AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

WYDZIAŁ INFORMATYKI, ELEKTRONIKI I TELEKOMUNIKACJI

Projekt dyplomowy

Aplikacja oparta o uczenie maszynowe do analizy wyników skoków narciarskich

Machine learning-based application to analyze ski jumping results

Autor: *Justyna Gręda*
Kierunek studiów: *Teleinformatyka*
Opiekun pracy: *dr inż. Jarosław Bułat*

Kraków, 2022

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Spis treści

Wstęp

Skoki narciarskie, pomimo niewątpliwej popularności w Europie Środkowej są mało rozpoznawalną dyscypliną sportu na skalę światową. Z tego względu, zasób analiz tej konkurencji pod kątem uczenia maszynowego jest bardzo niewielki. We wcześniejszych latach zespół z Japonii podjął próbę skonstruowania algorytmu do automatycznej oceny stylu zawodnika [5]. Wykorzystali oni w tym celu dane zgromadzone z czujników umieszczonych na ciele kilku skoczków i porównywali je z opinią niezależnego sędziego. Zgromadzone rezultaty były zadowalające i otworzyły furtkę do dalszych badań nad tematem ewaluacji stylu skoczków narciarskich bez obecności arbitrów.

Celem aplikacji jest predykcja łącznej oceny za styl, jaką otrzymał skoczek narciarski na podstawie pozostałych parametrów oddanego skoku - jego długości, rekompensaty za wiatr, belki startowej, punktów przyznanych za odległość i belkę oraz prędkości wiatru i wybicia z progu. Sędziowie oceniając próbę zawodnika zwracają uwagę na jego postawę w trakcie lotu i lądowania. Pozostałe parametry skoku nie są im znane, nie uzależniają zatem od nich swojej oceny, jest ona ściśle subiektywna. Ponadto, skok musi zostać oceniony tuż po jego zakończeniu, nie jest możliwe obejrzenie powtórki czy zmiana przyznanej noty. Dzięki wykorzystaniu uczenia maszynowego można przeanalizować dziesiątki tysięcy skoków i na ich podstawie przewidzieć oceny za styl następnych zawodników. Pozwoliłoby to zrezygnować z dokonywania ewaluacji przez sędziów, a co za tym idzie, uniknąć stronniczych sytuacji, w której arbiter przyznaje wyższe noty swoim rodakom.

W ramach projektu przygotowano trzy rozwiązania umożliwiające rozstrzygnięcie problemu predykcji noty skoczka narciarskiego. Dwa z nich oparte są na uczeniu maszynowym z wykorzystaniem sieci neuronowych, trzecie stanowi prostsze podejście do tematu z pomocą regresji liniowej. Po ich analizie i wybraniu najdokładniejszego automatycznego sędziego zostanie on użyty do przewidywania wyników zawodów, co będzie stanowić weryfikację jego działania.

W pierwszym rozdziale przedstawiono zagadnienie predykcji w uczeniu maszynowym od strony teoretycznej, a także wyjaśniono zasady oceniania obowiązujące w skokach narciarskich. Drugi rozdział opisuje przygotowanie danych do wprowadzenia ich do modeli, ich normalizację oraz korelację pomiędzy parametrami skoku narciarskiego. W trzecim rozdziale omawiana jest implementacja trzech modeli uczenia maszynowego, które posłużą do predykcji noty zawodnika. Czwarty rozdział przedstawia analizę zaimplementowanych rozwiązań celem wybrania najlepszego z nich, a także jego zastosowanie w przewidywaniu wyników konkursu i wykrywaniu upadków.

1. Omówienie problemu

Zaprojektowana aplikacja stanowi połączenie obszaru uczenia maszynowego ze światem sportu. Do poprawnej analizy zagadnienia od strony technicznej, niezbędne są także wiadomości teoretyczne dotyczące skoków narciarskich, a dokładnie sposobu oceny stylu zawodnika. Poniższy rozdział przybliża tematykę zawodów narciarskich, a także omawia zagadnienia uczenia maszynowego, które będą wykorzystywane podczas budowy i analizy modelu.

1.1. Zasady oceniania skoków narciarskich

Łączna ocena skoku zawodnika składa się z kilku elementów. Najbardziej znaczącymi składowymi noty są punkty przyznane za uzyskaną odległość oraz ocena sędziowska [25]. W celu zagwarantowania jak najbardziej wyrównanych warunków każdemu ze skoczków, wprowadzono także rekompensatę za wiatr (tj. ujemne punkty w przypadku zbyt korzystnych warunków i dodatnie przy niekorzystnym wietrze) oraz za wydłużony lub skrócony rozbieg.

Ocena sędziowska

Łączna nota sędziowska danego zawodnika nie może przekroczyć 60 punktów. Składają się na nią oceny pięciu sędziów w skali od 0 do 20 punktów (co 0.5 punktu), przy czym dwie skrajne noty, najwyższa i najniższa są odrzucane. Ewaluacji podlegają dwie ostatnie fazy skoku, lot oraz lądowanie i odjazd (oceniane osobno) według następujących kryteriów:

- **Lot**

  W tej fazie zawodnik powinien gwałtownie wybić się z progu, przejść szybko do optymalnej dla lotu pozycji, a w końcowym etapie rozpocząć przygotowania do lądowania. Sędziowie oceniają czy ciało zawodnika wraz z jego nartami tworzy spójny układ, symetrię ułożenia nart i kończyn, a także czy nogi skoczka są całkowicie proste. Za błędy popełnione w fazie skoku zawodnik może utracić maksymalnie **pięć punktów**.

- **Lądowanie**

  Zawodnik podczas lądowania powinien ustawić narty równolegle, ugiąć kolana i płasko wylądować w pozycji telemarku. Wśród aspektów lądowania ocenianych przez sędziów znajduje się

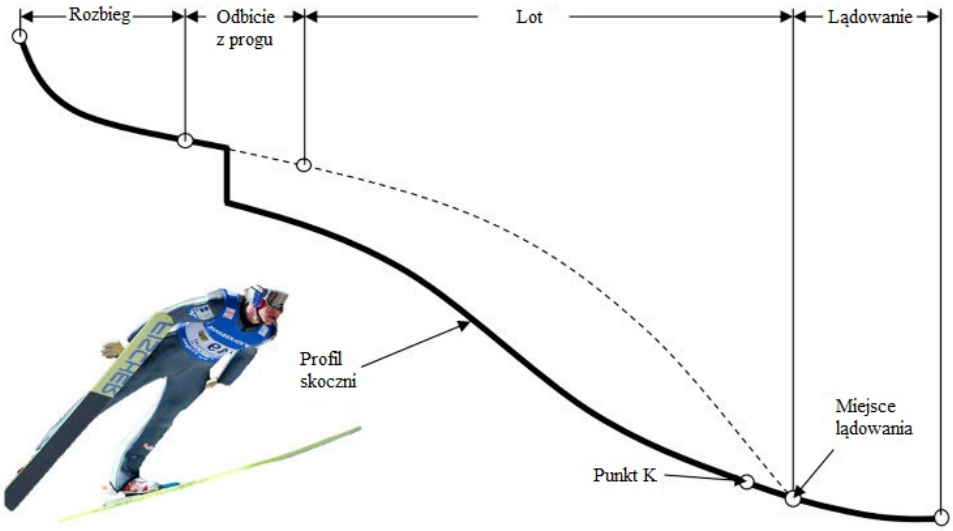
płynne przejście z pochylonej sylwetki w fazie lotu do wyprostowanej, ugięte i rozłączone kolana w momencie lądowania, gładkość lądowania wraz z redukcją prędkości zawodnika, poprawny telemark, a także równoległe ułożenie nart i zachowanie między nimi odległości mniejszej od ich dwukrotnej szerokości. Za błędy popełnione w czasie lądowania zawodnik może utracić maksymalnie **siedem punktów**, z czego dwa są odejmowane w przypadku braku telemarku.

– **Odjazd**

To faza skoku, w której zawodnik powinien pozostać w poprawnej pozycji telemarku aż do dotarcia do równej części zeskoku. Jego zadaniem jest też w stabilnej, ale zrelaksowanej pozycji minąć linię upadku (ang. *fall line*). Sędziowie oceniają, czy skoczek po wylądowaniu utrzymał telemark przez 10 – 15 sekund, jego narty były ułożone w pozycji równoległej z zachowaniem odległości mniejszej od ich dwukrotnej szerokości, a także ukończenie skoku w stabilnej pozycji z równym obciążeniem obu nóg. Za błędy popełnione w fazie odjazdu zawodnik może utracić maksymalnie **siedem punktów**, z czego pełne siedem jest odejmowane za przewrócenie się przed linią upadku. Między pół punktu a trzy punkty może zostać odjęte zawodnikowi za niepoprawną pozycję ciała w ostatniej fazie skoku. Za dotknięcie ręką zeskoku lub nart skoczkowi może zostać odjęte od czterech do pięciu punktów.

Punkty za uzyskaną odległość

Ocena długości skoku zawodnika jest powiązana z tzw. punktem K danej skoczni, którego położenie przedstawiono na rysunku 1.1. Za osiągnięcie punktu K skoczek otrzymuje 60 punktów. Punkt lądowania to miejsce, w którym obie stopy zawodnika zetknęły się z podłożem po skoku. Gdy tak jak w przypadku telemarku stopy skoczka są przesunięte względem siebie, za dokładny punkt mierzenia długości skoku przyjmuje się środek odległości między nimi. Jeżeli zawodnik upadnie, za miejsce wylądowania przyjmuje się punkt, w którym zetknął się on po raz pierwszy z ziemią po zakończeniu lotu.



Rys. 1.1. Fazy skoku narciarskiego a kluczowe punkty skoczni [1]

Długość skoku mierzona jest z dokładnością co pół punktu. Za każdy metr powyżej punktu K przyznaje się zawodnikowi punkty, zgodnie z tabelą 1.1:

Tabela 1.1. Liczba przyznawanych zawodnikowi punktów w zależności od punktu K skoczni

Długość punktu K (m)	Liczba punktów za metr
20 - 24	4.8
25 - 29	4.4
30 - 34	4.0
35 - 39	3.6
40 - 49	3.2
50 - 59	2.8
60 - 69	2.4
70 - 79	2.2
80 - 99	2.0
100 - 169	1.8
170 i więcej	1.2

W przypadku gdy zawodnik uzyska odległość mniejszą od punktu K, punkty przyznawane za metr dla danej wielkości skoczni są mnożone przez różnicę między punktem K a dystansem uzyskanym przez zawodnika i odejmowane od bazowej noty 60 punktów.

Rekompensata za wiatr

Punktacja za wiatr została wprowadzona, aby spróbować zagwarantować zawodnikom równe szanse i częściowo uniezależnić ich skok od warunków pogodowych. Punkty przyznane lub odjęte skoczkowi za wiatr obliczane są na podstawie wskaźnika TWS - prędkości wiatru stycznego (ang. *tangential wind speed*) [21] oraz stałych wyznaczonych przez siedem anemometrów ustawionych wzdłuż zakrzywienia skoczni.

Rekompensata za belkę

Ustawienie belki decyduje o prędkości zawodnika na progu - im niżej położona belka (a co za tym idzie, krótszy rozbieg), tym mniejsza prędkość skoczka w momencie wybicia z progu. Rekompensata za belkę wprowadzona została między innymi po to, aby wynagrodzić zawodnikom skrócenie rozbiegu. Może być ono zastosowane np. ze względów bezpieczeństwa, kiedy warunki pogodowe pozwalają na dalekie loty i organizator zawodów chce zapobiec upadkom czy kontuzjom. Gdy rozbieg zostanie skrócony, skoczkom dodawane są punkty, a gdy zostanie wydłużony (z reguły kiedy warunki pogodowe nie pozwalają na osiągnięcie zadowalającej odległości przy danej prędkości wybicia) punkty są odejmowane.

1.2. Predykcja w uczeniu maszynowym

Podział uczenia maszynowego

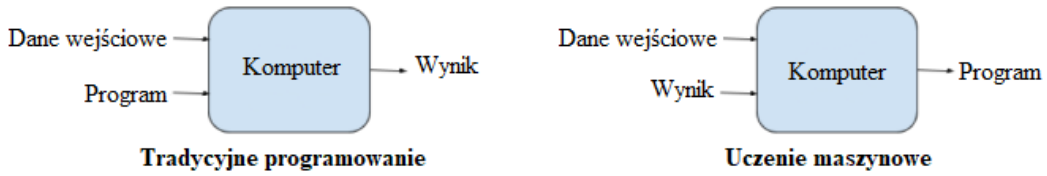
Uczenie maszynowe polega na zbudowaniu modelu na podstawie zaprezentowanych w trakcie etapu uczenia zbiorów danych [19]. Ze względu na sposób nauki, można je podzielić na trzy podstawowe kategorie:

- **uczenie nadzorowane** (ang. *supervised learning*) - w procesie uczenia dostępne są dane wejściowe i oczekiwane rezultaty,
- **uczenie nienadzorowane** (ang. *unsupervised learning*) - na podstawie przekazanych mu wyłącznie danych treningowych komputer musi znaleźć rozwiązanie problemu,
- **uczenie ze wzmocnieniem** (ang. *reinforcement learning*) - w trakcie procesu uczenia nie są dostarczane bezpośrednio ani dane wejściowe, ani oczekiwane rezultaty. Model jest trenowany poprzez interakcję z rzeczywistym bądź symulowanym środowiskiem.

Powyższym rodzajom uczenia maszynowego można przypisać charakterystyczne dla nich podejścia, co zostało przedstawione na rysunku 1.2.

Rys. 1.2. Rodzaje uczenia maszynowego wraz z przykładowymi podejściami [18][17]

W porównaniu do tradycyjnego programowania, którego przeznaczeniem również jest rozwiązanie zadanego mu problemu, uczenie maszynowe skupia się na przekazanych systemowi danych [2]. W klasycznym podejściu, do komputera trafiają informacje oraz kod programu, na podstawie którego obliczana jest odpowiedź układu. Całość jest w pełni zależna od napisanego przez człowieka algorytmu, który musi obsługiwać wyjątki i nie rozwinie się dalej opierając się na przekazanych mu danych. W przypadku uczenia maszynowego, do systemu trafiają dane wraz z oczekiwanym rezultatem, a wynikiem tego działania jest program (model), który może zostać użyty do wykonania predykcji w oparciu o dostarczone mu w przyszłości informacje. Na rysunku 1.3 przedstawiono różnicę pomiędzy podejściem klasycznym, a tym opartym o uczenie maszynowe.



Rys. 1.3. Porównanie tradycyjnego programowania z uczeniem maszynowym

Wykorzystanie uczenia maszynowego pozwala przetworzyć duże ilości danych, a co za tym idzie, w relatywnie krótkim czasie umożliwia stworzenie aplikacji opartej na znaczącej bazie informacji. Jest w stanie znaleźć nieoczywiste lub trudne do wykrycia zależności pomiędzy parametrami w wielowymiarowej przestrzeni. Jest zautomatyzowane - proces treningu po napisaniu kodu odbywa się bez ingerencji człowieka. Wymaga ono jednak dużego zbioru danych, na podstawie którego model zostanie wytrenowany. Dane wymagane do doskonalenia modelu nie zawsze są łatwe do pozyskania, mogą także być trudne w przetworzeniu czy interpretacji. Problem może także sprawić dostosowanie algorytmu do posiadanych informacji, co może doprowadzić do nieczytelnych, pozbawionych znaczenia wyników, lub spowodować, że model zamiast nauczyć się z danych, tylko je zapamięta. Przyczyni się to do jego przeuczenia i nieodpowiedniego działania (analogiczna sytuacja może wystąpić przy źle dobranych danych wejściowych).

Wybór uczenia maszynowego do rozwiązania problemu predykcji noty sędziowskiej skoczka narciarskiego można uzasadnić liczbą i wymiarem danych, jakie zostały pozyskane w celu budowy aplikacji. Poszukiwana zależność pomiędzy oceną wystawioną przez sędziów a pozostałymi parametrami skoku jest nieoczywista, a aplikacja ma za zadanie sprawdzić, czy jest ona deterministyczna, możliwa do przewidzenia.

Regresja liniowa

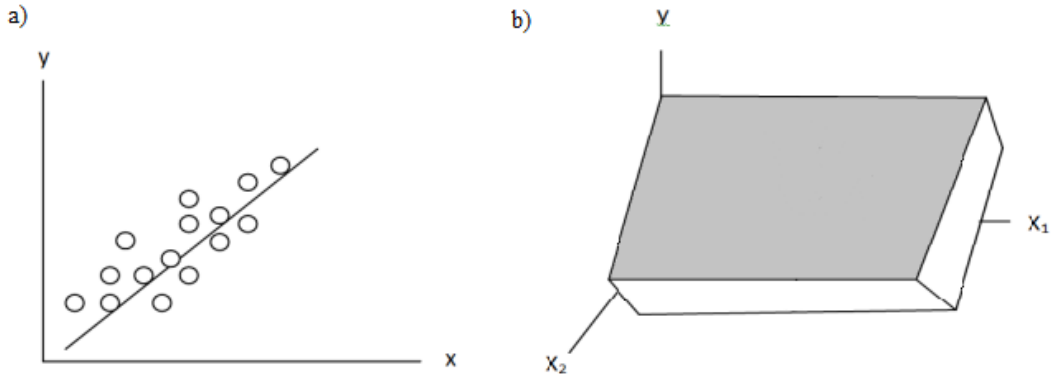
Jedna ze najprostszych metod uczenia nadzorowanego to regresja liniowa. Jest ona często używana do problemów predykcji lub zrozumienia relacji pomiędzy zmiennymi [20]. Zmienne używane do wyznaczania wyjścia określa się jako zmienne niezależne (ang. *independent variables*) [22]. W przypadku, gdy do przewidywania używana jest tylko jedna zmienna, wykorzystuje się regresję jednoparametrową (ang. *univariate regression*) (1.1):

$$y = a + bx \tag{1.1}$$

gdzie a jest stałą, x zmienną niezależną, a b współczynnikiem zmiennej x. Gdy prognoza odbywa się na podstawie wielu zmiennych, wykorzystywana jest regresja wieloparametrowa (ang. *multivariate regression*) (1.2):

$$y = a + b_1x_1 + b_2x_2 + ... + b_nx_n \tag{1.2}$$

gdzie a jest stałą, $x_1 ... x_n$ zmiennymi niezależnymi, a $b_1 ... b_n$ odpowiadającymi im współczynnikami. Rysunek 1.4 przedstawia wizualizację omówionych rodzajów regresji liniowej. Ze względu na liczbę zmiennych, regresja wieloparametrowa przedstawiana jest w formie wielowymiarowej płaszczyzny.



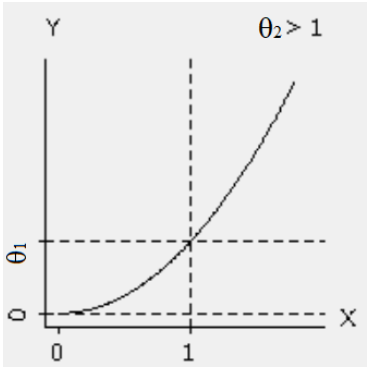
Rys. 1.4. Regresja jednoparametrowa (podpunkt a) i wieloparametrowa (podpunkt b) [22]

Inne typy regresji

Regresja nieliniowa stosowana jest w podobnych sytuacjach do wcześniej omówionej regresji liniowej. Używana jest, gdy relacja pomiędzy zmiennymi wejściowymi a wyjściem nie ma formy liniowej, więc liniowy odpowiednik regresji nie znalazłby dokładnego trendu. Odpowiednim przykładem może być zależność (1.3):

$$y = \theta_1 * x^{\theta_2} \tag{1.3}$$

Wówczas współczynnik θ_2 odpowiada za nieliniowość. Na rysunku 1.5 można zobaczyć wizualizację przedstawionej wyżej zależności.

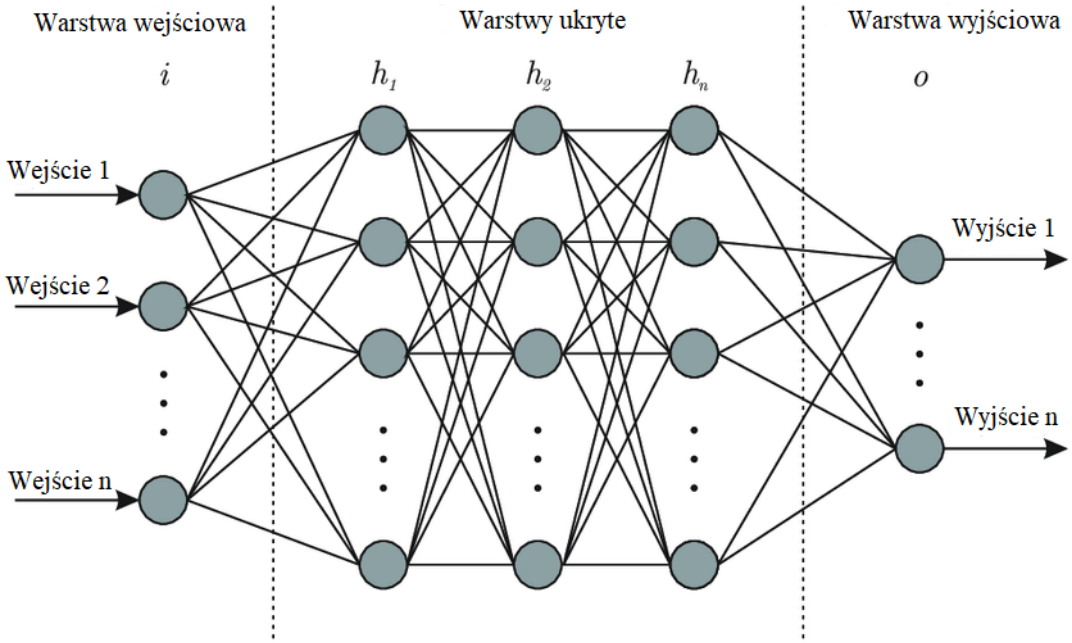


Rys. 1.5. Regresja nieliniowa [12]

Widać, że funkcja układa się w krzywą zależną od dwóch parametrów, θ_1, służącego za współczynnik stojący przy x oraz θ_2, który jest wykładnikiem potęgi.

Sztuczne sieci neuronowe

Struktura sztucznych sieci neuronowych (ang. *Artificial Neural Networks - ANN*) jest inspirowana działaniem i budową ludzkiego mózgu [8], dzięki czemu są one w stanie modelować nieliniowe i skomplikowane zależności. Architektura prostych sztucznych sieci neuronowych typu MLP (ang. *Multi Layer Perceptron*) zazwyczaj opiera się na warstwach, co zostało ukazane na rysunku 1.6. Warstwa wejściowa odpowiada za przekazanie zmiennych na wejście sieci neuronowej. Następnie są one przekazywane dalej z pewnymi wagami do funkcji aktywacji, a dalej do warstw ukrytych, aby na końcu mógł pojawić się wynik w warstwie wyjściowej. Jeżeli sieć jest typu *feed-forward* [16], powiązania między neuronami z kolejnych warstw są jednokierunkowe, a w samej sieci neuronowej nie występuje pętla. Modele ANN, w których występuje wiele warstw ukrytych, a na wejście podawane są surowe dane określa się jako głębokie sieci neuronowe (ang. *Deep Neural Networks*).

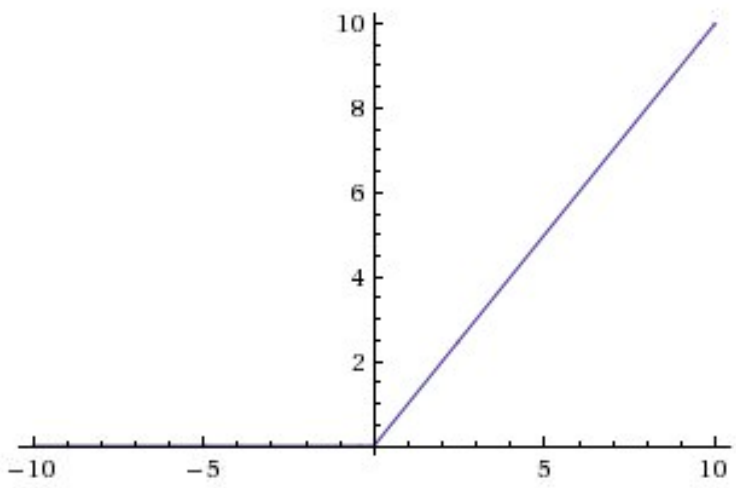


Rys. 1.6. Architektura MLP [4]

Między warstwami ANN często używana jest nieliniowa funkcja aktywacji (ang. *activation function*). Pozwala ona zamodelować nieliniową relację pomiędzy wejściem a wyjściem sieci. Bez jej wykorzystania model byłby bardzo zbliżony do regresji liniowej. Jedną z najpopularniejszych funkcji aktywacji używanych w sieciach neuronowych jest ReLU (ang. *Rectified Linear Unit*) (1.4). Zwraca ona 0, gdy podana jej wartość jest ujemna lub przekazaną jej liczbę, gdy liczba ta jest dodatnia.

$$f(x) = max(0, x) \tag{1.4}$$

Przykładowy wykres funkcji aktywacji ReLU przedstawiony jest na rysunku 1.7.



Rys. 1.7. Funkcja ReLU[3]

Zaletą funkcji ReLU jest to, że w przeciwieństwie do wcześniej popularnych sigmoidy i tangensu hiperbolicznego jest wrażliwa na szerszy przedział wartości [13]. Gdy do funkcji sigmoidalnych przekazywane są liczby inne niż z zakresu [0,1] ([-1,1] dla tanh), niezależnie od tego jak znacząco duże, na wyjściu zmieniane są w 1 (analogicznie dla bardzo małych liczb, dla których wynikiem przyporządkowania najmniejszą możliwą wartością może być 0 lub -1). Prowadzi to do zjawiska zwanego zanikającym gradientem (ang. *vanishing gradient*) - algorytm wstecznej propagacji błędu propaguje coraz mniejsze wartości, co dla większej liczby warstw może spowodować brak aktualizacji wag neuronów, a w konsekwencji zatrzymać naukę sieci neuronowej.

2. Dane

W poniższym rozdziale opisane zostanie w jaki sposób wybrano i przygotowano dane, które posłużyły do trenowania modelu. Na początku przedstawiony zostanie kod odpowiadający za przygotowanie danych wejściowych i ich podział na zbiory: testowy, treningowy i walidacyjny. W kolejnym kroku zostanie krótko opisana macierz korelacji między poszczególnymi elementami zbioru treningowego, która w dalszej części posłuży optymalizacji modelu, a także normalizacja wprowadzanych do modelu danych.

2.1. Przygotowanie plików z danymi

Dane dotyczące skoków pozyskano z bazy Kaggle [26]. Spośród około 290 tysięcy pojedynczych rekordów wybrano te obserwacje, które były kompletne, tj. zawierały wszystkie niezbędne do analizy dane dotyczące skoku - oprócz informacji dotyczących samego lotu, takich jak jego długość, pełna punktacja i rekompensaty niezbędne był także identyfikator konkursu czy zawodnika. Dodatkowo, ograniczono się wyłącznie do zawodów męskich, co wymagało dodania dodatkowej kolumny z płcią zawodnika (o zawartości `m/f`). Ograniczenie to wynika z faktu, że skoki narciarskie kobiet nie są dyscypliną popularną, co powoduje, że do modelu trafiłoby zbyt mało obserwacji. Ponadto, ze względu na inną budowę ciała, kobiety skaczą z niższej belki startowej, co wymusiłoby osobną analizę wyników kobiet i mężczyzn. Przy niewielkiej liczbie skoków kobiet możliwej do poddania badaniu, zbudowanie wiarygodnego modelu byłoby trudne. Ostatecznej analizie poddano około 70 tysięcy obserwacji, co jest wystarczającą liczbą do zbudowania wiarygodnego modelu.

Spośród pięciu plików, które znajdują się w bazie Kaggle skorzystano z dwóch:

– **`all_names.csv`** - Plik zawierający kody federacji narciarskiej FIS (ang. *Federation of International Skiing*) wszystkich skoczków oraz ich imiona i nazwiska. Do celów projektu został on uzupełniony o kolumnę z płcią zawodnika. W tabeli 2.1 przedstawione zostały kolumny występujące w pliku `all_names.csv` wraz z opisem ich zawartości.

Tabela 2.1. Struktura pliku `all_names.csv`

Nazwa kolumny	Opis zawartości
name	Imię i nazwisko zawodnika
codex	Unikalny kod FIS zawodnika
gender	Płeć zawodnika

– **`all_results.csv`** - Plik zawierający dane dotyczące konkretnych skoków, czyli pojedynczych obserwacji. Znajdują się w nim informacje na temat uzyskanej przez zawodnika odległości, punktów przyznanych mu za skok, a także rundy zawodów czy identyfikacji skoczka. W tabeli 2.2 opisano kolumny znajdujące się w pliku `all_results.csv` i ich zawartość.

Tabela 2.2. Struktura pliku `all_results.csv`

Nazwa kolumny	Opis zawartości
points	Łączna nota uzyskana przez zawodnika
speed	Prędkość zawodnika na progu
dist	Odległość uzyskana przez zawodnika
dist_points	Punkty przyznane za uzyskaną odległość
note_1	Ocena wystawiona przez pierwszego sędziego
note_2	Ocena wystawiona przez drugiego sędziego
note_3	Ocena wystawiona przez trzeciego sędziego
note_4	Ocena wystawiona przez czwartego sędziego
note_5	Ocena wystawiona przez piątego sędziego
note_points	Łączna nota sędziowska zawodnika
gate	Numer belki startowej
id	Unikalny numer ID konkursu
loc	Miejsce zajęte przez zawodnika
bib	Numer startowy zawodnika
round	Runda zawodów
wind	Prędkość wiatru w trakcie skoku zawodnika
wind_comp	Rekompensata za wiatr
gate_points	Rekompensata za belkę
codex	Unikalny kod FIS zawodnika

Do przygotowania danych posłużył kod w języku Python, wykorzystujący moduł `csv`. Plik `all_names.csv` podzielono na rzędy zawierające odpowiednio męskie i kobiece nazwiska, za co odpowiada kod z listingu 2.1.

Listing 2.1. Podział rzędów z nazwiskami zawodników

```
import csv

file_names = open('../all_names.csv')
csvreader2 = csv.reader(file_names)

men_rows = []
women_rows = []
for row in csvreader2:
    if row[2] == 'm':
        men_rows.append(row)
    else:
        women_rows.append(row)
```

Na podstawie tego podziału wybrano identyfikatory odpowiadające męskim zawodnikom i przechowano je jak w listingu 2.2, w odpowiedniej tablicy.

Listing 2.2. Wybranie identyfikatorów męskich zawodników

```
men_codex = []
for i in range (0, len(men_rows)):
    men_codex.append(men_rows[i][1])
```

Następnie na podstawie przygotowanej tablicy wybrano z pliku `all_results.csv` tylko rekordy, które dotyczyły zawodników płci męskiej i zapisano je do osobnego pliku. Listing 2.3 pokazuje cały proces.

Listing 2.3. Wybranie skoków oddanych przez mężczyzn i zapisanie ich do osobnego pliku

```
file = open('../all_results_colab2.csv')
csvreader = csv.reader(file)

men_jumps = []
for row in csvreader:
    if row[18] in men_codex:
        men_jumps.append(row)

with open('../jumps_men.csv', 'w', newline='') as f:
    write = csv.writer(f)
    write.writerows(men_jumps)
```

W kolejnym kroku podzielono obserwacje na zbiór treningowy, testowy i walidacyjny w stosunku 70:20:10. Aby model był trenowany na konkretnych zawodnikach, a nie przypadkowych obserwacjach, zamiast podziału pojedynczych skoków, do zbiorów przyporządkowano na początku identyfikatory zawodników, za co odpowiada listing 2.4.

Listing 2.4. Przyporządkowanie identyfikatorów zawodników do zbiorów treningowych, testowych i walidacyjnych

```
men_train_codex = []
men_test_codex = []
men_validation_codex = []

men_train_len = 0.7 * len(men_codex)
men_test_len = 0.2 * len(men_codex)
men_validation_len = 0.1 * len(men_codex)
```

Przy wybranym stosunku, skoki 972 zawodników znalazły się w zbiorze treningowym, do testowego przyporządkowano 278 skoczków, a do walidacyjnego 139. Spośród wszystkich 1389 zawodników rozlosowano 972 identyfikatory, które posłużyły trenowaniu modelu. Resztę przypisano do osobnej listy, z której wylosowano 278 identyfikatorów testowych. Pozostałe automatycznie przypisano jako walidacyjne (listing 2.5).

Listing 2.5. Losowanie identyfikatorów zawodników do odpowiednich zbiorów

```
import random

men_train_codex = random.sample(men_codex, 972)
men_other_codex = []
for i in range (0, len(men_codex)):
    if men_codex[i] not in men_train_codex:
        men_other_codex.append(men_codex[i])

men_test_codex = random.sample(men_other_codex, 278)
for i in range (0, len(men_other_codex)):
    if men_other_codex[i] not in men_test_codex:
        men_validation_codex.append(men_other_codex[i])
```

W ostatnim kroku przygotowania danych spośród wcześniej zapisanych do pliku `jumps_men.csv` rekordów wybrano skoki treningowe, testowe i walidacyjne, w oparciu o podział identyfikatorów zawodników. Na koniec, zapisano wybrane obserwacje do osobnych plików. Działania te można zauważyć w listingu 2.6.

Listing 2.6. Zapisanie skoków do odpowiednich zbiorów

```
file = open('../jumps_men.csv')
csvreader = csv.reader(file)

men_test_jumps = []
men_train_jumps = []
men_validation_jumps = []

for row in csvreader:
    if row[18] in men_train_codex:
```

```
        men_train_jumps.append(row)
    elif row[18] in men_test_codex:
        men_test_jumps.append(row)
    else:
        men_validation_jumps.append(row)

with open('../test_jumps_men.csv', 'w', newline='') as f:
    write = csv.writer(f)
    write.writerows(men_test_jumps)

with open('../train_jumps_men.csv', 'w', newline='') as f:
    write = csv.writer(f)
    write.writerows(men_train_jumps)

with open('../validation_jumps_men.csv', 'w', newline='') as f:
    write = csv.writer(f)
    write.writerows(men_validation_jumps)
```

Tak spreparowane pliki będą mogły służyć do treningu i testowania modelu predykcji łącznej noty sędziowskiej zawodnika, co jest celem omawianego projektu.

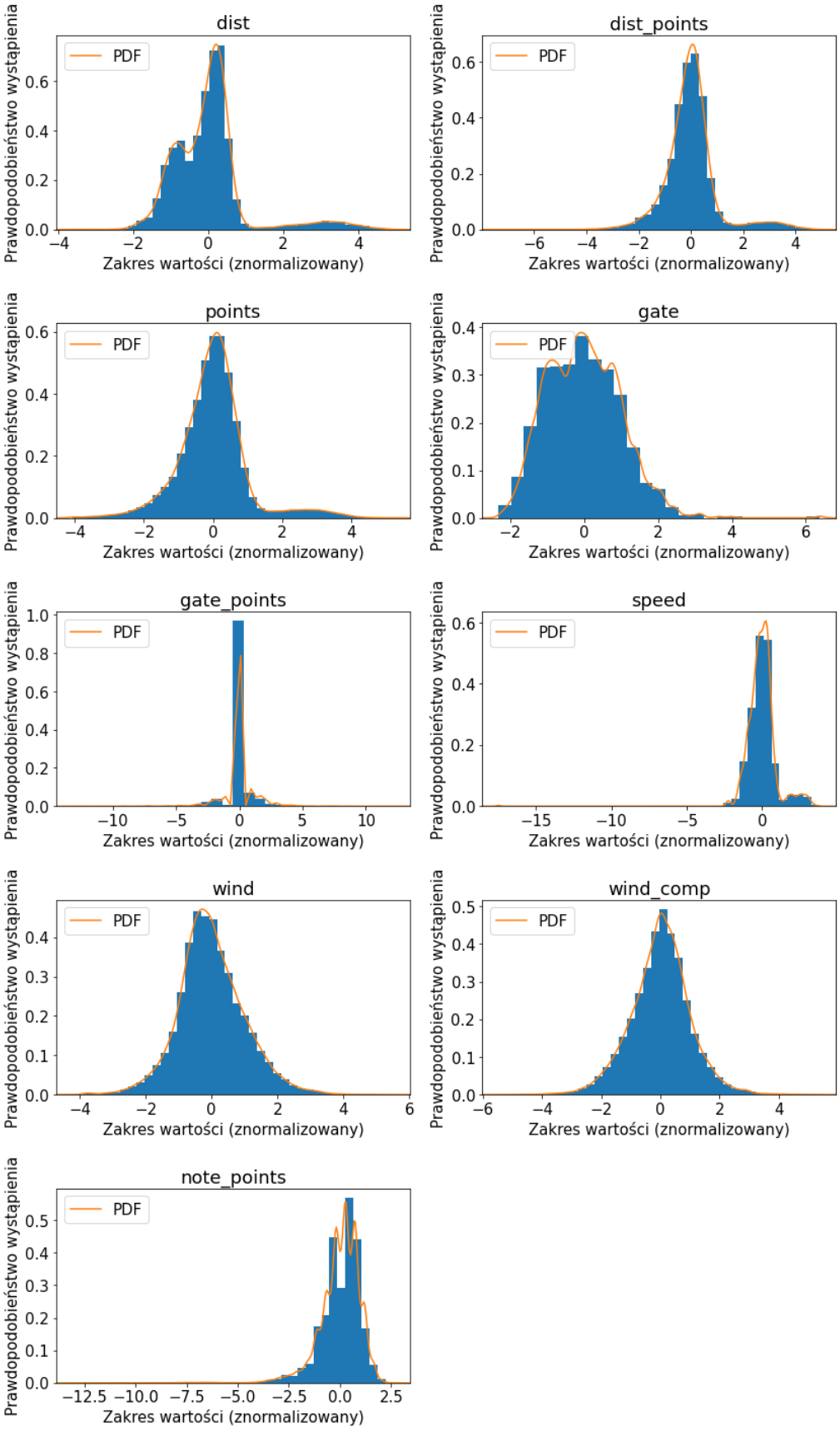
2.2. Normalizacja danych

Zanim dane trafiły do modelu, musiały zostać znormalizowane. Zabieg ten zwiększa możliwości obliczeniowe sieci neuronowej, co przyspiesza proces uczenia się modelu [24]. W tym przypadku zastosowana została tzw. normalizacja Z (ang. *z-score normalization*) (2.1) z użyciem średniej oraz odchylenia standardowego, określana także jako standaryzacja danych.

$$\hat{X}[:, i] = \frac{X[:, i] - \mu_i}{\sigma_i} \tag{2.1}$$

Proces ten sprowadza średnią danych do 0, a ich odchylenie standardowe do 1. Jeśli nie przedstawiały one uprzednio rozkładu normalnego, standaryzacja danych ich do niego nie sprowadzi. Rysunek 2.1 przedstawia histogramy każdej z cech modelu wraz z ich funkcją gęstości prawdopodobieństwa (ang. *PDF - Probability Density Function*).

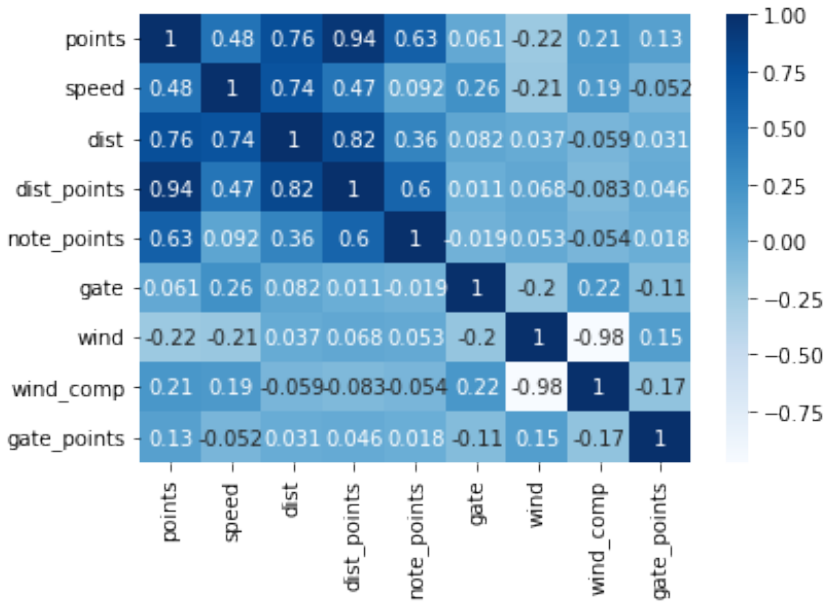
Część parametrów skoku nie ma rozkładu normalnego. Przykładowo `dist`, czyli odległość osiągnięta przez zawodnika nigdy nie ułoży się w kształcie krzywej Gaussa, ponieważ po najczęściej uzyskiwanym przedziale odległości 120-140 metrów (duża część zawodów odbywa się na tzw. skoczniach dużych, które umożliwiają loty na takie dystanse), praktycznie nigdy nie pojawiają się w konkursach skoki długości 150,160 czy 170 metrów. Jednak lot o długości powyżej 180 metrów jest odległością osiąganą na skoczni mamuciej, na której zawody odbywają się kilka razy w sezonie (wówczas przeciętny dystans lotu to 210-230 metrów). Na rozkładzie częstotliwości występowania danej odległości pojawi się więc "górka"przy dużych dystansach.



Rys. 2.1. Histogramy poszczególnych parametrów skoku

2.3. Macierz korelacji

Dla dokładniejszej analizy danych wejściowych i wagi poszczególnych parametrów skoku w modelu zastosowano funkcję `corr()` narzędzia `Pandas`. Z wykorzystaniem narzędzia `seaborn`, które służy do wizualizacji danych statystycznych sporządzono mapę korelacji cech zbioru treningowego. Z analizy wykluczono dane opisowe, takie jak runda i numer zawodów, numer skoczka, jego pozycję i numer startowy a także pojedyncze noty sędziowskie. Na rysunku 2.2 można zauważyć, które cechy najbardziej wpływają na siebie nawzajem.



Rys. 2.2. Macierz korelacji parametrów skoku

Macierz korelacji jest macierzą symetryczną z jedynkami na przekątnej (znajduje się na niej wynik korelacji elementu z sobą samym). Im bliżej wynikowi korelacji do 1 lub -1, tym parametry są ze sobą bardziej skorelowane. Wysoka korelacja dwóch parametrów oznacza, że reprezentują one tę samą informację, więc analizowanie obu z nich nie dostarcza żadnej dodatkowej wiedzy o danych. Na łączną notę zawodnika (tutaj, `points`) największy wpływ ma prędkość (`speed`, 0.48), odległość (`dist`, 0.76), punkty przyznane za odległość (`dist_points`, 0.94) oraz punkty przyznane przez sędziów (`note_points`, 0.63). Parametr taki jak belka, z której skacze zawodnik (`gate`) praktycznie nie ma wpływu na jego łączną notę (współczynnik korelacji dla niego wynosi 0.061). Z zasad oceniania wiadomo, że oddziałuje ona na punkty przyznane zawodnikowi w ramach rekompensaty. Stanowią one jednak znacznie mniejszą część łącznej noty niż ocena odległości i stylu. Z notą sędziowską skoczka, która będzie przewidywana przez model najsilniej skorelowane są: łączna nota zawodnika (współczynnik korelacji 0.63, ocena sędziowska stanowi jej część), przebyta odległość (0.36), i punkty przyznane za dystans (0.36). Wpływ pozostałych parametrów skoku jest niewielki. Warto także zauważyć, że na

odległość, jaką osiąga skoczek największy wpływ ma prędkość na progu, korelacja z wiatrem czy belką jest słaba.

3. Implementacja

W rozdziale dotyczącym implementacji przedstawiony zostanie sposób, w jaki zbudowano model. W dalszej części opisane będą także próby jego optymalizacji narzędziem KerasTuner oraz wynik zmniejszenia liczby parametrów wejściowych modelu.

3.1. Budowa modelu

Językiem wykorzystanym do budowy modelu jest Python. Został on wybrany ze względu na dobrą dostępność bibliotek przydatnych w uczeniu maszynowym, a także łatwość nauki czy dużą społeczność posługującą się nim. Do wczytania plików z parametrami skoków użyto modułu `Pandas`, który jest znanym narzędziem wspierającym analizę danych. Posiada on swój typ danych, dwuwymiarowy `Dataframe`, przypominający tablicę. Na wejście modelu przekazane zostały takie parametry skoków jak: odległość, punkty za odległość, numer belki startowej, rekompensata za belkę, prędkość wiatru, rekompensata za wiatr oraz prędkość zawodnika na progu. Przewidywaną wartością (wyjście modelu), jest łączna nota sędziowska.

Prosty model regresyjny

Na początku zbadano, czy problem predykcji noty sędziowskiej skoczka narciarskiego może zostać sprowadzony do prostego zagadnienia, które da się rozwiązać za pomocą regresji liniowej wieloparametrowej. Do zbudowania takiego modelu posłużyła biblioteka `scikit-learn` dysponująca wieloma narzędziami do analizy danych w języku Python. Kod odpowiadający za skonstruowanie modelu regresyjnego przedstawiono na listingu 3.1.

Listing 3.1. Budowa modelu regresyjnego

```
from sklearn import linear_model

X = train_data[['speed', 'dist', 'dist_points', 'gate', 'wind', 'wind_comp', '
    gate_points']]
y = train_data['note_points']
regr = linear_model.LinearRegression()
regr.fit(X, y)
```

Wytrenowany model wieloparametrowej liniowej regresji można przedstawić zależnością (3.1):

$$y = -3.022e^{-16} - 0.143x_1 - 0.25x_2 + 0.875x_3 + 0.02x_4 + 0.111x_5 + 0.134x_6 - 0.015x_7 \quad (3.1)$$

Wielkość współczynników wskazanych przez równanie regresji zgadza się z siłą korelacji z łączną notą sędziowską zawodnika odpowiadających im parametrów. W dalszym ciągu najbardziej wpływową cechą są punkty uzyskane za odległość (tutaj oznaczone jako x_3), a kolejno sama odległość i prędkość wybicia z progu (odpowiednio x_1 i x_2). Model uzyskał punktację R^2 równą 0.419, co oznacza, że zmienne niezależne odpowiadają za nieco ponad 40% zmian w zmiennej zależnej [10].

Regresja liniowa jest bardzo czuła na dane odbiegające od pozostałych (ang. *outliers*). Oznacza to, że w przypadku występowania obserwacji, które znacząco różnią się od typowych, linia regresji (lub w tym, wieloparametrowym przypadku wielowymiarowa płaszczyzna) zostaje przesunięta w ich stronę. W problemie predykcji noty skoczka narciarskiego takie elementy pojawiają się we wszystkich zbiorach danych. Mogą one oznaczać loty w bardzo nietypowych warunkach atmosferycznych, upadki lub błędne oceny sędziów, dzięki czemu są w stanie dostarczyć dodatkowych informacji. Z tego powodu nie zostaną one usunięte dla ewentualnej poprawy jakości modelu, ale wykorzystane do prognozy innych zdarzeń.

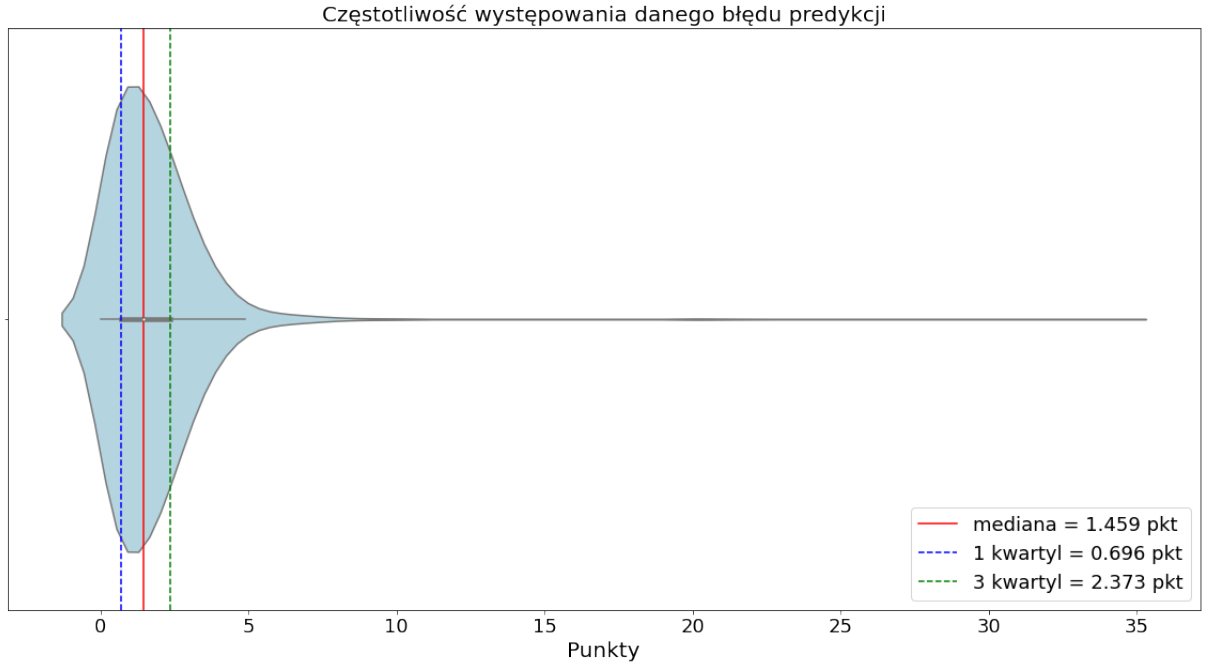
Metryką wykorzystaną do oceny jakości prostego modelu regresyjnego i porównania go z pozostałymi, bardziej zaawansowanymi jest średni błąd bezwzględny (*ang. MAE - mean absolute error*) (3.2).

$$MAE = \frac{1}{n}\sum |y_i - x_i| \quad (3.2)$$

W przypadku tego modelu, MAE dla zdenormalizowanych danych wyniósł 1.773. Oznacza to, że wyznaczona za pomocą regresji nota sędziowska zawodnika różni się średnio o 1.773 punktu od tej, jaką skoczek naprawdę otrzymał. Dla maksymalnej noty wynoszącej 60 punktów, przewidziana ocena może odbiegać od prawdziwej o około 3%. Dla średniej noty uzyskanej przez zawodnika, wynoszącej 51.648 punktów, różnica ta wyniesie około 3.5%.

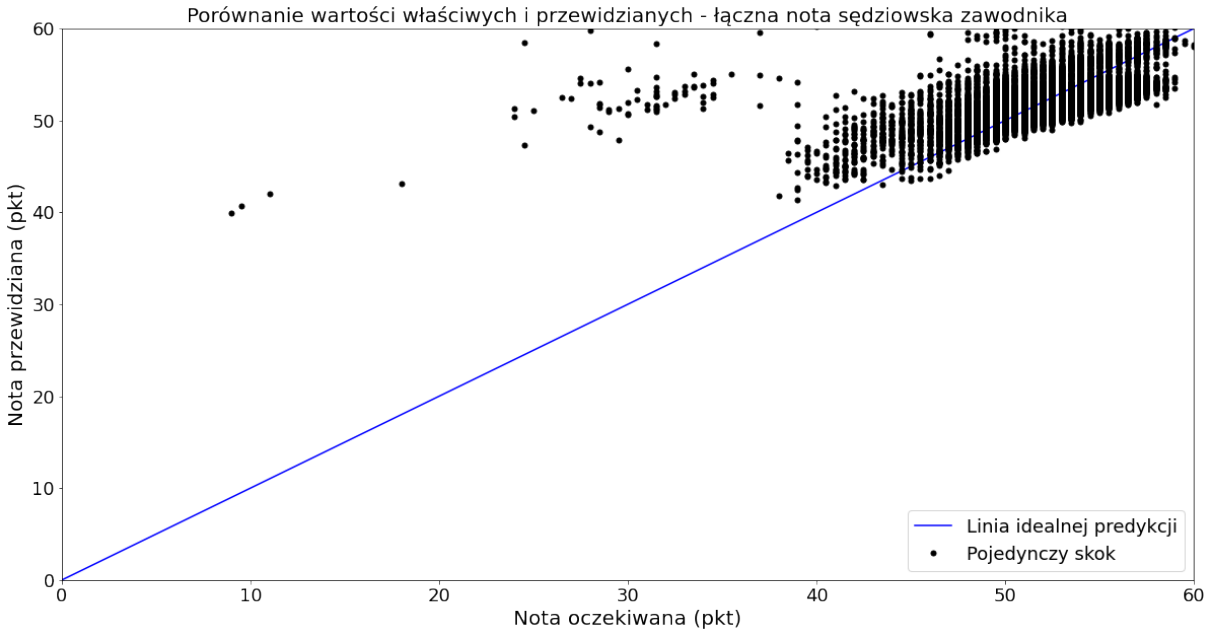
Mediana błędu wyniosła 1.459 punktu - spośród ponad 17.5 tysiąca not ze zbioru testowego, połowa z nich różni się od oceny przewidzianej modelem regresyjnym o niecałe 1.5 punktu. Rysunek 3.1 przedstawia rozkład częstotliwości występowania błędów sporządzony za pomocą funkcji `violinplot` biblioteki `seaborn`. Zaznaczono na nim także dolne i górne kwartyle zbioru błędów.

Można zauważyć, że 75% błędów przewidywań skupia się poniżej 2.373 punktu, czyli 4.6% średniej oceny za styl zawodnika. Jest to stosunkowo niewiele, jednak w praktyce dwa punkty mogą zmienić lokatę skoczka. Dla przykładu, w czasie zawodów inaugurujących sezon 2022/2023, różnica pomiędzy 10 a 15 miejscem wyniosła 2.6 punktu. W konkursie na skoczni normalnej podczas Igrzysk Olimpijskich w Pekinie w 2022 roku, różnica punktowa pomiędzy zdobywcą brązowego medalu a czwartym miejscem wyniosła zaledwie pół punktu. Warto wiedzieć, że wyższa nota sędziowska nie równa się wyższemu miejscu zajętemu w zawodach - w trakcie wspomnianej inauguracji, zawodnik z trzeciego miejsca osiągnął w drugiej serii ocenę za styl w wysokości 57 punktów, a zwycięzca o 1.5 punktu mniejszą.



Rys. 3.1. Wykres skrzypcowy błędu predykcji łącznej noty sędziowskiej skoczka dla modelu regresyjnego

Na rysunku 3.2 przedstawiono wykres różnic pomiędzy obserwacjami a ustaleniami modelu regresyjnego. Zaznaczono na nim także linię idealnej predykcji. Widać, że dla niższych not oczekiwanych, model przewiduje znacznie wyższą ocenę, a co za tym idzie, występują dla nich duże błędy. Warto zaobserwować również, że wykres not jest dyskretny - występują one co pół punktu.



Rys. 3.2. Porównanie noty przewidzianej z oczekiwaną - prosty model regresyjny

W pozostałej części projektu omówiony powyżej model będzie określany jako model **REGR**.

Sekwencyjny model MLP z optymalizacją hiperparametrów

Do budowy modelu użyta została biblioteka `Tensorflow`, która jest platformą umożliwiającą uczenie maszynowe przy zastosowaniu języka Python. Oprócz tego, wykorzystano narzędzie KerasTuner do optymalizacji hiperparametrów wybranego modelu, w tym przypadku sekwencyjnego. Jego trenowanie odbywało się w środowisku Google Colab.

KerasTuner umożliwia optymalizację hiperparametrów modelu celem jak najlepszego dopasowania ich do danych wejściowych [15]. Tutaj poszukiwano optymalnej liczby warstw ukrytych modelu oraz liczby neuronów w warstwie, osobnej dla każdej warstwy, a także prędkości uczenia modelu i liczby próbek podawanej jednocześnie do sieci (ang. *batch size*). Do poszukiwania najkorzystniejszych wartości danych parametrów wykorzystywana jest klasa `Tuner`. Oprócz niej, jako podstawowego narzędzia zdefiniowano także cztery specjalne tunery: `RandomSearch`, `Hyperband`, `BayesianOptimization` i `Sklearn`. Na potrzeby budowanego modelu użyto klasy `Hyperband`. Wykorzystuje ona algorytm Hyperband, który umożliwia przetestowanie wielu zbiorów hiperparametrów w krótkim czasie dzięki zastosowaniu techniki *early-stopping* i adaptacyjnej alokacji zasobów [14]. Za skonstruowanie obiektu klasy `Hyperband` wykorzystanego w optymalizacji omawianego modelu odpowiada kod źródłowy z listingu 3.2.

Listing 3.2. Stworzenie obiektu tunera

```
tuner = kt.Hyperband(buildmodel,
                     objective='val_mae',
                     max_epochs=100,
                     directory='my_dir',
                     project_name='jumps_men_tuner')
```

Tabela 3.1 przedstawia jakie hiperparametry modelu zostały przekazane do narzędzia KerasTuner oraz w jakim zakresie były one testowane. Tuning parametrów odbył się na zbiorze treningowym, a metrykę MAE analizowano na zbiorze walidacyjnym.

Tabela 3.1. Parametry optymalizowane przez KerasTuner i ich zakresy

Parametr	**Zakres**	**Krok**
Liczba warstw	2-5	1
Liczba neuronów w warstwie	32-512	16
Prędkość uczenia	$1e^{-5}$ - $1e^{-2}$	próbkowanie logarytmiczne
Batch size	16-256	16

Optymalnej liczby warstw poszukiwano pomiędzy wartościami 2 a 5. Biorąc pod uwagę jakość modelu regresyjnego, większa liczba warstw sieci neuronowej zbytnio by ją skomplikowała. Liczba neuronów w każdej warstwie sprawdzana była od wartości 32 do 512, z krokiem co 16. Prędkość uczenia się modelu poszukiwana była pomiędzy wartościami $1e^{-5}$ a $1e^{-2}$ z próbkowaniem logarytmicznym.

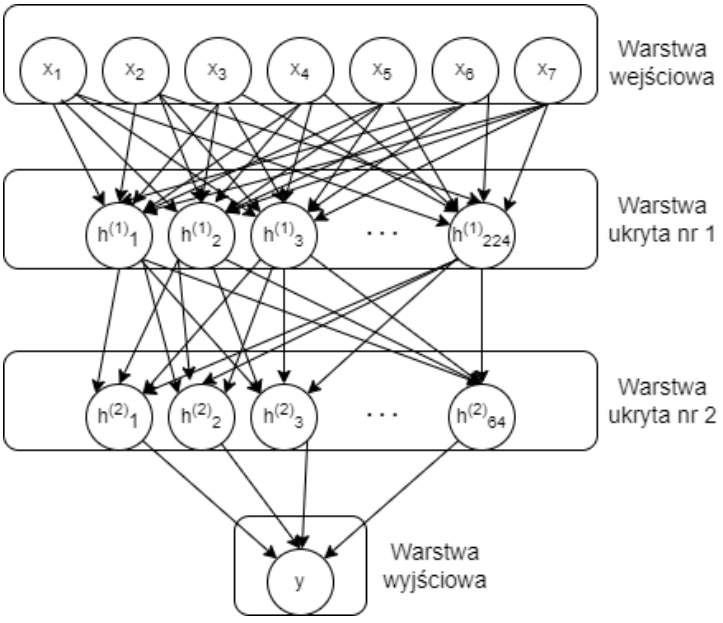
Batch size testowano od 16 do 256 próbek z krokiem co 16 (sprawdzono także jakość modelu dla wartości 4 oraz 8). W tabeli 3.2 sporządzono porównanie MAE dla danej liczby warstw oraz zebrano liczby neuronów w każdej warstwie.

Tabela 3.2. Wynik optymalizacji narzędziem KerasTuner. L - warstwa

	Liczba neuronów w warstwie					
Liczba warstw	**L1**	**L2**	**L3**	**L4**	**L5**	**MAE**
2	224	64				0.3686
3	240	240	368			0.3691
4	496	368	176	112		0.3687
5	320	256	128	64	128	0.3708

Najmniejszy błąd osiągany był przy dwóch warstwach. Niewiele mniejszy był ten dla czterech warstw, jednak przy podobnej wartości, dla maksymalnego uproszczenia modelu wybrano mniejszą liczbę warstw ukrytych. Jako prędkość uczenia się modelu wybrano 0.0003. Najkorzystniejszym *batch size* okazało się 16, co jest stosunkowo niewielką liczbą. Przy takiej wartości tego hiperparametru istniało ryzyko przeuczenia, jednak po przeanalizowaniu krzywych kosztu na zbiorze treningowym i walidacyjnym nie zaobserwowano go. Optymalizacja hiperparametrów dla sekwencyjnego modelu przewidywania noty sędziowskiej skoczka narciarskiego trwała około 2.5 godziny.

Tak przygotowane hiperparametry wraz ze znormalizowanymi danymi wejściowymi i wyjściowymi trafiły do sekwencyjnego modelu. Kod źródłowy odpowiedzialny za jego budowę przedstawiono jako listing 3.3, a schemat modelu znajduje się na rysunku 3.3.

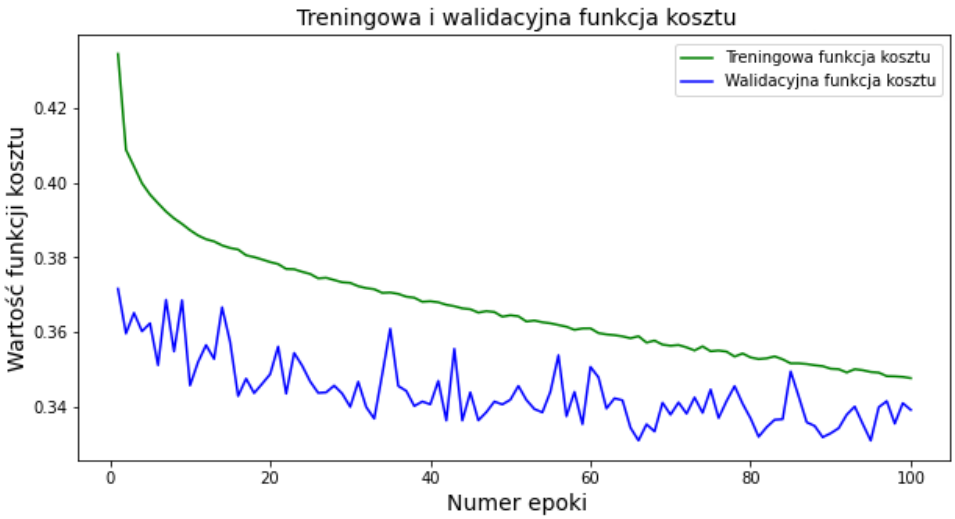


Rys. 3.3. Schemat trenowanego modelu MLP

Listing 3.3. Budowa modelu predykcji noty sędziowskiej skoczka narciarskiego

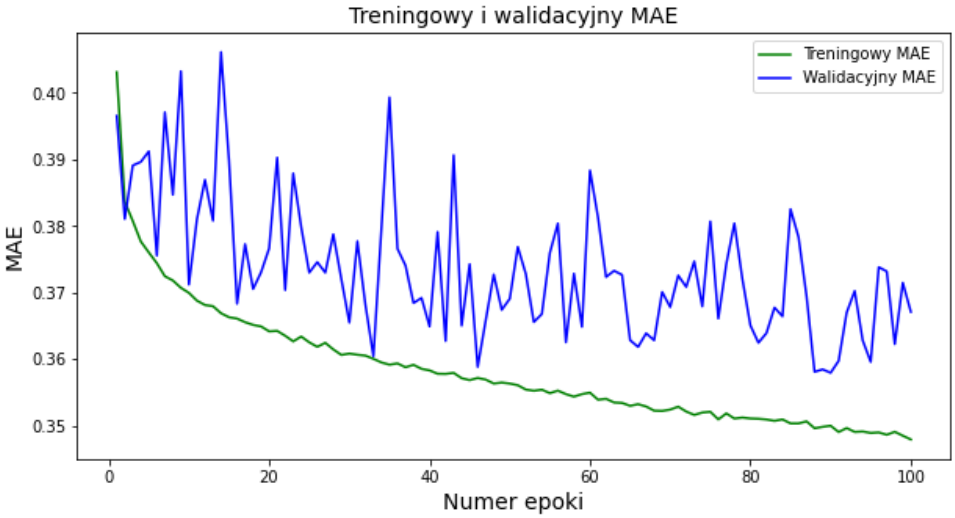
```
def buildmodel():
  d = train_data.shape[1]
  model = tf.keras.models.Sequential([
      tf.keras.layers.Dense(224, activation='relu', input_shape=(d,)),
      tf.keras.layers.Dense(64, activation='relu'),
      tf.keras.layers.Dense(1)
  ])
  optimizer = tf.keras.optimizers.Adam(lr=0.0003)
  model.compile(optimizer=optimizer, loss='mse', metrics=['mae'])
  return model
```

Trenowanie modelu zajęło średnio 11 minut. Zbiór treningowy przepuszczono przez algorytm 100 razy (ang. *epochs*). Wybraną dla modelu funkcją kosztu (ang. *loss*) jest średni błąd kwadratowy (ang. *MSE - mean squared error*). Parametr ten jest minimalizowany w trakcie uczenia. Treningowy *loss* stanowi sposób oceny błędów modelu w predykcji w trakcie uczenia, analogicznie walidacyjny dotyczy jakości modelu w trakcie jego walidacji. Na rysunku 3.4 przedstawiono wykres porównujący treningową i walidacyjną funkcję kosztu modelu w zależności od numeru epoki. Widać na nim, że model się uczy - MSE jest coraz mniejszy na obu zbiorach danych.



Rys. 3.4. Wykres kosztu w funkcji epoki

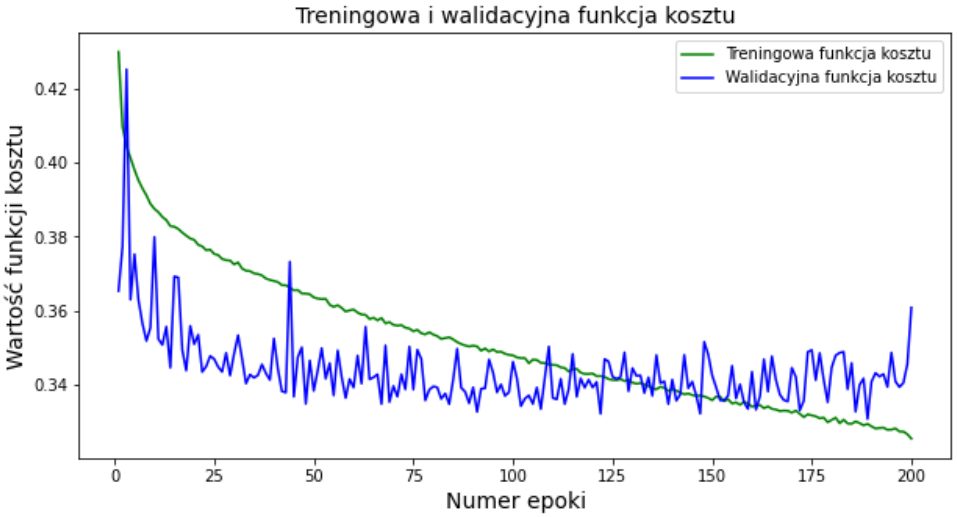
Metryką wybraną dla oceny jakości modelu (pole `metrics` w funkcji `compile` modelu) był podobnie jak dla modelu regresyjnego MAE. Tak jak funkcja kosztu, był on obliczony zarówno na zbiorze treningowym jak i walidacyjnym, pod koniec każdej epoki. Rysunek 3.5 przedstawia wykres MAE dla obu zbiorów danych w odniesieniu do numeru epoki.



Rys. 3.5. Wykres MAE w funkcji epoki

Na rysunkach 3.4 i 3.5 widać, że zarówno metryka jak i funkcja kosztu liczone na zbiorze walidacyjnym fluktuują. Może to wynikać z faktu, że w zbiorze znalazły się nietypowe obserwacje, których model nie zna. Jedną z przyczyn takiego zachowania może też być zbyt mała wielkość zbioru walidacyjnego, jednak w tym przypadku (prawie 4 tysiące próbek) nie stanowi to problemu. Powodem takiego działania mógłby być również zbyt mały *batch size*, jednak sprawdzono wykresy dla zarówno mniejszych jak i większych wartości tego parametru - fluktuacje nie znikały.

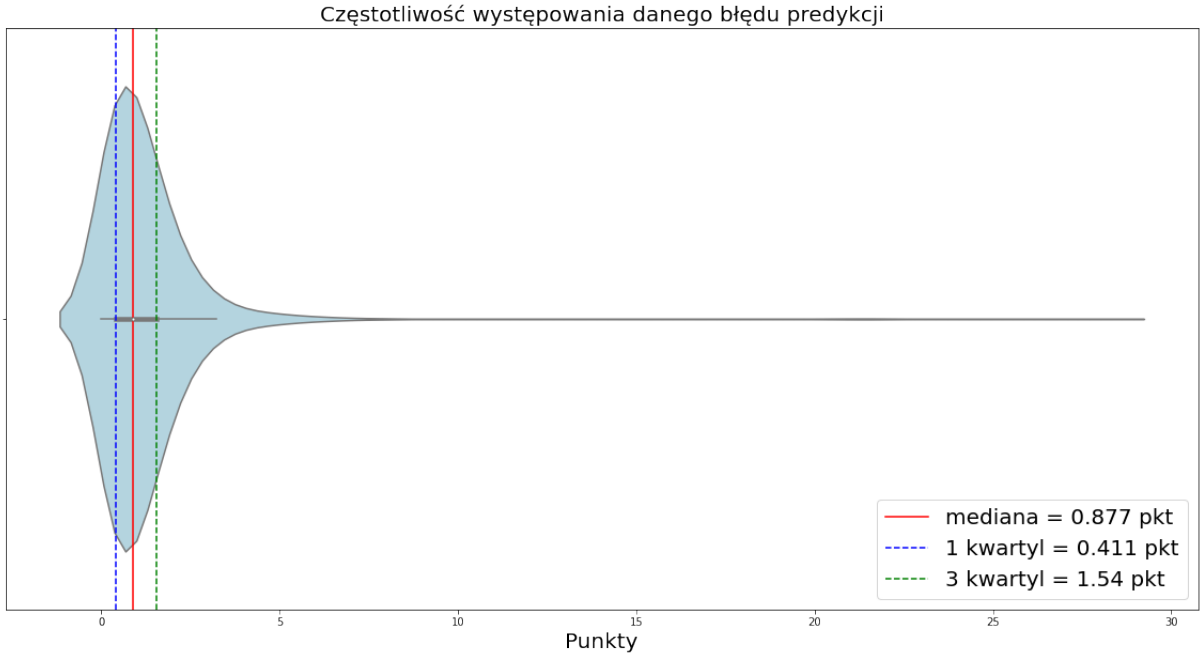
Celem dalszej eksploracji modelu, ustawiono liczbę epok na 200. Na wykresie funkcji straty z rysunku 3.6 można zauważyć, że przy nadal malejącej treningowej funkcji kosztu, walidacyjna rośnie. Jest to typowa oznaka przeuczenia modelu. Ograniczenie liczby epok do 100 jest zatem formą walki z przeuczeniem, zwaną *early-stopping* - w momencie gdy model już się nie uczy, trening kończy się.



Rys. 3.6. Wykres kosztu w funkcji epoki - 200 epok

Jako metrykę jakości prognozy po treningu, na zbiorze testowym obliczono *mean absolute error*, aby model mógł zostać porównany z jego regresyjnym odpowiednikiem. Średni błąd bezwzględny pomiędzy prawdziwymi obserwacjami a ich wyznaczonymi ekwiwalentami wyniósł 1.216 punktu. O tyle waha się nota przyznana skoczkowi przez sędziego od tej zdefiniowanej przez model. Oznacza to, że błąd stanowi około 2.36% średniej oceny zawodnika.

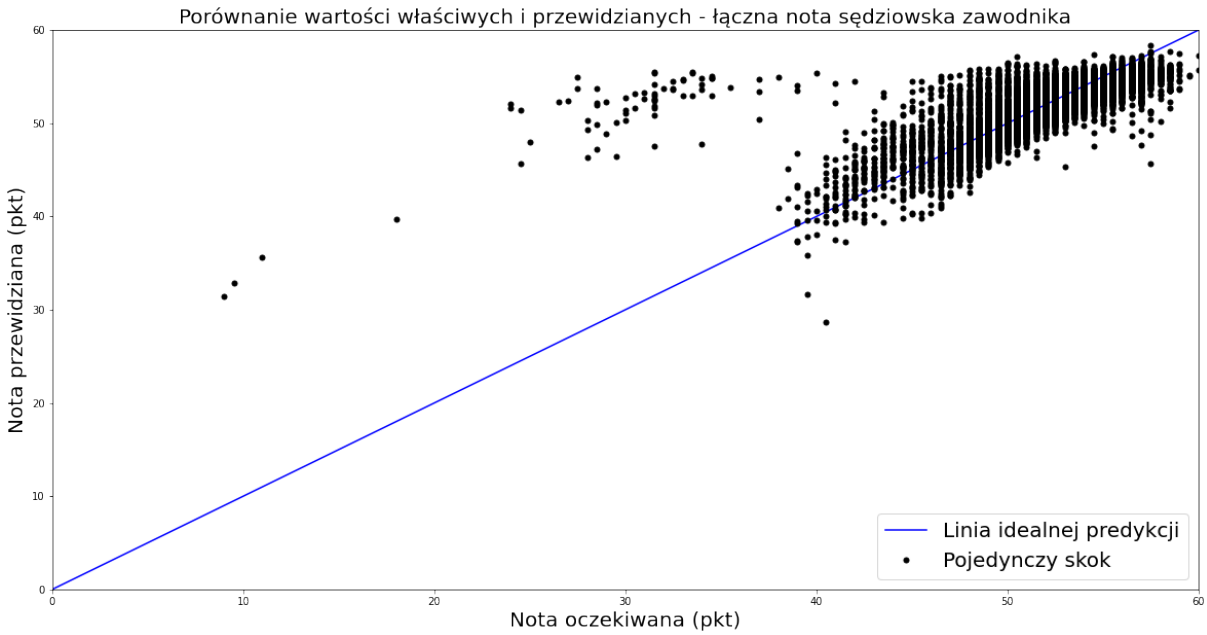
Na rysunku 3.7 nakreślono wykres skrzypcowy bezwzględnych różnic pomiędzy wartościami właściwymi a przewidzianymi. Mediana błędu sięgała 0.862 punktu - niecałe 9 tysięcy obserwacji różniło się o tyle lub mniej od ustaleń modelu. Osiągnięty wynik jest wystarczająco dokładny, aby model mógł zostać użyty. Na rysunku podkreślono również dolny i górny kwartyl zbioru.



Rys. 3.7. Wykres skrzypcowy błędu predykcji łącznej noty sędziowskiej skoczka dla modelu sekwencyjnego

Umiejscowienie górnego kwartylu na poziomie 1.54 punktu również satysfakcjonuje - 75% błędów przewidywań mieści się poniżej tej wartości. Chociaż stanowi ona 3% średniej noty sędziowskiej skoczka, należy pamiętać, że do łącznego wyniku zawodnika wliczają się także inne parametry. W przypadku najpopularniejszych skoczni, gdzie najczęściej osiągane są odległości rzędu 120-140 metrów, największą część całkowitej noty skoczka stanowią punkty za odległość. Dla przykładu, w czasie wspomnianej wcześniej inauguracji sezonu w Wiśle, łączne oceny zawodników wahały się między 120 a 140 punktów (za jedną serię). Wówczas błąd predykcji dla 75% zbioru to tylko 1.18% średniej noty zawodnika. Różnice pomiędzy czołowymi zawodnikami były większego rzędu.

Porównanie wartości przewidzianych z właściwymi stanowi wykres przedstawiony na rysunku 3.8. Widać na nim, że noty o typowych wartościach w okolicy 45-55 punktów układają się blisko linii idealnej predykcji - zostały przewidziane z najmniejszym błędem. Oceny znacznie poniżej tej wartości stanowią dla modelu większe wyzwanie. Błąd w ich wyznaczaniu jest już bardzo wyraźny.



Rys. 3.8. Porównanie noty przewidzianej z oczekiwaną - model sekwencyjny

W dalszej części pracy omówiony wyżej model sekwencyjny określany będzie jako **MLP1**.

3.2. Optymalizacja modelu

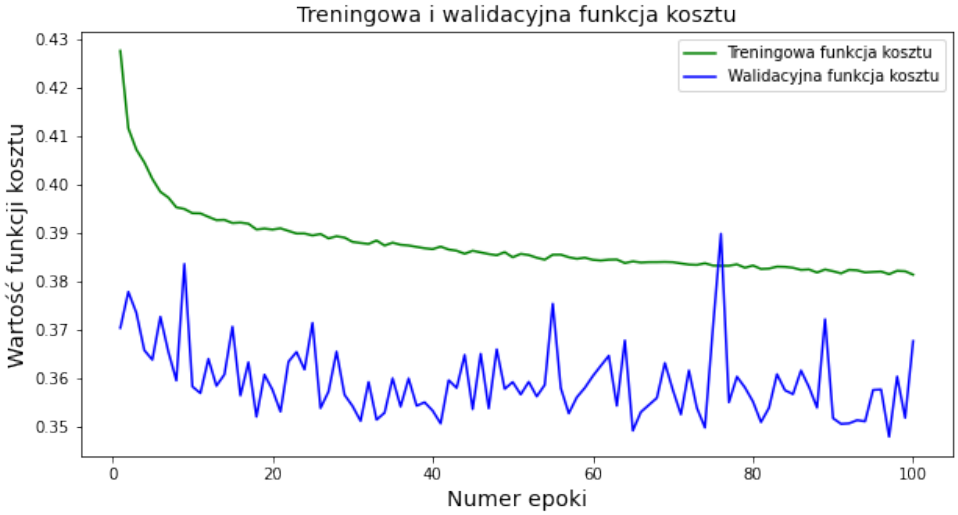
Analiza macierzy korelacji parametrów skoku oraz współczynników regresji liniowej wykazała, że niektóre cechy skoku, takie jak belka, wiatr oraz odpowiadające im rekompensaty punktowe mają niewielki wpływ na łączną ocenę sędziowską zawodnika. Dlatego celem próby zoptymalizowania (zmniejszenia) modelu uczenia maszynowego zdecydowano się wytrenować dodatkowy model, pozbawiony kilku cech na wejściu. Gdyby jego jakość okazała się podobna do modelu sekwencyjnego z optymalizacją hiperparametrów, prostsza sieć neuronowa zastąpiłaby go w dalszej analizie.

Mniejszy model zbudowano używając tych samych hiperparametrów, co w podsekcji 3.1 - prędkość uczenia ustawiono na 0.0003, jako *batch size* podawano 16 próbek, a sama sieć składała się z dwóch warstw ukrytych o liczbie neuronów odpowiednio 224 i 64. Tym razem jednak na wejście modelu trafiły tylko trzy parametry skoku: odległość (`dist`) i punkty za nią przyznane (`dist_points`) oraz prędkość zawodnika na progu (`speed`). Wybraną liczbą epok było 100, jak poprzednio. Trenowanie modelu zajęło około 11 minut. Od tego momentu będzie on określany jako **MLPOPT**.

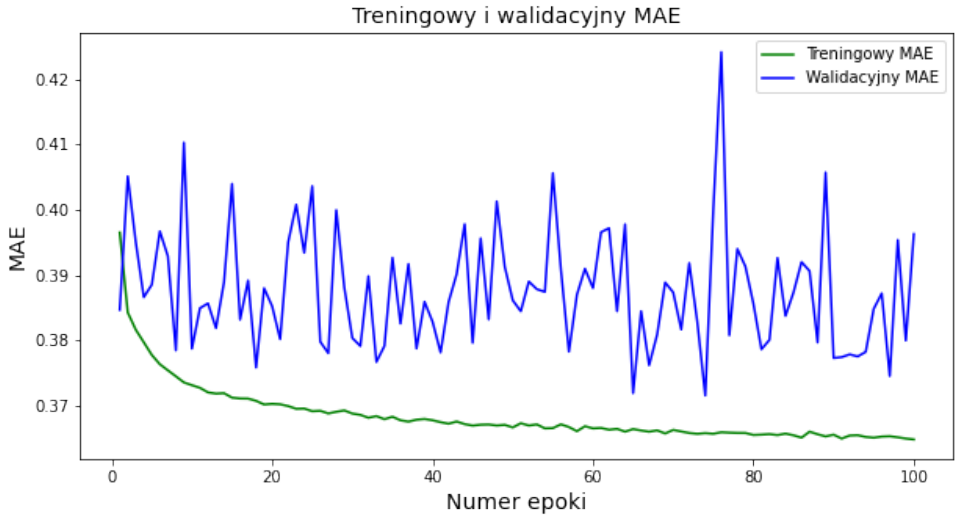
Po wykresie funkcji kosztu przedstawionym na rysunku 3.9 widać, że nowy model krócej się uczył. Treningowa funkcja kosztu maleje najbardziej przez pierwsze 20 epok, potem praktycznie się wypłaszcza. Walidacyjna funkcja kosztu nadal fluktuuje, tym razem jednak wokół tych samych wartości, nie zmniejsza się, jedynie przy pierwszych 20 epokach widać pewną tendencję spadkową.

Analogicznie w przypadku metryki, po 20 epoce krzywa treningowego *mean absolute error* zaczyna opadać w bardzo wolnym tempie. Walidacyjny MAE praktycznie się nie zmniejsza - pod koniec uczenia

oscyluje na poziomie tych samych wartości co na początku. Obie krzywe zostały zaprezentowane na rysunku 3.10.



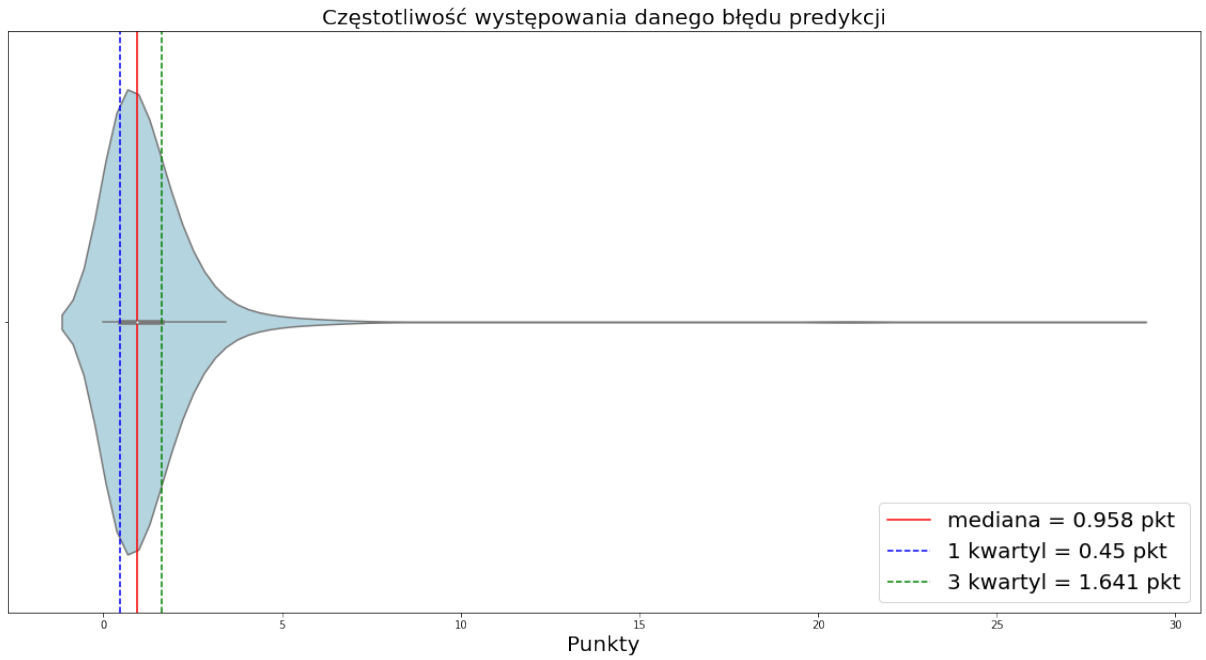
Rys. 3.9. Wykres kosztu w funkcji epoki



Rys. 3.10. Wykres MAE w funkcji epoki

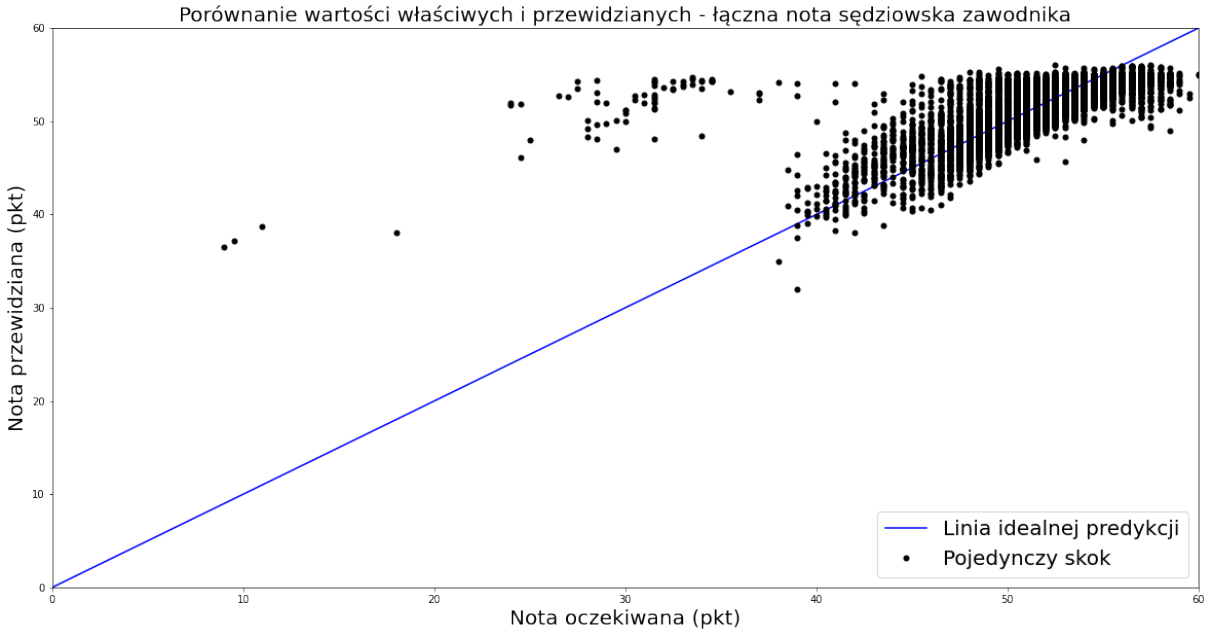
Do oceny jakości prognozy użyto, tak jak przy dwóch poprzednich modelach, MAE, który został ustalony na podstawie obliczonych przez funkcję `predict` ocen za styl skoków ze zbioru testowego. Jego wartość dla MLPOPT wyniosła 1.282. Sprawia to, że błąd przewidywań stanowi około 2.48% średniej noty sędziowskiej zawodnika.

Na rysunku 3.11 zaprezentowano wykres skrzypcowy dla błędu predykcji omawianego modelu. Mediana błędu wyniosła 0.958 punktu. 75% ocen za styl zostało przewidzianych z dokładnością nie mniejszą niż do 1.641 punktu.



Rys. 3.11. Wykres skrzypcowy błędu predykcji łącznej noty sędziowskiej skoczka dla modelu zoptymalizowanego

Wizualizację jakości przewidywań na zbiorze testowym stanowi wykres z rysunku 3.12. Zaznaczono na nim, oprócz uzyskanych i prawdziwych ocen za styl, także linię wyznaczającą idealną predykcję. Można zauważyć, że skoki, które powinny zostać ocenione bardzo nisko otrzymują od modelu znacznie więcej punktów niż powinny. Nieco mniejszy błąd, jednak też widoczny występuje w przypadku bardzo wysokich not - są one zaniżane przez model.



Rys. 3.12. Porównanie noty przewidzianej z oczekiwaną - model o mniejszej liczbie cech na wejściu

4. Analiza

W poniższym rozdziale zostaną wskazane podobieństwa i różnice pomiędzy trzema modelami zbudowanymi w poprzedniej części pracy. Po wybraniu najlepszego z nich, zostanie on poddany analizie na podstawie konkretnego konkursu skoków. Oprócz tego, poważne błędy przewidywań zostaną wykorzystane do wykrycia anomalii w obserwacjach.

4.1. Porównanie jakości modeli

Dla trzech modeli scharakteryzowanych na wcześniejszym etapie projektu wybrano te same metryki jakości. Wartość MAE obliczona na zbiorze testowym służy jako główny czynnik do oceny błędu. Oprócz niego, sprawdzono także medianę oraz górny i dolny kwartyl zbioru błędów. Atrybuty te zostały zebrane w tabeli 4.1.

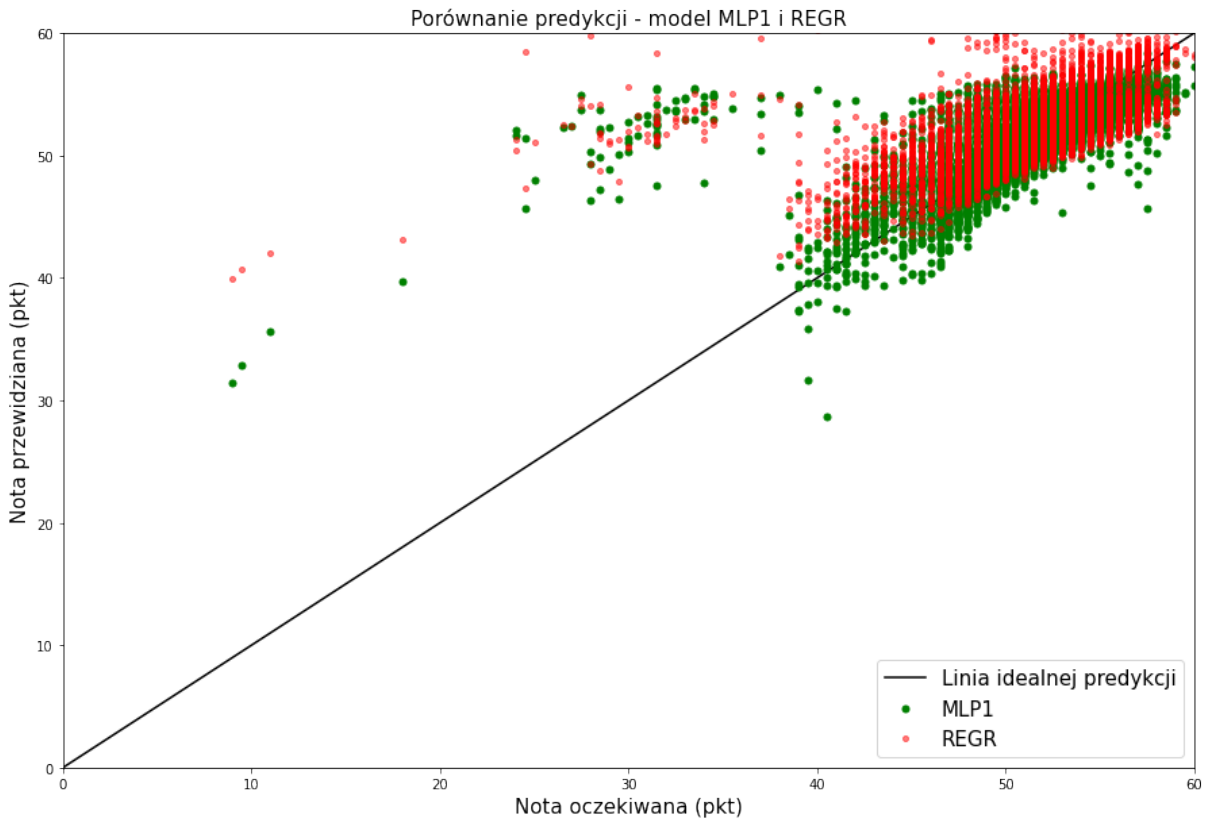
Tabela 4.1. Parametry błędu przewidywań trzech modeli

	REGR	**MLP1**	**MLPOPT**
MAE (pkt)	1.773	1.216	1.282
mediana błędu (pkt)	1.459	0.877	0.958
górny kwartyl (pkt)	0.696	0.411	0.45
dolny kwartyl (pkt)	2.373	1.54	1.641

Widać, że model regresyjny znacznie odbiega od pozostałych. Jego przewidywania są mniej dokładne, a mediana błędu prawie dwa razy wyższa niż w przypadku MLP1. Biorąc pod uwagę jakość pozostałych dwóch modeli można stwierdzić, że REGR jest niewystarczający do rozwiązania problemu predykcji oceny sędziowskiej skoczka narciarskiego.

Przed podjęciem ostatecznej decyzji co do jego eliminacji, zdecydowano się zestawić model regresyjny z MLP1. Rysunek 4.1 przedstawia jak oba modele różnią się pod względem określania oceny sędziowskiej skoczka w porównaniu z rzeczywistą, wyznaczoną przez arbitra notą. Warto zauważyć, że REGR, w przeciwieństwie do MLP1 radzi sobie z predykcją ocen wyższych niż 55 punktów. MLP1 praktycznie nie jest w stanie przewidzieć takiej noty, podczas gdy dla REGR pojawiają się nawet oceny 60-punktowe.

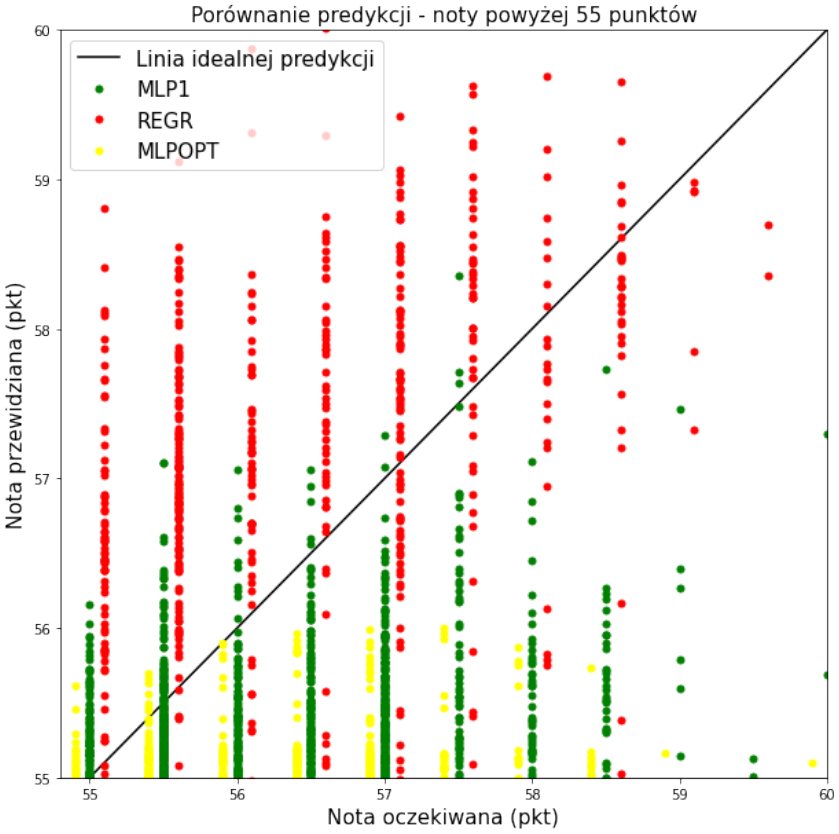
Również dla niższych not, które MLP1 ustalił na około 40 punktów, REGR wyznaczył ocenę rzędu 45-50 punktów. Pamiętając, że oba modele sekwencyjne miały duże błędy przy predykcji bardzo niskich not, widać, że model regresyjny ocenia nieudane skoki jeszcze wyżej. Można więc stwierdzić, że REGR znacznie zawyża niskie noty sędziowskie. Widać także, że w porównaniu z MLP1 o wiele rzadziej przewiduje on niższe noty od modelu sekwencyjnego.



Rys. 4.1. Zestawienie predykcji dwóch modeli - MLP1 i REGR

Sprawdzono także jak REGR, MLP1 i MLPOPT przewidują noty wyłącznie powyżej 55 punktów. Tak jak poprzednio, dokonano predykcji na zbiorze testowym przez wszystkie trzy modele. Wykres na rysunku 4.2, który przedstawia porównanie ich działania został celowo ograniczony do wartości [55,60], aby jak najlepiej oddać ich jakość. Ponadto, noty dwóch modeli zostały przesunięte od właściwej o $\pm$ 0.1 punktu, aby nie nachodziły na siebie na wykresie.

Widać, że jedynie oznaczony czeronymi punktami REGR jest w stanie przewidywać noty wyższe niż 57.5 punktu. MLP1 i MLPOPT mogły mieć problem z nauczeniem się, dla jakich skoków należy wystawić taką ocenę, ponieważ w zbiorze jest ich bardzo mało (jedynie 226). Możliwe także, że loty, które zostają bardzo dobrze ocenione nie odbywają się w charakterystycznych warunkach - daną obserwację od innych może wyróżniać jedynie subiektywna nota za styl, co w połączeniu z niewielką liczbą próbek utrudnia predykcję.



Rys. 4.2. Zestawienie predykcji wszystkich modeli dla wysokich not

Omówione wcześniej rysunki 3.2, 3.8 i 3.12 pokazują, że noty przewidziane przez MLP1 i MLPOPT w całym zakresie układają się lepiej wzdłuż linii idealnej predykcji niż te wyznaczone przez model regresyjny. REGR, chociaż zdolny do przewidywania wysokich ocen, również się w nich myli. Wyznaczając notę 58.5 - punktową zamiast 57, popełnia większy błąd niż modele sekwencyjne zdolne w tych warunkach do predykcji rzędu 56 punktów. Ostatecznie zdecydowano się go odrzucić ze względu na większe błędy i porównać ze sobą dwa pozostałe modele.

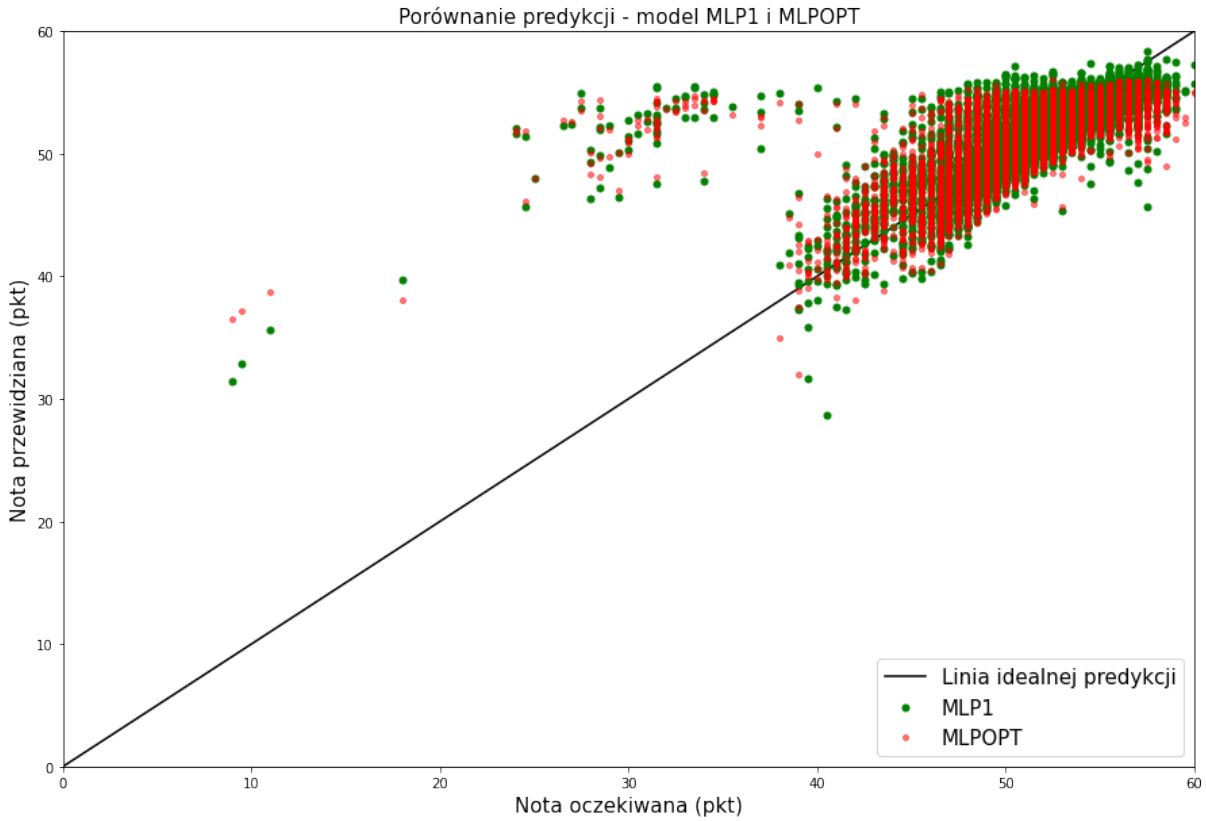
Oprócz MAE na zbiorze testowym, dla modeli uczonych za pomocą uczenia maszynowego wyznaczono także MAE na zbiorze walidacyjnym pod koniec treningu. Ponieważ i pod tym względem modele były bardzo podobne, każdy z nich poddano dziesięciokrotnemu uczeniu, a wielkość metryk pochodzących z każdego z treningów zgromadzono w tabeli 4.2.

Dla obu modeli MAE jest zbliżony przez te 10 treningów. MLP1 potrafi osiągnąć niższy wynik, rzędu 0.36-0.37, MLPOTP natomiast ma mniejsze wahania metryki, utrzymuje się ona w okolicach 0.38-0.39. Średni MAE po 10 treningach wyniósł 0.385 dla MLP1 i 0.389 dla MLPOPT. Jest to wynik nieznacznie korzystniejszy dla większego z modeli.

Tabela 4.2. Porównanie metryk (MAE) dla modeli MLP1 i MLPOPT

	MLP1	**MLPOPT**
1	0.3848	0.3835
2	0.3719	0.3818
3	0.3968	0.3830
4	0.3689	0.3884
5	0.3741	0.3827
6	0.4065	0.3980
7	0.3849	0.3963
8	0.3794	0.3975
9	0.3914	0.3968
10	0.3874	0.3804
Średnia	**0.385**	**0.389**

Sprawdzono również jak na wykresie prezentuje się porównanie predykcji dokonanych przez MLP1 i MLPOPT. Na rysunku 4.3 można zobaczyć jak modele są do siebie zbliżone pod względem przewidywań i jak odbiegają od rzeczywistych wartości.



Rys. 4.3. Zestawienie predykcji dwóch modeli - MLP1 i MLPOPT

Widać, że MLP1 i MLPOPT przewidują dużo bardziej zbliżone do siebie noty niż MLP1 i REGR. Punkty układają się bliżej linii idealnej predykcji. Różnice pomiędzy nimi są niewielkie. Można jednak odczytać z wykresu, że dla kilku bardzo niskich not, MLPOPT wyznaczył oceny wyższe niż jego odpowiednik o większej liczbie cech na wejściu. Nie można zapomnieć o tym, że noty wskazane przez model na około 30 punktów, w rzeczywistości były o wiele niższe. Oznacza to, że MLPOPT jeszcze bardziej zawyżył niskie oceny. Odwrotnie ma się sytuacja w przypadku not wysokich - MLPOPT przewiduje je na nieco niższym poziomie niż MLP1. Dla niektórych skoków, dla których MLP1 wyznacza ocenę rzędu 55-57 punktów, MLPOPT ustala nawet mniej niż 50 punktów. Należy pamiętać o tym, że oba modele mają problem z przyznawaniem zawodnikom wysokich ocen. Można więc stwierdzić, że MLPOPT je zaniża.

Biorąc pod uwagę zebrane wykresy i tabele zdecydowano się wykorzystać do dalszego użytku model MLP1. Chociaż wymaga on znajomości większej liczby parametrów skoku (siedem zamiast trzech) oraz dłuższego uczenia (100 epok zamiast 25) jest on dokładniejszy w swoich przewidywaniach od mniejszego odpowiednika. Nawet jeżeli różnice te są nieznaczne, mogą one odgrywać dużą rolę w obliczeniu ostatecznego wyniku konkursu, a co za tym idzie mieć wpływ na lokaty, punkty Pucharu Świata, a nawet wynagrodzenie zawodników.

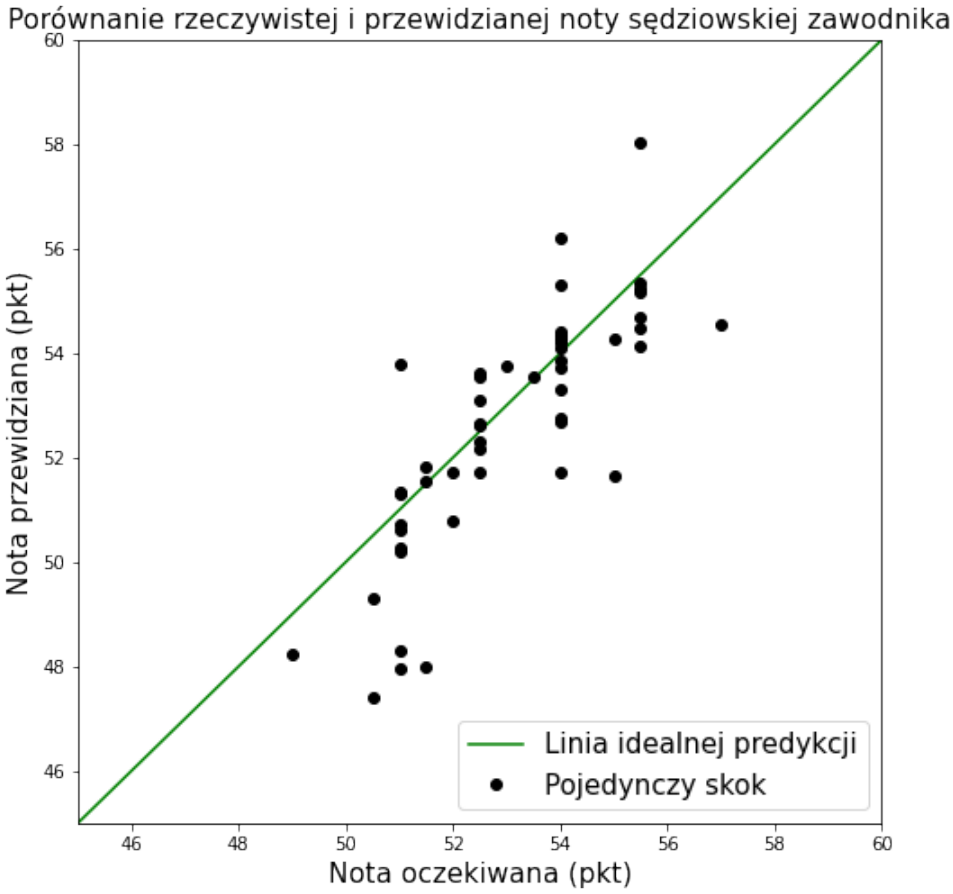
4.2. Predykcja wyniku konkursu skoków

Aby sprawdzić jak model radzi sobie w realnych warunkach, postanowiono zastosować go do wyznaczenia wyników konkursu Pucharu Świata w skokach narciarskich, który odbył się w Wiśle 6 listopada 2022 roku. Ze względu na późniejszą datę od daty pozyskania danych do modelu, skoki z konkursu nie znalazły się w żadnym z poprzednio analizowanych zbiorów. W zawodach wzięło udział 50 zawodników, 30 z nich awansowało do drugiej serii. Model miał za zadanie przewidzieć wyniki osobno pierwszej i drugiej serii. W przypadku, gdyby awansował zawodnik, który drugiego skoku nie oddał, jego wynik z pierwszej części zawodów zostałby zaliczony na poczet drugiej serii.

Dane dotyczące zawodników biorących udział w konkursie oraz oddanych przez nich skoków zostały pozyskane ze strony FIS [11]. Na ich podstawie utworzono plik `.csv`, który posiadał podobne kolumny, jak pliki użyte do trenowania modelu - łączną notę zawodnika, prędkość wybicia z progu, odległość oraz punkty za nią uzyskane, notę sędziowską, belkę, wiatr i odpowiadające im rekompensaty, a także uzyskaną lokatę (dotyczy tylko pierwszej serii) i nazwisko sportowca, dla sprawdzenia wyników przewidywań z rzeczywistymi. Z dwóch ostatnich oraz łącznej noty zawodnika zbudowano osobny zbiór, niezbędny do porównania wyników końcowych. Do predykcji użyty został model MLP1. Dane przed przepuszczeniem ich przez model zostały znormalizowane. Konkurs odbywał się w równych warunkach, wszyscy zawodnicy startowali z tej samej belki, a co za tym idzie rekompensata za nią wyniosła 0 punktów. Wpłynęło to na normalizację danych, ponieważ odchylenie standardowe obydwu tych parametrów wyniosło zero, a w zbiorze danych po normalizacji pojawiła się wartość NaN (ang. *Not a Number*). Została ona zastąpiona liczbą 0.

Pierwsza seria

Skoki 50 uczestników pierwszej rundy zostały wprowadzone do modelu. Rysunek 4.4 przedstawia, z jaką dokładnością obliczone zostały noty sędziowskie skoczków - porównuje on wynik działania modelu z rzeczywistymi danymi. Zarówno osie x jak i y wykresu zostały wyskalowane tak, aby zawierać jedynie wartości od 45 do 60 punktów, ponieważ w konkursie nie pojawiły się niższe noty.



Rys. 4.4. Porównanie noty przewidzianej z oczekiwaną - konkurs Pucharu Świata w Wiśle, pierwsza seria

Można zauważyć, że oceny leżą dość blisko zielonej linii oznaczającej idealną predykcję. Odchylenia występują w obie strony - wyznaczona przez MLP1 nota może być większa lub mniejsza od rzeczywistej. Najdokładniej wskazane zostały najliczniej występujące, przeciętne oceny.

W tabeli 4.3 zebrano pełne wyniki pierwszej serii zawodów - zarówno te autentyczne, na których bazował model, jak i te przez niego przewidziane. Porównano także lokaty, które zajęli skoczkowie. Co istotne, nota sędziowska wynikająca z obliczeń modelu nie została zaokrąglona do najbliższego pół punktu. Jako PU zostały oznaczone punkty, które zawodnik rzeczywiście uzyskał w zawodach, PP - punkty przewidziane, LU - lokata, którą zajął zawodnik po pierwszej serii, LP - lokata wynikająca z obliczeń modelu.

Tabela 4.3. Wyniki konkursu Pucharu Świata w Wiśle - 1 seria

Nazwisko	PU	PP	Różnica pkt	LU	LP	Różnica lokat
Kubacki	143.1	142.834	-0.266	1	1	-
Lanisek	141.9	139.431	-2.469	2	2	-
Lindvik	136.9	139.108	2.208	4	3	+1
Granerud	135.9	134.882	-1.018	5	6	-1
Kraft	133.9	133.090	-0.810	8	9	-1
Hoerl	135.1	137.628	2.528	6	4	+2
Tande	133.7	132.986	-0.714	9	10	-1
Zyla	129.1	127.793	-1.307	13	18	-5
Fettner	133.4	134.697	1.297	10	7	+3
Zajc	134.8	134.465	-0.335	7	8	-1
Nakamura	132.9	131.551	-1.349	11	11	-
Zschenwald	127.7	128.467	0.767	17	14	+3
Johansson	127.7	127.981	0.281	17	16	+1
Forfang	128.3	128.498	0.198	14	13	+1
Paschke	125.9	126.251	0.351	20	20	-
Prevc, Peter	128.2	127.871	-0.329	15	17	-2
Geiger	130.7	127.335	-3.365	12	19	-7
Prevc, Domen	128.2	128.299	0.099	15	15	-
Raimund	127.5	130.284	2.784	19	12	+7
Eisenbichler	123.8	123.666	-0.134	25	24	+1
Zografski	125.8	125.935	0.135	21	21	-
Wasek	124.9	125.299	0.399	23	22	+1
Hayboeck	123.8	123.120	-0.680	25	28	-3
Tschofenig	124.7	123.438	-1.262	24	27	-3
Wellinger	125.2	124.918	-0.282	22	23	-1
Schmid	121.7	119.425	-2.275	30	31	-1
Bresadola	122.5	123.620	1.120	29	26	+3
Kos	122.6	123.657	1.057	28	25	+3
Sundal	122.7	122.734	0.034	27	29	-2
Kobayashi	137.6	137.453	-0.147	3	5	-2
Juroszek	120.1	119.313	-0.787	31	32	-1
Aalto	119.4	119.998	0.598	32	30	+2
Kytosaho	118.9	119.025	0.125	33	33	-
Habdas	117.9	114.392	-3.508	34	40	-6
Nousiainen	117.7	118.019	0.319	35	34	+1

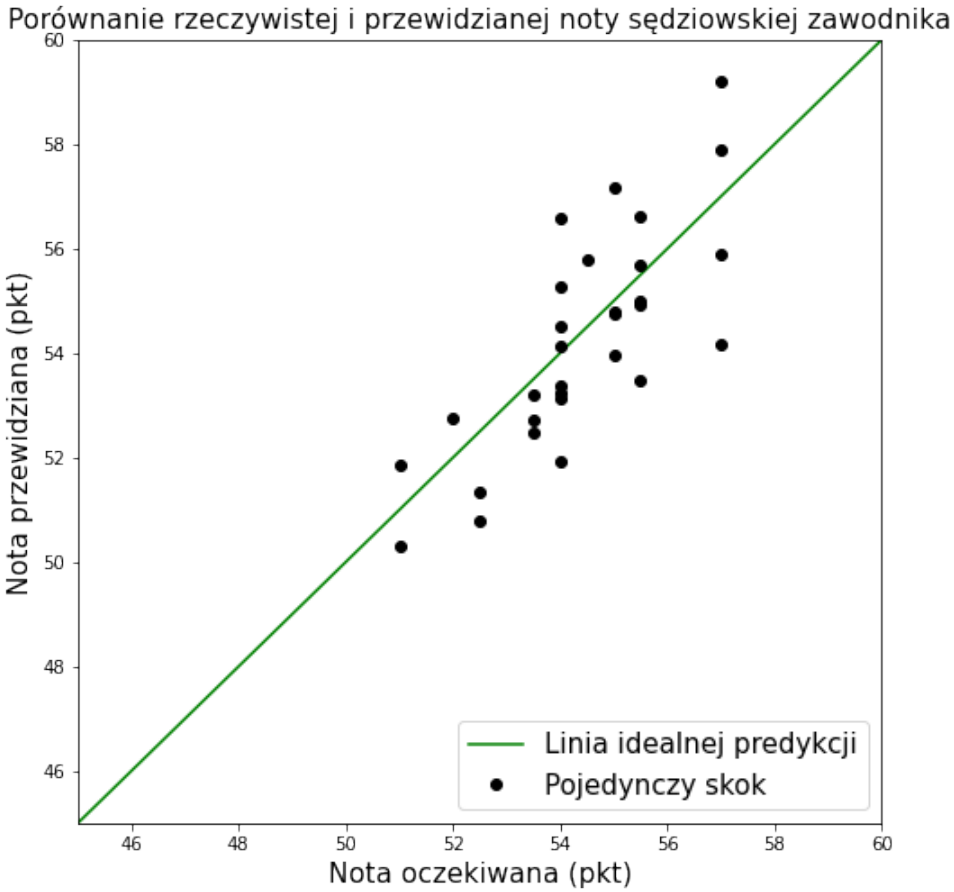
Insam	117.7	117.422	-0.278	35	35	-
Hula	116.1	114.905	-1.195	37	38	-1
Nikaido	115.9	115.945	0.045	38	37	+1
Sato	115.9	116.249	0.349	38	36	+2
Wolny	115.1	114.368	-0.732	40	41	-1
Jelar	114.4	114.212	-0.188	41	42	-1
Aigro	114.3	114.613	0.313	42	39	+3
Villumstad	113.8	113.513	-0.287	43	43	-
Vassilyev	111.9	109.196	-2.704	44	46	-2
Zniszczol	111.4	108.365	-3.035	45	47	-2
Palosaari	110.6	109.391	-1.209	46	44	+2
Ipcioglu	110.1	109.320	-0.780	47	45	+2
Deschwanden	108.7	108.312	-0.388	48	48	-
Sato	106.3	105.506	-0.794	49	49	-
Kot	99.8	96.700	-3.100	50	50	-

Rozbieżności punktowe pomiędzy rzeczywistymi wynikami, a tymi wskazanymi przez model są niewielkie. Największa z nich to 3.5 punktu, co przekłada się na 6 lokat. Należy pamiętać o tym, że jest to tylko pierwsza seria, więc zróżnicowanie ocen na poziomie 0.1 punktu, które ma przełożenie na zmianę lokaty może się zmienić po drugiej części zawodów. Gdyby brać pod uwagę wyniki wyznaczone przez MLP1 zawodnik z 32 miejsca, Antti Aalto awansowałby do dalszego etapu konkursu zamiast Constantina Schmida.

Seria finałowa

Do serii finałowej awansowało 30 zawodników. Uwzględniając różnice w predykcji, do modelu został przekazany jeden skoczek, który realnie nie osiągnął wystarczającego do promocji do drugiej rundy wyniku. Jako jego skok finałowy zaliczono ten z pierwszej serii.

Na rysunku 4.5 pokazano, podobnie jak w przypadku pierwszej serii porównanie rzeczywistych not sędziowskich i tych przewidzianych przez model. Osie wykresu zostały wyskalowane tak, aby nie pokazywać ocen mniejszych niż 45 punktów, ponieważ nie pojawiły się one w finałowym etapie konkursu. Noty wydają się leżeć jeszcze bliżej linii idealnej predykcji niż miało to miejsce w przypadku poprzedniej rundy.



Rys. 4.5. Porównanie noty przewidzianej z oczekiwaną - konkurs Pucharu Świata w Wiśle, seria finałowa

Tabela 4.4 zawiera przewidziane wyniki łączne konkursu. Dla drugiej serii nie badano już osobnych lokat, zdecydowano się na podstawie obliczonych rezultatów z pierwszego etapu zawodów i predykcji not z finałowej części konkursu wyznaczyć jego ostateczne wyniki. Podobnie jak w przypadku tabeli 4.3, jako PU oznaczono punkty, które zawodnik uzyskał w zawodach, PP - punkty przewidziane przez model, LU - lokata, którą zajął zawodnik w konkursie, LP - lokata wynikająca z obliczeń modelu.

Spośród 30 oddanych zawodników, 11 z nich zajęłoby inną lokatę, gdyby wziąć pod uwagę wynik wyjściowy z modelu. Największy błąd punktowy przewidywań obydwu części zawodów wyniósł 2.819 punktu. Co ciekawe, nie wpłynął on na zajęte przez zawodnika miejsce - wygrał on z przewagą ponad ośmiu punktów nad skoczkiem z drugiego miejsca. Bardzo możliwe, że jakość jego skoków wpłynęła na błąd modelu - w poprzedniej sekcji rozdziału zostało wspomniane, że MLP1 ma problem z wyznaczaniem wysokich not sędziowskich, a zwycięzca konkursu za swój drugi skok osiągnął ocenę 57 punktów. Przewidywania były bardzo zbliżone do rzeczywistych wyników u zawodników, którzy oddali najgorsze skoki. Większe błędy predykcji pojawiły się u skoczków, którzy zajęli miejsca w pierwszej dziesiątce zawodów. Zaszłaby nawet zmiana na podium, pomiędzy drugim a trzecim miejscem.

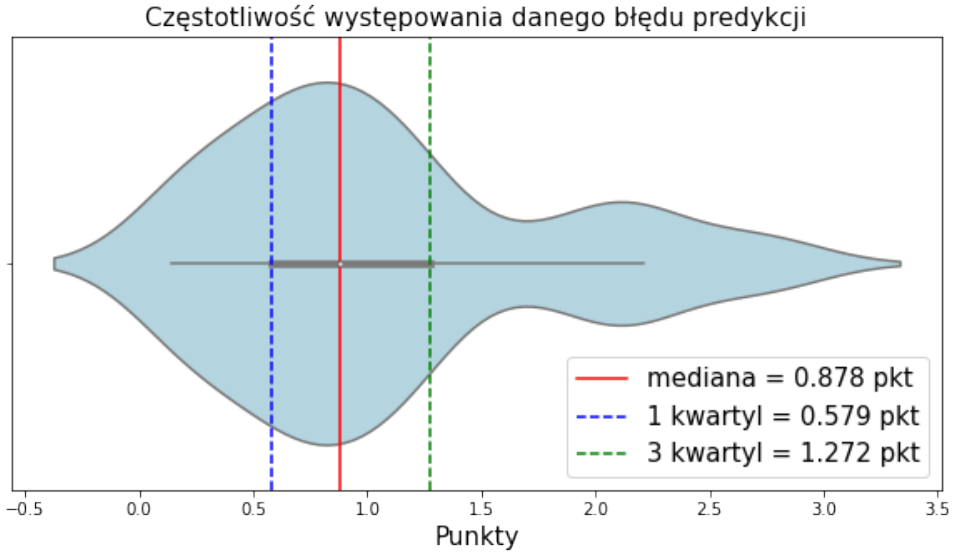
Tabela 4.4. Wyniki konkursu Pucharu Świata w Wiśle - cały konkurs

Nazwisko	PU	PP	Różnica pkt	LU	LP	Różnica lokat
Kubacki	287.0	284.181	-2.819	1	1	-
Lanisek	278.7	277.578	-1.122	2	3	-1
Lindvik	277.0	279.209	2.209	3	2	+1
Granerud	273.5	273.674	0.174	4	6	-2
Kobayashi	273.4	275.575	2.175	5	4	+1
Kraft	272.7	275.296	2.596	6	5	+1
Hoerl	272.0	272.905	0.905	7	7	-
Tande	268.8	269.931	1.131	8	8	-
Zyla	267.7	265.682	-2.018	9	9	-
Fettner	263.7	264.975	1.275	10	10	-
Zajc	263.4	262.892	-0.508	11	11	-
Nakamura	262.1	261.532	-0.568	12	12	-
Aschenwald	258.1	257.898	-0.202	13	14	-1
Johansson	257.6	256.569	-1.031	14	16	-2
Forfang	257.3	258.561	1.261	15	13	+2
Paschke	257.1	257.622	0.522	16	15	+1
Prevc, Peter	256.5	256.245	-0.255	17	17	-
Geiger	256.4	255.789	-0.611	18	18	-
Prevc, Domen	254.9	255.038	0.138	19	19	-
Raimund	252.8	251.767	-1.033	20	20	-
Eisenbichler	250.4	250.089	-0.311	21	21	-
Zografski	249.8	249.048	-0.752	22	22	-
Wasek	249.1	248.249	-0.851	23	24	-1
Hayboeck	248.1	248.858	0.758	24	23	+1
Tschofenig	247.9	247.107	-0.793	25	25	-
Wellinger	247.7	245.626	-2.074	26	26	-
Bresadola	242.9	243.751	0.851	27	27	-
Kos	242.4	241.239	-1.161	28	28	-
Aalto	238.8	237.096	-1.704	29	29	-
Sundal	236.9	236.218	-0.682	30	30	-

Po pierwszej części zawodów, wiele lokat przewidzianych przez model nie zgadzało się z tymi, które skoczkowie rzeczywiście zajęli. Jednak miejsca zajęte w pierwszej serii odzwierciedlają jedynie jakość skoku oddanego przez zawodnika i definiują kolejność startową serii finałowej. Jako ostateczny wynik konkursu uznaje się się miejsce zajęte po drugiej części zawodów, wyznaczone poprzez sumę punktów

z obu etapów. Dla przykładu zawodnicy Karl Geiger oraz Philipp Raimund, których lokaty po pierwszej serii najbardziej odbiegały od rzeczywistych, zajęli w zawodach przewidziane przez model pozycje.

Rysunek 4.6 przedstawia wykres skrzypcowy błędu predykcji wyników punktowych konkursu w Wiśle. Ma on nieco inny kształt niż te tworzone dla oceny jakości modeli - ze względu na mniejszą ilość danych i rozbieżność błędów, lepiej widoczny jest ich rozkład. Na wykresie zaznaczono także medianę oraz górny i dolny kwartyl.



Rys. 4.6. Rozkład błędów predykcji punktów w konkursie skoków w Wiśle

Wynik działania modelu jest odpowiedni. MAE predykcji konkursu wyniósł 1.083 punktu, a połowa wyników została przewidziana z dokładnością nie mniejszą niż 0.9 punktu. Nie satysfakcjonuje jedynie zmiana na podium, ponieważ oznaczałaby ona stratę pieniężną oraz punktową w klasyfikacji generalnej dla zawodnika "pokrzywdzonego"przez model.

4.3. Wykrywanie anomalii

Żaden z porównywanych ze sobą w projekcie modeli nie potrafił przewidzieć niskich ocen sędziowskich zawodników. Sytuacje, kiedy skoczek otrzymuje notę poniżej około 35 punktów zdarzają się bardzo rzadko i często w rzeczywistości oznaczają upadek. Podczas omawiania problemu wskazano, że jeśli zawodnik przewróci się przed uprzednio wyznaczoną linią, straci on siedem punktów z oceny każdego sędziego - wówczas maksymalna nota za styl, jaką może uzyskać to już tylko 39 punktów. Zamiast usuwać z modelu niskie wyniki celem polepszenia jego działania dla częściej występujących not zdecydowano się pozostawić i przeanalizować wszystkie przypadki, dla których MLP1 przewidział znacznie wyższą notę niż zawodnik powinien otrzymać. W ten sposób, na podstawie wyniku modelu można stwierdzić, czy wystąpił upadek, czy skoczek normalnie wylądował. Jako próg wykrywania anomalii

ustalono błąd predykcji w wysokości co najmniej 20 punktów. Dalsza analiza ma na celu zbadanie jak dobrze model radzi sobie z wykryciem upadku.

Na etapie przetwarzania danych wejściowych do modelu utworzono dodatkowy obiekt typu `Dataframe`, w którym zapisano identyfikator konkursu i zawodnika, aby móc je później odnaleźć w archiwalnych nagraniach i wynikach. Każda z anomalii musiała zostać sprawdzona ręcznie - tylko poprzez obejrzenie skoku czy przeczytanie informacji można stwierdzić, że zakończył się on upadkiem. Zbadanie bezwzględnych różnic pomiędzy wartością przewidzianą przez MLP1 wskazało 52 anomalie, czyli błędy predykcji powyżej 20 punktów. Najbardziej znaczący z nich wyniósł 28.07 punktu. Poniżej zostanie omówione szczegółowo 5 największych anomalii. Pozostałe zostaną jedynie sprawdzone pod kątem tego, czy wystąpił upadek i ujęte we wspólnym zestawieniu.

1. Największą rozbieżność w stosunku do noty rzeczywistej skoczka osiągnął zawodnik o identyfikatorze 5891 - Jaka Hvala w konkursie Pucharu Kontynentalnego (zawodów niższej rangi niż Puchar Świata) w 2017 roku w Sapporo. Jego łączna nota sędziowska wyniosła wówczas jedynie 24 punkty. MLP1 przewidział ocenę za styl w wysokości 52.07 punktu, czyli przeciętną. Ze względu na niższą rangę zawodów, a także ich lokalizację (konkursy w Japonii nie przyciągają wielu widzów ze względu na nocną porę emisji) trudno było znaleźć nagranie lub informacje na temat tego skoku. Komentarze pod artykułem dotyczącym tych zawodów [7] wskazują jednak na to, że zakończył się on upadkiem.

2. Różnica wielkości 27.62 punktu pomiędzy notą przewidzianą a oczekiwaną została wyznaczona dla zawodnika z identyfikatorem 4801. Był to skok Romana Koudelki w konkursie Pucharu Świata w Willingen w 2018 roku, za który dostał ocenę od sędziów w wysokości 24 punktów. Na podstawie znalezionych nagrań z tych zawodów [9] można stwierdzić, że zawodnik upadł w czasie lądowania, co widoczne jest na rysunku 4.7.

Rys. 4.7. Upadek Romana Koudelki

3. Wyznaczona nota dla zawodnika o identyfikatorze 7835 była niższa od rzeczywistej o 27.48 punktu. Jest to ocena za skok Kacpra Juroszka w zawodach FIS Cup (najniższej rangi) w Kuopio w 2021 roku. W artykule opisującym konkurs [6] można znaleźć informacje o tym, że skok zakończył się niegroźnym upadkiem.

4. Czwarta anomalia rzędu 26.9 punktu dotyczy zawodnika o identyfikatorze 6671 - Kevin Bickner w trakcie zawodów Pucharu Świata na skoczni mamuciej w Vikersund nie ustał swojej próby i został zniesiony ze skoczni na noszach. Rysunek 4.8 pochodzący z nagrań zawodów [23] prezentuje jego upadek.

Rys. 4.8. Upadek Kevina Bicknera

5. Na temat piątej znacznej rozbieżności między wynikami nie udało się znaleźć informacji. Na 27.5 punktu został oceniony skok Floriana Altenburgera w zawodach Pucharu Kontynentalnego, jednak tak jak w przypadku pierwszej anomalii, konkurs odbywał się w porze nocnej w Sapporo, więc dostęp do danych był utrudniony. Nie rozstrzygnięto, czy skok zakończył się upadkiem. Na podstawie teorii oceniania skoków narciarskich z pierwszego rozdziału projektu można jednak przypuszczać, że zawodnik nie ustał swojego próby - gdy przewróci się on przed linią upadku może uzyskać maksymalnie notę w wysokości 39 punktów. Ocena uzyskana przez Altenburgera wyniosła jeszcze mniej - 27.5 punktu.

Spośród wszystkich wyznaczonych anomalii, upadkami okazało się 41. 79% not przewidzianych przez model, które różnią się od tych rzeczywistych o co najmniej 20 punktów nie zakończyło się poprawnym lądowaniem. Biorąc pod uwagę opisaną w pierwszym rozdziale teorię oceniania skoku narciarskiego, bardzo możliwe, że wszystkie ze wspomnianych anomalii oznaczało upadki. Jednak ze względu na brak odpowiednich informacji, wynikających z niskiej popularności tej dziedziny sportu w niektórych krajach lub niskiej rangi zawodów, zachowano pewien procent niepewności.

Im mniejsza rozbieżność pomiędzy notą rzeczywistą zawodnika, a tą wyznaczoną przez model, tym mniejsze szanse na to, że skok zakończył się upadkiem. Błędy rzędu trzech punktów wynikają raczej z niedostosowania modelu, te kilkunastopunktowe mają już większą szansę oznaczać nieudane lądowanie.

Podsumowanie

W ramach projektu został zbudowany model umożliwiający przewidywanie noty sędziowskiej skoczka narciarskiego na podstawie cech oddanego przez niego skoku. W celu rozwiązania zadanego problemu powstały trzy modele - jeden prosty - regresyjny i dwa typu MLP oparte o SNN. Każdy z nich był w stanie ustalić ocenę za styl skoczka z pewną dokładnością. Najlepszym modelem okazał się MLP1 o siedmiu cechach na wejściu - jego ustalenia najbardziej odpowiadały rzeczywistym wartościom wyznaczonym przez arbitrów. Działanie MLP1 zostało zweryfikowane na przykładzie zawodów, które nie znalazły się w żadnym ze zbiorów służących do treningu czy testowania modelu. Osiągał w nich niewielkie błędy, średnio 1.2 punktu, co stanowi około 2.36% przeciętnej oceny zawodnika. Są to wartości wystarczająco niskie, aby móc stwierdzić, że rozwiązuje on zadany problem.

Największą trudnością okazała się predykcja zarówno bardzo wysokich not, jak i tych skrajnie niskich. W obu przypadkach obserwacje o zadanych ocenach rzadko pojawiają się w zbiorach danych - wybitne skoki czy upadki nie zdarzają się w czasie każdych zawodów. Kwestię przewidywania not powyżej 55 punktów najlepiej rozstrzygał model regresyjny, jednak z powodu większych różnic pomiędzy jego ustaleniami a wartościami rzeczywistymi zrezygnowano z jego użycia. Znaczące błędy zostały wykorzystane do wykrywania lotów zakończonych nieudanym lądowaniem. Wykazano, że jeśli rozbieżność między notą przyznaną przez arbitra, a tą wyznaczoną przez model jest istotna, to zawodnik najpewniej nie ustał swojej próby.

Model predykcji oceny sędziowskiej skoczka narciarskiego może zostać rozwinięty. W pierwszej kolejności należałoby się skupić na rozwiązaniu kwestii wyznaczania not skrajnych. Kolejno, można poprawiać dokładność MLP1 i badać związek not za styl z parametrami, które nie zostały wzięte pod uwagę w tym projekcie, takimi jak skocznia (jej położenie geograficzne, wielkość) czy cechy charakterystyczne skoczków (wiek, wzrost, waga). Odpowiednio dostosowany model będzie mógł z jeszcze większą precyzją wypełniać zadanie arbitrów.

Bibliografia

[1] Firoz Alam, Harun Chowdhury i Hazim Moria. „A review on aerodynamics and hydrodynamics in sports". W: *Energy Procedia* 160 (lut. 2019), s. 798–805. DOI: *10.1016/j.egypro.2019.02.158*.

[2] Abdulrahman Alassadi i Tadas Ivanauskas. „Classification Performance Between Machine Learning and Traditional Programming in Java". W: (2019).

[3] Dan Becker. *Rectified Linear Units (ReLU) in Deep Learning*. [online]. Paź. 2022. URL: *https://www.kaggle.com/code/dansbecker/rectified-linear-units-relu-in-deep-learning/notebook*.

[4] Facundo Bre, Juan Gimenez i Víctor Fachinotti. „Prediction of wind pressure coefficients on building surfaces using Artificial Neural Networks". W: *Energy and Buildings* 158 (list. 2017). DOI: *10.1016/j.enbuild.2017.11.045*.

[5] Heike Brock, Yuji Ohgi i Kazuya Seo. „Development of an Automated Motion Evaluation System from Wearable Sensor Devices for Ski Jumping". W: *Procedia Engineering* 147 (grud. 2016), s. 694–699. DOI: *10.1016/j.proeng.2016.06.248*.

[6] Adam Bucholz. *FC w Kuopio: Zwycięstwo Lacknera, upadki Polaków*. [online]. List. 2022. URL: *https://www.skijumping.pl/wiadomosci/30063/fc-w-kuopio-zwyciestwo-lacknera-upadki-polakow/*.

[7] Adam Bucholz. *PK w Sapporo: Triumf Aignera, Zniszczoł trzeci!* [online]. List. 2022. URL: *https://www.skijumping.pl/wiadomosci/22600/pk-w-sapporo-triumf-aignera-zniszczol-trzeci/*.

[8] Mingzhe Chen i in. „Artificial Neural Networks-Based Machine Learning for Wireless Networks: A Tutorial". W: *IEEE Communications Surveys and Tutorials* 21.4 (2019), s. 3039–3071. DOI: *10.1109/COMST.2019.2926625*.

[9] DominSkiJump. *Wilingen 2018 Roman Koudelka upadek/fall*. [online]. List. 2022. URL: *https://www.youtube.com/watch?v=Pvx0-COMnmw%5C&ab_channel=DominSkiJump*.

[10] Jason Fernando, Andy Smith i Yarilet Perez. *R-Squared Formula, Regression, and Interpretations*. [online]. List. 2022. URL: *https://www.investopedia.com/terms/r/r-squared.asp*.

[11] *FIS Ski Jumping - Calendar*. [online]. List. 2022. URL: *https://www.fis-ski.com/DB/ski-jumping/calendar-results.html?noselection=true*.

[12] Jim Frost. *The Difference between Linear and Nonlinear Regression Models*. [online]. List. 2022. URL: *https://statisticsbyjim.com/regression/difference-between-linear-nonlinear-regression-models/*.

[13] Ian Goodfellow, Yoshua Bengio i Aaron Courville. *Deep Learning*. *http://www.deeplearningbook.org*. MIT Press, 2016.

[14] *Introduction to the Keras Tuner*. [online]. List. 2022. URL: *https://www.tensorflow.org/tutorials/keras/keras_tuner?hl=en*.

[15] Luca Invernizzi i in. *Getting started with KerasTuner*. [online]. Paź. 2022. URL: *https://keras.io/guides/keras_tuner/getting_started/*.

[16] A.K. Jain, Jianchang Mao i K.M. Mohiuddin. „Artificial neural networks: a tutorial". W: *Computer* 29.3 (1996), s. 31–44. DOI: *10.1109/2.485891*.

[17] Ajitesh Kumar. *Reinforcement Learning Real-world examples*. Grud. 2022. URL: *https://vitalflux.com/reinforcement-learning-real-world-examples/*.

[18] Panos Louridas i Christof Ebert. „Machine Learning". W: *IEEE Software* 33.5 (2016), s. 110–115. DOI: *10.1109/MS.2016.114*.

[19] Batta Mahesh. „Machine learning algorithms-a review". W: *International Journal of Science and Research (IJSR).[Internet]* 9 (2020), s. 381–386.

[20] Dastan Maulud i Adnan M Abdulazeez. „A review on linear regression comprehensive in machine learning". W: *Journal of Applied Science and Technology Trends* 1.4 (2020), s. 140–147.

[21] Virmavirta Mikko i Kivekäs Juha. „The effect of wind on jumping distance in ski jumping depends on jumpers' aerodynamic characteristics". W: *Journal of Biomechanics* 137 (2022), s. 111101. ISSN: 0021-9290. DOI: *https://doi.org/10.1016/j.jbiomech.2022.111101*. URL: *https://www.sciencedirect.com/science/article/pii/S0021929022001531*.

[22] Kavitha S, Varuna S i Ramya R. „A comparative analysis on linear regression and support vector regression". W: *2016 Online International Conference on Green Engineering and Technologies (IC-GET)*. 2016, s. 1–5. DOI: *10.1109/GET.2016.7916627*.

[23] Iga Sankiewicz. *KEVIN BICKNER 234,5m FATALNY UPADEK VIKERSUND 2017*. [online]. List. 2022. URL: *https://www.youtube.com/watch?v=7wI--39dZxo%5C&ab_channel=IgaSankiewicz*.

[24] M. Shanker, M.Y. Hu i M.S. Hung. „Effect of data standardization on neural network training". W: *Omega* 24.4 (1996), s. 385–397. ISSN: 0305-0483. DOI: *https://doi.org/10.1016/0305-0483(96)00010-2*. URL: *https://www.sciencedirect.com/science/article/pii/0305048396000102*.

[25] International Ski i Snowboard Federation FIS. „SKI JUMPING". W: *THE INTERNATIONAL SKI COMPETITION RULES* 3 (maj 2022).

[26] *Ski jumping results database*. [online]. Paź. 2022. URL: *https://www.kaggle.com/datasets/wrotki8778/ski-jumping-results-database-2009now?resource=download*.