Софийски университет „Св. Кл. Охридски”

Факултет по математика и информатика

Курсов Проект

на тема: „Предсказване на краен резултат от партия шах”

Студенти: **Радомир Минков**

Ф.Н. **7MI0800016**

Курс: **„Изкуствен Интелект“, КН** Учебна година: **2024/25**

Преподаватели: **проф. Иван Койчев, Борис Величков**:

==========

Декларация за липса плагиатство:

- Плагиатство е да използваш, идеи, мнение или работа на друг, като претендираш, че са твои. Това е форма на преписване.
- Тази курсова работа е моя, като всички изречения, илюстрации и програми от други хора са изрично цитирани.
- Тази курсова работа или нейна версия не са представени в друг университет или друга учебна институция.
- Разбирам, че ако се установи плагиатство в работата ми ще получа оценка “Слаб”.

7.2.25 г. Подпис на студента:

Съдържание

1. Увод

В последните години методите на машинното обучение (Machine Learning) и дълбокото обучение (Deep Learning) намират широко приложение в различни сфери. С навлизането на шахът като спорт във все повече големи турнири и първенства нуждата от добри ChessBot-ове се засилва. Една от интересните области в шаха като игра, е крайният резултат. Затова този проект показва система за предсказване на изхода от шахматна партия на база исторически данни за рейтингите на играчите, използваните дебюти и други характеристики на партията. Проектът цели автоматизирано класифициране на крайния победител в една шахматна партия (бял, черен или реми), използвайки база данни от Kaggle („Chess dataset“). В проекта са разгледани три основни подхода:

- Логистична регресия (Logistic Regression)
- XGBoost (градиентно-бустнато дърво)
- Невронни мрежи (Multi-Layer Perceptrons) с различна дълбочина (1, 2 и 3 скрити слоя)

2. Преглед на областта (Отговаряне на научни въпроси със затворен отговор)

2.1. Подходи и методи:

Предсказването на шахматен резултат може да се разглежда като задача за многокласова класификация (3 класа: бял, черен, реми). Това е задача, която основно и най-точно се решава с XGBoost алгоритъма, но за сравнение ще използваме както Logistic Regression така и Neural Network

- ***XGBoost(Extreme Gradient Boosting)***

  XGBoost (Extreme Gradient Boosting) е един от най-мощните алгоритми за класификация и регресия върху таблични данни. Той използва Gradient Boosting върху дървета на решенията (Decision Trees), като подобрява представянето на модела чрез итеративно изграждане на по-добри дървета. Основната идея на XGBoost е, че той комбинира множество слаби модели (дървета на решенията) в един силен ансамблов модел, като на всяка стъпка намалява грешките на предишните дървета.

За да подобрим точността и да оценим предимствата на различни подходи, сравняваме XGBoost с няколко други модела за класификация на данните:

- ***Logistic Regression***

  Логистичната регресия е основен модел за класификация, който предсказва вероятността даден пример да принадлежи към определен клас. За разлика от линейната регресия, която прогнозира непрекъснати стойности, логистичната регресия използва логистична функция (Sigmoid function), за да ограничи резултатите между 0 и 1. При многокласовата класификация, какъвто е нашият случай (бял, черен, реми), се използва Softmax функция, която трансформира резултатите така, че сумата от вероятностите за трите класа да бъде равна на 1.

- ***MLP Neural Network***

MLP е вид изкуствена невронна мрежа, която се състои от fully connected layers. В нашия проект използвах мрежи с 1, 2 и 3 скрити слоя, като тествах различни архитектури и хиперпараметри. Основната разлика между MLP и XGBoost е, че MLP може да научи по-сложни зависимости, но изисква повече данни и внимателна настройка.

2.2. Съществуващи решения:

Има доста уебсайтове и инструменти, които опитват да предвидят резултат от шахматна партия въз основа на рейтинги, предишни партии и т.н. В общия случай те използват рейтинг формули (като Elo, Glicko), статистически модели или готови **chess engines**. Нашият подход е изцяло базиран на машинното самообучение, без да използваме вградена логика за оценка на позиции..

2.3. Сравнителен анализ:

Често XGBoost постига висока точност при таблични данни. Логистичната регресия е базова линия, която обикновено се справя добре при по-„линейни“ задачи. Невронните мрежи могат да са ефективни, но понякога изискват по-сложна настройка и повече данни. В хода на проекта се установи, че различните архитектури дават различни резултати в зависимост от това колко добре улавят рейтинговата разлика, избран дебют и др.

3. Проектиране

3.1. Анализ на изискванията:

Целта на проекта е да се създаде система, която ще може по начално зададени характеристики и параметри, които спохождат една партия шах да предвижда какъв ще е крайният резултат с една доста голяма точност, като вариантите биват 3: победа за бели, победа за черни и равенство

Обща архитектура:

Проектът ще бъде реализиран чрез следната архитектура:

1. **Обработка на данни**: Зареждане и подготвяне, изтриване на данни от *Kaggle dataset-a*. Като те биват в CSV формат и съдържат характеристики за над 2000 партии с някакви излишъци.
2. **Модели:** Създаване на архитектури за 3 вида модели, които ще бъдат използвани.

3. **Обучение на моделите**: След подготовка на данните, моделите подлежат на обучение като за целта им се дава достъп до 80% от наличните данни
4. **Тестване на точността**: Извършване на тестове за оценка на производителността и точността на моделите с останалите 20%.
5. **Parameter tuning**: След като се види какъв е резултатът започва ново обучение на данните, което има за цел да намери най-подходящите и оптимални стойности за хиперпараметрите на всеки модел
6. **Модел на данните:**

- Входни данни: текстови характеристики на шахматни игри взети от *Kaggle*.
- Изходни данни: победителя в партията

7. **Основни характеристики: Точност (Accuracy) и/или F1 (macro) заради 3-те класа (особено ако клас „реми“ е малък).**

4. Реализация/Провеждане на експерименти:

4.1. Реализация, тестване/експерименти

Процесът на реализация включва:

- **Събиране и обработка на данни** – изтегляне на базата данни от *Kaggle* и премахването на ненужните колони от нея.
- **Генериране на нови колпни**
- **Балнасиране на класовете поради това, че някои класове се срещат доста повече от други –** чрез SMOTE
- **Обучение на модели** – настройване на параметри за всеки от моделите поотделно.
- **Logistic Regression** - (StandardScaler) + GridSearchCV за C и penalty
- **XGBoost** - Tuning на max_depth, learning_rate и и други хиперпараметри.
- **MLP** - С 1, 2 и 3 слоя. Ползване на PyTorch или scikit-learn MLPClassifier със съответно early stopping.
- **Тестване на модела** – Измерване на macro-F1, confusion matrix и анализ на грешките.

4.2. Използвани технологии, платформи и библиотеки:

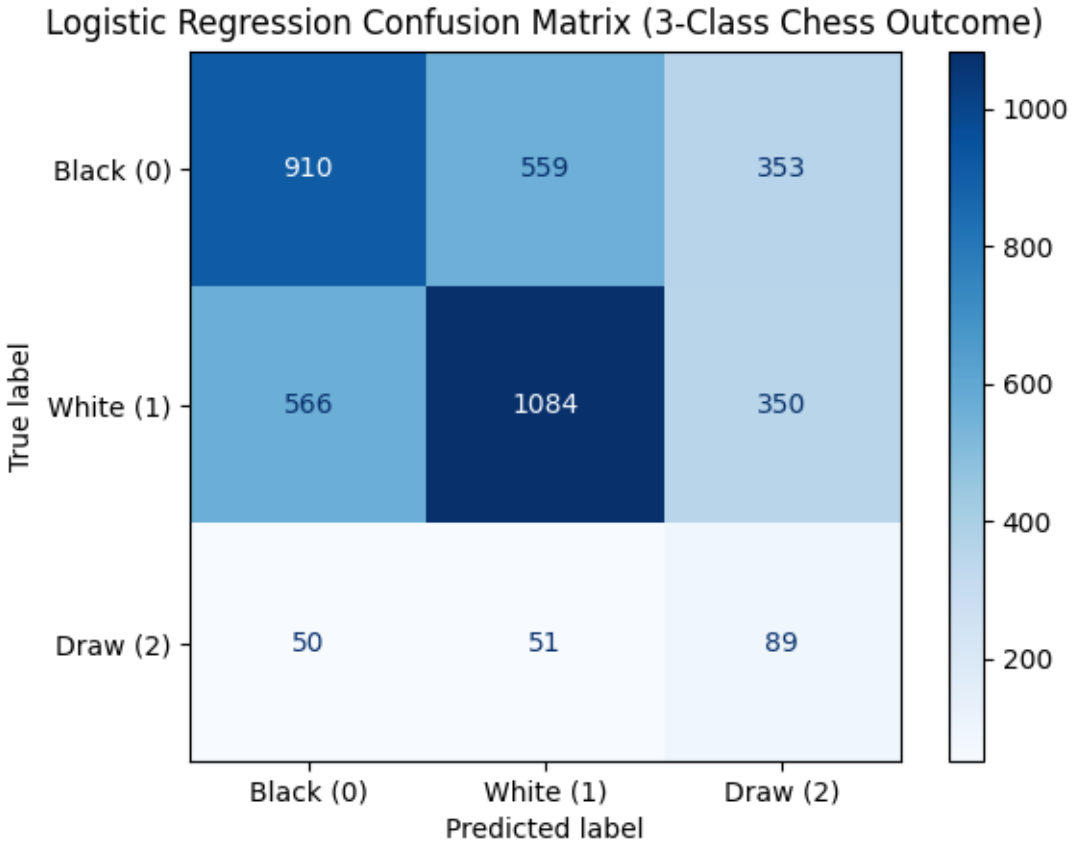
- ***Scikit-learn*** – за допълнителни функции като разделяне на данни и оценка на точността.
- ***PyTorch*** – библиотека за машинно и дълбоко обучение, която позволява изграждане, обучение и тестване на невронни мрежи.

Изборът на тези технологии се обяснява с тяхната ефективност и популярност в областта на машинното обучение и обработката на текст.

5. Получени резултати и анализ

5.1. Логистична регресия

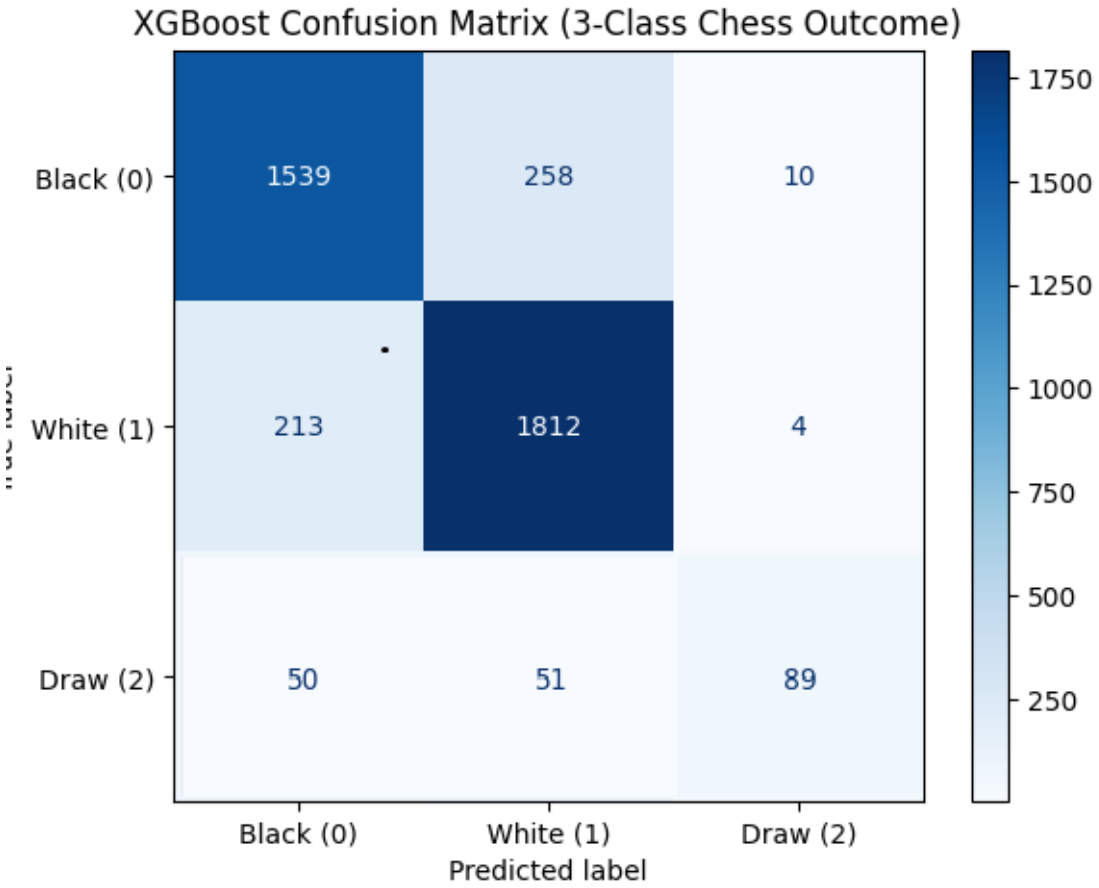
- Логистичната регресия има сравнително добри предсказания за победа на бели и черни, но често бърка ремитата.
- Често класифицира реми като черна или бяла победа, тъй като този клас е рядък в обучителните данни.
- Като цяло, по-простите зависимости между рейтингите на играчите и крайния резултат са уловени, но моделът не е в състояние да разбере по-сложните фактори като дебютите и темпото на партията.



5.2. XGBoost

- XGBoost има най-висока точност и Macro-F1 сред всички модели.
- Добре разграничава белите и черните победи, като допуска по-малко грешки в сравнение с MLP и Logistic Regression.
- Най-добър при ремитата – макар все още да има известна неточност, той превъзхожда невронните мрежи в балансираното разпределение на класовете.

- Постига добър баланс между бързина и точност.



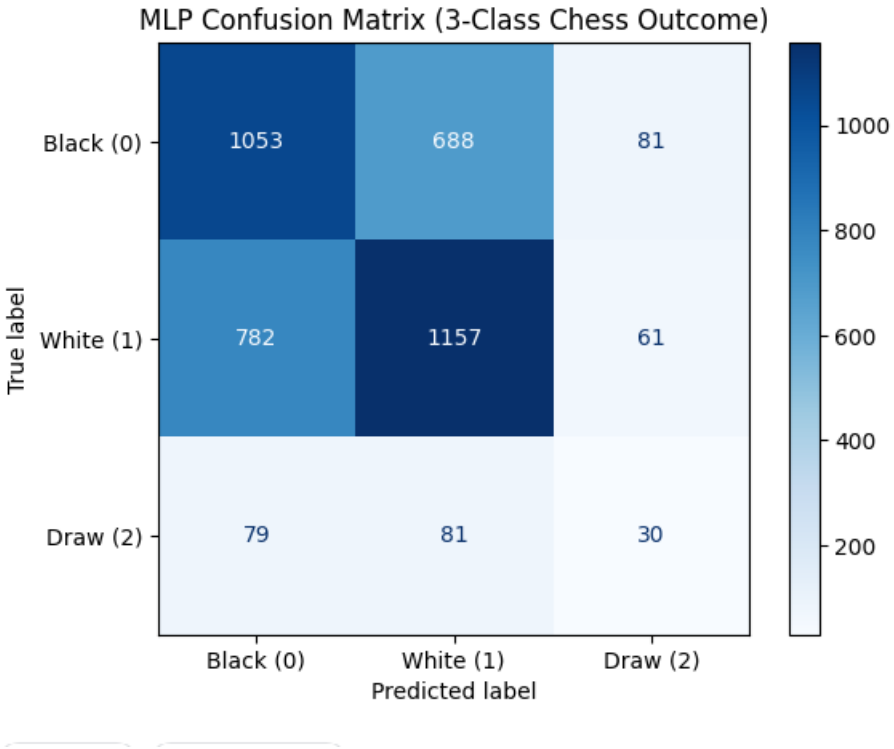
5.3. MLP (1 слой)

- Много малко подобрение спрямо логистична регресия

```
Iteration 248, loss = 0.19439178
Iteration 249, loss = 0.20116264
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.
Classification Report:
              precision    recall  f1-score   support

   Black (0)       0.55      0.58      0.56      1822
   White (1)       0.60      0.58      0.59      2000
    Draw (2)       0.17      0.16      0.17       190

    accuracy                           0.56      4012
   macro avg       0.44      0.44      0.44      4012
weighted avg       0.56      0.56      0.56      4012
```



5.4. MLP (2 слоя)

- По-силен от логистичната регресия, но още не улавя сложните зависимости толкова добре, колкото XGBoost.
- Има слаби резултати при ремитата – често ги бърка с черна или бяла победа**.**
- След допълнително тунинговане (втори тест)**,** успява да повиши точността си**,** но все още остава зад XGBoost.

```
Best trial value (Val F1): 0.5095144853154973
Best hyperparams: {'hidden_dim1': 64, 'hidden_dim2': 128, 'dropout': 0.2, 'activation': 'relu', 'lr': 0.0014821061268725242, 'weight_decay': 5.055724450895964e-05, 'n_epochs': 25}
Final Test macro-F1: 0.47872297564924066
Confusion Matrix:
 [[1172  484  166]
 [ 742 1110  148]
 [  65   65   60]]

Classification Report:
              precision    recall  f1-score   support

   Black (0)       0.59      0.64      0.62      1822
   White (1)       0.67      0.56      0.61      2000
    Draw (2)       0.16      0.32      0.21       190

    accuracy                           0.58      4012
   macro avg       0.47      0.50      0.48      4012
weighted avg       0.61      0.58      0.59      4012
```

5.5. MLP (3 слоя)

- *Най-добрият MLP модел* – След оптимизация на хиперпараметрите (брой неврони, dropout, learning rate и optimizer), мрежата се представя по-добре
- По-добри резултати за ремитата (но все още по-слаби от XGBoost).
- Обучението изисква повече ресурси – повече епохи и оптимизация с Optuna са необходими, за да достигнем добър баланс.
- Все пак изостава спрямо XGBoost по отношение на общата точност и Macro-F1.

```
Best trial value (Val F1): 0.5121484891857077
Best hyperparams: {'batch_size': 64, 'hidden_dim1': 256, 'hidden_dim2': 128, 'hidden_dim3': 128, 'dropout1': 0.5,
'dropout2': 0.3000000000000004, 'dropout3': 0.3000000000000004, 'activation': 'relu', 'n_epochs': 54, 'optimize
r': 'Adam', 'lr': 0.001168361442110532, 'weight_decay': 1.6162285373995382e-06}
Final Test macro-F1: 0.4804291957773364
Confusion Matrix:
 [[ 900  784  138]
 [ 436 1418  146]
 [  43   89   58]]

Classification Report:
              precision    recall  f1-score   support

   Black (0)       0.65      0.49      0.56      1822
   White (1)       0.62      0.71      0.66      2000
    Draw (2)       0.17      0.31      0.22       190

    accuracy                           0.59      4012
   macro avg       0.48      0.50      0.48      4012
weighted avg       0.61      0.59      0.60      4012
```

	Модел	Точност (Accuracy)	Macro-F1	Основни наблюдени
1	Logistic Regression	0.55	0.44	Бърз, но често бърка ремитата
2	XGBoost	0.65	0.58	Най-добър модел – добър баланс между точност и скорост
3	MLP (1 layer)	0.56	0.44	Слабо подобрение спрямо Logistic Regression
4	MLP (2 layers)	0.58	0.48	По-добър баланс, но все още проблеми при ремитата
5	MLP (3 layers)	0.59	0.5	Най-добрият MLP модел, но все още по-слаб от XGBoost

6. Заключение

Проектът показва ефективността и предизвикателствата при прогнозиране на победителя в шахматна партия. Основните изводи, които може да направим са:

- XGBoost често е най-силен при таблични характеристики и постига най-добри резултати сред разгледаните методи.
- Логистичната регресия е бърз и интерпретируем модел, но страда при сложни зависимости (особено при опити да различи реми от победа).
- MLP с повече слоеве може да достигне резултати, близки до XGBoost, при сериозен оптимизиране на хиперпараметрите (дълбочина, dropout, learning rate), но продължава да бърка някои случаи, особено ако клас „реми“ е рядък и трудно разпознаваем.

6.1. Идеи за развитие

- Изпробване на други модели (Random Forest, LightGBM) и по-сложни и обемни невронни мрежи като се дава на всеки слой точно ясна и определена задача.
- Сегментиране на задачата на етапи (първо „дали е реми“, после „черен/бял“).
- Включване на опция да се следят ходовете по време на игра и в реално време да се дава по-вероятният победител като се прави оценка на позицията в реално време

7. Използвана литература

1 **Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD.**

2 https://research.google/blog/transformer-a-novel-neural-network-architecture-for-language-understanding/

3 Effective XGBoost: Optimizing, Tuning, Understanding, and Deploying Classification Models (Treading on Python)

4 Multilayer Perceptrons: Theory and Applications