**ENTREGA 1**

*Análise de Dados e Big Data*

*Inteligência Artificial*

**Conforme solicitado na documentação do projeto, foi enviado para o email do professor Evandro as informações relacionadas a essa entrega.**

**ENTREGA 2**

*Análise de Dados e Big Data*

*Inteligência Artificial*

A base de dados que utilizamos é [**Chess Game Dataset (Lichess)**](https://www.kaggle.com/datasets/datasnaek/chess), que consiste em um conjunto de pouco mais de 20.000 partidas coletadas de uma seleção de usuários do site Lichess.org.

Cada partida de xadrez contém uma grande quantidade de detalhes. O xadrez é, principalmente, um jogo baseado em padrões, e a ciência de dados se concentra em identificar esses padrões nos dados. Por isso, o xadrez tem sido uma das áreas mais exploradas no desenvolvimento de Inteligência Artificial no passado.

Nesse conjunto de dados em particular, temos informações sobre mais de 20.000 partidas. Essas informações são organizadas de uma maneira que facilita muito a análise. Por exemplo, podemos descobrir qual é a probabilidade do jogador x ganhar do jogador y. Também podemos explorar até que ponto fatores que não estão diretamente relacionados ao jogo em si afetam o resultado de uma partida.

Utilizamos um total de 10.000 partidas da base de dados para coletar informações e variáveis como:

* created\_at
  + Tipo **Int**
  + Armazena o horário de início da partida.
* last\_move\_at
  + Tipo **Int**
  + Armazena o horário de término da partida.
* winner
  + Tipo **String**
  + Armazena quem é o vencedor da partida.
* white\_id
  + Tipo **String**
  + Armazena o ID do jogador branco.
* white\_rating
  + Tipo **Int**
  + Armazena a classificação do jogador branco.
* black\_id
  + Tipo **String**
  + Armazena o ID do jogador preto.
* black\_rating
  + Tipo **Int**
  + Armazena a classificação do jogador preto.

A base de dados foi filtrada com base na relevância e importância para a análise dos dados, onde várias variáveis foram excluídas para o treinamento de Machine Learning.

Nove novas variáveis foram criadas para auxiliar na obtenção de uma melhor precisão para os resultados esperados da IA, dentre elas, estão:

* game\_duration
  + Tipo **Float**
  + Armazena a duração total da partida.
* white\_average\_rating
  + Tipo **Float**
  + Armazena a média da classificação do jogador branco.
* white\_win\_rate
  + Tipo **Float**
  + Armazena a média de vitórias do jogador branco.
* white\_average\_duration
  + Tipo **Float**
  + Armazena a duração média das partidas do jogador branco.
* white\_total\_games\_played
  + Tipo **Int**
  + Armazena o total de partidas jogadas pelo jogador branco.
* black\_average\_rating
  + Tipo **Float**
  + Armazena a média da classificação do jogador preto.
* black\_win\_rate
  + Tipo **Float**
  + Armazena a média de vitórias do jogador preto.
* black\_average\_duration
  + Tipo **Float**
  + Armazena a duração média das partidas do jogador preto.
* black\_total\_games\_played
  + Tipo **Int**
  + Armazena o total de partidas jogadas pelo jogador preto.

**ENTREGA 3**

*Inteligência Artificial*

A variável target, no contexto da análise de partidas de Xadrez, é crucial para a definição do objetivo de previsão de um modelo de IA.

A variável target escolhida é "winner", que registra o resultado da partida, indicando qual jogador (Branco, Preto ou Empate) saiu vitorioso. A escolha dessa variável target é justificada, pois o propósito principal do modelo é prever o resultado de partidas de Xadrez, fornecendo insights sobre estratégias vencedoras e a performance dos jogadores.

**ENTREGA 3**

*Análise de Dados e Big Data*

A hipótese utilizada nesse projeto define que jogadores com uma maior quantidade de jogos possuem uma taxa maior de vitória. Uma amostra contendo os jogos de 40 jogadores com o maior número de partidas jogadas foi obtida (amostra A) e, a partir dessa amostra, calculou-se uma média de X vitórias para esses jogares, com um desvio padrão de I.

Uma segunda amostra, de 40 jogadores selecionados aleatoriamente (sem considerar os 40 jogadores da amostra A) foi selecionada (amostra B), possuindo uma média de Y vitórias e um desvio padrão de J". Neste caso, a hipótese alternativa seria: A média de vitórias da amostra A é maior do que a média de vitórias da amostra B. Enquanto a hipótese nula seria o inverso: A média de vitórias da amostra A não é maior que a da amostra B

Com essa hipótese levantada, utilizamos as seguintes variáveis para prever esse resultado:

* white\_win\_rate
* black\_win\_rate

**ENTREGA 4**

*Inteligência Artificial*

A partir dos modelos desenvolvidos, utilizamos como o primeiro Método o HistGradientBoostingClassifier.

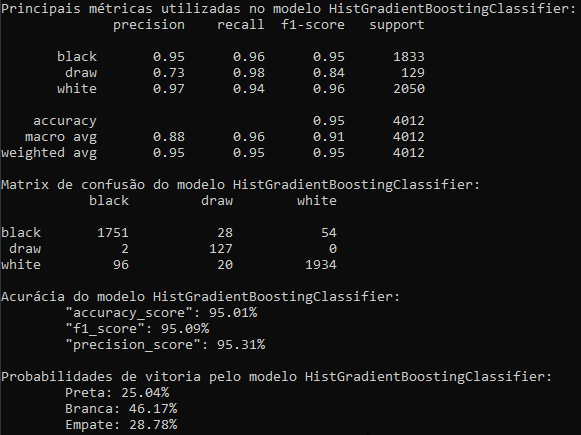
O **HistGradientBoostingClassifier** é um algoritmo de aprendizado de máquina baseado em árvores de decisão que pertence à família de métodos ensemble, mais especificamente, ao grupo de gradient boosting.

Ensemble Learning é uma abordagem em que vários modelos individuais são combinados para formar um modelo mais robusto e geralmente mais preciso

A utilização de todos os métodos se encontra no arquivo MAIN.PY, onde o método com os parâmetros iniciais é abordado na linha de código **23 a 25**.

Além dos parâmetros mencionados anteriormente, o **HistGradientBoostingClassifier** possui outros ajustáveis, como a taxa de aprendizagem (**learning\_rate**), a profundidade máxima (**max\_depth**), entre outros.

Os resultados obtidos a partir do método acima se encontra na foto abaixo



**ENTREGA 4**

*Análise de Dados e Big Data*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Média** | **Mediana** | **Moda** | **Des. Padrão** | **Variância** |
| **turns** | 61,25 | 56,00 | 53,00 | 34,22 | 1170,97 |
| **white\_rating** | 1618,00 | 1594,00 | 1500,00 | 289,28 | 83680,70 |
| **black\_rating** | 1610,00 | 1587,00 | 1500,00 | 286,86 | 82291,05 |
| **game\_duration** | 859,23 | 660,50 | 4,00 | 1149,11 | 1320457,63 |
| **white\_total\_games\_played** | 8,73 | 2,00 | 1,00 | 12,13 | 147,18 |
| **white\_win\_rate** | 0,51 | 0,50 | 1,00 | 0,38 | 0,14 |
| **white\_average\_rating** | 1619,33 | 1594,82 | 1500,00 | 290,32 | 84285,87 |
| **white\_average\_duration** | 874,43 | 700,00 | 462,00 | 798,14 | 637020,61 |
| **black\_total\_games\_played** | 8,74 | 2,00 | 1,00 | 12,30 | 151,39 |
| **black\_win\_rate** | 0,48 | 0,50 | 0,00 | 0,38 | 0,14 |
| **black\_average\_rating** | 1610,40 | 1590,02 | 1500,00 | 288,79 | 83400,54 |
| **black\_average\_duration** | 869,02 | 693,09 | 583,00 | 1057,52 | 1118346,61 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  | **MIN** | **MAX** | **Intervalo** | **Assimetria** | **Curtose** |
| **turns** | 1 | 349 | 348,00 | 0,94 | 1,81 |
| **white\_rating** | 784 | 2700 | 1916,00 | 0,26 | 0,09 |
| **black\_rating** | 789 | 2621 | 1832,00 | 0,19 | 0,00 |
| **game\_duration** | 1 | 72349 | 72348,00 | 25,27 | 1416,72 |
| **white\_total\_games\_played** | 1 | 120 | 119,00 | 2,40 | 9,19 |
| **white\_win\_rate** | 0 | 1 | 1,00 | -0,05 | -1,36 |
| **white\_average\_rating** | 793 | 2700 | 1907,00 | 0,25 | 0,09 |
| **white\_average\_duration** | 2 | 21301 | 21299,00 | 5,39 | 64,19 |
| **black\_total\_games\_played** | 1 | 119 | 118,00 | 2,50 | 9,65 |
| **black\_win\_rate** | 0 | 1 | 1,00 | 0,07 | -1,36 |
| **black\_average\_rating** | 795 | 2621 | 1826,00 | 0,17 | -0,03 |
| **black\_average\_duration** | 2 | 72349 | 72347,00 | 0,94 | 1968,60 |

Essas estatísticas oferecem uma visão abrangente do desempenho, comportamento e características dos jogadores em partidas de xadrez. Elas incluem informações sobre a duração das partidas, classificações dos jogadores, taxas de vitória, número total de partidas jogadas e outros aspectos relevantes.

Gráfico

Descrição gerada automaticamente

Gráfico

Descrição gerada automaticamente

Uma imagem contendo Gráfico

Descrição gerada automaticamente

Gráfico, Gráfico de pizza

Descrição gerada automaticamente

Gráfico, Gráfico de pizza

Descrição gerada automaticamente

**ENTREGA 5**

*Inteligência Artificial*

Para o segundo método de aplicação de Inteligência Artificial, decidimos utilizar o **RandomForestClassifier.**

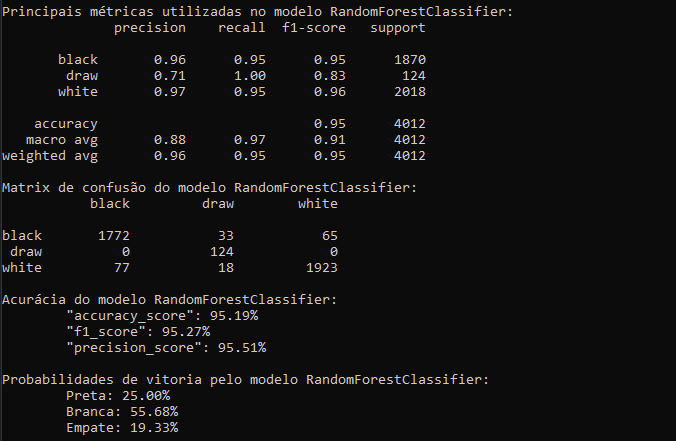
O **RandomForestClassifier** pertence à categoria de métodos ensemble, especificamente à classe de algoritmos conhecida como Floresta Aleatória (Random Forest). O **RandomForestClassifier** cria várias árvores de decisão durante o treinamento e combina suas previsões para obter uma decisão mais estável e geral.

Ensemble Learning é uma abordagem em que vários modelos individuais são combinados para formar um modelo mais robusto e geralmente mais preciso.

A utilização de todos os métodos se encontra no arquivo MAIN.PY, onde o método com os parâmetros iniciais é abordado na linha de código **26 a 28**.

Além dos parâmetros mencionados anteriormente, o Random Forest possui outros ajustáveis, como o número de árvores (**n\_estimators**), a profundidade máxima de cada árvore (**max\_depth**), entre outros.

Os resultados obtidos a partir do método acima se encontra na foto abaixo



**ENTREGA 5**

*Análise de Dados e Big Data*

**ENTREGA 6**

*Inteligência Artificial*

Para o terceiro método que utilizamos no nosso projeto para encontrar nossa variável target, foi usado o método **LogisticRegression.**

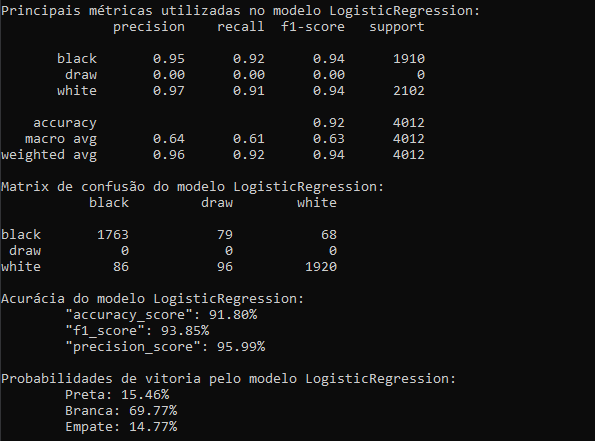
**LogisticRegression** é um algoritmo de aprendizado de máquina para tarefas de classificação binária e multiclasse. Apesar do nome "regressão", a **LogisticRegression** é utilizada para problemas de classificação.

O modelo calcula a probabilidade de pertencer a uma classe específica.

Um limiar de decisão é escolhido para converter as probabilidades em rótulos de classe (por exemplo, se a probabilidade for maior que 0,5, a classe é atribuída como 1; caso contrário, é atribuída como 0).

Utilizamos o Multi-class case para o treinamento do algoritmo, para problemas multiclass, apenas ‘newton-cg’, ‘sag’, ‘saga’ e ‘lbfgs’ lidam com perda multinomial, isso é, usa-se a perda de entropia cruzada se a opção 'multi\_class' estiver definida como 'multinomial ', a perda minimizada é a perda multinomial ajustada em toda a distribuição de probabilidade, mesmo quando os dados são binários.

Os resultados obtidos a partir do método acima se encontra na foto abaixo



**Comparação entre os métodos utilizados**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **accuracy\_score** | **f1\_score** | **precision\_score** |
| **HistGradientBoostingClassifier** | 95.01% | 95.09% | 95.31% |
| **RandomForestClassifier** | 95.19% | 95.27% | 95.51% |
| **LogisticRegression** | 91.80% | 93.85% | 95.99% |
| **DecisionTreeClassifier** | 92.10% | 92.11% | 92.12% |

Com base nos resultados apresentados, o método que apresentou o melhor desempenho em termos de métricas de avaliação foi o RandomForestClassifier. Ele obteve as pontuações mais altas em 2 das 3 métricas comparadas, sendo elas: accuracy\_score (95.19%), f1\_score (95.27%). Esses valores indicam que o RandomForestClassifier teve uma performance superior na tarefa de classificação em comparação com os outros métodos (HistGradientBoostingClassifier, LogisticRegression e DecisionTreeClassifier) considerados no projeto.

O RandomForestClassifier pode ter superado os outros métodos devidos a várias características inerentes ao algoritmo e à natureza do conjunto de dados. Aqui estão algumas razões pelas quais o RandomForestClassifier pode ter se destacado:

1. Ensemble Learning:
   * O RandomForestClassifier é um algoritmo de aprendizado ensemble que combina várias árvores de decisão. Essa abordagem permite que o modelo aproveite a diversidade das árvores para obter uma decisão mais robusta e geral.
2. Redução de Overfitting:
   * O RandomForestClassifier lida bem com o overfitting, um problema comum em modelos de árvores de decisão. A combinação de múltiplas árvores treinadas em subconjuntos diferentes do conjunto de dados ajuda a reduzir o sobreajuste e a melhorar a generalização do modelo.
3. Robustez:
   * O RandomForest é robusto em relação a outliers e ruído nos dados. A combinação de várias árvores ajuda a mitigar o impacto de pontos atípicos, tornando-o mais adequado para conjuntos de dados com características variadas.
4. Ajuste Automático de Hiperparâmetros:
   * O RandomForest permite a otimização automática de hiperparâmetros durante o treinamento por meio de métodos como o GridSearchCV. Isso pode resultar em um ajuste fino que melhora o desempenho geral.

**ENTREGA 6**

*Análise de Dados e Big Data*