą, ze względu na zamianę nazw w poprzednim kroku.

3. **Losowa zamiana miejsc zawodników:** Aby uniknąć stronniczości modelu wynikającej z przypisania zwycięzcy zawsze jako `player_1`, w losowo wybranej połowie wierszy zamieniono miejscami dane dotyczące `player_1` i `player_2` oraz target z 1 na 0, ponieważ teraz `player_2` stał się w nich wygranym. Dzięki temu w połowie wierszy `player_1` jest zwycięzcą meczu, a w drugiej połowie `player_2` jest zwycięzcą.

5.3. Metody walidacji

5.3.1. Walidacja krzyżowa

Walidacja krzyżowa (ang. cross-validation) polega na:

1. podzieleniu zbioru danych na k równych, rozłącznych podzbiorów,
2. wytrenowaniu modelu k-krotnie, każdorazowo wykorzystując $k-1$ podzbiorów jako dane treningowe i 1 podzbiór jako dane walidacyjne,
3. obliczeniu średniej wartości metryki ewaluacyjnej dla wszystkich iteracji stanowiącej ostateczną ocenę modelu.

Walidacja krzyżowa, choć jest powszechnie stosowaną metodą oceny modeli, ma istotne ograniczenia w przypadku danych w postaci szeregów czasowych. Wynika to z charakterystycznej cechy tych danych, gdzie obserwacje są zależne od siebie w czasie, a dane późniejsze mogą zawierać informacje o wcześniejszych.

W klasycznej walidacji krzyżowej dane są dzielone na podzbiory w sposób losowy. Taki podział prowadzi do sytuacji, w której dane treningowe mogą pochodzić z późniejszych okresów niż dane walidacyjne. Narusza to kluczowe założenie analizy szeregów czasowych, w którym model powinien być trenowany wyłącznie na danych z przeszłości i testowany na danych z przyszłości.

5.3.2. Podział zbioru ze względu na lata

Aby uniknąć problemów związanych z klasyczną walidacją krzyżową w przypadku danych szeregów czasowych, zastosowano podejście oparte na podziale danych zgodnie z porządkiem czasowym. W tym podejściu dane zostały podzielone na trzy rozłączne zbiory:

- **Zbiór treningowy (2018–2021):** Obejmuje dane z meczów rozgrywanych w latach 2018–2021. Służy do trenowania modelu.
- **Zbiór walidacyjny (2022):** Obejmuje dane z meczów rozgrywanych w roku 2022. Służy do oceny skuteczności modelu na danych, które nie były uwzględnione podczas procesu trenowania.
- **Zbiór testowy (2023):** Obejmuje dane z meczów rozgrywanych w roku 2023. Służy do końcowej ewaluacji modelu na nowych, nieznanych danych, co pozwala oszacować jego rzeczywistą zdolność do generalizacji.

Takie podejście zapewnia, że model trenuje się wyłącznie na danych z przeszłości, a jego skuteczność jest testowana na danych z przyszłości. Dzięki temu zachowany jest naturalny porządek czasowy danych, co eliminuje ryzyko „przecieku informacji" między zbiorami treningowym, walidacyjnym i testowym. Dodatkowo, taki podział pozwala lepiej odwzorować rzeczywiste warunki, w których model będzie wykorzystywany, gdzie prognozy są dokonywane na podstawie danych historycznych.

Mimo licznych zalet to podejście ma również swoje ograniczenia. Jednym z głównych problemów jest ryzyko nadmiernego dopasowania modelu do zbioru walidacyjnego. W sytuacji, gdy model walidowany jest tylko na jednym zbiorze, może dojść do sytuacji, w której model jest „przystosowany" do dobrego działania na nim, lecz jego zdolność do uogólnienia spada na zbiorze testowym. Takie zachowanie było również zauważone w przeprowadzonych eksperymentach, gdzie wyniki na zbiorze walidacyjnym były znacząco lepsze niż na zbiorze testowym.

Taki problem jest w pewnym stopniu rozwiązywany przez klasyczną walidację krzyżową, która dzięki wielokrotnemu podziałowi i testowaniu na różnych podzbiorach danych pozwala na bardziej kompleksową ocenę zdolności modelu do generalizacji.

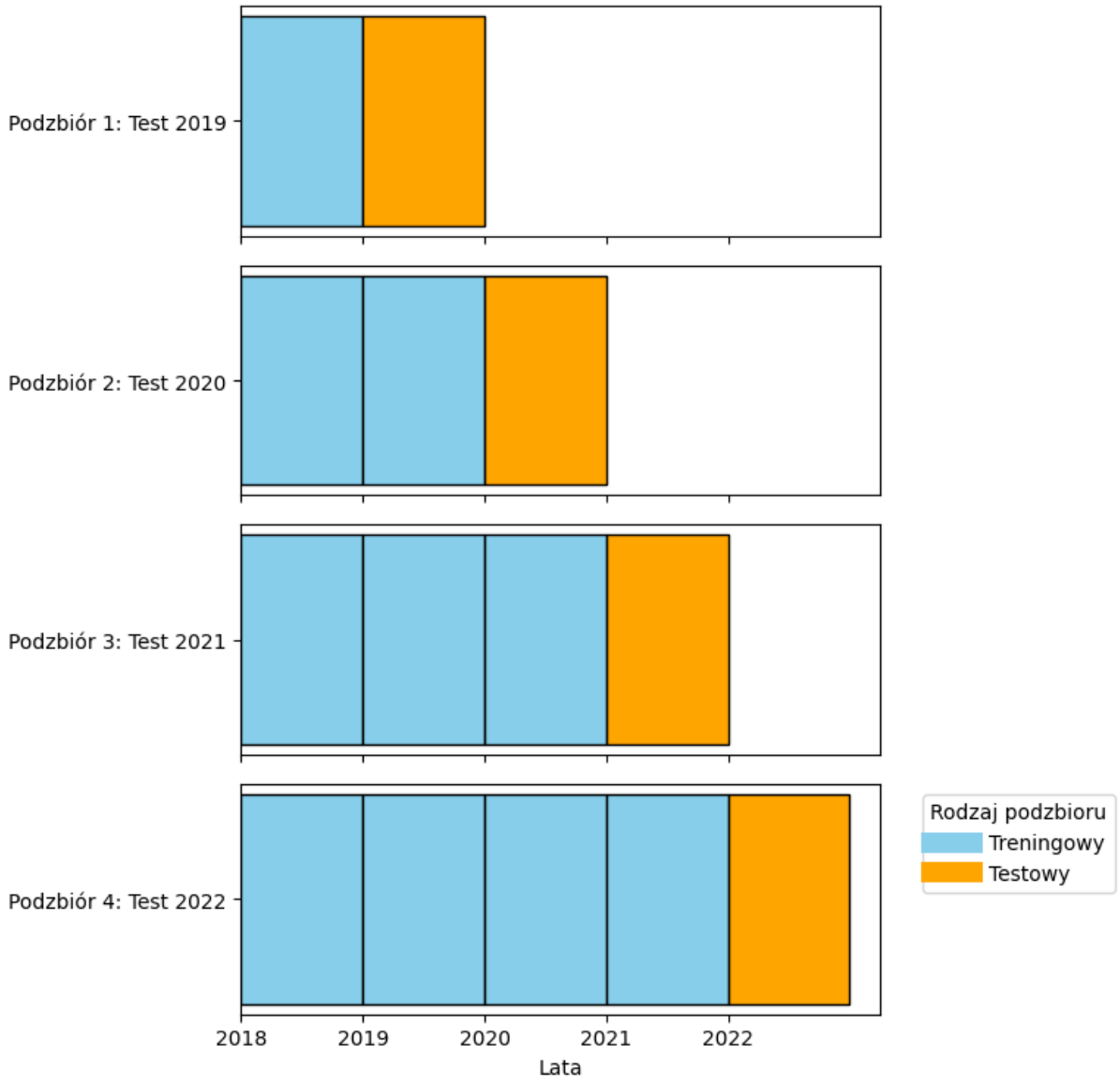
5.3.3. Zagnieżdżona walidacja krzyżowa

Zagnieżdżona walidacja krzyżowa (ang. nested cross-validation) stanowi połączenie klasycznej walidacji krzyżowej oraz podziału danych na lata w celu zachowania porządku czasowego i uniknięcia problemu przecieku informacji między zbiorami treningowym a walidacyjnym. W tym podejściu dane są iteracyjnie dzielone na zbiory treningowe i walidacyjne w sposób progresywny. W każdej kolejnej iteracji zbiór treningowy jest stopniowo rozszerzany o dane z kolejnych lat, a zbiór walidacyjny obejmuje dane z bezpośrednio następującego roku. Rozwiązanie to jest stosowane, aby uniknąć wycieku danych, które prowadziłyby do zbyt optymistycznego oszacowania błędu [3] oraz aby rozwiązać problem zbytniego dopasowywania się do zbioru walidacyjnego, gdy jest nim tylko rok 2022. W przypadku przedstawianego rozwiązania zbiór danych został podzielony tak, aby każdy podzbiór odpowiadał meczom rozegranym w danym roku kalendarzowym. Na rysunku 5.1 przedstawiona jest wizualizacja strategii dla przedstawianego rozwiązania.

5.4. Selekcja cech z wykorzystaniem wartości SHAP

Selekcja cech (ang. feature selection) jest kluczowym etapem w procesie analizy danych, który pozwala na eliminację mniej istotnych zmiennych w celu poprawy interpretowalności modelu oraz jego efektywności obliczeniowej. W przedstawianym rozwiązaniu do oceny ważności cech w przypadku modeli regresji logistycznej oraz XGBoost zostały wykorzystane wartości SHAP (Shapley Additive Explanations).

Wartości SHAP bazują na koncepcji wartości Shapleya z teorii gier kooperacyjnych, które mierzą wpływ każdej zmiennej na wynik modelu. SHAP dostarcza interpretowalnych i spójnych wartości istotności cech poprzez analizę wkładu danej cechy w przewidywanie modelu. Proces selekcji cech z wykorzystaniem wartości SHAP składa się z następujących kroków:



Rys. 5.1. Wizualizacja zagnieżdżonej walidacji krzyżowej na zbiorach danych

1. **trenowanie modelu**: model uczony jest na pełnym zestawie cech,
2. **obliczenie wartości SHAP**: dla każdej cechy obliczane są wartości SHAP, które odzwierciedlają jej wpływ na predykcję,
3. **ranking cech**: cechy są sortowane według średniej wartości bezwzględnej SHAP,
4. **eliminacja najmniej istotnych cech**: cechy o najniższej wartości SHAP mogą być usunięte z modelu,
5. **ponowna ewaluacja**: model jest ponownie trenowany i oceniany na zmniejszonym zestawie cech w celu sprawdzenia wpływu redukcji wymiarów na jakość predykcji.

W przypadku przedstawianego rozwiązania, podczas trenowania modeli, ustalono określone progi

procentowe dotyczące liczby wybieranych cech. Przyjęte wartości to: 50%, 65%, 75%, 85% i 100% wszystkich dostępnych cech.

5.5. Optymalizacja bayesowska

Optymalizacja bayesowska polega na budowaniu probabilistycznego modelu funkcji celu i iteracyjnym wybieraniu punktów w przestrzeni hiperparametrów, które maksymalizują oczekiwaną poprawę modelu. Dzięki temu proces poszukiwania optymalnych wartości hiperparametrów staje się bardziej efektywny, ponieważ w przeciwieństwie do metod losowych skupia się na najbardziej obiecujących obszarach przestrzeni parametrów [11].

5.6. Zbiory testowe

Model oceniono na trzech różnych podzbiorach zbioru testowego:

1. **Zbiór 1** – cały zbiór testowy,
2. **Zbiór 2** – podzbiór zawierający wyłącznie wiersze z zawodnikami znajdującymi się w rankingu na lepszym miejscu niż pięćdziesiąte,
3. **Zbiór 3** – podzbiór zawierający 50% wierszy z najmniejszą wartością dla kolumny `CO_uncertainty` oraz zawodnikami znajdującymi się w rankingu na lepszym miejscu niż pięćdziesiąte.

Ocena modelu na zbiorze 2 wynika z faktu, że zawodnicy poniżej rankingu 50 rozgrywają, z większą lub mniejszą częstotliwością, turnieje rangi niższej niż ATP, które nie zostały uwzględnione w przedstawianym rozwiązaniu.

5.7. Regresja logistyczna

Regresja logistyczna jest jednym z najczęściej stosowanych modeli w uczeniu maszynowym do problemów klasyfikacyjnych, oferując interpretowalność oraz solidność w analizie zależności między zmiennymi. W kontekście przewidywania wyników meczów tenisowych jej zastosowanie jest uzasadnione z kilku powodów:

- **Prostota i interpretowalność:** Regresja logistyczna dostarcza przejrzystego modelu probabilistycznego, który pozwala na łatwą interpretację wpływu poszczególnych cech na

prawdopodobieństwo zwycięstwa danego zawodnika. W porównaniu do bardziej złożonych metod, takich jak sieci neuronowe, pozwala to lepiej zrozumieć mechanizmy stojące za przewidywaniami modelu.

- **Szybkość trenowania i małe wymagania obliczeniowe:** W porównaniu do bardziej złożonych modeli, takich jak drzewa decyzyjne czy sieci neuronowe, regresja logistyczna jest relatywnie szybka w trenowaniu i nie wymaga dużej mocy obliczeniowej. Dzięki temu możliwe jest szybkie testowanie różnych wariantów modelu i dostrajanie jego parametrów.

Mimo swoich zalet regresja logistyczna ma również pewne ograniczenia, które należy wziąć pod uwagę przy jej stosowaniu:

- **Założenie liniowości w przestrzeni logitowej:** Model zakłada, że zależność między zmiennymi wejściowymi a logarytmem szans (ang. log-odds) jest liniowa. W rzeczywistości zależności w sporcie, w tym w tenisie, mogą być nieliniowe, co ogranicza skuteczność regresji logistycznej w niektórych przypadkach.
- **Brak zdolności do automatycznego wykrywania interakcji między zmiennymi:** Regresja logistyczna nie jest w stanie samodzielnie wykrywać i modelować interakcji między cechami, chyba że zostaną one jawnie wprowadzone do modelu. W skomplikowanych problemach sportowych, gdzie wiele czynników współdziała ze sobą, może to stanowić istotne ograniczenie.

De Seranno [3] oraz Sipko [10] w swoich badaniach również wykorzystali modele regresji logistycznej. Jednak w przedstawianej pracy zastosowano bardziej rozbudowaną siatkę hiperparametrów, co pozwala na dokładniejsze dostrojenie modelu i potencjalnie lepsze wyniki predykcyjne. Szersza eksploracja przestrzeni hiperparametrów umożliwia optymalizację kluczowych ustawień, takich jak metoda optymalizacji, waga klas, siła regularyzacji oraz jej rodzaj.

5.7.1. Uproszczenie modelu

Regresja logistyczna jest modelem liniowym, co oznacza, że wyniki przewidywań modelu są w całości oparte wyłącznie na liniowej kombinacji dostarczanych cech. Dzięki tej zależności liczba parametrów modelu regresji logistycznej może zostać znacznie zredukowana.

Wyraz wolny

W przypadku, gdy dane przedmeczowe nie są znane, model opiera swoją predykcję wyłącznie na wartości wyrazu wolnego. W takim przypadku prawdopodobieństwo wybrania któregokolwiek z zawodników jako zwycięzcy meczu powinno być jednakowe i równe 0,5. Dlatego zgodnie

z zależnością

$$\theta(I) = 0{,}5 \iff I = 0, \tag{5.3}$$

gdzie:

- $\theta(I)$ – prawdopodobieństwo wygranej zawodnika,
- I – wartość wyrazu wolnego,

wartość wyrazu wolnego została ustawiona na 0, aby uniknąć jego wytrenowania do szumu.

Zmienne środowiskowe

Rozpatrzmy wpływ zmiennych środowiskowych na wyniki przewidywania modelu. Dla ustalenia uwagi przyjmijmy, że rozpatrywaną zmienną będzie nawierzchnia kortu. Rodzaj nawierzchni pozostaje niezmienny niezależnie od zamiany miejscami danych wejściowych dla `Player1` i `Player2`. Załóżmy, że rodzaj nawierzchni ma niezerowy wpływ na zwiększenie prawdopodobieństwa wygrania `Player1`, oznaczałoby to jednak, że w przypadku zamiany miejscami danych wejściowych dla zawodników, ten sam rodzaj nawierzchni faworyzowałby `Player2`, co prowadzi do sprzeczności. W związku z tym zmienne środowiskowe nie są użyteczne w modelu regresji logistycznej i nie zostały uwzględnione.

Cechy zawodników

Przyjmijmy:

1. x_{p1}, x_{p2} – cechy zawodników,
2. β_{p1}, β_{p2} – odpowiadające im wagi.

Zauważmy, że aby spełnione było założenie o symetryczności modelu, musi zachodzić:

$$\theta(\beta_{p1}x_{p1} + \beta_{p2}x_{p2}) = 1 - \theta(\beta_{p1}x_{p2} + \beta_{p2}x_{p1}). \tag{5.4}$$

Wykorzystując własność funkcji logistycznej:

$$\theta(x) = 1 - \theta(-x), \tag{5.5}$$

możemy przekształcić równanie w następujący sposób:

$$\beta_{p1}x_{p1} + \beta_{p2}x_{p2} = -(\beta_{p1}x_{p2} + \beta_{p2}x_{p1}). \tag{5.6}$$

Po przekształceniu otrzymujemy:

$$\beta_{p1}(x_{p1} + x_{p2}) = -\beta_{p2}(x_{p1} + x_{p2}), \tag{5.7}$$

$$\beta_{p1} = -\beta_{p2}. \tag{5.8}$$

Oznacza to, że współczynniki dla jednej cechy zawodnika obu graczy powinny być przeciwne co do wartości. W wyniku tej obserwacji wykonane zostało kolejne uproszczenie modelu – wszystkie cechy zawodników w postaci `Player1_`, `Player2_` zostały przekształcone do kolumn `diff_`, redukując liczbę cech o połowę.

5.7.2. Skalowanie zmiennych

Dla zwiększenia efektywności modelu regresji logistycznej zmienne zostały odpowiednio przeskalowane. W tym celu zostały one uprzednio rozróżnione ze względu na swój rozkład. Wyróżnione zostały dwie grupy cech: grupa zmiennych o rozkładzie Gaussa oraz grupa zmiennych, których rozkład nie był normalny. Do dokonania wspomnianego podziału wykorzystany został test **Kołmogorowa–Smirnowa**.

Test Kołmogorowa–Smirnowa testuje hipotezę zerową, że rozkład danej cechy jest taki sam jak zadany rozkład (w tym przypadku rozkład normalny). W tym celu porównuje rozkład empiryczny danej cechy z teoretycznym rozkładem. Test został wykonany na poziomie istotności $\alpha = 0{,}05$, w związku z czym:

- Jeśli p-wartość $> 0{,}05$, to hipoteza zerowa nie została odrzucona, co oznacza, że cecha nie różniła się istotnie od rozkładu normalnego.
- Jeśli p-wartość $\leqslant 0{,}05$, to hipoteza zerowa została odrzucona, co oznacza, że cecha różniła się istotnie od rozkładu normalnego.

Na zmiennych, które zostały sklasyfikowane jako gaussowskie, przeprowadzono proces standaryzacji. W wyniku tego procesu średnia takich zmiennych wynosi 0, a odchylenie standardowe wynosi 1. Na drugiej grupie zmiennych przeprowadzony został proces skalowania metodą Min Max, czyli przeskalowania wartości w taki sposób, aby znajdowały się w przedziale $[0, 1]$.

5.7.3. Proces uczenia

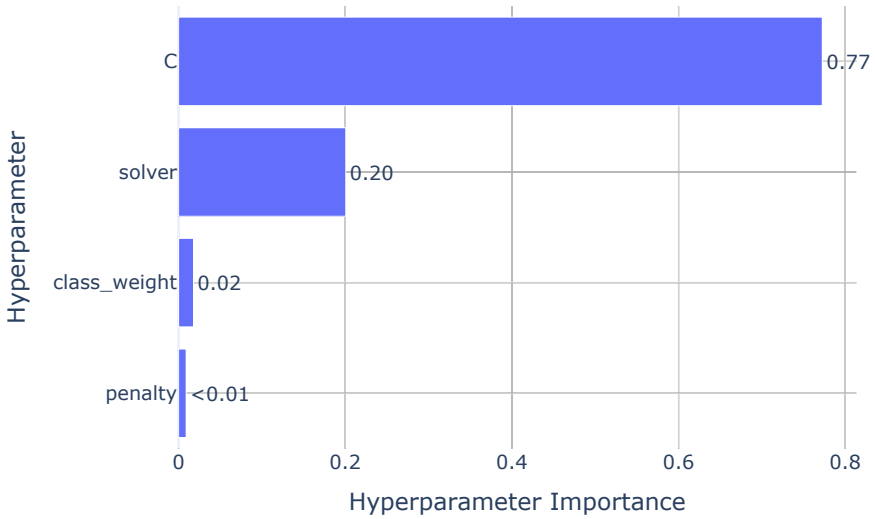
Walidacja krzyżowa

Przeskalowane dane zostały podzielone na zestaw testowy – mecze, które odbyły się w roku 2023 oraz treningowe – pozostałe mecze. Do trenowania modelu została zdefiniowana funkcja celu, która wykorzystywała 5-krotną walidację krzyżową na zestawie treningowym. W przypadku podziału zbioru treningowego zapewniony został podział z zachowaniem proporcji klasy zmiennej objaśnianej, aby zapewnić równomierny rozkład celu w podzbiorach. Ponadto została zdefiniowana siatka hiperparametrów, przedstawiona w tabeli 5.1.

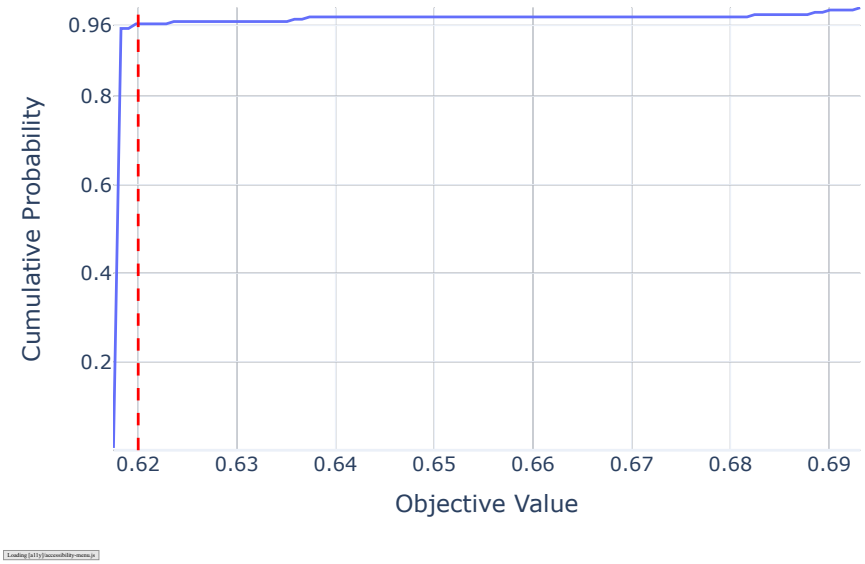
Tab. 5.1. Siatka hiperparametrów

Hiperparametr	Zakres
C (siła regularyzacji)	10^{-4} do 10^{4}
penalty (rodzaj regularyzacji)	`null, l1, l2`
solver (metoda optymalizacji)	`libliner, saga, lbfgs`
class_weight (waga klas)	`null, balanced`

Z analizy uczenia wynika, że największy wpływ na rezultaty miała metoda optymalizacji oraz siła regularyzacji. Poziom istotności hiperparametrów przedstawiony został na rysunku 5.2.
Model na zbiorze treningowym osiągnął najlepszy wynik logarytmicznej funkcji straty – 0,6174 i nie wykazał już predyspozycji do poprawy, ponieważ dla wartości wynoszącej 0,62, według rozkładu empirycznego (rys. 5.3) dystrybuanta przyjmuje już wartość 0,96.



Rys. 5.2. Poziom istotności hiperparametrów dla walidacji krzyżowej w modelu regresji logistycznej



Rys. 5.3. Empiryczny rozkład wartości funkcji celu dla walidacji krzyżowej w modelu regresji logistycznej

Tab. 5.2. Wartości hiperparametrów, dla których model osiągnął najlepsze wyniki na zbiorze treningowym

Hiperparametr	Wartość
C	49,17579
penalty	`l2`
solver	`lbfgs`
class_weight	`null`

Tab. 5.3. Wybrane przez model cechy

1.	**diff_age**
2.	**diff_rank**
3.	**diff_entry_Q**
4.	**diff_is_seeded**
5.	**diff_1st_serve_in_pct_avg**
6.	**diff_CO_1st_serve_win_pct_avg**
7.	**diff_CO_2nd_serve_in_pct_avg**
8.	**diff_service_games_won_pct_avg**
9.	**diff_CO_service_games_won_pct_avg**
10.	**diff_CO_2nd_serve_return_win_pct_avg**
11.	**diff_CO_return_games_win_pct_avg**
12.	**diff_bp_won_pct_avg**
13.	**diff_CO_bp_saved_pct_avg**
14.	**diff_elo**
15.	**diff_surface_elo**
16.	**diff_blended_elo**
17.	**diff_h2h_wins**
18.	**diff_win_pct_last_10_surface**
19.	**diff_Game_Diff_Tournament**
20.	**diff_total_wins_tournament_history**

Wartości hiperparametrów modelu, który osiągnął najlepsze wyniki, przedstawione zostały w tabeli 5.2. W wyniku selekcji cech w końcowym modelu znalazło się 20 cech przedstawionych w tabeli 5.3.

Tab. 5.4. Wyniki modelu regresji logistycznej

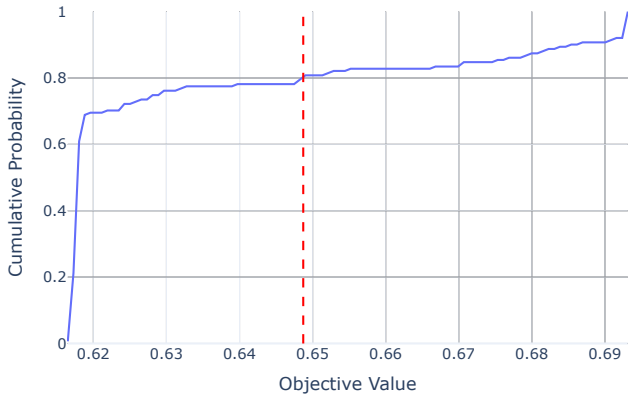
Zbiór testowy	Dokładność	Logarytmiczna funkcja straty	Wskaźnik Briera
Zbiór 1	0,64695	0,62002	0,21663
Zbiór 2	0,67697	0,64884	0,22037
Zbiór 3	0,68704	0,64062	0,21675

Na podstawie wyników (tab. 5.4) można zauważyć, że model na całym zbiorze testowym osiąga wartość logarytmicznej funkcji straty 0,62002. Jednak dla **zbioru 2** i **zbioru 3**, mimo większej dokładności, wartość logarytmicznej funkcji straty w przypadku tych zbiorów znacznie wzrosła. Oznacza to, że model regresji logistycznej dla zawodników z pierwszej pięćdziesiątki rankingu przewiduje zwycięzcę ze znacznie większą pewnością, co w wyniku błędnej predykcji skutkuje zwiększoną karą.

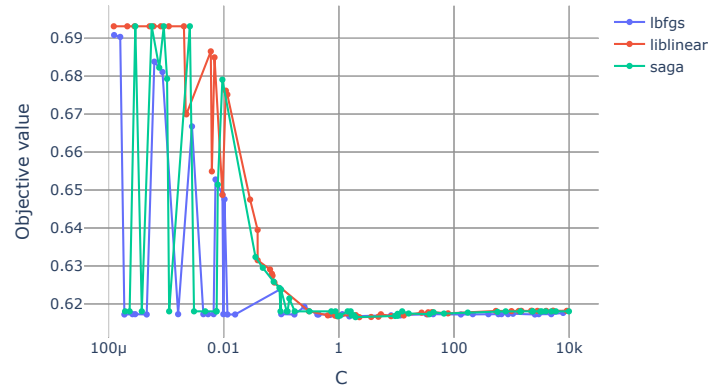
Zagnieżdżona walidacja krzyżowa

Proces trenowania modelu dla zagnieżdżonej walidacji krzyżowej był analogiczny do procesu przeprowadzonego na zwykłej walidacji. W wyniku procesu uczenia model oparty na zagnieżdżonej walidacji krzyżowej osiągnął na zbiorze treningowym wynik logarytmicznej funkcji straty równy 0,61689.

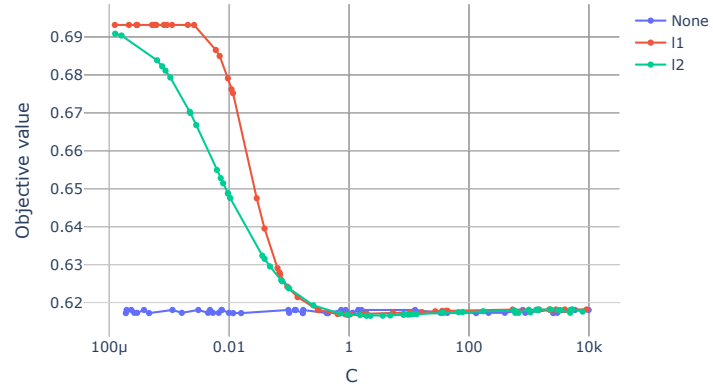
Wynik ten na zbiorze treningowym już jest lepszy niż w przypadku zwykłej walidacji, jednak z empirycznego rozkładu wartości funkcji celu (rys. 5.4) wynika, że przy aktualnej siatce hiperparametrów model poświęca wiele prób na przeszukiwanie obszarów, w których wartość funkcji jest znacznie większa. Dystrybuanta przyjmuje wartość 0,8 dopiero dla funkcji celu równej 0,649.



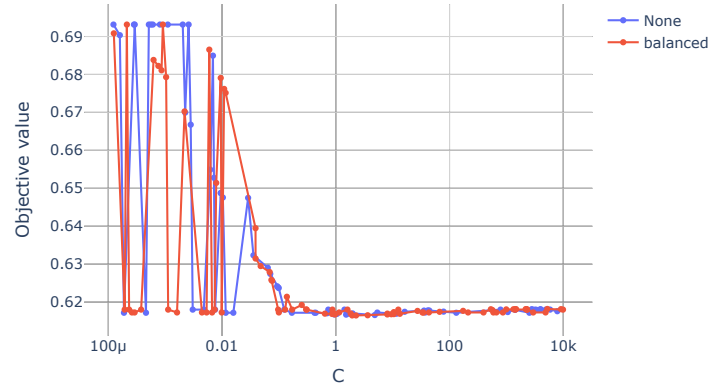
Rys. 5.4. Empiryczny rozkład wartości funkcji celu



Rys. 5.5. Zależność między metodą optymalizacji a siłą regularyzacji dla wartości funkcji celu



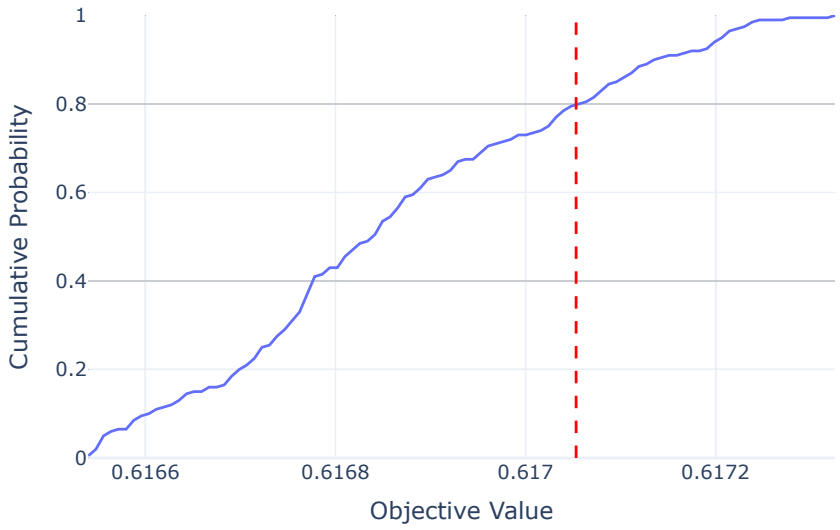
Rys. 5.6. Zależność między rodzajem regularyzacji a siłą regularyzacji dla wartości funkcji celu



Rys. 5.7. Zależność między wagą klas a siłą regularyzacji dla wartości funkcji celu

Tab. 5.5. Zmodyfikowana siatka hiperparametrów

Hiperparametr	Zakres
C (siła regularyzacji)	10^{-1} do 10
penalty (rodzaj regularyzacji)	`l1, l2`
solver (metoda optymalizacji)	`libliner, saga, lbfgs`
class_weight (waga klas)	`null, balanced`



Rys. 5.8. Empiryczny rozkład wartości funkcji celu dla zmodyfikowanej siatki hiperparametrów dla zagnieżdżonej walidacji krzyżowej w modelu regresji logistycznej

Z analizy zależności, jak pary hiperparametrów wpływają na wartość funkcji celu przedstawionej na wykresach 5.5, 5.6 oraz 5.7 wynika, że model osiąga słabe rezultaty dla wartości `C` z zakresu 10^{-4}–10^{-1}. Co więcej, przy braku regularyzacji model osiąga względnie niskie wartości logarytmicznej funkcji straty, jednak nie zbliża się do najmniejszych wartości osiągniętych przez model. W wyniku tych obserwacji siatka hiperparametrów została odpowiednio zmodyfikowana (tab. 5.5), aby model poświęcał więcej prób w bardziej obiecujących obszarach.

W wyniku trenowania na zmodyfikowanej siatce hiperparametrów udało się minimalnie poprawić na zbiorze treningowym wartość logarytmicznej funkcji straty, osiągając wartość 0,61649. Dla tak zmodyfikowanej siatki dystrybuanta przyjmuje wartość 0,8 dla funkcji celu równej 0,61704, co zostało przedstawione na rysunku 5.8.

W wyniku selekcji cech w końcowym modelu znalazły się te same cechy, co w przypadku modelu opartego na zwykłej walidacji, przedstawione w tabeli 5.3. Wartości hiperparametrów modelu, który osiągnął najlepsze wyniki, znajdują się w tabeli 5.6 poniżej.

Tab. 5.6. Ustawienia hiperparametrów, dla których model osiągnął najlepsze wyniki na zbiorze treningowym

Hiperparametr	**Wartość**
C	2,26940
penalty	`l2`
solver	`liblinear`
class_weight	`balanced`

5.7.4. Podsumowanie wyników

Zgodnie z wynikami tabeli 5.7 model oparty na zagnieżdżonej walidacji krzyżowej osiągnął nieznacznie lepsze rezultaty na każdym podzbiorze testowym, dla każdej metryki. Oba modele mają widocznie większą wartość logarytmicznej funkcji straty dla obu zbiorów z zawodnikami z pierwszej pięćdziesiątki rankingu. Niezależnie od metody walidacji, modele przewidują zwycięzcę ze znacznie większą pewnością, czego skutkiem w wyniku błędnej predykcji jest zwiększona właśnie wspomniana wcześniej wartość.

Tab. 5.7. Porównanie wyników modelu regresji logistycznej dla walidacji krzyżowej i zagnieżdżonej walidacji krzyżowej

Zbiór testowy	**Metoda**	**Dokładność**	**Logarytmiczna funkcja straty**	**Wskaźnik Briera**
Zbiór 1	Walidacja krzyżowa	0,64695	0,62002	0,21663
	Zagnieżdżona walidacja krzyżowa	**0,64700**	**0,61976**	**0,21647**
Zbiór 2	Walidacja krzyżowa	0,67697	0,64884	0,22037
	Zagnieżdżona walidacja krzyżowa	**0,67753**	**0,64271**	**0,21931**
Zbiór 3	Walidacja krzyżowa	0,68704	0,64062	0,21675
	Zagnieżdżona walidacja krzyżowa	**0,68889**	**0,63318**	**0,21526**

5.8. XGBoost

XGBoost to jeden z algorytmów uczenia maszynowego, który w ostatnich latach cieszy się dużą popularnością ze względu na swoją wysoką skuteczność, szybkość działania oraz zdolność do modelowania skomplikowanych zależności między danymi. W kontekście przewidywania wyników meczów tenisowych XGBoost wydaje się dobrym wyborem z następujących powodów:

- **Modelowanie złożonych zależności**: Wynik meczu tenisowego zależy od wielu czynników, takich jak styl gry zawodników, ich aktualna forma, typ nawierzchni czy warunki atmosferyczne. XGBoost potrafi skutecznie wykrywać i wykorzystywać te zależności, nawet jeśli są one nieliniowe i skomplikowane.
- **Obsługa dużej liczby cech**: W analizie wyników sportowych często wykorzystuje się wiele różnych zmiennych – od statystyk zawodników po dane o turnieju. XGBoost dobrze radzi sobie z dużą liczbą cech i potrafi automatycznie określić, które z nich mają największy wpływ na wynik.
- **Odporność na przeuczenie**: Dzięki wbudowanej regularyzacji XGBoost lepiej unika przeuczenia niż tradycyjne drzewa decyzyjne. To oznacza, że model dobrze generalizuje na nowych danych i nie dopasowuje się nadmiernie do specyfiki zbioru treningowego.

Mimo wielu zalet XGBoost ma również pewne ograniczenia, które warto uwzględnić:

- **Mniejsza interpretowalność**: Chociaż XGBoost pozwala analizować znaczenie poszczególnych cech, jego wewnętrzne działanie nie jest tak intuicyjne, jak w przypadku np. regresji logistycznej. Może to utrudniać wyjaśnianie decyzji modelu, zwłaszcza osobom niezaznajomionym z uczeniem maszynowym.
- **Złożoność optymalizacji hiperparametrów**: Aby osiągnąć najlepsze wyniki, konieczne jest odpowiednie dostrojenie parametrów modelu, takich jak liczba drzew, głębokość drzewa czy szybkość uczenia. Proces ten wymaga przeprowadzenia wielu eksperymentów i może być czasochłonny.

5.8.1. Skalowanie zmiennych

Model XGBoost jest oparty o drzewa decyzyjne, które są niezależne od skali zmiennych, ponieważ bazują na podziałach, a nie wartościach liczbowych. W związku z tym algorytm automatycznie określa optymalne punkty podziału niezależnie od wielkości liczb, skalowanie zmiennych w przypadku modelu XGBoost zostało pominięte.

5.8.2. Obliczanie prawdopodobieństwa

Niestety dla modelu XGBoost nie udało się spełnić założenia symetryczności modelu. W przeciwieństwie do sieci neuronowych, gdzie można zmodyfikować metodę `forward` i wpłynąć na sposób propagacji w całym procesie uczenia i predykcji, w XGBoost oferowanym przez bibliotekę *scikit-learn* istnieje ograniczenie dostępu do wewnętrznego mechanizmu generowania predykcji podczas treningu.

Chociaż można dostosować metodę *predict_proba*, to model w trakcie uczenia korzysta z własnego mechanizmu predykcji, do którego użytkownik nie ma bezpośredniego dostępu. W efekcie nie ma możliwości pełnej kontroli nad sposobem, w jaki generowane są przewidywania w czasie treningu, co uniemożliwia zapewnienie symetryczności modelu.

Niemniej jednak, rezultaty modelu wciąż oferują wysoką skuteczność predykcji, a jego realizację uznajemy za wartościowy element przedstawianego rozwiązania. Zapewnienie symetryczności modelu stanowi jeden z elementów przyszłych prac (zob. rozdz. 7).

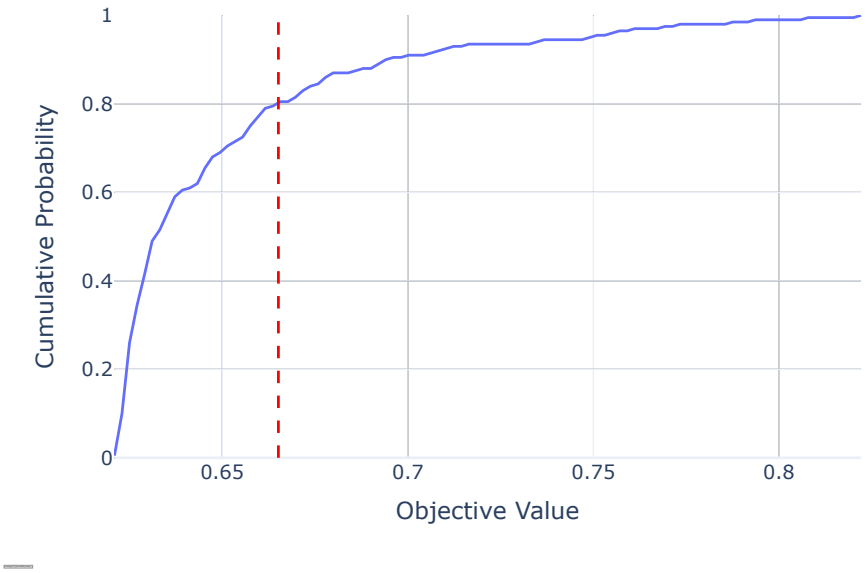
5.8.3. Proces uczenia

Walidacja krzyżowa

Proces trenowania modelu był analogiczny do tego przedstawionego dla regresji logistycznej. Dla procesu optymalizacji zdefiniowana została siatka hiperparametrów przedstawiona w tabeli 5.8.

Tab. 5.8. Siatka hiperparametrów

Hiperparametr	Zakres	Opis
max_depth	3–10	Maksymalna głębokość drzewa
learning_rate	0,01–0,3	Tempo uczenia
n_estimators	100–1000	Liczba estymatorów (drzew) w modelu
subsample	0,5–1	Frakcja próbek używanych do budowy każdego drzewa
gamma	0–5	Minimalna redukcja funkcji straty wymagana do podziału węzła
colsample_bytree	0,5–1	Frakcja cech wybieranych do budowy każdego drzewa
min_child_weight	1–10	Minimalna suma wag próbek w węźle liściowym
lambda	10^{-3}–10	Współczynnik regularyzacji L_2
alpha	10^{-3}–10	Współczynnik regularyzacji L_1



Rys. 5.9. Empiryczny rozkład wartości funkcji celu

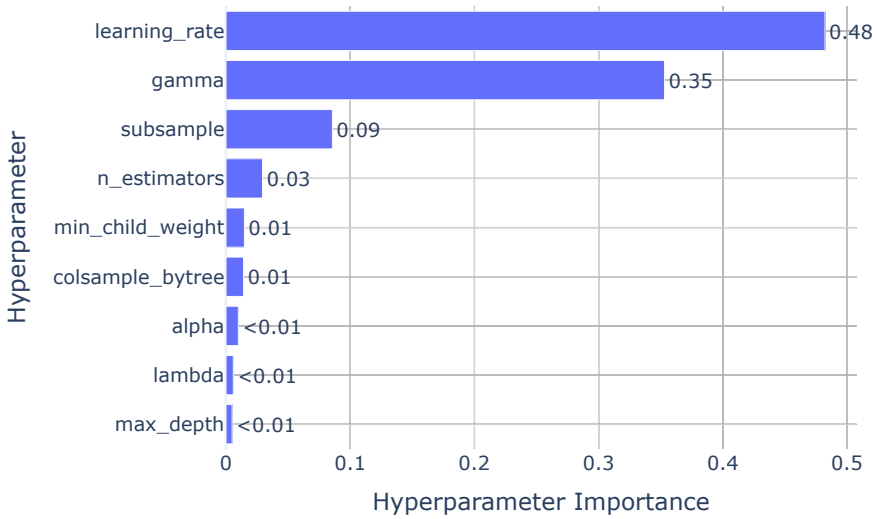
W wyniku pierwszego uczenia model na zbiorze treningowym osiągnął najlepszą wartość logarytmicznej funkcji straty wynoszącą 0,621. Z empirycznego rozkładu wartości funkcji celu (rys. 5.9) wynika, że model ma predyspozycje do poprawy, ponieważ osiąga wartość dystrybuanty równą 0,8 dopiero dla wartości funkcji celu równej 0,665.

Z analizy istotności hiperparametrów (rys. 5.10) wynika, że największy wpływ na rezultaty modelu miały hiperparametry `gamma`, `learning rate` oraz `subsample`. Dla tych trzech najważniejszych hiperparametrów została ustalona zależność pomiędzy ich wartościami a wartością funkcji straty (rys. 5.11, rys. 5.12). Model osiąga gorsze rezultaty, gdy `learning rate` przyjmuje wartości większe niż 0,05. Dla hiperparametru `gamma` widoczna jest tendencja spadkowa wartości funkcji celu wraz ze wzrostem wartości parametru, dodatkowo model osiąga najlepsze rezultaty, gdy `subsample` znajduje się w przedziale 0,8–1.

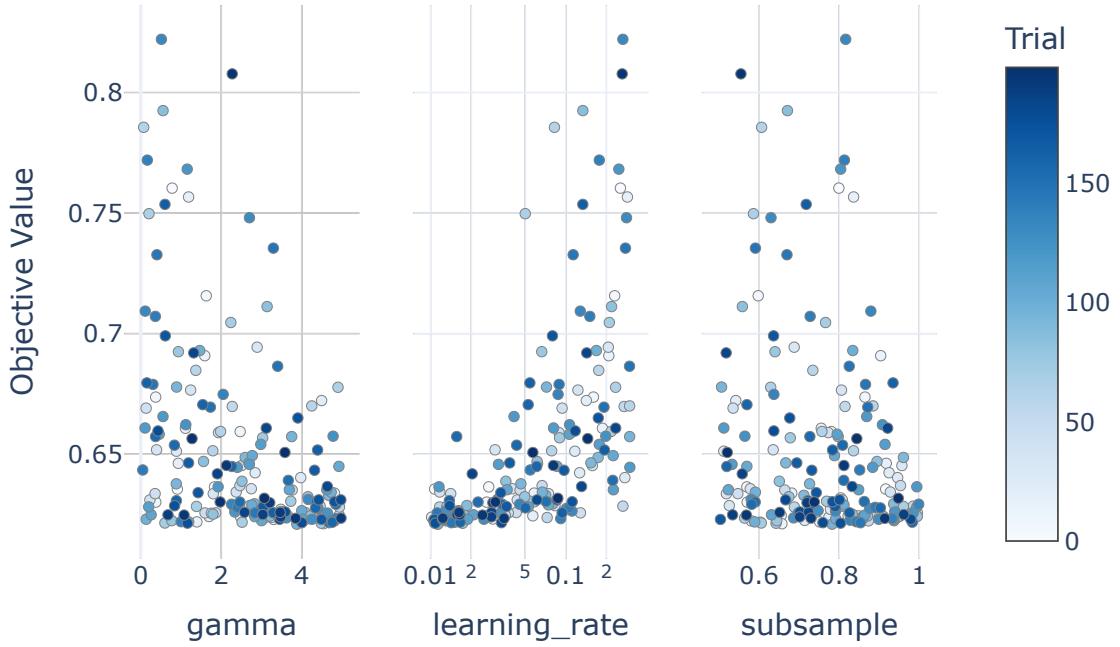
Na tej podstawie dla wskazanych hiperparametrów wartości w siatce zostały odpowiednio zmodyfikowane (tab. 5.9).

Tab. 5.9. Zmiany w siatce hiperparametrów

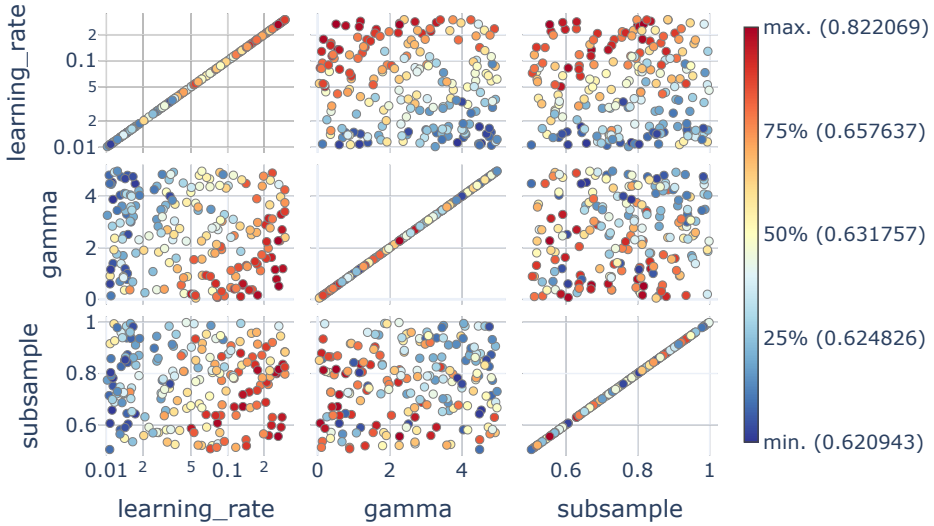
Hiperparametr	**Zakres**
learning_rate	0,01–0,04
gamma	3,5–8
subsample	0,8–1



Rys. 5.10. Wpływ hiperparametrów na rezultaty modelu



Rys. 5.11. Wpływ wartości hiperparametru na logarytmiczną funkcję straty

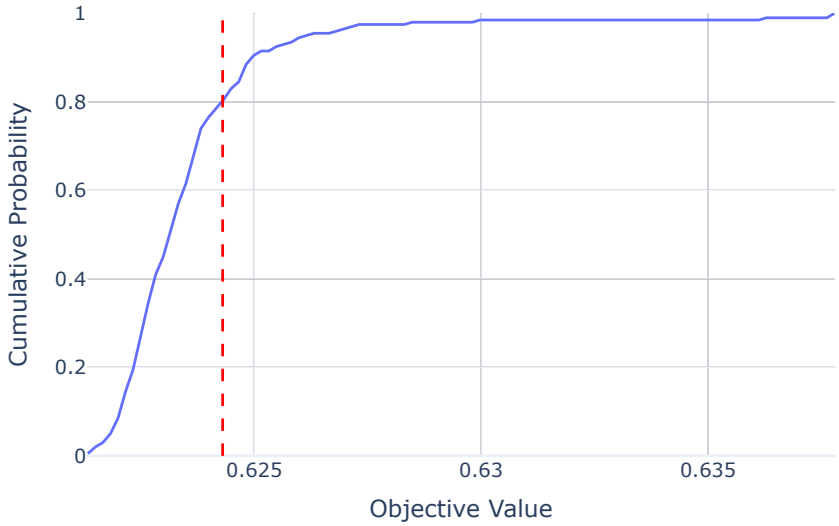


Rys. 5.12. Zależność pomiędzy wartościami hiperparametrów a logarytmiczną funkcją straty

Model po ponownym trenowaniu na zmodyfikowanej siatce hiperparametrów osiągnął średnio lepsze wyniki, o czym świadczy wartość dystrybuanty 0,8 już dla wartości funkcji celu równej 0,624 widocznej na empirycznym rozkładzie rezultatów (rys. 5.13). Mimo średniej poprawy wyników, model nie osiągnął jednak lepszego rezultatu niż podczas trenowania na oryginalnej siatce hiperparametrów. W związku z tym ustawienia po pierwszym procesie uczenia zostały zapisane jako najlepsze i przedstawione w tabeli 5.10.

Tab. 5.10. Wartości hiperparametrów, dla których model osiągnał najlepsze wyniki

Hiperparametr	**Wartość**
learning_rate	0,0392
gamma	4,5694
subsample	0,9344
max_depth	6
n_estimators	318
colsample_bytree	0,8808
min_child_weight	8
lambda	0,0296
alpha	0,3382



Rys. 5.13. Empiryczny rozkład wartości funkcji celu dla zmodyfikowanej siatki hiperparametrów

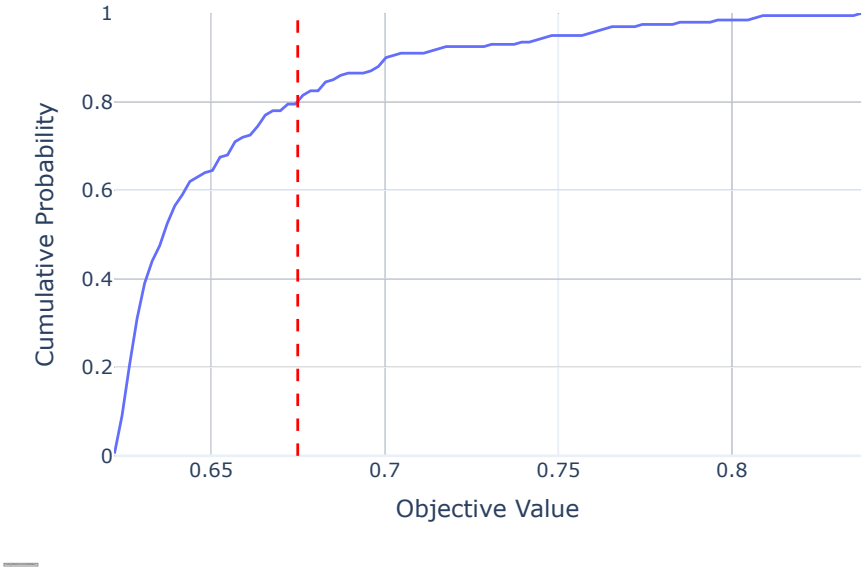
Tab. 5.11. Wyniki modelu XGBoost opartej na walidacji krzyżowej

Zbiór testowy	Dokładność	Logarytmiczna funkcja straty	Wskaźnik Briera
Zbiór 1	0,63727	0,62525	0,21834
Zbiór 2	0,68729	0,59092	0,20287
Zbiór 3	0,69314	0,58797	0,20151

W wyniku selekcji model osiągający najlepsze rezultaty wykorzystywał wszystkie cechy dostępne w oryginalnej ramce danych. Z wyników przedstawionych w tabeli 5.11 można zauważyć, że model na całym zbiorze testowym osiągnął gorsze wyniki niż model regresji logistycznej. Niemniej jednak, w przeciwieństwie do modelu regresji w przypadku **zbioru 2** i **3**, rezultaty są dużo bardziej satysfakcjonujące. Zauważalna jest wyraźna poprawa pod względem każdej z metryk.

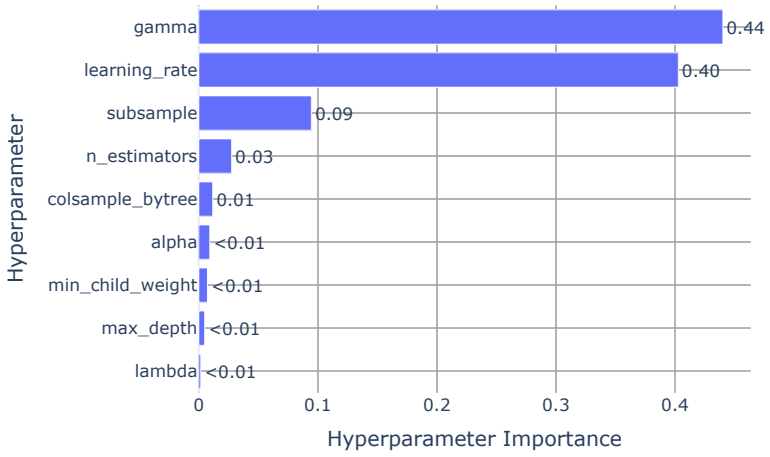
Zagnieżdżona walidacja krzyżowa

Model oparty o zagnieżdżoną walidację krzyżową wykazał analogiczne problemy do modelu opartego o zwykłą walidację. Wytrenowany został na identycznej siatce hiperparametrów i po pierwszym uczeniu można z rozkładu wyników wartości funkcji celu (rys. 5.14) zauważyć, że model poświęcił wiele prób na przeszukiwanie obszarów, które nie przyniosły satysfakcjonujących efektów. Wartość dystrybuanty jest równa 0,8 dopiero dla wartości funkcji celu równej 0,676.

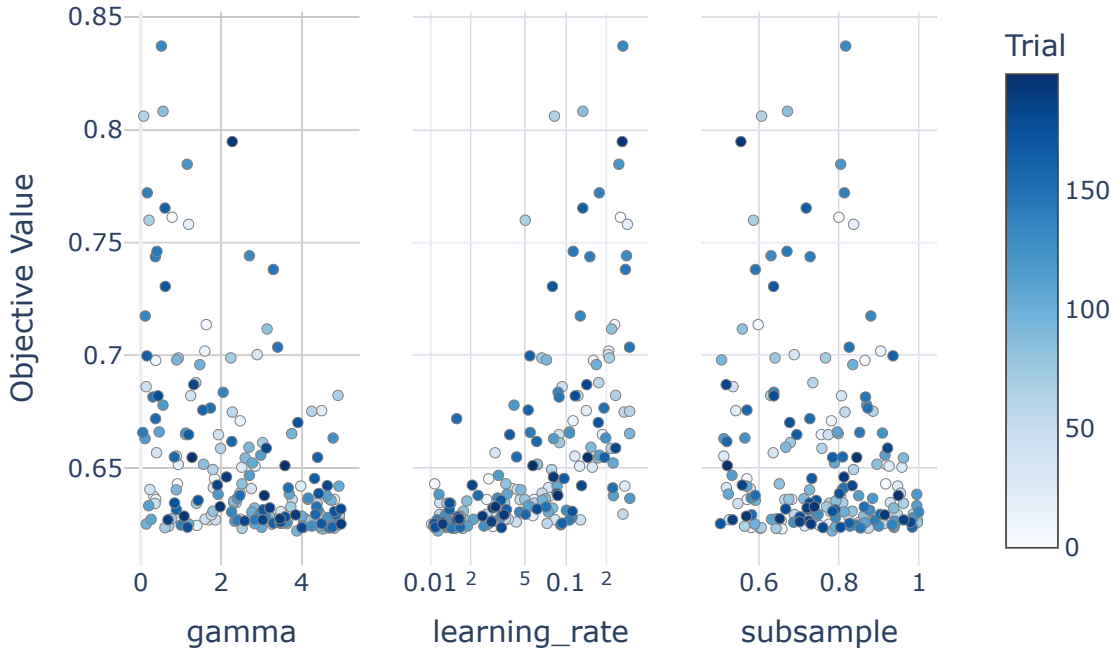


Rys. 5.14. Empiryczny rozkład wartości funkcji celu

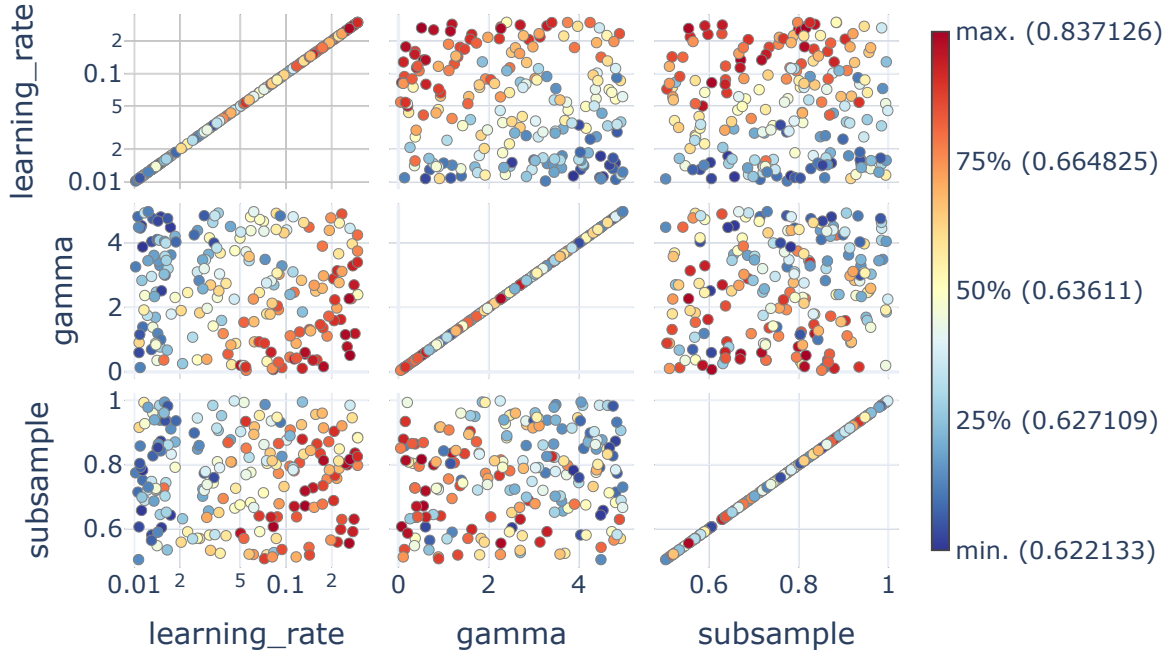
Analiza wpływu poszczególnych parametrów (rys. 5.15), jako te najbardziej istotne, wskazała te same trzy czynniki, co zwykła walidacja: `gamma`, `learning rate` oraz `subsample`.
Z wykresów zależności pomiędzy wartościami tych parametrów a wartością funkcji celu (rys. 5.16, rys. 5.17) można również wyciągnąć jednakowe wnioski w kwestii modyfikacji siatki hiperparametrów, co w przypadku modelu opartego o zwykłą walidację.



Rys. 5.15. Wpływ hiperparametrów na rezultaty modelu



Rys. 5.16. Zależność pomiędzy wartościami hiperparametrów a logarytmiczną funkcją straty



Rys. 5.17. Zależność pomiędzy wartościami hiperparametrów a logarytmiczną funkcją straty

W związku z tym dla drugiego procesu trenowania modelu siatka hiperparametrów uległa modyfikacji tak samo, jak zostało to przedstawione w tabeli 5.9. W wyniku uczenia modelu udało się na zbiorze treningowym osiągnąć wynik logarytmicznej funkcji straty równy 0,622.

Główną różnicą między modelami opartymi o różne metody walidacji był wybór cech. Model oparty o zagnieżdżoną walidację krzyżową, który osiągnął najlepsze wyniki, zrezygnował z cech przedstawionych w tabeli 5.12. Optymalne wartości hiperparametrów dla tego modelu przedstawiono w tabeli 5.13.

Tab. 5.12. Niewykorzystane przez model cechy

1.	**right_handed**
2.	**h2h_surface_wins**
3.	**is_seeded**
4.	**h2h_wins**
5.	**injury_score**
6.	**surface_clay**,**surface_grass**

Tab. 5.13. Wartości hiperparametrów, dla których model osiągnał najlepsze wyniki

Hiperparametr	**Wartość**
learning_rate	0,0113
gamma	3,8776
subsample	0,7824
max_depth	4
n_estimators	525
colsample_bytree	0,5328
min_child_weight	5
lambda	0,1251
alpha	0,0579

Tab. 5.14. Porównanie wyników modelu XGBoost dla walidacji krzyżowej i zagnieżdżonej walidacji krzyżowej

Zbiór testowy	Metoda	Dokładność	Logarytmiczna funkcja straty	Wskaźnik Briera
Zbiór 1	Walidacja krzyżowa	0,63727	0,62525	0,21834
	Zagnieżdżona walidacja krzyżowa	**0,64843**	**0,62137**	**0,21657**
Zbiór 2	Walidacja krzyżowa	0,68729	0,59092	0,20287
	Zagnieżdżona walidacja krzyżowa	**0,69398**	**0,58816**	**0,20122**
Zbiór 3	Walidacja krzyżowa	0,69314	0,58797	0,20151
	Zagnieżdżona walidacja krzyżowa	**0,70216**	**0,58670**	**0,20049**

5.8.4. Podsumowanie wyników

Na podstawie uzyskanych wyników można zauważyć, że zastosowanie zagnieżdżonej walidacji krzyżowej w modelu XGBoost przynosi nieznaczną poprawę wyników w porównaniu do standardowej walidacji krzyżowej. W każdym z testowanych zbiorów testowych zagnieżdżona walidacja krzyżowa prowadzi do wyższej dokładności oraz nieco lepszych wartości funkcji straty logarytmicznej i wskaźniku Briera. Oba modele radzą sobie bardzo dobrze w przypadku zbiorów drugiego oraz trzeciego. Oznacza to, że klasyfikator XGBoost osiąga bardzo dobre wyniki dla zawodników z pierwszej pięćdziesiątki rankingu, co sugeruje, że może być szczególnie przydatny do przewidywania wyników meczów na najwyższym poziomie rozgrywek.

5.9. Sieć neuronowa

Sieci neuronowe są jednymi z najbardziej zaawansowanych algorytmów w dziedzinie uczenia maszynowego, oferując elastyczność i zdolność do modelowania złożonych zależności nieliniowych. W kontekście problemu przewidywania wyników meczów tenisowych ich zastosowanie jest uzasadnione z kilku powodów:

- **Modelowanie nieliniowych zależności:** Wyniki meczów tenisowych zależą od wielu zmiennych, takich jak cechy zawodników, warunki meczu czy historia wcześniejszych spotkań. Zależności te są często nieliniowe, co sprawia, że proste modele statystyczne mogą być niewystarczające. Sieci neuronowe doskonale nadają się do uchwycenia takich skomplikowanych relacji.
- **Obsługa dużej liczby cech:** Zestaw cech używanych w modelu obejmuje informacje o zawodnikach, dane pogodowe, statystyki historyczne i inne. Sieci neuronowe potrafią skutecznie przetwarzać dane wielowymiarowe, umożliwiając analizę dużej liczby zmiennych wejściowych.
- **Uniwersalność:** Dzięki swojej elastyczności sieci neuronowe mogą być dostosowane do różnych problemów, takich jak klasyfikacja, regresja czy szeregowanie. To czyni je wszechstronnym narzędziem o szerokim zastosowaniu, w tym w analizie wyników sportowych.

Mimo wielu zalet sieci neuronowe niosą ze sobą również pewne trudności i wyzwania, które należy uwzględnić podczas ich projektowania i trenowania:

- **Złożoność architektury:** Sieci neuronowe, szczególnie głębokie, mogą wymagać zaprojektowania złożonej architektury, która uwzględnia liczbę warstw, liczbę neuronów w warstwach ukrytych oraz funkcje aktywacji. Wybór odpowiednich parametrów często wymaga eksperymentów i dogłębnego zrozumienia problemu, co jest procesem czasochłonnym.
- **Wymagania obliczeniowe:** Trenowanie sieci neuronowych, zwłaszcza na dużych zbiorach danych, wymaga znacznych zasobów obliczeniowych. Ograniczone zasoby mogą wydłużać czas trenowania i utrudniać przeprowadzanie wielu eksperymentów.
- **Interpretowalność wyników:** Sieci neuronowe często są uważane za „czarne skrzynki", co oznacza, że trudno jest zrozumieć, jak dokładnie model podejmuje decyzje. W kontekście analizy danych sportowych może to utrudniać interpretację wyników i uzasadnienie prognoz modelu.

- **Optymalizacja hiperparametrów:** Proces trenowania sieci wymaga doboru wielu hiperparametrów, takich jak szybkość uczenia, wielkość partii czy liczba epok. Znalezienie optymalnych wartości wymaga przeprowadzenia licznych eksperymentów, co dodatkowo zwiększa złożoność procesu.

De Seranno [3] oraz Sipko [10] przeprowadzili badania nad zastosowaniem sieci neuronowych do przewidywania wyników meczów tenisowych. W obu przypadkach sieć neuronowa okazała się najskuteczniejszym modelem predykcyjnym w porównaniu z innymi metodami. W niniejszej pracy podejście to zostało rozwinięte poprzez uwzględnienie większej liczby cech wejściowych, takich jak dane pogodowe, które mogą mieć istotny wpływ na wyniki meczów. Ponadto w eksperymentach wykorzystano bardziej rozbudowane architektury sieci, zawierające większą liczbę neuronów oraz warstw. Dzięki temu możliwe było dokładniejsze modelowanie złożonych zależności w danych, co potencjalnie zwiększa skuteczność predykcji.

5.9.1. Struktura sieci

Model został skonstruowany jako sekwencyjna sieć neuronowa składająca się z następujących elementów:

- **Warstwa wejściowa**: Przyjmuje dane wejściowe, które są połączeniem wektorów cech obu graczy oraz środowiska. Rozmiar wejścia to suma:
  - cech gracza pierwszego,
  - cech gracza drugiego,
  - cech środowiska.
- **Warstwy ukryte**: Są tworzone dynamicznie w zależności od zadanej listy rozmiarów (`hidden_sizes`). Każda warstwa ukryta składa się z:
  - liniowej transformacji,
  - funkcji aktywacji ReLU,
  - warstwy odrzucania neuronów, co pozwala na redukcję przetrenowania modelu.
- **Warstwa wyjściowa**: Składa się z pojedynczego neuronu, który zwraca wartość prawdopodobieństwa dzięki zastosowaniu funkcji aktywacji sigmoidalnej.

5.9.2. Obliczanie prawdopodobieństwa

Aby spełnić założenie o symetryczności modelu, wykorzystano podejście opracowane przez De Seranno [3]. Opiera się ono na dwukrotnym przetwarzaniu wejściowych danych w różnych kolejnościach:

1. Wejścia są konkatenowane w dwóch wariantach:
   - (gracz 1, gracz 2, środowisko),
   - (gracz 2, gracz 1, środowisko).
2. Obliczane są prawdopodobieństwa dla obu wariantów wejść:

$$P_1 = \text{wynik sieci dla wejścia (gracz 1, gracz 2, środowisko)},$$

$$P_2 = \text{wynik sieci dla wejścia (gracz 2, gracz 1, środowisko)}.$$

3. W celu uwzględnienia symetryczności, model zwraca średnią wartość wyliczoną jako:

$$P_{\text{finalne}} = \frac{P_1 + (1 - P_2)}{2}.$$

Taki sposób obliczeń zapewnia symetrię wyników w odniesieniu do kolejności graczy. Wizualizacja tej metody została zaprezentowana na rysunku 5.18.

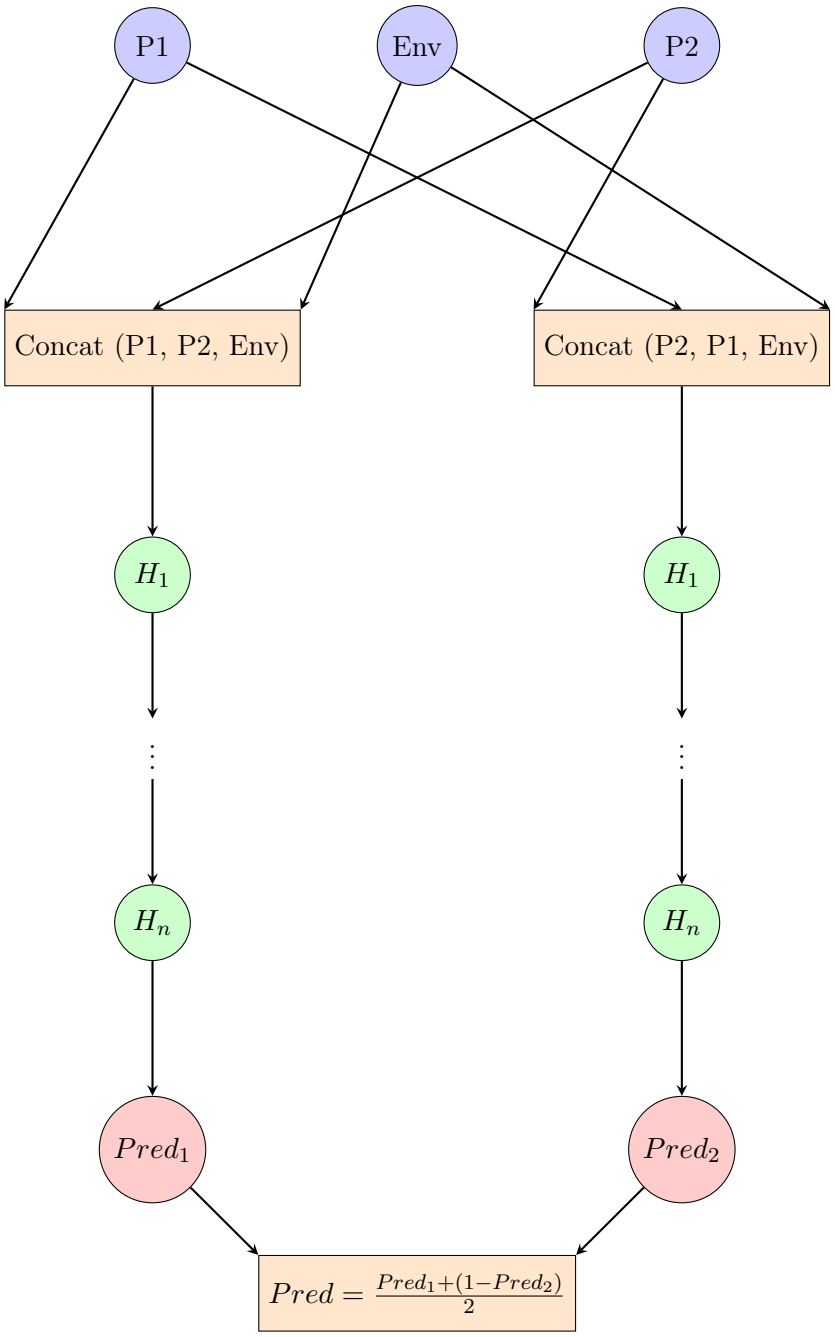
5.9.3. Zastosowanie biblioteki PyTorch

Do budowy sieci wykorzystano bibliotekę PyTorch. Model korzysta z mechanizmów PyTorch, takich jak:

- `torch.cat`, który umożliwia łączenie tensorów wzdłuż zadanych wymiarów,
- `nn.Module` jako bazowej klasy do definiowania architektury sieci,
- `nn.Sequential` do budowania sieci w sposób modularny,
- funkcji aktywacji, takich jak `ReLU` i `Sigmoid`, które odpowiadają za wprowadzenie nieliniowości oraz skalowanie wyników.

5.9.4. Skalowanie zmiennych

W procesie trenowania sieci neuronowej kluczowym krokiem jest skalowanie zmiennych wejściowych. Zdecydowano dla wszystkich zmiennych zastosować skalowanie Min-Max, które przekształca wartości cech do zakresu $[0, 1]$. Wybór skalowania Min-Max został uzasadniony następującymi czynnikami:



Rys. 5.18. Wizualizacja architektury sieci neuronowej

- **Efektywność optymalizacji:** Skalowanie wartości cech do jednorodnego zakresu $[0, 1]$ poprawia efektywność algorytmów optymalizacyjnych, takich jak gradient prosty, co przyspiesza proces trenowania modelu.

- **Zgodność z funkcjami aktywacji:** Funkcje aktywacji, takie jak `sigmoid` czy `ReLU`, działają najlepiej, gdy dane wejściowe są w ograniczonym zakresie, co umożliwia bardziej stabilne propagowanie gradientów w sieci.

- **Redukcja dominacji cech o dużych wartościach:** Bez skalowania zmienne o większych

Tab. 5.15. Obszerna siatka hiperparametrów użyta w optymalizacji sieci neuronowej

Hiperparametr	Zakres	Opis
hidden_layers	{[1024], [512, 256], [1024, 512], [512, 256, 128], [128, 64, 32], [512], [256], [128], [64, 32, 16], [128, 64]}	Konfiguracja liczby i wielkości warstw ukrytych w sieci neuronowej.
dropout	0,05–0,4	Prawdopodobieństwo 'wyłączenia' neuronu podczas trenowania
lr (learning rate)	10^{-6}–10^{-3} (logarytmiczna skala)	Tempo uczenia
weight_decay	10^{-6}–10^{-4} (logarytmiczna skala)	Współczynnik regularyzacji wag
batch_size	{32, 64, 128, 256, 512, 1024}	Liczba próbek w pojedynczej partii
optimizer	{Adam, SGD, RMSprop}	Algorytm optymalizacji

wartościach mogą zdominować proces trenowania, prowadząc do nieoptymalnych wyników. Skalowanie Min-Max równoważy wpływ wszystkich cech na model.

Skalowanie Min-Max jest często rekomendowane w kontekście sieci neuronowych, co zostało również potwierdzone w literaturze naukowej. Na przykład w pracy [4], autorzy wskazują, że ich sieć neuronowa osiągnęła najlepsze wyniki po użyciu skalowania metodą Min-Max.

5.9.5. Proces uczenia

Podobnie jak dla pozostałych modeli, sieć optymalizowana była z użyciem Optuny. Jako główną metrykę określającą jakość modelu wybrano wskaźnik Briera.

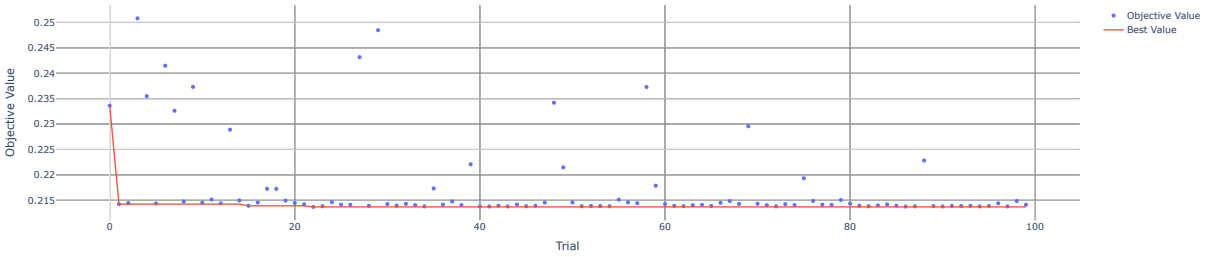
Zagnieżdżona walidacja krzyżowa

Na podstawie uzasadnienia przedstawionego w podrozdz. 5.3.3, jak i lepszych wyników dla poprzednich modeli, dla sieci neuronowej zdecydowano od razu zastosować metodę zagnieżdżonej walidacji krzyżowej. Przy procesie optymalizacji najpierw zastosowano obszerną siatkę hiperparametrów, widoczną w tabeli 5.15. Na podstawie otrzymanych wyników (tab. 5.16) oraz wykresów przedstawiających historię optymalizacji, ważność hiperparametrów oraz zależności między hiperparametrami a wartością funkcji celu, dokonano szczegółowej analizy.

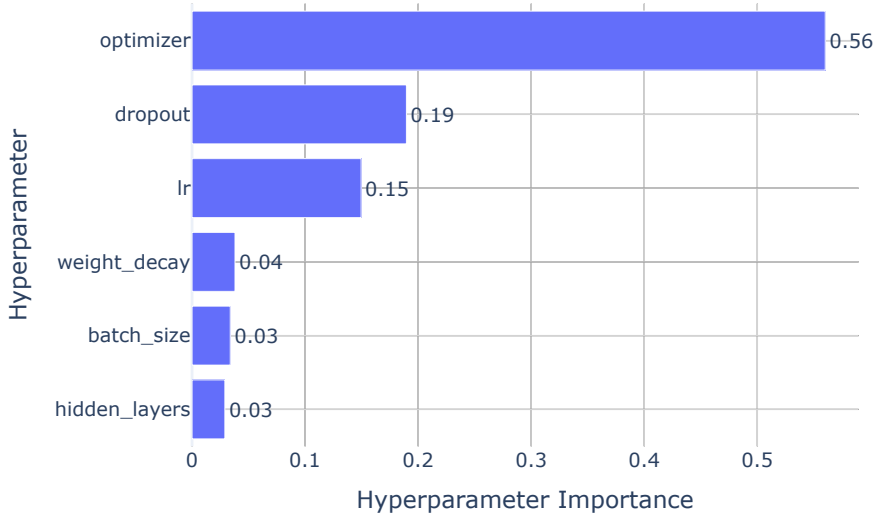
Historia optymalizacji (Rysunek 5.19) ukazuje szybki spadek wartości funkcji celu. Już po drugiej próbie osiągnięto wynik bliski najlepszemu (**0,2137**), co sugeruje, że pozostałe próby miały marginalny wpływ na poprawę wyników.
Ważność hiperparametrów (Rysunek 5.20) wskazuje, że największy wpływ na wynik miały: **Optimizer** (56%), **Dropout** (19%) oraz **Learning Rate** (15%). Pozostałe hiperparametry, takie jak *batch size*, *hidden layers* i *weight decay*, miały niewielki wpływ na wynik modelu.

Tab. 5.16. Najlepsze hiperparametry oraz wynik modelu dla obszernej siatki

Hiperparametr	**Wartość**
Warstwy ukryte (*hidden layers*)	[1024, 512]
Dropout	0,2837
Współczynnik uczenia (*learning rate*)	0,0003048
Weight decay	$2{,}01 \cdot 10^{-6}$
Wielkość partii (*batch size*)	32
Optymalizator (*optimizer*)	Adam
Najlepszy wynik (Brier score)	**0,2137**



Rys. 5.19. Historia optymalizacji funkcji celu



Rys. 5.20. Ważność hiperparametrów w optymalizacji

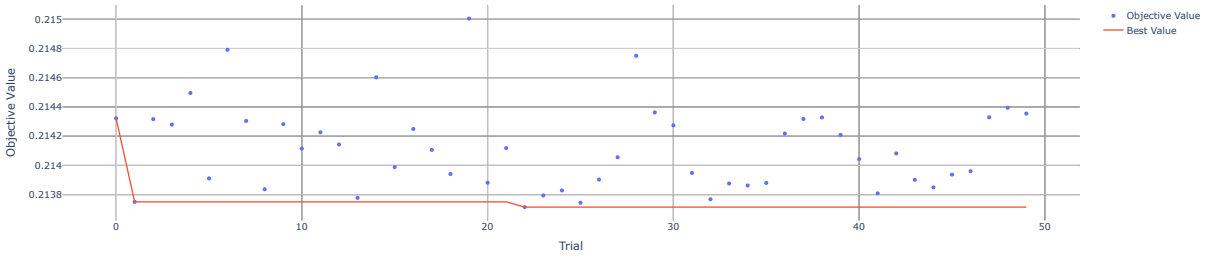
Tab. 5.17. Zawężona siatka hiperparametrów użyta prz optymalizacji sieci neuronowej

Hiperparametr	Zakres	Opis
hidden_layers	{[512, 256], [1024, 512], [512, 256, 128]}	Konfiguracja liczby i wielkości warstw ukrytych w sieci neuronowej.
dropout	0,2–0,3	Prawdopodobieństwo 'wyłączenia' neuronu podczas trenowania
lr (learning rate)	$3 \cdot 10^{-4}$–$5 \cdot 10^{-4}$ (logarytmiczna skala)	Tempo uczenia
weight_decay	$2{,}01 \cdot 10^{-6}$ (stała wartość)	Współczynnik regularyzacji wag
batch_size	{32, 256, 512}	Liczba próbek w pojedynczej partii
optimizer	{Adam}	Algorytm optymalizacji

Na podstawie szczegółowej analizy zależności hiperparametrów od wartości funkcji celu oraz ich ważności, wyciągnięto następujące wnioski:

- **Liczba warstw ukrytych (*hidden layers*)**: Najlepsze wyniki osiągnięto dla konfiguracji **2** (architektura [1024, 512]). Inne obiecujące konfiguracje to **1** (architektura [512, 256]) oraz **3** (architektura [512, 256, 128]),
- **Dropout**: Najlepsza wartość wyniosła **0,2837**, co sugeruje, że wartości w zakresie **0,2–0,3** są najbardziej obiecujące,
- **Współczynnik uczenia (*learning rate*)**: Najlepszy wynik uzyskano dla wartości **0,000304**. Optymalny zakres do dalszej analizy to **0,0003–0,0005**,
- **Optymalizator (*optimizer*)**: Najlepsze wyniki uzyskano przy użyciu **Adam**, co uzasadnia eliminację innych optymalizatorów, takich jak *SGD* i *RMSprop*,
- **Weight Decay**: Najlepszy wynik osiągnięto przy wartości $2{,}01 \cdot 10^{-6}$. Ze względu na niski wpływ tego parametru, warto ustawić jego wartość jako stałą,
- **Wielkość partii (*batch size*)**: Najlepszy wynik uzyskano dla **batch size = 32**, jednak wartości **256** i **512** również dały zadowalające rezultaty.

Na podstawie tej analizy stworzono nową siatkę hiperparametrów (tab. 5.17), która pozwala skupić się na najbardziej obiecujących wartościach, co mogłoby potencjalnie zwiększyć efektywność optymalizacji i umożliwić dalsze doskonalenie modelu. Jednakże najlepsza wartość wskaźniku Briera osiągnięta dla tej siatki wynosiła minimalnie mniej, **0,2136**, od wyniku na większej siatce. Mimo iż poprawa jest znikoma, to poprzez sam jej fakt, zdecydowano jako finalny zestaw najlepszych hiperparametrów wybrać ten uzyskany po optymalizacji na mniej obszernej siatce, przedstawionych w tabeli 5.18. Dalsze próby konstrukcji kolejnych siatek hiperparametrów nie zostały podjęte, gdyż historia optymalizacji przedstawiona na rysunku 5.21 wskazuje, że najlepsza wartość funkcji celu osiągnięta została już przy 22. próbie.



Rys. 5.21. Historia optymalizacji funkcji celu

Tab. 5.18. Najlepsze hiperparametry oraz wynik modelu dla obszernej siatki

Hiperparametr	**Wartość**
Warstwy ukryte (*hidden layers*)	[1024, 512]
Dropout	0,2713
Współczynnik uczenia (*learning rate*)	0,0003258
Weight decay	$2{,}01 \cdot 10^{-6}$
Wielkość batcha (*batch size*)	32
Optymalizator (*optimizer*)	Adam
Najlepszy wynik (Brier score)	**0,2136**

5.9.6. Podsumowanie wyników

Wyniki sieci neuronowej osiągnięte na trzech zbiorach testowych zostały przedstawione w tabeli 5.19. Dla zbioru **1** uzyskano wartość wskaźniku Briera na poziomie **0,2162**, co stanowi podstawę do porównania z pozostałymi ograniczonymi zbiorami. Wprowadzenie ograniczenia w zbiorze **2** pozwoliło na obniżenie wskaźniku Briera do **0,1975**, co wskazuje na wyraźną poprawę jakości predykcji. Z kolei w zbiorze **3**, poprzez dodatkowe filtrowanie przypadków o niższej niepewności predykcji, osiągnięto najniższy wskaźnik Briera (**0,1962**) oraz najwyższą dokładność (**68,89%**).

Poprawa wyników w ograniczonych zbiorach (2 i 3) wskazuje, że model lepiej radzi sobie w warunkach, gdzie predykcje są bardziej pewne lub gdy analizuje zawodników z wyższym rankingiem.

Wyniki potwierdzają skuteczność wprowadzenia ograniczeń w zbiorach testowych oraz wskazują na potencjalne zastosowanie modelu w scenariuszach, gdzie prognozowanie jest bardziej przewidywalne, np. w meczach zawodników z wyższych pozycji rankingowych.

Tab. 5.19. Najlepsze wyniki osiągnięte dla sieci neuronowej

Zbiór	Dokładność	Logarytmiczna funkcja straty	Wskaźnik Briera
1	0,64956	0,61906	0,21620
2	0,68557	0,57509	0,19748
3	0,68889	0,57245	0,19616

Tab. 5.20. Zestawienie rezultatów wszystkich modeli oraz kursów bukmacherskich

Zbiór	Model	Dokładność	Logarytmiczna funkcja straty	Wskaźnik Briera
1	**Regresja logistyczna**	0,64700	0,61976	0,21647
	XGBoost	0,64843	0,62137	0,21657
	Sieć neuronowa	0,64956	0,61906	0,21620
	Bukmacherzy	**0,69444**	**0,59048**	**0,20385**
2	**Regresja logistyczna**	**0,67753**	0,64271	0,21931
	XGBoost	**0,69398**	0,58816	0,20122
	Sieć neuronowa	**0,68557**	0,57509	0,19748
	Bukmacherzy	**0,69759**	**0,56373**	**0,19171**
3	**Regresja logistyczna**	**0,68889**	0,63318	0,21526
	XGBoost	**0,70216**	0,58670	0,20049
	Sieć neuronowa	**0,68889**	0,57245	0,19616
	Bukmacherzy	**0,69444**	**0,56625**	**0,19296**

5.10. Rezultaty

5.10.1. Porównanie modeli

W tabeli 5.20 zestawiono wyniki uzyskane dla trzech modeli: regresji logistycznej, XGBoost oraz sieci neuronowej, a także porównano je z wynikami uzyskanymi na podstawie kursów bukmacherskich. Najlepsze rezultaty dla każdego zbioru zostały pogrubione. Dodatkowo, w odniesieniu do dokładności, pogrubione zostały wartości, które nie różnią się istotnie statystycznie od najlepszego modelu w ramach danego zbioru, co zostało ocenione za pomocą z-testu dla dwóch niezależnych proporcji na poziomie istotności $\alpha = 0{,}05$.

Regresja logistyczna charakteryzowała się umiarkowaną skutecznością na wszystkich zbiorach. Najlepszy wynik osiągnięto na zbiorze **3**, gdzie dokładność wyniosła **68,89%**, a wskaźnik Briera **0,2153**. Niemniej jednak model ten był wyraźnie gorszy od pozostałych metod pod względem jakości predykcji.

XGBoost wykazał znaczącą poprawę względem regresji logistycznej, szczególnie na zbiorach

2 i **3**. Wskaźnik Briera dla zbioru **3** wyniósł **0,2005**, co wskazuje na lepsze dopasowanie predykcji modelu do rzeczywistych wyników. Model ten przewyższał regresję logistyczną we wszystkich miarach, ale pozostawał wciąż wyraźnie słabszy od wyników bukmacherów.

Sieć neuronowa osiągnęła najlepsze wyniki spośród trzech porównywanych modeli. Wskaźnik Briera dla zbioru **3** wyniósł **0,19616**, co stanowi najbliższy wynik w stosunku do bukmacherów (**0,19296**). Różnica między tymi wynikami jest stosunkowo niewielka (**0,00324**) i można ją uznać za satysfakcjonującą, zważywszy na fakt, że predykcje bukmacherów opierają się nie tylko na zaawansowanych modelach, ale również na danych rynkowych, takich jak ruchy kursów czy wiedza ekspercka. Sieć neuronowa zbliżyła się do wyników bukmacherów, co podkreśla jej potencjał w dalszym zastosowaniu.

5.10.2. Kalibracja najlepszego modelu

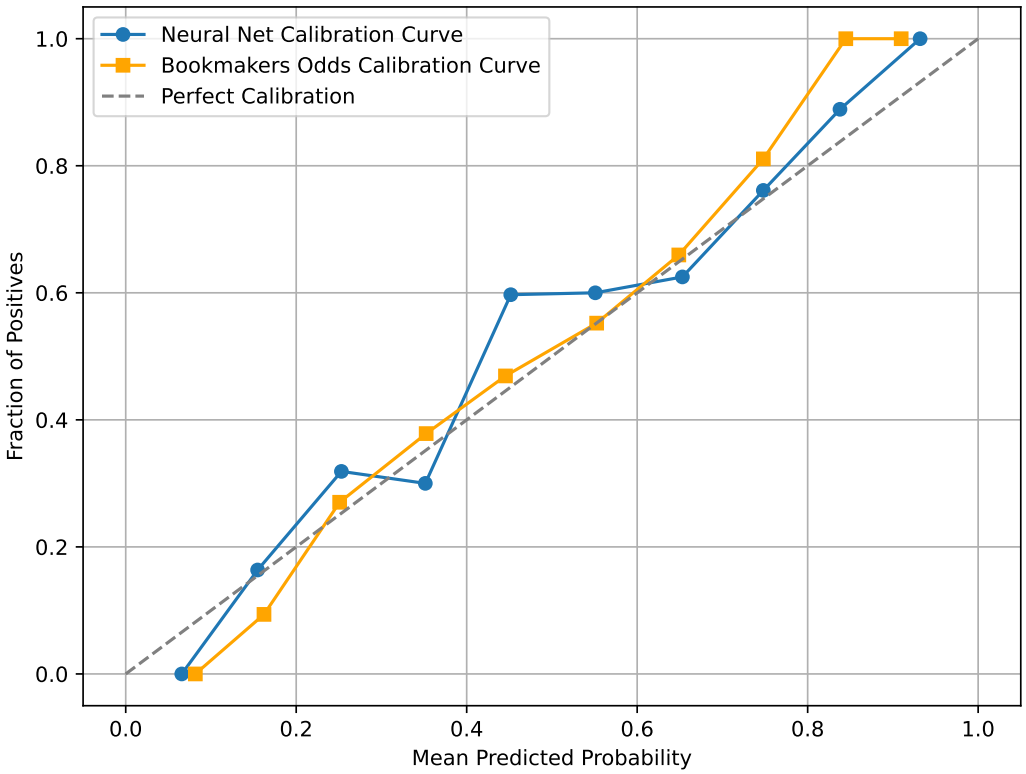
Aby dodatkowo ocenić jakość predykcji najlepszego modelu (sieci neuronowej), przeprowadzono analizę kalibracji, której wyniki przedstawiono na rysunku 5.22. Wykres kalibracji przedstawia zależność pomiędzy średnią przewidywaną wartością prawdopodobieństwa a rzeczywistą częstością pozytywnych zdarzeń w danych testowych. Idealnie skalibrowany model powinien znajdować się na linii „Perfect Calibration", która odpowiada sytuacji, w której przewidywane prawdopodobieństwa dokładnie odpowiadają rzeczywistym częstościom.

Analiza wykresu wskazuje, że model sieci neuronowej wykazuje dobrą kalibrację w dwóch kluczowych obszarach:

- Dla małych wartości przewidywanego prawdopodobieństwa (poniżej 0,2), gdzie krzywa modelu (niebieska linia) leży blisko idealnej linii kalibracji.
- W przedziałach wyższych przewidywań (powyżej 0,7), gdzie model również dobrze odzwierciedla rzeczywiste częstości zdarzeń.

Problematycznym obszarem jest jednak przedział przewidywanego prawdopodobieństwa w okolicach 0,5, gdzie krzywa modelu wyraźnie przekracza linię idealnej kalibracji. Oznacza to, że w tym zakresie model ma tendencję do przeszacowywania prawdopodobieństw, co może wynikać z niewystarczającej ilości informacji w cechach, przez co model przewiduje „bezpieczne" wartości w okolicy 50%.

Porównanie z krzywą kalibracji bukmacherów (żółta linia) wskazuje, że model bukmacherów w tym obszarze (około 0,5) jest bardziej stabilny, natomiast w wyższych przedziałach prawdopodobieństw (powyżej 0,7) oraz w niskich prawdopodobieństwach (poniżej 0,2), sieć neuronowa przewyższa bukmacherów pod względem kalibracji.

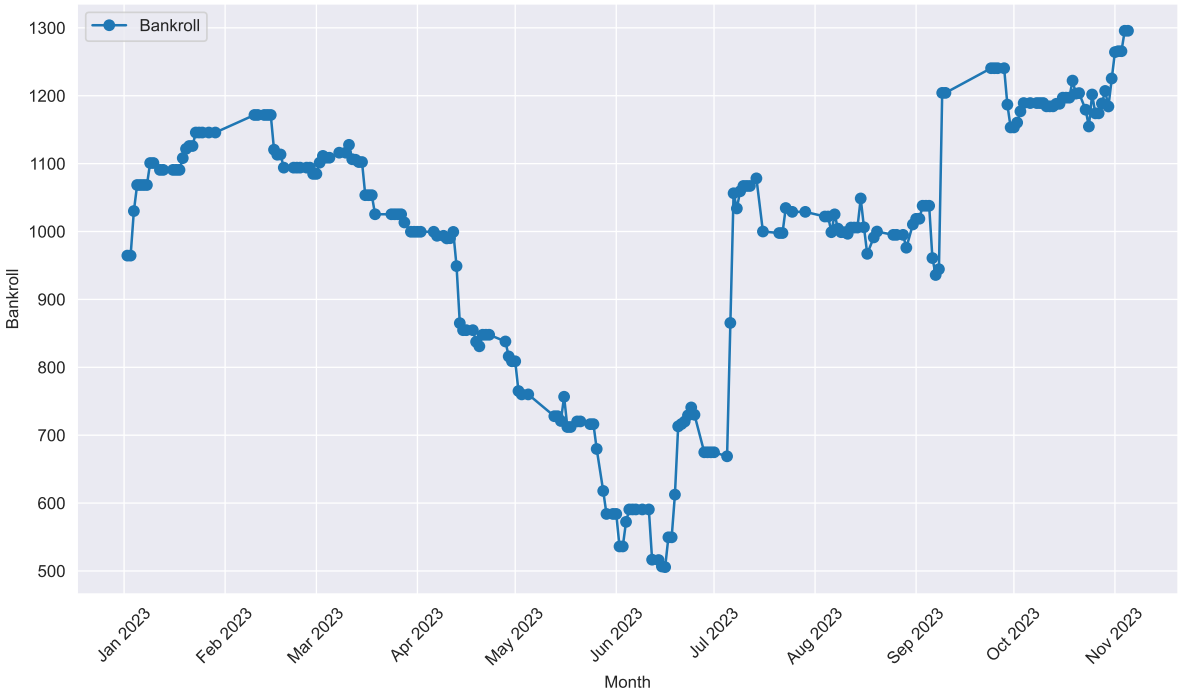


Rys. 5.22. Wykres kalibracji dla sieci neuronowej i przewidywań bukmacherów

5.10.3. Symulacja

Aby ocenić praktyczne zastosowanie modelu sieci neuronowej, przeprowadzono symulację obstawiania zakładów bukmacherskich na meczach ze **zbioru testowego 3** z wykorzystaniem prognoz generowanych przez model. Strategia obstawiania została oparta na kryterium Kelly'ego z maksymalną stawką zakładu ograniczoną do 200 jednostek. Wyniki symulacji przedstawiono na rysunku 5.23, który ilustruje zmiany kapitału w czasie oraz w tabeli 5.21.

Warto zaznaczyć, że ograniczenie maksymalnej stawki zakładu do 200 jednostek jest arbitralnym założeniem i wymagałoby dalszych badań w celu optymalizacji tego parametru. Ograniczenie to ma wpływ na dynamikę wzrostu funduszy oraz zmienia wartość końcowych rezultatów.



Rys. 5.23. Zmiany kapitału w czasie przy użyciu strategii Kelly

Tab. 5.21. Podsumowanie wyników zakładów

Parametr	**Wartość**
Początkowy budżet	1000
Całkowita suma stawek	5257
Liczba postawionych meczów	193
Zwrot z inwestycji (ROI)	5,62%
Procentowy zysk	29,57%

Zastosowanie modelu sieci neuronowej pozwoliło na osiągnięcie procentowego zysku na poziomie **29,57%** oraz zwrotu z inwestycji wynoszącego **5,62%**, co potwierdza wysoką skuteczność modelu w praktycznym zastosowaniu. Symulacja wykazała, że model jest zdolny do identyfikacji wartościowych zakładów, co przełożyło się na stabilny wzrost kapitału w dłuższym horyzoncie czasowym.

Należy jednak zauważyć, że symulacja została przeprowadzona na ograniczonym zbiorze testowym 3, co spowodowało, że liczba stawianych zakładów była stosunkowo niewielka. Mimo tego wysoki poziom ROI potwierdza skuteczność modelu, a zatem można przypuszczać, że w przyszłości, po rozwiązaniu problemów, które powodują potrzebę stworzenia ograniczonego zbioru testowego, potencjalna liczba obstawianych meczów mogłaby znacząco wzrosnąć. Większa liczba zakładów mogłaby przyczynić się do dalszego zwiększenia procentowego zysku.

Podsumowując, analiza zwrotu z inwestycji wskazuje na praktyczny potencjał zaproponowanego modelu sieci neuronowej w kontekście zakładów bukmacherskich. Wysoki poziom ROI potwierdza, iż prognozy modelu są skuteczne pod względem ich zastosowania w rzeczywistych warunkach, a dalszy rozwój i eliminacja ograniczeń zbioru testowego mogą prowadzić do jeszcze bardziej imponujących wyników.

6. Strona internetowa projektu

W ramach pracy stworzono stronę internetową z wykorzystaniem biblioteki **Django**, która umożliwia prezentację listy meczów ukończonych oraz zaplanowanych na dany dzień. Dla każdego meczu użytkownik może zobaczyć różne statystyki, zarówno dotyczące samego meczu, jak i statystyki przedmeczowe wykorzystywane przez model predykcyjny. Ponadto strona wyświetla prognozy wygenerowane przez model, pozwalając na sprawdzenie działania modelu na danych w żadnym stopniu niedostępnych wcześniej oraz na potencjalne wykorzystanie predykcji przez użytkowników.

Aby strona mogła wyświetlać nadchodzące mecze wraz z aktualnymi statystykami i wiarygodnymi predykcjami modelu, konieczne było zaprojektowanie zaawansowanej architektury przepływu danych. Głównym wyzwaniem było zapewnienie, że wszystkie dane potrzebne do działania modelu będą na bieżąco aktualizowane i dostępne w czasie rzeczywistym. W tym celu konieczne było opracowanie nowego systemu zbierania, przetwarzania i udostępniania danych.
Ponieważ sieć neuronowa uzyskała najlepsze wyniki, zdecydowano wykorzystać jej predykcje do prezentacji na stronie.

6.1. Zbieranie danych

W przeciwieństwie do wcześniejszych etapów pracy, gdzie korzystano z gotowych, archiwalnych zbiorów danych do analizy i trenowania modelu, konieczne było stworzenie mechanizmu pozwalającego na pozyskiwanie informacji o nadchodzących meczach oraz ich aktualnych statystykach. Rozwiązaniem tego problemu było stworzenie skryptu, który automatycznie pobiera dane ze strony internetowej.

Skrypt ten został zaprojektowany w taki sposób, aby zbierać wszystkie informacje, które były dostępne w archiwalnych zbiorach danych, takie jak:

- statystki meczowe (np. liczba asów, procent trafionego pierwszego serwisu),
- informacje o zawodnikach (np. wiek, kraj pochodzenia),

- dane o turnieju (np. lokalizacja).

6.2. Procesowanie danych

Zebrane dane wymagały odpowiedniego przetworzenia, aby ich struktura odpowiadała tej, która była używana w archiwalnych zbiorach danych. W tym celu stworzono dedykowany skrypt, który automatycznie przekształca zebrane informacje do odpowiedniego formatu. Taka transformacja danych była konieczna, aby zapewnić kompatybilność z funkcjami i metodami przetwarzania danych wykorzystywanymi na wcześniejszych etapach pracy.

6.3. Iteracyjne tworzenie cech

Po przetworzeniu danych pozyskanych za pomocą skrobania do formatu zgodnego z oryginalnymi zbiorami danych kolejnym krokiem było iteracyjne tworzenie cech. Proces ten wymagał modyfikacji istniejących funkcji odpowiedzialnych za generowanie cech, aby dostosować je do pracy w dynamicznym środowisku, gdzie dane są aktualizowane na bieżąco.

W dotychczasowym podejściu funkcje tworzące cechy działały na pełnej ramce danych, obliczając nowe kolumny (cechy) na podstawie dostępnych informacji w tej ramce. Jednak w nowym podejściu, z uwagi na konieczność uwzględnienia danych historycznych (od roku 2018), funkcje te zostały zmodyfikowane, aby obsługiwały dwa zestawy danych:

- **Istniejącą ramkę danych historycznych:** Zawiera wszystkie mecze od roku 2018, w tym również wcześniej obliczone cechy.
- **Nowe dane meczowe:** Zawierają dane dla najnowszych meczów pozyskane ze strony internetowej.

Funkcje tworzące cechy zostały tak zaprojektowane, aby obliczać nowe kolumny na podstawie informacji dostępnych zarówno w nowych meczach, jak i w pełnej historii meczów. Po obliczeniu cech dla nowych danych ramka historyczna jest aktualizowana poprzez dodanie nowych meczów wraz z uzupełnionymi kolumnami cech. Dzięki temu proces ten ma charakter iteracyjny i umożliwia dynamiczne rozszerzanie pełnej historii danych wraz z pojawieniem się nowych meczów. To podejście pozwoliło również zachować pełną spójność między historycznymi i nowymi danymi oraz zapewnić, że wszystkie cechy są obliczane w sposób jednolity.

Aby zrealizować iteracyjny proces przetwarzania danych, opracowano dedykowaną klasę `IterativeDataFrameProcessor` (rys. 6.1). Procesor ten umożliwił dynamiczne obliczanie no-

```
class IterativeDataFrameProcessor:
    def __init__(self, initial_df, feature_functions):
        # Inicjalizacja procesora z ramk  danych historycznych
        # oraz list  funkcji tworz cych cechy.
        self.df = initial_df
        self.feature_functions = feature_functions

    def calculate_new_columns(self, new_rows):
        # Tworzenie nowych kolumn dla nowych wierszy.
        for func in self.feature_functions:
            new_rows = func(new_rows, self.df)
        return new_rows

    def add_rows(self, new_rows):
        # Przetwarzanie nowych wierszy i aktualizacja historii.
        processed_rows = self.calculate_new_columns(new_rows)

        # Aktualizacja istniej cych wierszy.
        rows_to_update = processed_rows[
        processed_rows['match_id'].isin(self.df['match_id'])
        ]
        self.df.update(rows_to_update)

        # Dodanie nowych wierszy do ramki historycznej.
        rows_to_append = processed_rows[
        ~processed_rows['match_id'].isin(self.df['match_id'])
        ]
        self.df = pd.concat([self.df, rows_to_append], ignore_index=True)
```

Rys. 6.1. Pseudokod iteracyjnego procesora danych

wych cech na podstawie zarówno danych historycznych, jak i nowych wierszy danych. Klasa została zaprojektowana w sposób modułowy, co pozwala na łatwe dodawanie funkcji tworzących cechy oraz zarządzanie zbiorami danych.

6.4. Przygotowanie danych do predykcji i integracja z bazą danych Django

Po iteracyjnym procesie dodawania cech, zaktualizowane dane przechodzą przez kolejne etapy przetwarzania, które przygotowują je do wygenerowania predykcji modelu.

6.4.1. Przygotowanie danych do modelu

Nowe dane, wzbogacone o cechy dodane w poprzednim kroku, trafiają do skryptu przygotowującego je do pracy z modelem predykcyjnym. Proces ten obejmuje szereg transformacji, które opisane są w podrozdz. 5.2.

6.4.2. Transformacje dla modelu i generowanie predykcji

Dane przygotowane w poprzednim kroku są następnie poddawane dalszym transformacjom w kolejnym skrypcie. Obejmuje to wszystkie transformacje używane na danych podczas konstruowania modeli, takie jak skalowanie zmiennych, czy np. tworzenie tensorów dla sieci neuronowej.
Następnie, wytrenowany wcześniej model jest wczytywany z pliku w formacie `.pth`. Model ten wykorzystuje przetworzone dane do wygenerowania predykcji dla nowych meczów. Wynikiem tego etapu jest ramka danych, zawierająca predykcje dla każdego meczu.

6.4.3. Tworzenie ramek do bazy danych Django

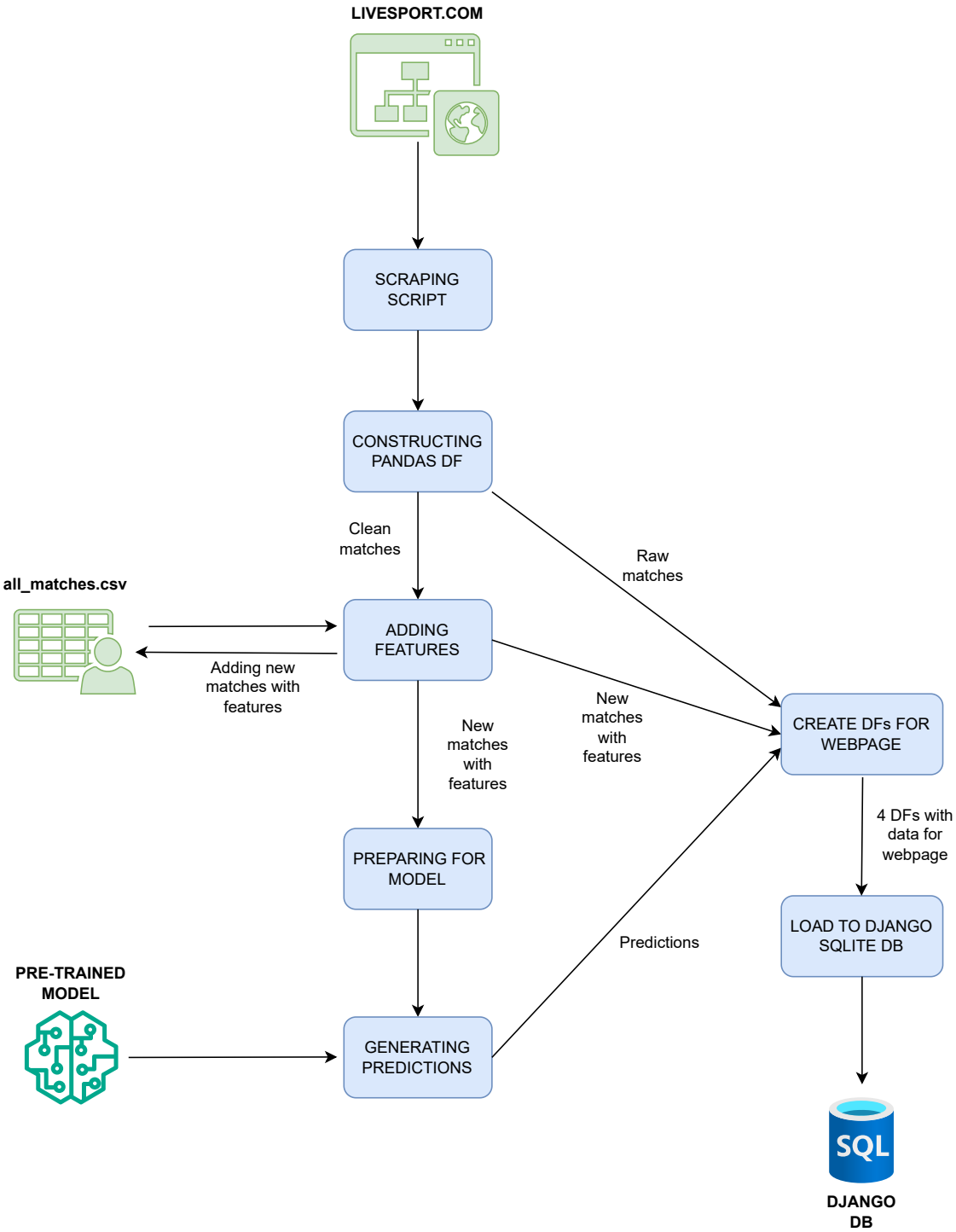
W ostatnim kroku, wygenerowane w poprzednich etapach ramki danych są przekazywane do kolejnego skryptu, który przygotowuje je do załadowania do bazy danych Django. Proces ten obejmuje:

- **Wybór odpowiednich kolumn:** Z ramki danych wybierane są tylko te kolumny, które mają być wyświetlane na stronie internetowej.
- **Podział na tabele:** Dane są rozdzielane na różne tabele w zależności od ich przeznaczenia, takie jak tabela meczów, statystyk przedmeczowych czy zawodników.

Po przetworzeniu danych ramki są eksportowane do bazy danych Django za pomocą specjalnej komendy, która ładuje dane do systemu. Dzięki temu dane są gotowe do wyświetlenia na stronie internetowej w przejrzystej formie.

6.5. Całość procesu

Wszystkie opisane powyżej kroki, od pozyskania danych poprzez generowanie predykcji, aż po integrację z bazą danych Django, tworzą złożony przepływ danych. Przepływ ten umożliwia bieżącą aktualizację informacji na stronie internetowej, zapewniając jednocześnie spójność między danymi historycznymi a nowymi informacjami generowanymi przez model predykcyjny.

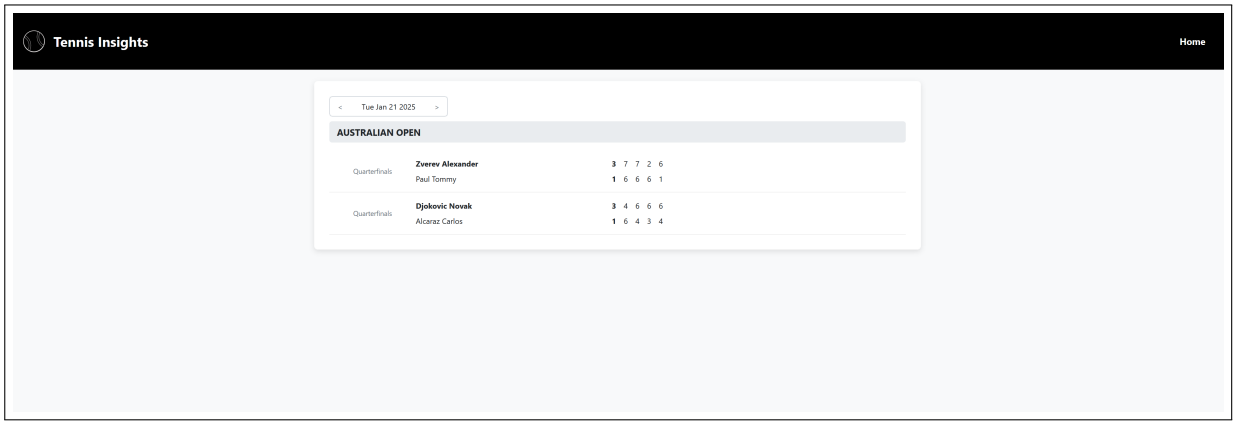


Rys. 6.2. Wizualizacja przepływu danych od źródła do strony internetowej

6.6. Podręcznik użytkownika

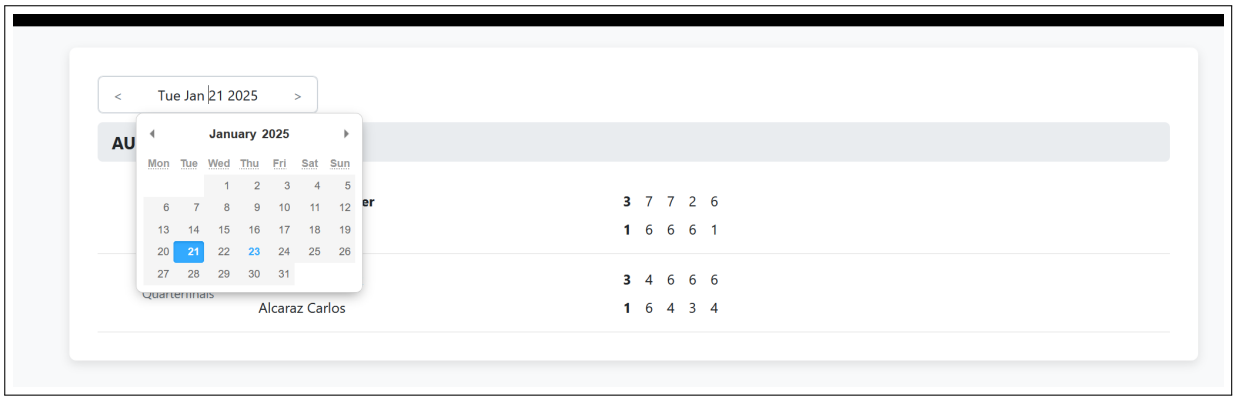
Obecnie aplikacja działa wyłącznie lokalnie, co oznacza, że jest dostępna tylko na komputerze, na którym została uruchomiona. W związku z tym dane widoczne w interfejsie odpowiadają stanowi z momentu ostatniego pakowania plików, co oznacza, że nie będą one automatycznie aktualizowane w czasie rzeczywistym bez wykonania odpowiednich skryptów.

Po uruchomieniu aplikacji i otworzeniu w przeglądarce wskazanego linku `http://127.0.0.1:8000/stats/results` użytkownik znajduje się w interfejsie głównym, gdzie może zobaczyć rezultaty meczów, jakie odbyły się (bądź odbędą się) danego dnia z podziałem na turnieje wraz z etapem turnieju (rys. 6.3).



Rys. 6.3. Strona główna

Użytkownik może wybrać interesującą go datę poprzez klikanie strzałek, wtedy cofnie się lub pójdzie do przodu o jeden dzień, lub najpierw klikając w przycisk, który wyłania kalendarz, a potem klikając na wybrany dzień (rys. 6.4).



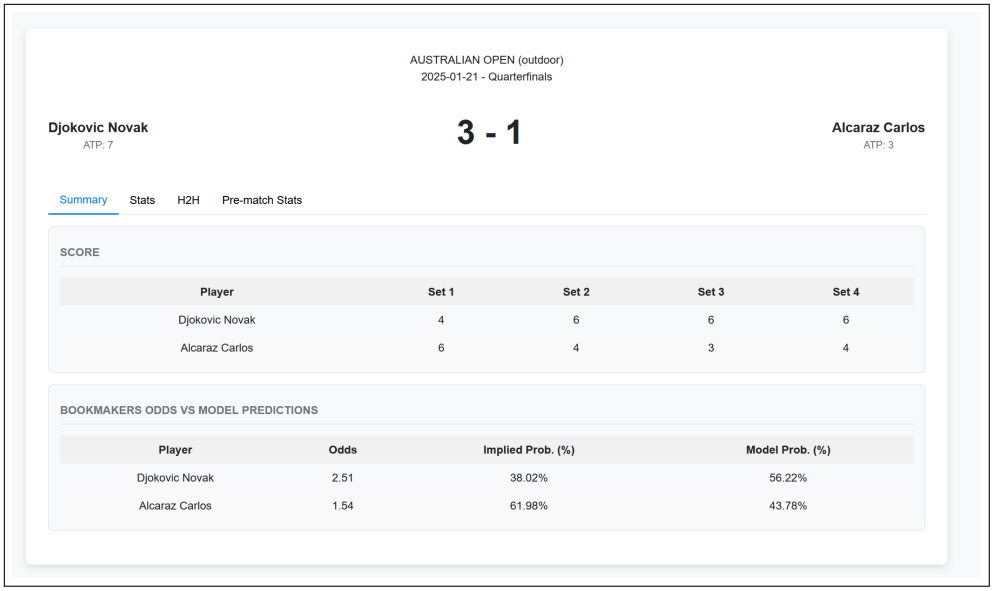
Rys. 6.4. Wybór daty

Użytkownik może następnie, poprzez kliknięcie na interesujący go mecz, przejść do strony ze szczegółowymi statystykami meczu oraz predykcjami modeli.

Strona meczowa ma do wyboru kilka zakładek:

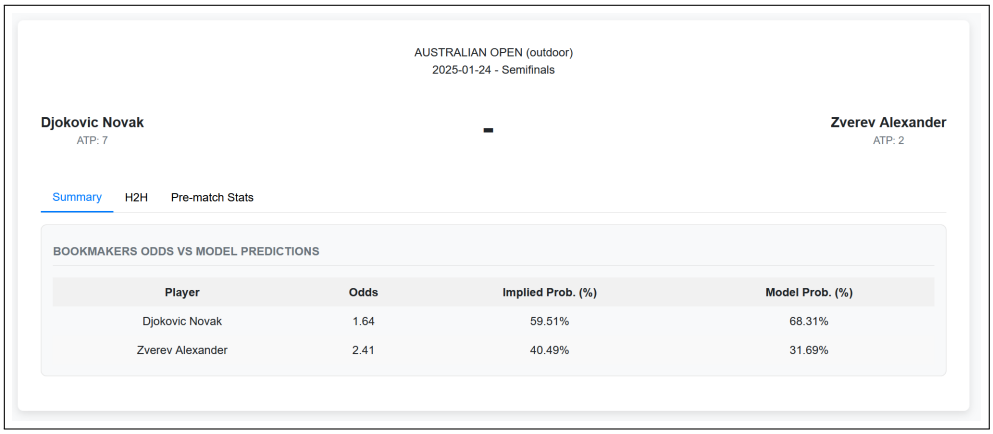
1. Summary

Jest to domyślna zakładka po wybraniu meczu interesującego użytkownika. Dla meczu, który się już odbył, widnieje wynik meczu wraz ze szczegółowym podziałem na sety oraz sekcja, gdzie można zobaczyć, jakie prawdopodobieństwo wynikało z kursów bukmacherskich oraz jakie prawdopodobieństwo przewidział model (rys. 6.5).



Rys. 6.5. Zakładka „Summary" dla zakończonego meczu

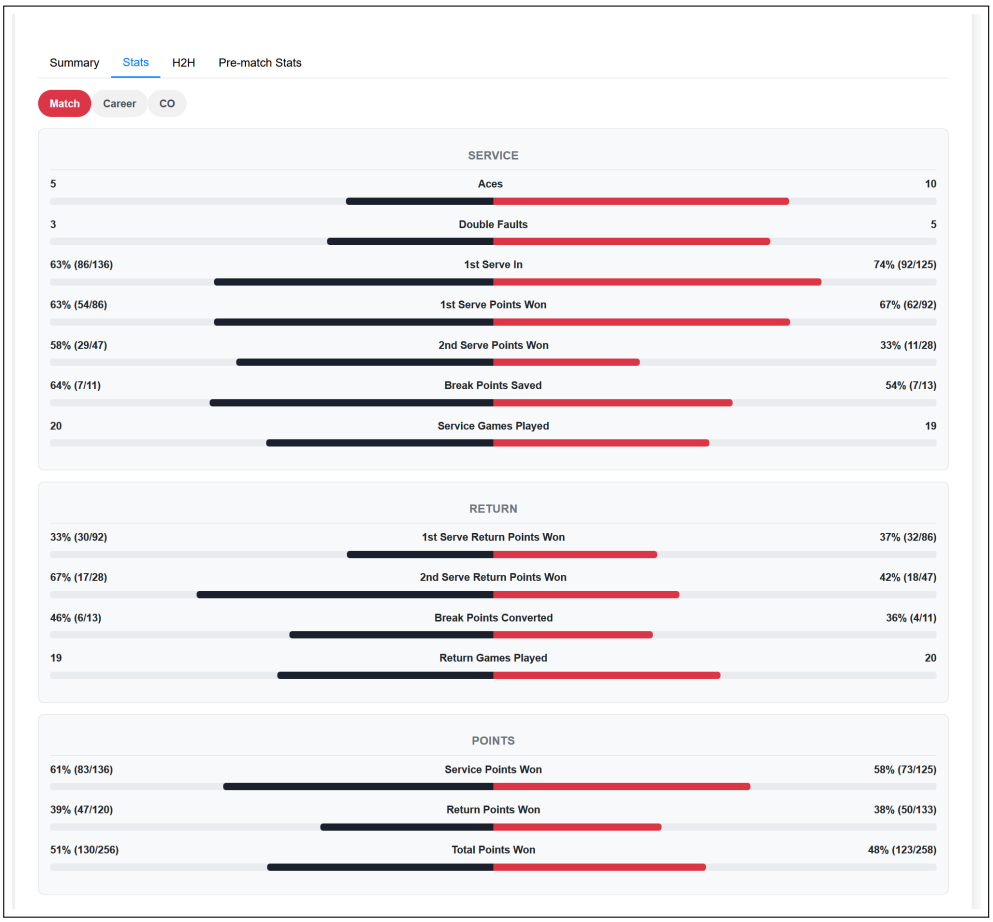
Jeżeli wybrano mecz, który się jeszcze nie zakończył, z oczywistych powodów w zakładce nie ujrzymy informacji dotyczących wyników. Nadal dostępne za to będą przewidywania modelu oraz bukmacherów (rys. 6.6).



Rys. 6.6. Zakładka „Summary" dla niezakończonego meczu

2. Stats

W zakładce „Stats” znajdują się szczegółowe statystyki dla danego meczu pod warunkiem, że ten już się odbył (rys. 6.7).



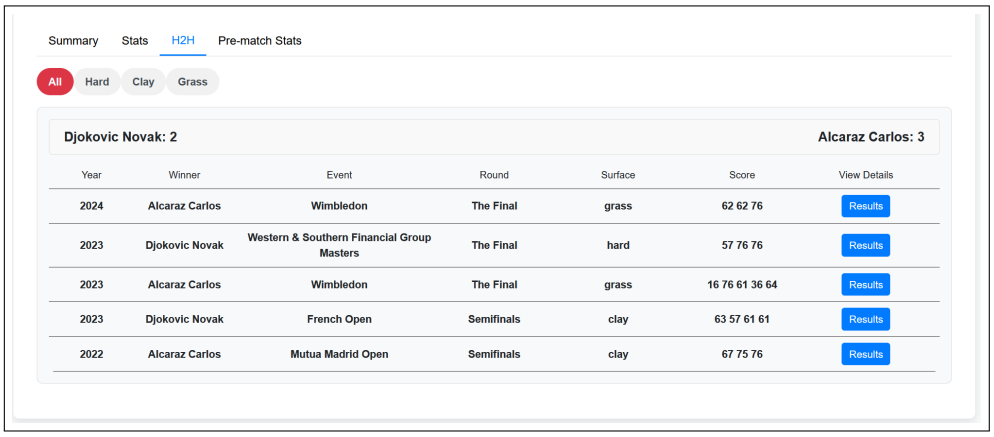
Rys. 6.7. Zakładka „Stats-Match”

Dostępne są również dwie inne opcje „Career”, która przedstawia zagregowane statystyki każdego z zawodników jako średnie wartości na podstawie całej dotychczasowej kariery oraz „CO”, która dodatkowo używa agregacji z użyciem metody Wspólnych Przeciwników (odpowiadają one „stat avg” i „CO stat avg” w (4.25)). Wygląd zakładki przedstawiono na rysunku 6.8.

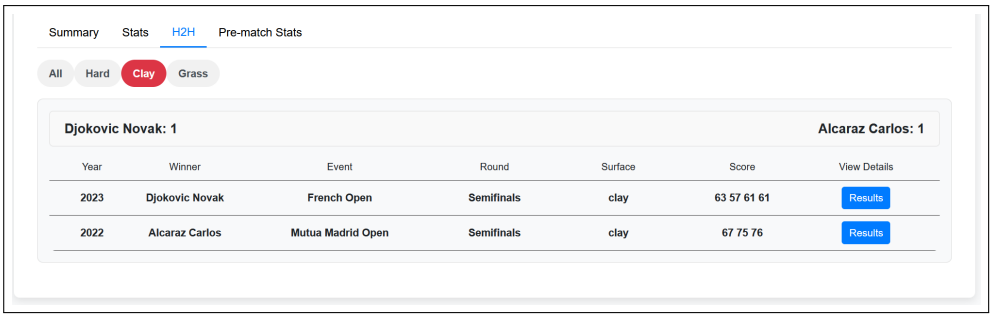
Rys. 6.8. Zakładka „Stats-Career”

3. H2H

W zakładce „H2H” znajdują się rezultaty wszystkich meczów pomiędzy danymi zawodnikami oraz łączny bilans. Użytkownik może wyświetlić wszystkie rozegrane spotkania lub filtrować mecze ze względu na nawierzchnię, na której były rozgrywane (rys. 6.9, rys. 6.10).



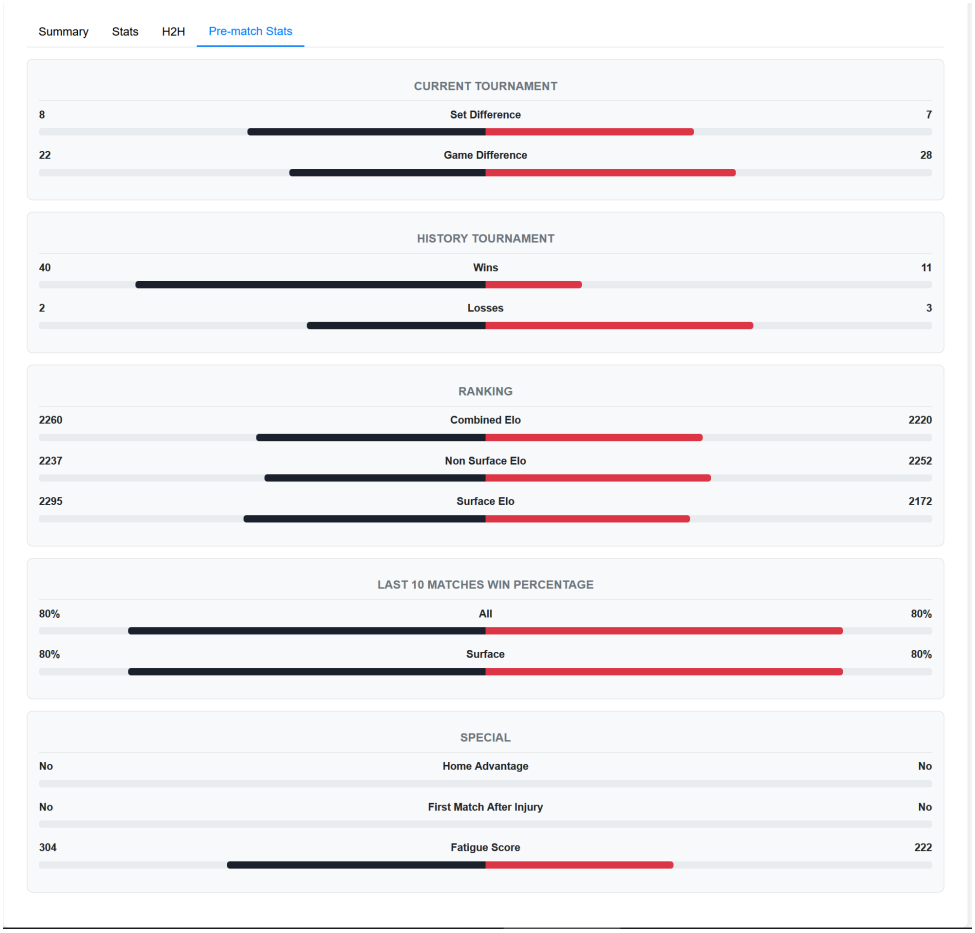
Rys. 6.9. Zakładka „H2H” z wyborem wszystkich meczów



Rys. 6.10. Zakładka „H2H" z wyborem meczu rozgrywanych na mączce

4. Pre-match Stats

W tej zakładce, przedstawionej na rysunku 6.11, znajdują się szczegółowe przedmeczowe statystyki dla danego meczu, przedstawiające ostatnią formę obydwu zawodników oraz ich punkty rankingu Elo. Dodatkowo użytkownik może się dowiedzieć, czy któryś z zawodników rozgrywał mecz w kraju, z którego pochodzi, czy jest to pierwszy mecz po powrocie z kontuzji oraz jaki jest wskaźnik zmęczenia każdego z zawodników (wszystkie te statystyki odpowiadają cechom stworzonym w rozdz. 4).



Rys. 6.11. Zakładka „Pre-match Stats"

7. Podsumowanie i wnioski

Projekt skoncentrował się na opracowaniu modelu predykcyjnego do prognozowania wyników meczów tenisowych, z wykorzystaniem różnych algorytmów uczenia maszynowego. Przeprowadzone testy wykazały, że sieć neuronowa osiągnęła najlepsze wyniki w porównaniu do innych metod, co świadczy o jej wysokiej efektywności w tego rodzaju zadaniach predykcyjnych.

Przeprowadzona symulacja osiągnęła zwrot z inwestycji (ROI) na poziomie **5,62%** oraz procentowy zysk wynoszący **29,57%**, co potwierdza praktyczną wartość opracowanego podejścia w obszarze analizy sportowej.

Dodatkowo stworzono platformę internetową, która w prosty i przystępny sposób prezentuje prognozy modelu oraz kluczowe statystyki związane z meczami i zawodnikami. Dzięki zastosowaniu metod pozyskiwania danych w czasie rzeczywistym i ich przetwarzania, system jest w stanie dostarczać wartościowych informacji użytkownikom.

Projekt pokazuje, że połączenie skutecznych algorytmów predykcyjnych z intuicyjną aplikacją internetową stanowi mocną podstawę dla rozwoju w dziedzinie analizy danych sportowych. W przyszłości warto kontynuować rozwój modelu, testując dodatkowe techniki uczenia maszynowego oraz rozszerzając funkcjonalności platformy o nowe opcje analizy i wizualizacji danych.

Przyszłe prace

Wytworzenie bardziej złożonych cech zawodnika

Jednym z potencjalnych kierunków poprawy działania modelu jest stworzenie złożonych, pochodnych cech, które lepiej oddają dynamikę meczu tenisowego. Na przykład, zamiast wykorzystywać surowe statystyki serwisowe (takie jak procent pierwszego serwisu, liczba asów czy podwójnych błędów), można rozważyć stworzenie złożonych metryk, takich jak „jakość serwisu". Taka cecha mogłaby łączyć różne statystyki związane z serwisem, dostarczając bardziej spójnej informacji na temat skuteczności tego elementu gry.

Podobne podejście mogłoby zostać zastosowane do innych aspektów meczu, takich jak gra przy siatce, skuteczność returnu czy odporność psychiczna zawodników, co mogłoby prowadzić do bardziej precyzyjnych prognoz.

Modyfikacja modelu XBGoost

Zmodyfikowanie modelu XGBoost w taki sposób, aby spełniał on założenia o symetryczności, potencjalnie mogłoby w znaczący sposób poprawić jego działanie, wymaga to niestety jednak dużych zmian w kodzie dostępnego modelu lub nawet implementacji własnego modelu „od zera".

Uwzględnienie turniejów niższych rang

Innym ważnym kierunkiem przyszłych badań jest pozyskanie danych z turniejów niższej rangi niż ATP, takich jak turnieje Challenger lub Futures. Obecny zbiór danych obejmuje jedynie mecze z turniejów ATP, co powoduje brak informacji o wynikach zawodników w niższych klasach rozgrywek. Jest to szczególnie problematyczne, ponieważ wielu graczy z rankingu ATP regularnie uczestniczy w takich turniejach, a brak tych danych może wpływać na niepełny obraz ich formy i możliwości.

Konsekwencją tego ograniczenia było konieczność rozpatrywania wyników na zbiorze testowym ograniczonym do zawodników z rankingiem poniżej **50**, ponieważ gracze z tej grupy rzadko uczestniczą w turniejach niższej rangi, co minimalizuje problem brakujących danych. Rozszerzenie zbioru danych o wyniki z turniejów niższej rangi mogłoby znacząco zwiększyć zakres analiz i poprawić jakość predykcji dla zawodników spoza czołówki rankingu.

Rozwój strony internetowej

Kolejnym obszarem rozwoju projektu jest ulepszenie strony internetowej zarówno pod względem wizualnym, jak i funkcjonalnym. Aktualna wersja strony skupia się na prezentacji wyników modeli oraz statystyk meczowych, jednak istnieje potencjał do dalszego rozwoju w następujących kierunkach:

Poprawa wizualna: Projekt strony mógłby zostać wzbogacony o bardziej nowoczesny i atrakcyjny wizualnie interfejs użytkownika, co uczyniłoby ją bardziej przejrzystą i przyjazną w użytkowaniu.

Nowa funkcjonalność: Ważnym elementem dalszego rozwoju byłoby dodanie funkcji rekomendacji zakładów. Funkcja ta, bazując na predykcjach modelu oraz optymalnej strategii obstawiania, mogłaby sugerować użytkownikowi, czy i ile postawić na danego zawodnika. Mechanizm ten mógłby uwzględniać następujące elementy:

- budżet użytkownika, który byłby podawany jako dane wejściowe,
- kursy bukmacherskie na mecz,

- wyliczone prawdopodobieństwa wyniku na podstawie modelu,
- optymalną strategię obstawiania (np. wykorzystanie strategii Kelly'ego).

Przykładowo, po podaniu budżetu i wprowadzeniu kursów bukmacherskich, strona mogłaby generować rekomendację, że użytkownik powinien postawić określoną kwotę na jednego z zawodników, maksymalizując potencjalny zwrot z inwestycji przy jednoczesnym ograniczeniu ryzyka.

Wystawienie strony na serwer i przejście na API: Obecnie strona działa wyłącznie lokalnie, co ogranicza jej dostępność do jednego urządzenia. Aby umożliwić szerszy dostęp do wyników modeli i statystyk, należałoby wdrożyć aplikację na serwerze, np. za pomocą platformy AWS lub serwera VPS. Wdrożenie wymagałoby dostosowania konfiguracji bazy danych oraz procesu regularnej aktualizacji danych wejściowych, aby strona prezentowała najnowsze informacje o meczach.

Ponadto warto rozważyć przejście z obecnej metody skrobania stron internetowych na korzystanie z oficjalnego tenisowego API, które dostarcza rzetelne i aktualne dane o meczach, statystykach i kursach bukmacherskich. Takie rozwiązanie byłoby bardziej niezawodne i mniej podatne na błędy związane ze zmianami struktury stron internetowych. Jednak zarówno hosting aplikacji na platformie chmurowej, jak i dostęp do profesjonalnych API wiąże się z dodatkowymi kosztami. W związku z tym wdrożenie tego rozwiązania wymagałoby pewnej inwestycji finansowej, ale w dłuższej perspektywie znacząco zwiększyłoby stabilność i efektywność systemu.

Bibliografia

[1] T. Barnett, G. Pollard. How the tennis court surface affects player performance and injuries. *Medicine and Science in Tennis*, 12(1):34–37, 2007.

[2] G. Berthelot, S. Len, P. Hellard, i in. Exponential growth combined with exponential decline explains lifetime performance evolution in individual and human species. *AGE*, 34:1001–1009, 2012.

[3] A. De Seranno, T. De Pessemier, L. Martens. Predicting tennis matches using machine learning. *Master of Science in Computer Science Engineering*, strony 26–27, 2020.

[4] . Lj. Djordjević, M. I Jordović-Pavlović, Ž.M. Ćojbašić, S.P. Galović, M. N. Popović, M. V Nešić, D. D. Markushev. Influence of data scaling and normalization on overall neural network performances in photoacoustics. *Optical and Quantum Electronics*, 54(8):501, 2022.

[5] Arpad E Elo, Sam Sloan. The rating of chessplayers: Past and present. *(No Title)*, 1978.

[6] John L Kelly. A new interpretation of information rate. *the bell system technical journal*, 35(4):917–926, 1956.

[7] W. J. Knottenbelt, Demetris S., A. M. Madurska. A common-opponent stochastic model for predicting the outcome of professional tennis matches. *Computers & Mathematics with Applications*, 64(12):3820–3827, 2012.

[8] M. Mlakar, T. Tušar. Analyzing and predicting peak performance age of professional tennis players. *Proceedings of the 13th International Symposium on Operational Research*, strony 99–104, 2015.

[9] A. Painsky, G. Wornell. On the universality of the logistic loss function. *2018 IEEE International Symposium on Information Theory (ISIT)*, strony 936–940. IEEE, 2018.

[10] M. Sipko, W. Knottenbelt. Machine learning for the prediction of professional tennis matches. *MEng computing-final year project, Imperial College London*, 2, 2015.

[11] W. Zhang, C. Wu, H. Zhong, Y. Li, L. Wang. Prediction of undrained shear strength using extreme gradient boosting and random forest based on bayesian optimization. *Geoscience Frontiers*, 12(1):469–477, 2021.