**Dokumentacja aplikacji**

**bukmacherskiej**

**Checkmate Prophet**

**Warszawa, Czerwiec 2024**

Obraz zawierający symbol, godło, logo, szkic

Opis wygenerowany automatycznie

# 1. Informacje o dokumencie

## 1.1 Cel dokumentu

Dokument zawiera opis dokumentacji projektowej aplikacji „Checkmate Prophet” służącej do wsparcia procesów bukmacherskich związanych z platformą szachową „Lichess”.

## 1.2 Rejestr zmian

W niniejszym dokumencie dokonano następujących zmian:

|  |  |  |  |
| --- | --- | --- | --- |
| **Wersja** | **Data weryfikacji** | **Opis zmian** | **Osoby odpowiedzialne** |
| 0.01 | 25.05.2024 | Utworzenie dokumentu | Adam Kaczkowski (s23020),  Patryk Siedlik (s22811),  Wiktor Snochowski (s22748) |
| 1.00 | 23.06.2024 | Publikacja dokumentu | Adam Kaczkowski (s23020),  Patryk Siedlik (s22811),  Wiktor Snochowski (s22748) |

## 1.3 Zakresy odpowiedzialności

Osoby odpowiedzialne za wytworzenie dokumentacji:

|  |  |  |
| --- | --- | --- |
| **Osoba** | **Rola** | **Zakres odpowiedzialności** |
| Wiktor Snochowski | Autor Projektu | Realizacja projektu. |
| Adam Kaczkowski | Autor Projektu | Realizacja projektu. |
| Patryk Siedlik | Autor Projektu | Realizacja projektu. |

## 1.4 Spis treści

[1. Informacje o dokumencie 2](#_Toc170063152)

[1.1 Cel dokumentu 2](#_Toc170063153)

[1.2 Rejestr zmian 2](#_Toc170063154)

[1.3 Zakresy odpowiedzialności 2](#_Toc170063155)

[1.4 Spis treści 3](#_Toc170063156)

[2. Wprowadzenie 4](#_Toc170063157)

[2.1 Cel projektu 4](#_Toc170063158)

[2.2 Kontekst biznesowy 4](#_Toc170063159)

[2.3 Charakterystyka użytych technologii 5](#_Toc170063160)

[3. Metoda 8](#_Toc170063161)

[3.1 Parametry modeli ML 8](#_Toc170063162)

[3.2 Opis funkcjonalności 10](#_Toc170063163)

[4. Logika aplikacji 12](#_Toc170063164)

[4.1 Diagram sekwencji 12](#_Toc170063165)

[4.2 Diagram aktywności dla main.py 13](#_Toc170063166)

[5. Ewaluacja 14](#_Toc170063167)

[5.1 Moduł evaluate\_model.py 14](#_Toc170063168)

[6. Podsumowanie 16](#_Toc170063169)

# 2. Wprowadzenie

## 2.1 Cel projektu

Celem projektu jest stworzenie aplikacji, służącej do predykcji wyników partii szachowych, rozgrywanych na platformie internetowej „Lichess”. Oprogramowanie wykorzystując wdrożony model uczenia maszynowego, pozwoli użytkownikom na wprowadzenie danych dotyczących konkretnej partii szachowej, na podstawie których dokonana zostanie analiza i próba przewidzenia, który z zawodników ma większe szanse na zwycięstwo.

Aplikacja ma stanowić wsparcie dla procesów predykcyjnych funkcjonujących w branży bukmacherskiej.

## 2.2 Kontekst biznesowy

Wraz z dynamicznym rozwojem branży bukmacherskiej, rośnie zapotrzebowanie na różnego rodzaju narzędzia predykcyjne, które umożliwią dokładniejszą ocenę ryzyka i zwiększą szanse na trafne przewidzenie wyników obstawianych rozgrywek.

Szczególnie istotne jest to w przypadku szachów, gdzie tradycyjne czynniki, takie jak ranking Elo, mogą okazać się niewystarczające, a nawet mylące. W szachach kluczową rolę odgrywają również inne aspekty, takie jak doświadczenie zawodników, format rozgrywki, historia pojedynków, a nawet preferowane style gry czy stosowane debiuty. Ogromna ilość tego typu informacji oraz ich złożoność, sprawiają, że analiza partii szachowych staje się wyzwaniem dla przeciętnego użytkownika, co przekłada się na mniejsze zainteresowanie obstawianiem tego typu rozgrywek.

Na rynku brakuje obecnie kompleksowych rozwiązań, które skutecznie sprostałyby temu problemowi. W odpowiedzi na tę lukę, powstał projekt „Checkmate Prophet” – aplikacja oparta na algorytmach uczenia maszynowego, która ma na celu dostarczenie bukmacherom, graczom oraz entuzjastom szachów narzędzia do analizy i predykcji wyników partii. Dzięki „Checkmate Prophet” możliwe będzie uwzględnienie szerokiego spektrum czynników wpływających na wynik partii, co pozwoli na podejmowanie bardziej świadomych decyzji zarówno w kontekście zakładów bukmacherskich, jak i doskonalenia umiejętności szachowych. Aplikacja ta ma potencjał zrewolucjonizować sposób, w jaki analizujemy i przewidujemy wyniki partii szachowych, przyczyniając się do rozwoju branży bukmacherskiej, jak i samej dyscypliny szachów.

## 2.3 Charakterystyka użytych technologii

Do najważniejszych technologii (narzędzi i bibliotek) użytych do przygotowywanej w ramach projektu aplikacji należą:

* GitHub
* Python
* Conda
* PowerShell
* Pandas
* Sklearn
* AutoGluon
* Argparse
* Streamlit
* Lichess API
* Weights&Biases (Wandb)

|  | **Charakterystyka użytych technologii** |
| --- | --- |
| **1.** | **GitHub** |
| 1.1 | **Opis rozwiązania:**  Platforma do hostowania kodu źródłowego i współpracy nad projektami, używana do kontroli wersji i zarządzania kodem. |
| 1.2 | **Zastosowanie w projekcie:**  Całość repozytorium projektowego aplikacji składowana jest na GitHub:  *https://github.com/git3st/PJA\_SUML\_11c\_gr4* |
| **2.** | **Python** |
| 2.1 | **Opis rozwiązania:**  opularny język programowania, wykorzystywany do tworzenia skryptów i aplikacji, w tym aplikacji do uczenia maszynowego. |
| 2.2 | **Zastosowanie w projekcie:**  Zastosowano Python w wersji 3.11.8 |
| **3**. | **Conda** |
| 3.1 | **Opis rozwiązania:**  Menedżer środowisk i pakietów dla Pythona, ułatwiający instalację i zarządzanie zależnościami (pakietami) projektu. |
| 3.2 | **Zastosowanie w projekcie:**  Conda stanowi główne rozwiązanie dotyczące zarządzania pakietami w projekcie. Procedura importu środowiska wymaganego do uruchomienia aplikacji, została opisana w repozytorium GitHub. |
| 4. | **PowerShell** |
| 4.1 | **Opis rozwiązania:**  Narzędzie Microsoftu służące do automatyzacji zadań i zarządzania konfiguracją, używane do uruchamiania skryptów oraz zarządzania środowiskiem. |
| 4.2 | **Zastosowanie w projekcie:**  Skrypt importujący środowisko Condy |
| **5**. | **Pandas** |
| 5.1 | **Opis rozwiązania:**  Biblioteka Pythona do analizy i manipulacji danymi, szczególnie przydatna do pracy z danymi tabelarycznymi. |
| 5.2 | **Zastosowanie w projekcie:**  Operacje na danych tabelarycznych. Wykorzystywane przede wszystkim w fazie przygotowania danych. |
| **6**. | **Sklearn** |
| 6.1 | **Opis rozwiązania:**  Biblioteka Pythona zawierająca narzędzia do uczenia maszynowego, takie jak modele, algorytmy i metody ewaluacji. |
| 6.2 | **Zastosowanie w projekcie:**  Zastosowano do normalizacji danych oraz wytrenowania, ewaluacji i udostępnienia modelu ML. |
| **7**. | **AutoGluon** |
| 7.1 | **Opis rozwiązania:**  Narzędzie do automatycznego uczenia maszynowego, upraszczające proces tworzenia modeli predykcyjnych. |
| 7.2 | **Zastosowanie w projekcie:**  Wytrenowanie, ewaluacja i udostępnienie modeli AutoML. |
| **8**. | **Argparse** |
| 8.1 | **Opis rozwiązania:**  Moduł Pythona do tworzenia interfejsów wiersza poleceń, pozwalający na przekazywanie argumentów do skryptów. |
| 8.2 | **Zastosowanie w projekcie:**  Parametryzacja części związanej z charakterystyką trenowania i ewaluacji modeli ML oraz AutoML. |
| **9**. | **Streamlit** |
| 9.1 | **Opis rozwiązania:**  Framework do tworzenia interaktywnych aplikacji webowych opartych na danych, używany do wizualizacji i udostępniania wyników. |
| 9.2 | **Zastosowanie w projekcie:**  Interfejs użytkownika w aplikacji. |
| **10**. | **Lichess API** |
| 10.1 | **Opis rozwiązania:**  Interfejs programistyczny (API) platformy Lichess, umożliwiający dostęp do danych o grach, użytkownikach i turniejach. Endpoint dostępny pod adresem:  <https://lichess.org> |
| 10.2 | **Zastosowanie w projekcie:**  Dostęp do natywnych funkcjonalności oferowanych przez platformę Lichess, np.: odczytywanie informacji z publicznego profilu gracza i rating w poszczególnych trybach gry. |
| **11**. | **Weights & Biases (Wandb)** |
| 11.1 | **Opis rozwiązania:**  Platforma do monitorowania i śledzenia eksperymentów uczenia maszynowego |
| 11.2 | **Zastosowanie w projekcie:**  Rejestrowanie parametrów modelu, metryk, wizualizacji i artefaktów, co ułatwia analizę i porównywanie wyników kolejnych iteracji procesu uczenia maszynowego. |

# 3. Metoda

## 3.1 Parametry modeli ML

Parametryzacja z poziomu konsoli

W pliku main.py dostępne są następujące opcje konfiguracyjne, które można ustawić przy użyciu argumentów w konsoli:

* ***--file\_prefix*** (str): Prefiks plików CSV do scalenia. Domyślnie: data/01\_raw\_data/games\_metadata\_profile\_2024\_01
* ***--num\_files*** (int): Liczba plików do scalenia. Domyślnie: 16
* ***--output\_file*** (str): Ścieżka do wynikowego pliku CSV. Domyślnie: data/01\_raw\_data/full\_dataset.csv
* ***--use\_automl*** (bool): Flaga wskazująca użycie AutoGluon. Domyślnie: True
* ***--train*** (float): Proporcja danych do użycia w treningu. Domyślnie: 0.8
* ***--test*** (float): Proporcja danych do użycia w teście. Domyślnie: 0.10
* ***--validation*** (float): Proporcja danych do użycia w walidacji. Domyślnie: 0.10
* ***--seed*** (int): Losowy seed dla powtarzalności wyników. Domyślnie: 50
* ***--n\_samples*** (int): Liczba próbek do użycia w treningu AutoGluon. Domyślnie: 500
* ***--time\_limit*** (int): Limit czasu dla AutoGluon w sekundach. Domyślnie: 60
* ***--n\_estimators*** (int): Liczba drzew w Random Forest. Domyślnie: 100
* ***--n\_estimators\_pipeline*** (int): Liczba drzew w Random Forest dla pipeline. Domyślnie: 100
* ***--random\_state\_pipeline*** (int): Losowy seed dla pipeline. Domyślnie: 42
* ***--n\_samples\_evaluate*** (int): Liczba próbek do użycia w ewaluacji. Domyślnie: 100
* ***--random\_state\_evaluate*** (int): Losowy seed dla próbek ewaluacyjnych. Domyślnie: 0
* ***--wandb\_project*** (str): Nazwa projektu WandB.
* ***--wandb\_api\_key*** (str): Klucz API WandB.

W funkcji transform\_data z pliku data\_preprocessing.py znajduje się wiele parametrów, które mogą być modyfikowane bezpośrednio w kodzie, ale nie są dostępne do zmiany z poziomu konsoli linii poleceń. Poniżej przedstawiono szczegółowy opis tych parametrów:

1. **cols\_to\_remove**:
   * Typ: List[str]
   * Opis: Lista kolumn, które mają zostać usunięte z zestawu danych. Domyślnie obejmuje takie kolumny jak WhiteRatingDiff, BlackRatingDiff, Black\_count\_all, Black\_createdAt, Black\_is\_deleted, Black\_playTime\_total, Black\_profile\_flag, Black\_title, Black\_tosViolation, Date, ECO, GameID, Moves, Opening, Round, Site, Termination, Time, TimeControl, TotalMoves, White\_count\_all, White\_createdAt, White\_is\_deleted, White\_playTime\_total, White\_profile\_flag, White\_title, White\_tosViolation.
2. **cols\_to\_fill\_numbers**:
   * Typ: Dict[str, str]
   * Opis: Słownik mapujący nazwy kolumn na typy danych, określający, które kolumny numeryczne mają być uzupełniane średnią wartością, jeśli zawierają braki. Domyślne kolumny to WhiteElo, WhiteRatingDiff, White\_playTime\_total, White\_count\_all, BlackElo, BlackRatingDiff, Black\_playTime\_total, Black\_count\_all.
3. **fill\_string\_values**:
   * Typ: Dict[str, str]
   * Opis: Słownik mapujący nazwy kolumn na wartości, którymi mają być uzupełniane braki w kolumnach tekstowych. Domyślne wartości to White\_profile\_flag jako "Unknown", White\_title jako "None", Black\_profile\_flag jako "Unknown", Black\_title jako "None", Opening jako "Unknown".
4. **clean\_outliers**:
   * Typ: bool
   * Opis: Flaga wskazująca, czy usunąć wartości odstające z danych. Domyślnie ustawiona na False.
5. **cols\_to\_transform**:
   * Typ: Dict[str, Dict[str, str]]
   * Opis: Słownik mapujący nazwy kolumn na słowniki zamian wartości w tych kolumnach. Domyślnie obejmuje zamiany w kolumnach Result (z wartości "1-0" na "White", "0-1" na "Black", "1/2-1/2" na "Draw") oraz Event (z wartości zawierających "tournament" na "tournament", "swiss" na "swiss").
6. **clean\_missing\_vals**:
   * Typ: bool
   * Opis: Flaga wskazująca, czy usunąć wiersze z brakującymi wartościami. Domyślnie ustawiona na True.
7. **cols\_to\_normalize**:
   * Typ: List[str]
   * Opis: Lista kolumn, które mają być znormalizowane do zakresu [0, 1]. Domyślnie wartość ta jest ustawiona na None.
8. **categorical\_features**:
   * Typ: List[str]
   * Opis: Lista kolumn kategorycznych do zakodowania za pomocą OneHotEncoder. Domyślnie obejmuje takie kolumny jak Event, Day, Time\_TimeOfDay, White\_is\_deleted, White\_profile\_flag, White\_title, Black\_is\_deleted, Black\_profile\_flag, Black\_title, ECO, Opening, TimeControl, Termination.

## 3.2 Opis funkcjonalności

***create\_error\_logger***

Funkcja *create\_error\_logger* służy do tworzenia loggera przeznaczonego do rejestrowania błędów występujących podczas wykonania logiki aplikacji. Zwraca skonfigurowany obiekt typu *logging.Logger*.

***get\_elo***

Funkcja *get\_elo* umożliwia pobranie aktualnego elo użytkownika na platformie Lichess dla określonego trybu gry przy wykorzystaniu API Lichess.

**Argumenty:**

* *user (str)*: Nazwa użytkownika na Lichess, dla którego ma zostać pobrane elo.
* *game\_mode (str)*: Tryb gry, dla którego ma zostać pobrane elo (np. bullet, blitz, rapid, classical).
* *api\_key (str)*: Klucz API Lichess wymagany do autoryzacji.

**Zwraca:**

* *int*: Elo użytkownika dla określonego trybu gry.
* *None*: W przypadku wystąpienia błędu podczas pobierania elo.

**Podnosi wyjątki:**

* *requests.exceptions.RequestException*: W przypadku błędu związanego z żądaniem HTTP.
* *KeyError*: Jeśli dane dotyczące elo nie są dostępne w odpowiedzi API.
* *ValueError*: Jeśli elo nie może zostać przekonwertowane na *int*.
* *Exception*: Dla innych nieoczekiwanych błędów.

**Obsługa Błędów:**

Funkcja rejestruje błędy za pomocą wcześniej skonfigurowanego obiektu loggera. Dodatkowo, odpowiednie komunikaty błędów są wyświetlane przy użyciu funkcji *st.error* z biblioteki Streamlit.

***fake\_loader***

Funkcja *fake\_loader* służy do symulacji animacji ładowania, wyświetlając kolejne komunikaty podczas procesu pobierania danych i przetwarzania. Komunikaty są wyświetlane sekwencyjnie z 2-sekundowym opóźnieniem między nimi, przy użyciu funkcji *st.toast* z biblioteki Streamlit. Po zakończeniu procesu, funkcja wyświetla komunikat sukcesu za pomocą funkcji *st.success*.

**Argumenty:**

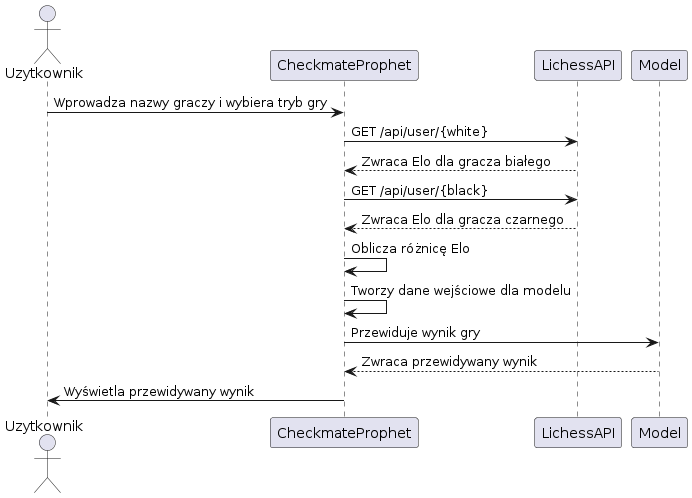
* *final\_result (str)*: Ostateczny komunikat wynikowy, który ma zostać wyświetlony po zakończeniu ładowania.

***Dodatkowe elementy aplikacji***

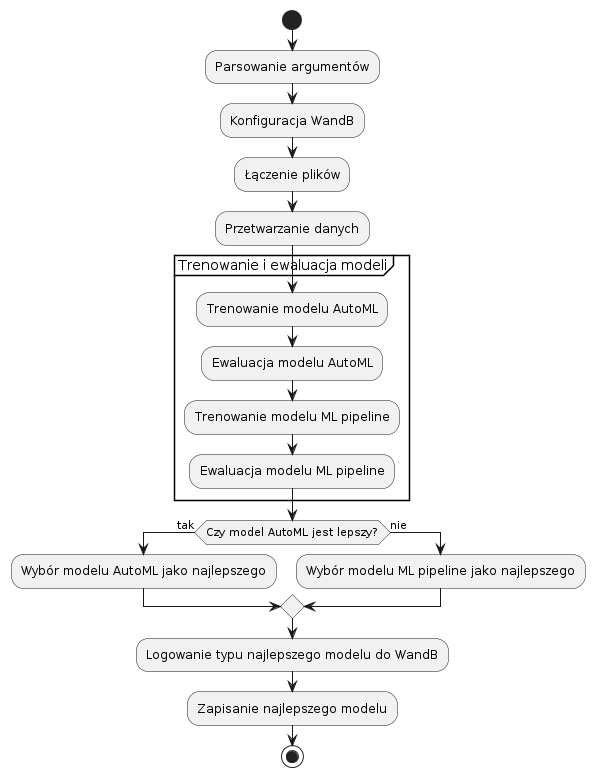
* Parser argumentów (*argparse.ArgumentParser*): Komponent służy do parsowania argumentów wiersza poleceń, w tym klucza API Lichess (*--api*).
* Ładowanie Modelu *(pickle.load):* Komponent odpowiedzialny za ładowanie wstępnie wytrenowanego modelu uczenia maszynowego z określonego katalogu.
* Interfejs Streamlit: Interfejs wyświetla tytuł oraz wprowadzenie za pomocą funkcji *st.title* i *st.write* z biblioteki Streamlit. Ponadto, przyjmuje dane wejściowe od użytkownika, takie jak nazwy graczy, strony po których grają oraz tryb gry, przy użyciu funkcji *st.text\_input* i *st.selectbox*. Po kliknięciu przycisku *Predict* (*st.button*), interfejs wykonuje logikę przewidywania, obliczając różnice Elo i przewidując wynik gry.

# 4. Logika aplikacji

## 4.1 Diagram sekwencji



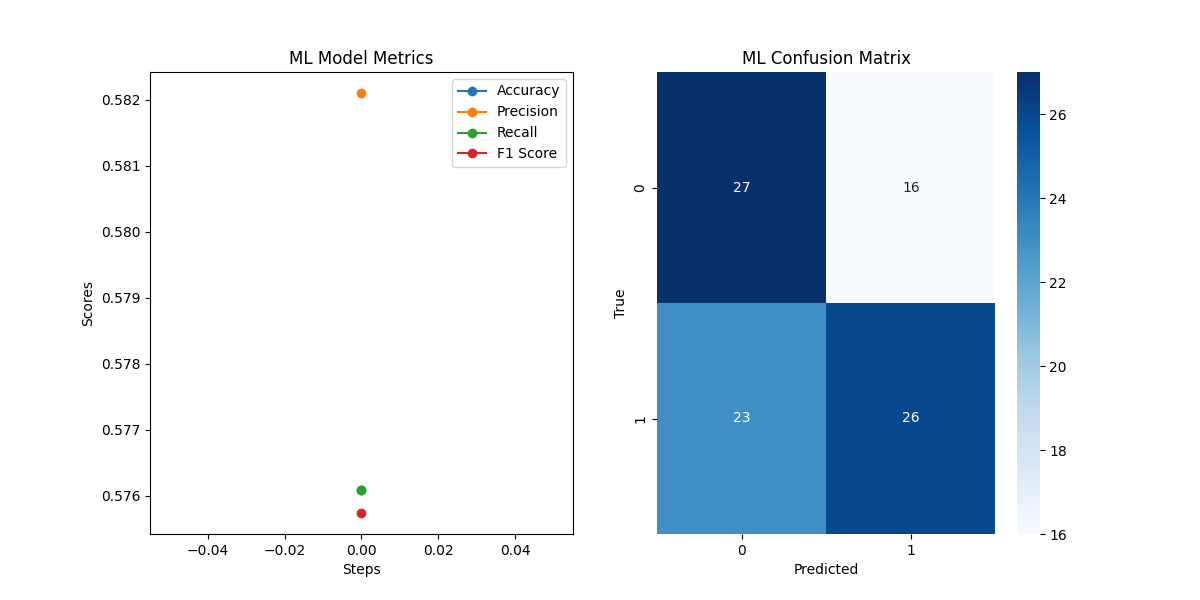
## 4.2 Diagram aktywności dla main.py

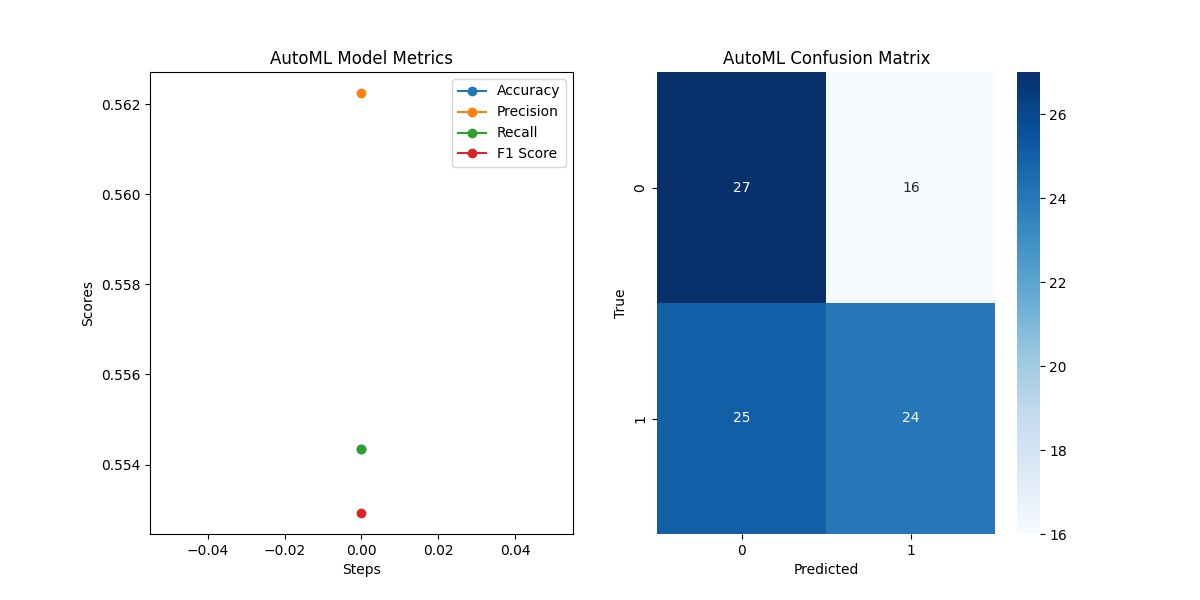


# 5. Ewaluacja

## 5.1 Moduł evaluate\_model.py

1. **Argumenty wejściowe**: Funkcja evaluate\_model przyjmuje dane testowe (x\_test, y\_test), wytrenowany model (predictor), liczbę próbek do ewaluacji (n\_samples\_evaluate), stan losowy (random\_state\_evaluate), oraz typ modelu (model\_type).
2. **Próbkowanie danych testowych**: Wybiera losową próbkę danych testowych o rozmiarze n\_samples\_evaluate.
3. **Predykcja**:
   * **AutoML**: Jeśli model jest typu TabularPredictor, dokonuje predykcji na danych testowych.
   * **Pipeline**: Jeśli model jest pipeline'em, najpierw przetwarza dane testowe za pomocą kroku preprocessor, a następnie dokonuje predykcji za pomocą kroku classifier.
4. **Obliczanie metryk**: Funkcja oblicza różne metryki ewaluacyjne, takie jak:
   * accuracy\_score: Dokładność modelu.
   * precision\_score: Precyzja modelu.
   * recall\_score: Czułość modelu.
   * f1\_score: Wskaźnik F1.
   * confusion\_matrix: Macierz pomyłek.
5. **Generowanie raportu klasyfikacji**: Tworzy raport klasyfikacji za pomocą classification\_report, który zawiera szczegółowe informacje o metrykach dla każdej klasy.
6. **Logowanie do WandB**: Metryki oraz raport klasyfikacji są logowane do Weights & Biases.
7. **Wizualizacja**: Funkcja plot\_metrics generuje wykresy przedstawiające metryki oraz macierz pomyłek, które są następnie logowane do WandB jako obrazy.





# 6. Podsumowanie

"Checkmate Prophet" to nie tylko narzędzie predykcyjne, ale także platforma do analizy partii szachowych, która może pomóc graczom w doskonaleniu swoich umiejętności. Projekt stanowi rozwiązanie całkowicie nowe na rynku, otwierając nowe możliwości zarówno dla branży bukmacherskiej, jak i świata szachów.

W niniejszej dokumentacji przedstawiono opis projektu, począwszy od kontekstu biznesowego, przez charakterystykę użytych technologii, aż po dokładną metodologię zastosowaną do budowy modelu. Opisano również funkcjonalności aplikacji, umożliwiając użytkownikom pełne zrozumienie jej możliwości.

Przyszłe prace nad rozwiązaniem mogą skupić się na dalszym rozwoju modelu, uwzględnieniu jeszcze większej liczby czynników wpływających na wynik partii, a także na rozszerzeniu funkcjonalności aplikacji, na przykład o możliwość analizy partii w czasie rzeczywistym.