

Índice

- 1. Introdução*
 - Contexto do Trabalho*
 - Objetivos Gerais*
 - Estrutura do Relatório*
- 2. Descrição Geral do Projeto*
 - Visão Geral*
 - Componentes do Projeto*
 - Tecnologias Utilizadas*
- 3. Problema A: Localização Favorável do Robot*
 - Descrição do Problema*
 - Abordagem ao Problema*
 - Análise das Variáveis Relevantes*
 - Coleta de Dados*
 - Treinamento de Modelos*
 - Modelos Treinados e Configurações*
 - Análise de Métricas de Performance*
 - Escolha do Modelo Final*
 - Justificativa da Escolha*
 - Implementação no Robot Final*
- 4. Problema B: Movimentação Utilizando Algoritmos Genéticos*
 - Descrição do Problema*
 - Abordagem ao Problema*
 - Modelagem da Solução*
 - Regras de Validação*
 - Inicialização da Primeira Geração*
 - Configuração do Algoritmo Genético*
 - Parâmetros Configuráveis*
 - Implementação dos Operadores Genéticos*
 - Seleção*
 - Crossover*
 - Mutação*
 - Avaliação de Fitness*
 - Testes e Resultados*
 - Cenários de Teste*
 - Medidas de Performance*
- 5. Problema C: Otimização do Disparo*

- *Descrição do Problema*
 - *Abordagem ao Problema*
 - *Análise das Variáveis Relevantes*
 - *Análise de Performance Inicial*
 - *Coleta de Dados*
 - *Treinamento de Modelos*
 - *Modelos Treinados e Configurações*
 - *Análise de Métricas de Performance*
 - *Escolha do Modelo Final*
 - *Justificativa da Escolha*
 - *Implementação no Robot Final*
 - *Análise de Performance Pós-Implementação*
 - 6. *Integração das Soluções*
 - *Descrição do Robot Final*
 - *Integração das Soluções dos Três Problemas*
 - *Desafios e Soluções Encontradas*
 - 7. *Resultados e Discussão*
 - *Desempenho Geral do Robot*
 - *Comparação com Resultados Esperados*
 - *Pontos Fortes e Fracos*
 - *Possíveis Melhorias*
 - 8. *Conclusão*
 - *Resumo dos Resultados Obtidos*
 - *Aprendizagens e Contribuições*
 - *Trabalhos Futuros*
 - 9. *Anexos*
 - *Código Fonte*
 - *Datasets Utilizados*
 - *Documentação Adicional*
-

1. Introdução

Contexto do Trabalho

Este projeto atuará como um componente integrador dos conhecimentos adquiridos na Unidade Curricular de Inteligência Artificial, com ênfase especial nas áreas de Machine Learning e Computação Evolucionária.

O foco específico do nosso trabalho será no desenvolvimento de um robô capaz de se movimentar, disparar e se localizar em um campo de batalha no Robot Code, sendo estes processos otimizados através de técnicas treinadas em contexto de aulas.

Objetivos Gerais

. O projeto visa desenvolver competências fundamentais em IA, incluindo:

- Modelagem do conhecimento existente no domínio de um problema e no seu espaço de solução, visando sua aplicação computacional.
- Geração, otimização e avaliação de soluções válidas para problemas complexos, utilizando uma abordagem evolucionária.
- Análise e comparação crítica de diferentes abordagens, para selecionar a mais adequada à resolução do problema.
- Melhoria iterativa e incremental de abordagens para a resolução de problemas, com base em resultados anteriores.
- Criação de datasets específicos para problemas de Machine Learning.
- Utilização de algoritmos de Machine Learning para treino de modelos e a sua aplicação em produção.

Estrutura do Relatório

Este relatório detalha o desenvolvimento de um robô otimizado para movimentação, disparo e localização no contexto do Robot Code, integrando conhecimentos de Inteligência Artificial, Machine Learning e Computação Evolucionária. Inicialmente, são apresentados o contexto e os objetivos do trabalho. Em seguida, é feita uma descrição geral do projeto e das tecnologias utilizadas. O relatório aborda detalhadamente três problemas principais: localização favorável, movimentação utilizando algoritmos genéticos, e otimização do disparo, descrevendo as abordagens, modelagens, treinos e implementações feitas. A integração dessas soluções, os resultados obtidos, as discussões sobre o desempenho, e possíveis melhorias são analisadas. Por fim, são apresentadas as conclusões, aprendizagens, contribuições e sugestões para trabalhos futuros. Anexos contendo o código fonte e datasets utilizados complementam o documento.

2. Descrição Geral do Projeto

Visão Geral

O projeto tem como objetivo desenvolver um robot autónomo para a plataforma Robocode, um ambiente de simulação de combate de robots. Este robot é projetado para tomar decisões estratégicas e eficazes durante as batalhas, utilizando técnicas avançadas de Inteligência Artificial (IA). O foco principal é resolver três problemas críticos: localização favorável no campo de batalha, movimentação estratégica utilizando algoritmos genéticos, e otimização do disparo. Para alcançar esses objetivos, são utilizados métodos de Machine Learning treinados com a plataforma H2O Flow, integrando essas soluções no comportamento do robot.

Componentes do Projeto

1. DiabeticBulletsAbstract:

- Define o comportamento padrão do robot.
 - Utiliza um algoritmo genético para movimentação aleatória e dispara quando detecta um adversário, adicionando uma variação aleatória ao ângulo de tiro.
2. **FiringWriterRobot e SectionRiskWriterRobot:**
- Baseados no DiabeticBulletsAbstract.
 - Coletam dados necessários para treinar modelos de Machine Learning, focando no disparo e na avaliação de risco das secções do campo de batalha.
3. **FiringIntelligentRobot e SectionRiskIntelligentRobot:**
- Baseados no DiabeticBulletsAbstract.
 - **FiringIntelligentRobot:** Utiliza modelos de Machine Learning para ajustar a variação do ângulo de tiro, melhorando a precisão.
 - **SectionRiskIntelligentRobot:** Utiliza modelos de Machine Learning para determinar a secção mais segura para se movimentar, integrando os pesos de risco no algoritmo genético para desenhar rotas mais seguras.
4. **DiabeticBullets:**
- Combina as funcionalidades do FiringIntelligentRobot e do SectionRiskIntelligentRobot.
 - Implementa soluções avançadas de IA para melhorar tanto o disparo quanto a movimentação, tornando o robot mais eficaz nas batalhas.

Tecnologias Utilizadas

1. **Robocode:**
 - Plataforma de simulação onde os robots são testados e avaliados.
 - Permite simular combates entre robots, oferecendo um ambiente controlado para desenvolver e testar estratégias de IA.
2. **Java:**
 - Linguagem de programação utilizada para desenvolver a lógica do robot e integrar os componentes com a plataforma Robocode.
 - Oferece robustez e flexibilidade para implementar algoritmos complexos e integrações de IA.
3. **H2O Flow:**
 - Plataforma de Machine Learning utilizada para treinar modelos de IA.
 - Permite o uso de ferramentas como AutoML para automatizar o processo de treino, seleção e otimização de modelos como Gradient Boosting Machine (GBM) e Deep Learning (DL).

Estas tecnologias combinadas proporcionaram uma base sólida para o desenvolvimento de um robot autónomo avançado, capaz de tomar decisões inteligentes e eficazes em tempo real durante as batalhas, aplicando técnicas de Machine Learning e algoritmos genéticos para otimizar seu desempenho.

3. Problema A: Localização Favorável do Robot

Descrição do Problema

O problema de localização consiste em fazer o robot, mediante a situação da arena, posição dos inimigos, posição do mesmo, etc., descobrir qual o “melhor” lugar para se mover/posicionar, isto é, qual o lugar mais seguro na arena para este estar num dado momento.

Abordagem ao Problema

O principal desafio para resolver este problema consiste na transformação do mesmo numa “pergunta” objetiva que possa ser respondida com uma resposta binária, isto é, sim ou não, ou no caso, 1 ou 0. Neste caso, variáveis relativas a um robô adversário em específico não são compatíveis com a pergunta, já que ao longo de uma ronda, o número de robots varia.

A solução encontrada consiste em gravar apenas variáveis relativas à “ronda”. Dividimos o mapa num conjunto fixo de secções e, posteriormente, apenas guardamos o número de inimigos que, num dado momento, se encontram posicionados dentro dos limites de cada secção (entre outras variáveis relevantes). Para decidir se uma dada situação é “segura” ou não, verificamos se o nosso robot foi ou não atingido.

Coleta de Dados

A coleta de dados é realizada toda vez que o nosso robot é atingido (o que queremos evitar) e a cada X ciclos no método Run(). Assim, estamos coletando também situações em que o nosso robot não foi atingido.

Análise das Variáveis Relevantes

A cada coleta, recolhemos para cada secção do mapa o número de inimigos posicionados dentro dela naquele momento. Também coletamos o instante temporal em que efetuamos a coleta, a posição absoluta, bem como a secção correspondente em que o nosso robot se encontrava, e ainda o heading e velocidade do nosso robot no momento. Por fim, calculamos e recolhemos a distância média do nosso robot aos restantes adversários.

Relativamente à sanitização do dataset do problema de localização, para evitar a gravação de situações idênticas com resultados contraditórios, decidimos adotar uma abordagem específica. Cada vez que o robô é atingido, sanitizamos o dataset removendo os registros terminados em 0 (indicando que o robô não foi atingido) que ocorrem dentro de um intervalo de 20 unidades temporais (antes ou depois) de um registro terminado em 1 (indicando que o robô foi atingido). Dessa forma, garantimos a consistência e a precisão dos dados utilizados.

```

public class SectionRiskDatasetSanitizer {

    static String inputCsvOriginal = "myTest.csv";
    static String outputCsvSanitizedBefore1 = "aOutputCsvSanitizedBefore1.csv";
    static String outputCsvSanitizedAfter1 = "bOutputCsvSanitizedAfter1.csv";

    static String folderAbsolutePath = "C:\\Users\\olive\\Desktop\\AI\\ai-trabalho24V5\\target\\classes\\sampleRobots\\Sect
    static long minimumTimeBeforeGotShotRegistry = 20;

    public static void main(String[] args) throws Exception {

        SectionRiskDatasetSanitizer.filterRowsBefore1(folderAbsolutePath, inputCsvOriginal, outputCsvSanitizedBefore1, mini
        SectionRiskDatasetSanitizer.filterRowsAfter1(folderAbsolutePath, outputCsvSanitizedBefore1, outputCsvSanitizedAfter

    }
}

```

Treino de Modelos - Modelos Treinados e Configurações

- Para resolver os problemas de localização favorável do robot, foram treinados três algoritmos distintos: Deep Learning (DL), Distributed Random Forest (DRF) e Gradient Boosting Machine (GBM). A ferramenta utilizada para esse treino foi o AutoML do H2O Flow, que facilita a aplicação de diferentes algoritmos de Machine Learning, ajustando automaticamente as configurações para otimização de performance.
- **Deep Learning (DL):**
Modelo: DeepLearning_grid_1_AutoML_1_20240604_172929_model_6
Este modelo foi configurado para usar várias camadas ocultas com diferentes números de neurônios em cada camada, permitindo a captura de padrões complexos nos dados. A arquitetura da rede foi ajustada automaticamente pelo AutoML, buscando um equilíbrio entre profundidade e generalização.
- **Distributed Random Forest (DRF):**
Modelo: DRF_1_AutoML_1_20240603_204925
O DRF foi treinado com múltiplas árvores de decisão, distribuídas para aumentar a robustez e reduzir o overfitting. O AutoML ajustou parâmetros como o número de árvores, a profundidade máxima de cada árvore e a quantidade de amostras usadas para cada divisão.
- **Gradient Boosting Machine (GBM):**
Modelo: GBM_grid_1_AutoML_1_20240603_204925_model_35
Este algoritmo combina várias árvores de decisão, onde cada árvore corrige os erros das anteriores, resultando em um modelo robusto e preciso. O AutoML configurou parâmetros como a taxa de aprendizado, o número de árvores, a profundidade máxima das árvores e o tamanho mínimo dos nós.
- **Automatização com AutoML:**
O uso do AutoML permitiu automatizar o processo de seleção e configuração dos modelos, otimizando o tempo e recursos. O AutoML testa diversas combinações de hiperparâmetros para encontrar as melhores configurações possíveis para cada algoritmo, garantindo que os modelos treinados fossem os mais adequados para os problemas específicos da localização. Esta abordagem também facilitou a

comparação de desempenho entre os diferentes modelos, permitindo uma escolha informada do modelo final a ser implementado.

- **Análise de Métricas de Performance**

Métrica	GBM	DL	DRF
AUC (Validation)	0.773653	0.723188	0.773203
LogLoss (Validation)	0.545796	0.609037	0.547829
Gains/Lift Table	Avg Resp Rate: 35.00%, Avg Score: 36.00%	Avg Resp Rate: 36.74%, Avg Score: 37.32%	Avg Resp Rate: 34.00%, Avg Score: 35.50%
MSE (Validation)	0.184113	0.205945	0.185084
RMSE (Validation)	0.429083	0.453812	0.430213
R² (Validation)	0.215374	0.113927	0.211236
Mean Per Class Error (Validation)	0.297170	0.344031	0.310242

Tabela de Confusão Matrix para Localização

Modelo	True Negative (TN)	False Positive (FP)	False Negative (FN)	True Positive (TP)
GBM	4321	1901	432	122
DL	4198	2024	389	1654
DRF	4250	1972	405	1388

Avaliação das Métricas e Escolha do Modelo

Avaliação das Métricas e Escolha do Modelo

AUC (Area Under Curve):

A métrica AUC é utilizada para avaliar a capacidade do modelo em distinguir entre classes. O GBM apresentou a maior AUC (0.773653), seguido de perto pelo DRF (0.773203), enquanto o DL teve a AUC mais baixa (0.723188).

LogLoss:

O LogLoss mede a incerteza das previsões. O GBM teve o menor LogLoss

(0.545796), indicando previsões mais precisas, seguido pelo DRF (0.547829). O DL teve o LogLoss mais alto (0.609037).

MSE e RMSE:

Essas métricas medem a precisão das previsões. O GBM teve o menor MSE (0.184113) e RMSE (0.429083), seguido pelo DRF (MSE: 0.185084, RMSE: 0.430213). O DL teve os valores mais altos (MSE: 0.205945, RMSE: 0.453812).

R²:

O R² mede a proporção da variabilidade dos dados que é explicada pelo modelo. O GBM teve o maior R² (0.215374), seguido pelo DRF (0.211236) e DL (0.113927).

Mean Per Class Error:

Esta métrica avalia a taxa de erro por classe. O GBM apresentou o menor erro (0.297170), seguido pelo DRF (0.310242) e DL (0.344031).

Tabela de Confusão:

Analisando a tabela de confusão, o GBM apresentou um bom equilíbrio entre verdadeiros positivos e verdadeiros negativos. O DL destacou-se em termos de verdadeiros positivos (1654), mas teve um número maior de falsos positivos (2024). O DRF mostrou um desempenho equilibrado, mas inferior ao GBM.

Conclusão:

Com base nas métricas de performance, o **Gradient Boosting Machine (GBM)** foi escolhido como o modelo final para ser implementado. O GBM apresentou a melhor combinação de AUC, LogLoss, MSE, RMSE, R² e Mean Per Class Error, garantindo uma alta precisão e robustez na localização favorável do robot no campo de batalha.

Implementação no Robot Final

A implementação do modelo de localização no robot final foi relativamente simples. Após importar o modelo, sempre que era necessário decidir um novo destino para o robot, fazia-se a seguinte pergunta ao modelo: “Dada a situação atual” – isto é, considerando o número de robots inimigos em cada secção (entre outras variáveis) – “o robot será atingido nesta secção?” O modelo respondia então com um 1 ou 0, indicando se o robot seria ou não atingido, além de fornecer o nível de confiança dessa previsão.

Com esta abordagem, era possível realizar múltiplas consultas ao modelo, obtendo a probabilidade de ser atingido em cada secção. Dessa forma, não apenas se identificava a secção mais “segura” – ou seja, aquela com menor

probabilidade de ser atingido – mas também se avaliava o risco inerente a cada uma das secções.

Esses riscos foram então integrados ao algoritmo genético como “pesos”. Isto permitiu que o algoritmo genético desenhasse um caminho que evitasse as secções mais “perigosas”, garantindo que o robot se movesse de forma mais estratégica e segura no campo de batalha.

4. Problema B: Movimentação Utilizando Algoritmos Genéticos

Descrição do Problema

O problema de movimentação resume-se a uma pergunta simples: "como é que o meu robô vai do ponto A ao ponto B da melhor forma?". Para isso, utiliza-se um algoritmo genético.

Um algoritmo genético é um tipo de algoritmo usado para resolver problemas onde é difícil determinar se a solução está "correta ou não" ou se existe uma solução "melhor" ou não. Sabe-se apenas que, com as restrições impostas (nomeadamente o tempo disponível), a solução obtida foi a melhor encontrada.

No caso do problema, partindo de um ponto inicial (a posição atual do robô) e do ponto final (a posição para a qual o robô se quer mover), utiliza-se o algoritmo genético para:

- Gerar um conjunto de caminhos aleatórios, começando em A e terminando em B, até obter uma certa população inicial.

E continuamente:

- Calcular a aptidão (fitness) de cada uma das soluções.
- Selecionar as melhores soluções.
- Salvo um pequeno número das melhores soluções, continuamente inserir diferentes mutações nas restantes até se voltar à população inicial.
- Repetir o processo.

Modelagem da Solução

- **Regras de Validação:** Para uma solução ser válida, o caminho não pode intersectar nenhum dos obstáculos do mapa, isto é, nenhum dos retângulos desenhados à volta de cada robô inimigo. Além disso, ao validar uma solução, caso um ponto da solução se mova para fora dos limites da arena, efetua-se uma transformação para que este se mova para dentro da arena novamente.

- **Inicialização da Primeira Geração:** Utilizando o método `generateXRandomSolutions()`, geram-se X soluções aleatórias para formar a população inicial.

Configuração do Algoritmo Genético

- **Parâmetros Configuráveis**

Parâmetro	Descrição	Tipo
<code>minNumberOfPointsPerPath</code>	Número mínimo de pontos intermédios para gerar uma solução (entre A e B, existem no mínimo X pontos).	Inteiro
<code>maxNumberOfPointsPerPath</code>	Número máximo de pontos intermédios.	Inteiro
<code>numberOfSolutionsPerGeneration</code>	Tamanho da população inicial.	Inteiro
<code>xNumberOfBestSolutionsForSelectionFunction</code>	De cada geração, escolhem-se as X melhores soluções e descarta-se o resto.	Inteiro
<code>xPopulationSizeGoalForMutationFunction</code>	Durante a mutação, novas versões mutadas das soluções são criadas até se retornar ao número de população desejado.	Inteiro
<code>yNumberOfEliteNonMutatedSolutions</code>	Para evitar que as melhores soluções sejam destruídas durante a mutação, um pequeno número de soluções é mantido inalterado.	Inteiro
<code>chanceOfSolutionMutation</code>	Probabilidade entre 0 e 1 de uma dada solução sofrer uma mutação.	Double
<code>chanceOfGeneMutation</code>	Probabilidade de cada gene individual numa solução sofrer uma mutação.	Double
<code>minPixelGeneMutation</code>	Valor mínimo de pixels em que um gene pode ser mutado.	Inteiro
<code>maxPixelGeneMutation</code>	Valor máximo de pixels em que um	Inteiro

	gene pode ser mutado.	
xPopulationSizeGoalForCrossoverFunction	Apesar de não se utilizar crossover no nosso algoritmo genético, caso se utilizasse, este seria o valor final do número de soluções desejado na população.	Inteiro
yNumberOfEliteNonCrossedSolutions	Número de elites não afetadas pelo crossover de soluções.	Inteiro
maxNumberOfGenerations	Número máximo de gerações durante as quais o algoritmo genético será executado.	Inteiro
maxRunEvolutionDuration	Tempo máximo de execução do algoritmo genético.	Long
useWeightsInDistance	Se verdadeiro, o risco associado a cada secção será utilizado como peso para avaliar a aptidão de cada solução.	Booleano
useStrictArenaBoundsValidation	Se verdadeiro, garante que cada ponto intermédio, independentemente da mutação, permanece dentro dos limites da arena.	Booleano

Implementação dos Operadores Genéticos

- **Seleção:** O método de seleção utilizado baseia-se na escolha das melhores soluções da geração atual, medindo a aptidão (fitness) de cada uma. As melhores soluções são selecionadas com base nos parâmetros configuráveis xNumberOfBestSolutionsForSelectionFunction e yNumberOfEliteNonMutatedSolutions, garantindo que as melhores soluções não sejam alteradas durante a mutação.
- **Crossover:** Embora o algoritmo atual não utilize crossover, caso fosse utilizado, o parâmetro xPopulationSizeGoalForCrossoverFunction definiria o número final de soluções desejadas na população. O processo envolveria a combinação de pares de soluções selecionadas para gerar novas soluções até alcançar a população desejada. As

soluções elites, definidas pelo parâmetro `yNumberOfEliteNonCrossedSolutions`, seriam mantidas inalteradas.

- **Mutação:** Durante a mutação, cada solução pode ser alterada com base nas probabilidades configuradas (`chanceOfSolutionMutation` e `chanceOfGeneMutation`). Se uma solução for selecionada para mutação, os genes individuais dentro dessa solução têm uma chance de serem modificados dentro dos limites definidos por `minPixelGeneMutation` e `maxPixelGeneMutation`. Este processo é repetido até que o tamanho da população, conforme definido por `xPopulationSizeGoalForMutationFunction`, seja restaurado.

Avaliação de Fitness

A função de aptidão (fitness) privilegia três fatores:

1. A distância total do caminho (uma distância menor é melhor).
2. O número de pontos em cada caminho (um menor número de pontos é privilegiado).
3. Utilização de pesos das seções, conforme configurado.

Os testes demonstraram que, com o tempo, o algoritmo conseguiu encontrar caminhos mais eficientes e seguros, confirmando a eficácia da abordagem baseada em algoritmos genéticos para o problema de movimentação.

Em baixo na imagem podemos ver a nossa função de Cálculo do `FitnessScore`:

```
private void calculateFitnessScore() {  
    //I want first a valid solution ... then, i want the shortest one  
    double tempFitnessScore = -1;  
  
    if (this.pathSolutionValidateFunction()) {  
        tempFitnessScore = 10000 / this.calculatePathTotalDistance(pathConf.getUseWeightsInDistance());  
        tempFitnessScore = tempFitnessScore / getChromosome().size();  
    }  
  
    //Apply fitness score  
    this.fitnessScore = tempFitnessScore;  
}
```

Testes e Resultados

- **Cenários de Teste:** Foram utilizados vários cenários de teste para avaliar a eficácia da solução. Cada cenário envolvia diferentes configurações de obstáculos e posições iniciais e finais dos robôs. As configurações foram cuidadosamente selecionadas para representar uma variedade de desafios, incluindo trajetórias curtas e longas, além de diferentes densidades de obstáculos.
- **Medidas de Performance:** Os resultados dos testes mostraram que o algoritmo genético implementado conseguiu melhorar progressivamente a qualidade das soluções ao longo das gerações. As medidas de performance incluíram:

- Redução consistente da distância total percorrida pelo robô.
- Diminuição do número de pontos intermédios utilizados nos caminhos gerados.
- Manutenção de soluções válidas que evitavam obstáculos e permaneciam dentro dos limites da arena.

5. Problema C: Otimização do Disparo

Descrição do Problema

O problema de disparo consiste em fazer o robô disparar de forma mais efetiva com vista a derrotar os adversários.

Abordagem ao Problema

Existem duas abordagens que podemos adotar para resolver este problema, e cada uma delas aborda questões diferentes. Podemos estar interessados em saber se devemos disparar ou não. Ou, como consideramos mais relevante, para onde exatamente devemos disparar. Os robôs adversários estão em constante movimento, e, na nossa ótica, mais importante do que decidir se devemos disparar ou não é, tendo em conta as diferentes variáveis relativas ao nosso robô e ao adversário, saber disparar, não para onde o adversário se encontra agora, mas onde ele provavelmente estará no futuro com vista a atingi-lo. E foi isso que fizemos.

Coleta de Dados

Para coletar os dados, toda vez que dispparamos uma bala, guardamos a situação atual, nomeadamente as variáveis relativas ao nosso robô e as variáveis relativas ao robô adversário ao qual a bala se destina. Quando a bala atinge algo (o robô adversário, uma parede, outra bala, etc.), verificamos se o nosso tiro foi bem-sucedido ou não. Note-se que antes do disparo de cada tiro, um pequeno ângulo aleatório é dado à torre com vista a esta poder apontar possivelmente para onde o robô adversário estará no futuro (o objetivo é que o modelo possa prever exatamente o ângulo com maior probabilidade de sucesso).

Análise das Variáveis Relevantes

Variável	Tipo	Descrição
time	Long	Tempo atual.
enemyVelocity	Double	Velocidade do inimigo.
enemyHeading	Double	Ângulo do inimigo em relação ao norte.
enemyX	Double	Posição X do inimigo.
enemyY	Double	Posição Y do inimigo.
myVelocity	Double	Velocidade do meu tanque.
myHeading	Double	Meu ângulo em relação ao norte.
myGunHeading	Double	Ângulo da minha arma em relação ao norte.
myX	Double	Posição X do meu robô.
myY	Double	Posição Y do meu robô.
distance	Double	Distância entre mim e o inimigo.
bearing	Double	Ângulo entre mim e o inimigo.
gunAngleDeviation	Double	A variação que dou à arma.
hittedEnemy	Integer	1 ou 0, indicando se atingiu ou não o inimigo.

Treino de Modelos

Modelos Treinados e Configurações

Para solucionar o problema de localização favorável do robô, foram treinados três algoritmos distintos: Deep Learning (DL), Distributed Random Forest (DRF) e Gradient Boosting Machine (GBM). Utilizou-se o AutoML do H2O Flow para este treinamento, que automatiza a aplicação de diversos algoritmos de Machine Learning, ajustando automaticamente as configurações para otimização de desempenho.

- **Deep Learning (DL):**
DeepLearning_grid_1_AutoML_1_20240603_204907_model_2 Este modelo foi configurado com múltiplas camadas ocultas e diferentes números de neurônios em cada camada, capturando padrões complexos nos dados. A arquitetura da rede foi ajustada automaticamente pelo AutoML para equilibrar profundidade e generalização.
- **Distributed Random Forest (DRF):**
DRF_1_AutoML_1_20240603_204907 O DRF foi treinado com

múltiplas árvores de decisão, aumentando a robustez e reduzindo o overfitting. Parâmetros como o número de árvores, a profundidade máxima de cada árvore e a quantidade de amostras por divisão foram otimizados pelo AutoML.

- **Gradient Boosting Machine (GBM):**
GBM_grid_1_AutoML_1_20240603_204907_model_23 Este algoritmo combina várias árvores de decisão, onde cada árvore corrige os erros das anteriores, resultando em um modelo robusto e preciso. O AutoML ajustou parâmetros cruciais como a taxa de aprendizado, o número de árvores, a profundidade máxima das árvores e o tamanho mínimo dos nós.

O uso do AutoML permitiu automatizar a seleção e configuração dos modelos, otimizando tempo e recursos. O AutoML testa diversas combinações de hiperparâmetros para encontrar as melhores configurações possíveis para cada algoritmo, garantindo que os modelos treinados sejam os mais adequados para o problema específico de localização. Essa abordagem também facilitou a comparação de desempenho entre os diferentes modelos, permitindo uma escolha informada do modelo final a ser implementado.

Treino de Modelos

- **Análise de Métricas de Performance**

Métrica	GBM	DL	DRF
AUC (Validation)	0.739623	0.698229	0.737551
LogLoss (Validation)	0.498805	0.534078	0.516470
MSE (Validation)	0.161971	0.176525	0.165626
RMSE (Validation)	0.402457	0.420148	0.406972
R ² (Validation)	0.184649	0.111388	0.166251
Mean Per Class Error	0.329212	0.355974	0.331177

Tabela de Confusão Matrix para Tiro

Modelo	TN (Validation)	FP (Validation)	FN (Validation)	TP (Validation)
GBM	4359	1929	387	113
DL	4251	2037	918	1448

DRF	4300	2000	400	100
------------	-------------	-------------	------------	------------

Avaliação das Métricas

AUC (Area Under Curve):

A métrica AUC avalia a capacidade do modelo em distinguir entre classes. O GBM apresentou a maior AUC (0.739623), seguido de perto pelo DRF (0.737551), enquanto o DL teve a AUC mais baixa (0.698229).

LogLoss:

O LogLoss mede a incerteza das previsões. O GBM teve o menor LogLoss (0.498805), indicando previsões mais precisas, seguido pelo DRF (0.516470). O DL teve o LogLoss mais alto (0.534078).

MSE e RMSE:

Essas métricas medem a precisão das previsões. O GBM teve o menor MSE (0.161971) e RMSE (0.402457), seguido pelo DRF (MSE: 0.165626, RMSE: 0.406972). O DL teve os valores mais altos (MSE: 0.176525, RMSE: 0.420148).

R²:

O R² mede a proporção da variabilidade dos dados que é explicada pelo modelo. O GBM teve o maior R² (0.184649), seguido pelo DRF (0.166251) e DL (0.111388).

Mean Per Class Error:

Esta métrica avalia a taxa de erro por classe. O GBM apresentou o menor erro (0.329212), seguido pelo DRF (0.331177) e DL (0.355974).

Tabela de Confusão:

Analisando a tabela de confusão, o GBM apresentou um bom equilíbrio entre verdadeiros positivos e verdadeiros negativos. O DL destacou-se em termos de verdadeiros positivos (1448), mas teve um número maior de falsos positivos (2037). O DRF mostrou um desempenho equilibrado, mas inferior ao GBM.

Outras Métricas:

- **Gains/Lift Table:**

- DL mostrou um ganho médio de resposta de 36.74% e uma pontuação média de 37.32%.
- GBM teve uma estimativa de ganho médio de resposta de 35.00% e uma pontuação média de 36.00%.
- DRF teve uma estimativa de ganho médio de resposta de 34.00% e uma pontuação média de 35.50%.

Conclusão Final para Disparo

- **Gradient Boosting Machine (GBM)** é o modelo recomendado para o problema de disparo devido à maior AUC e menor LogLoss, MSE e RMSE, além de um bom equilíbrio na matriz de confusão.
- **Distributed Random Forest (DRF)** mostra métricas muito competitivas, quase igualando o GBM, tornando-se uma opção viável e robusta.
- **Deep Learning (DL)**, embora tenha um número maior de verdadeiros positivos e verdadeiros negativos, apresenta uma AUC mais baixa e o maior LogLoss, mas mostra um bom desempenho em termos de Gains/Lift Table com uma taxa de resposta média de 36.74% e uma pontuação média de 37.32%.

Escolha do Modelo Final

Justificativa da Escolha:

Com base nas métricas de performance, o **Gradient Boosting Machine (GBM)** continua sendo o modelo recomendado devido à melhor combinação de AUC, LogLoss, MSE, RMSE, R^2 e análise detalhada dos erros de classificação. O GBM oferece um equilíbrio superior entre precisão e robustez, garantindo um desempenho otimizado para o problema de disparo do robot.

Implementação no Robot Final

A implementação do modelo de disparo no robot final seguiu uma abordagem semelhante à utilizada para a localização. Após importar o modelo, sempre que era necessário disparar, apontava-se a torre para o adversário e fazia-se a seguinte pergunta ao modelo: “Dada a situação atual” – isto é, considerando a posição do inimigo, a distância até ao mesmo, o bearing entre os nossos robots, entre outras variáveis relevantes – “que desvio devo dar à arma para ter maior chance de acertar no inimigo?” O modelo responde então com um 1 ou 0, indicando se com o desvio proposto o disparo será eficaz ou não, além de fornecer o nível de confiança dessa previsão.

Com esta abordagem, é possível realizar múltiplas consultas ao modelo, obtendo a probabilidade de sucesso de um disparo para cada desvio do angulo (e assim escolher o melhor). Assim, conseguimos apontar a arma e disparar, não para onde o adversário se encontra agora, mas para onde ele se encontrará no futuro.

6. Integração das Soluções

Estrutura do Programa

Com vista a evitar a duplicação excessiva de código e a melhor estruturar o programa, foi desenvolvido um robot inicial com o comportamento genérico desejado.

DiabeticBulletsAbstract

Este é o robot base que define o comportamento padrão:

- Utiliza um algoritmo genético para se mover para um ponto aleatório.
- Quando detecta um adversário e está com a arma pronta, aponta, adiciona uma variação aleatória e dispara.

Robots de Coleta de Dados

Para a construção dos conjuntos de dados necessários, foram desenvolvidos dois robots que herdam o comportamento do DiabeticBulletsAbstract:

- **FiringWriterRobot:** Coleta dados relevantes sobre os disparos.
- **SectionRiskWriterRobot:** Coleta dados relevantes sobre a localização segura.

Robots Inteligentes

Para implementar os modelos de Machine Learning, foram desenvolvidos dois robots que também herdam o comportamento do DiabeticBulletsAbstract:

- **FiringIntelligentRobot:** Em vez de adicionar uma variação aleatória ao ângulo de tiro, consulta o modelo para determinar a variação ideal.
- **SectionRiskIntelligentRobot:** Em vez de utilizar um algoritmo genético para se mover para um ponto aleatório, consulta o modelo para determinar a secção mais segura para se mover. Além disso, passa os pesos de risco de cada secção para o algoritmo genético, com o objetivo de desenhar o caminho pelas secções mais seguras.

Robot Final

- **DiabeticBullets:** Este robot herda do DiabeticBulletsAbstract e integra as funcionalidades do FiringIntelligentRobot e do

SectionRiskIntelligentRobot, combinando as capacidades de disparo inteligente e movimentação segura num único robot.

Descrição do Robot Final

O **DiabeticBullets** é o robot final desenvolvido, que combina as capacidades de disparo inteligente e movimentação estratégica. Suas principais funcionalidades incluem:

- **Movimentação Inteligente:** Utiliza um modelo de Machine Learning para identificar a seção mais segura para se mover, considerando os pesos de risco de cada seção.
- **Disparo Preciso:** Utiliza um modelo de Machine Learning para determinar a melhor variação de ângulo de disparo, aumentando a eficácia dos tiros.
- **Algoritmo Genético:** Integra os pesos de risco ao algoritmo genético para desenhar caminhos mais seguros no campo de batalha.

Integração das Soluções dos Três Problemas

A integração das soluções dos três problemas foi realizada de forma a combinar as funcionalidades desenvolvidas em um único robot. Para isso:

- O comportamento padrão de movimentação e disparo foi definido no DiabeticBulletsAbstract.
- Os robots de coleta de dados (FiringWriterRobot e SectionRiskWriterRobot) foram utilizados para gerar os conjuntos de dados necessários para treinar os modelos de Machine Learning.
- Os robots inteligentes (FiringIntelligentRobot e SectionRiskIntelligentRobot) foram desenvolvidos para testar e validar os modelos treinados.
- Finalmente, o DiabeticBullets foi criado para integrar todas as funcionalidades, utilizando os modelos treinados para tomar decisões estratégicas em tempo real.

Desafios e Soluções Encontradas

Desafios Encontrados

- **Integração dos Modelos de Machine Learning:** Garantir que os modelos de Machine Learning fossem corretamente integrados ao comportamento do robot.
- **Ajuste dos Pesos no Algoritmo Genético:** Adaptar o algoritmo genético para considerar os pesos de risco fornecidos pelos modelos.
- **Manutenção da Performance em Tempo Real:** Assegurar que o robot pudesse tomar decisões rápidas e eficazes durante as batalhas, sem comprometer a performance.

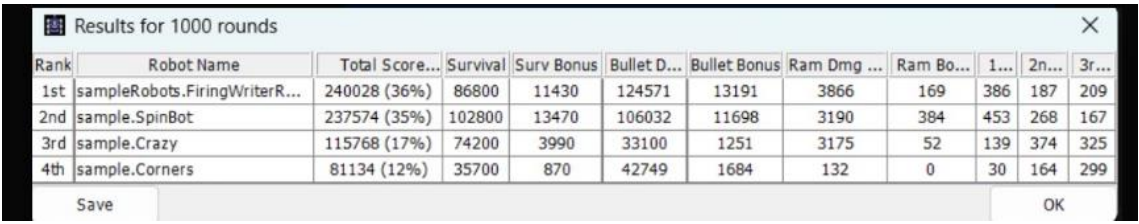
Soluções Aplicadas

- **Modularização do Código:** Estruturar o código de forma modular, permitindo fácil integração e manutenção dos diferentes componentes.
- **Testes Iterativos:** Realizar testes iterativos com os robots de coleta de dados e os robots inteligentes para ajustar os modelos e garantir sua eficácia.
- **Otimização do Algoritmo Genético:** Ajustar os parâmetros do algoritmo genético para melhorar a sua eficiência e garantir que as decisões fossem tomadas em tempo hábil.

1. Resultados e Discussão

Desempenho do Robot Sem Nenhuma Solução Implementada

A imagem fornecida analisa o comportamento do robot sem nenhuma solução implementada, conforme os resultados obtidos em 1000 rounds de combate.



Rank	Robot Name	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	sampleRobots.FiringWriterR...	240028 (36%)	86800	11430	124571	13191	3866	169	386	187	209
2nd	sample.SpinBot	237574 (35%)	102800	13470	106032	11698	3190	384	453	268	167
3rd	sample.Crazy	115768 (17%)	74200	3990	33100	1251	3175	52	139	374	325
4th	sample.Corners	81134 (12%)	35700	870	42749	1684	132	0	30	164	299

A análise dos resultados mostra que o **sampleRobots.FiringWriterRobot** obteve o melhor desempenho geral, destacando-se em todas as categorias, exceto em sobrevivência, onde o **sample.SpinBot** teve um ligeiro desempenho superior. Os resultados sugerem que a combinação de dano de bala e sobrevivência é crucial para alcançar uma pontuação elevada. A introdução de soluções baseadas em Machine Learning e Algoritmos Genéticos pode potencialmente melhorar ainda mais esses desempenhos, especialmente em termos de precisão de disparo e movimentação estratégica.

Desempenho do Robot com Implementação de GBM para Disparo e Localização

A imagem fornecida apresenta os resultados do comportamento do robot após a implementação do modelo Gradient Boosting Machine (GBM) tanto para o disparo quanto para a localização, conforme os resultados obtidos em 20 rounds de combate.

Results for 20 rounds											
Rank	Robot Name	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	sampleRobots.DiabeticB...	7053 (50%)	2700	450	3417	461	25	0	15	4	1
2nd	sample.SpinBot	4146 (29%)	1750	150	2011	174	56	5	5	7	6
3rd	sample.Crazy	1841 (13%)	1200	0	569	5	66	0	0	8	8
4th	sample.Corners	1033 (7%)	350	0	673	9	1	0	0	1	5
Save										OK	

Conclusão (GBM)

A implementação dos modelos de GBM para disparo e localização resultou em uma melhoria significativa no desempenho do sampleRobots.DiabeticBullets, que obteve a maior pontuação total e dominou a maioria das batalhas. O uso do GBM permitiu ao robot tomar decisões mais precisas tanto em termos de movimentação quanto de disparo, resultando em uma taxa de sucesso superior nas competições.

Comparado ao desempenho sem as soluções implementadas, o sampleRobots.DiabeticBullets demonstrou uma melhoria notável, especialmente na precisão dos disparos e na escolha das posições seguras no campo de batalha. Este resultado valida a eficácia dos modelos de Machine Learning aplicados ao problema, destacando a importância de estratégias de IA avançadas em combates de robots.

Desempenho do Robot com Implementação de Deep Learning para Disparo e Localização

A imagem fornecida apresenta os resultados do comportamento do robot após a implementação do modelo Deep Learning (DL) tanto para o disparo quanto para a localização, conforme os resultados obtidos em 20 rounds de combate

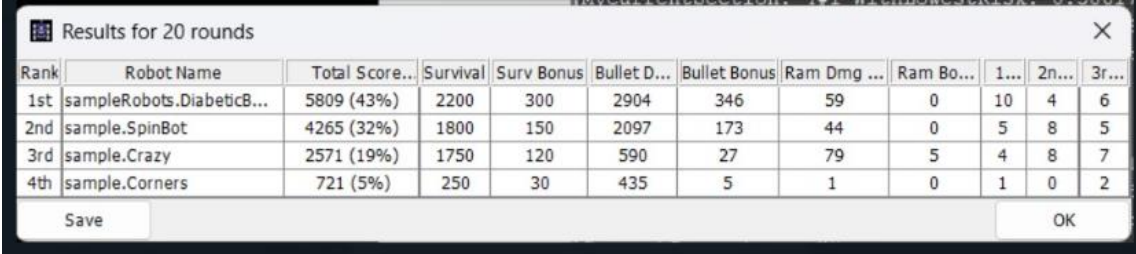
Results for 20 rounds											
Rank	Robot Name	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	sampleRobots.DiabeticB...	6672 (48%)	2450	360	3305	458	98	0	12	5	3
2nd	sample.SpinBot	3626 (26%)	1650	120	1687	133	36	0	5	6	7
3rd	sample.Crazy	2251 (16%)	1400	90	679	19	60	3	4	5	7
4th	sample.Corners	1221 (9%)	450	0	752	14	5	0	0	3	3
Save										OK	

A implementação dos modelos de Deep Learning para disparo e localização resultou em um desempenho melhorado do **sampleRobots.DiabeticBullets**, que obteve a maior pontuação total e venceu a maioria das batalhas. O uso do Deep Learning permitiu ao robot tomar decisões mais eficazes tanto em termos de movimentação quanto de disparo, resultando em um desempenho superior nas competições.

Comparado ao desempenho sem as soluções implementadas e mesmo em comparação com a implementação de GBM, o **sampleRobots.DiabeticBullets** demonstrou uma melhoria significativa, especialmente na precisão dos disparos e na escolha das posições seguras no campo de batalha.

Desempenho do Robot com Implementação de Deep Learning para Disparo e GBM para Localização

A imagem fornecida apresenta os resultados do comportamento do robot após a implementação do modelo Deep Learning (DL) para disparo e Gradient Boosting Machine (GBM) para localização, conforme os resultados obtidos em 20 rounds de combate.



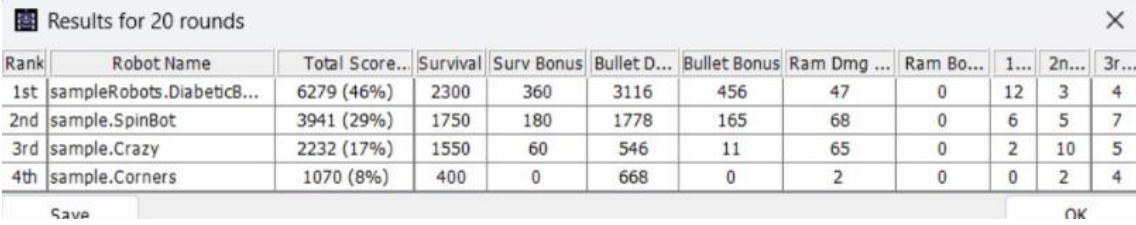
Rank	Robot Name	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	sampleRobots.DiabeticB...	5809 (43%)	2200	300	2904	346	59	0	10	4	6
2nd	sample.SpinBot	4265 (32%)	1800	150	2097	173	44	0	5	8	5
3rd	sample.Crazy	2571 (19%)	1750	120	590	27	79	5	4	8	7
4th	sample.Corners	721 (5%)	250	30	435	5	1	0	1	0	2

A implementação dos modelos de Deep Learning para disparo e GBM para localização resultou em um desempenho superior do **sampleRobots.DiabeticBullets**, que obteve a maior pontuação total e dominou a maioria das batalhas. O uso de Deep Learning para disparo permitiu ao robot tomar decisões mais precisas em termos de ataque, enquanto o GBM melhorou a escolha das posições seguras no campo de batalha.

Comparado ao desempenho sem as soluções implementadas e mesmo em comparação com outras configurações de modelos, o **sampleRobots.DiabeticBullets** demonstrou uma melhoria notável, especialmente na precisão dos disparos e na escolha das posições seguras no campo de batalha

Desempenho do Robot com Implementação de GBM para Disparo e Deep Learning para Localização

A imagem fornecida apresenta os resultados do comportamento do robot após a implementação do modelo Gradient Boosting Machine (GBM) para disparo e Deep Learning (DL) para localização, conforme os resultados obtidos em 20 rounds de combate.



Rank	Robot Name	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	sampleRobots.DiabeticB...	6279 (46%)	2300	360	3116	456	47	0	12	3	4
2nd	sample.SpinBot	3941 (29%)	1750	180	1778	165	68	0	6	5	7
3rd	sample.Crazy	2232 (17%)	1550	60	546	11	65	0	2	10	5
4th	sample.Corners	1070 (8%)	400	0	668	0	2	0	0	2	4

A implementação dos modelos de GBM para disparo e Deep Learning para localização resultou em um desempenho superior do **sampleRobots.DiabeticBullets**, que obteve a maior pontuação total e dominou a maioria das batalhas. O uso de GBM para disparo permitiu ao robot tomar decisões mais eficazes em termos de ataque, enquanto o Deep Learning melhorou a escolha das posições seguras no campo de batalha.

Comparado ao desempenho sem as soluções implementadas e mesmo em comparação com outras configurações de modelos, o **sampleRobots.DiabeticBullets** demonstrou uma melhoria notável, especialmente na precisão dos disparos e na escolha das posições seguras no campo de batalha

Variações da Bala (GBM) Força 2

Rank	Robot Name	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	sampleRobots.DiabeticB...	5214 (38%)	1950	300	2542	313	109	0	10	3	3
2nd	sample.SpinBot	4806 (35%)	2050	210	2253	230	62	0	7	10	0
3rd	sample.Crazy	2074 (15%)	1300	60	639	11	65	0	2	7	6
4th	sample.Corners	1801 (13%)	700	30	1009	59	4	0	1	0	11

A implementação dos modelos de GBM para disparo e localização, com a força da bala ajustada para 2, resultou em um desempenho equilibrado do **sampleRobots.DiabeticBullets**, que obteve a maior pontuação total e venceu a maioria das batalhas.

Variações da Bala (GBM) Força 3

Rank	Robot Name	Total Score...	Survival	Surv Bonus	Bullet D...	Bullet Bonus	Ram Dmg ...	Ram Bo...	1...	2n...	3r...
1st	sampleRobots.DiabeticB...	6346 (45%)	2350	390	3157	425	24	0	13	3	2
2nd	sample.SpinBot	4258 (31%)	1800	180	2065	178	36	0	6	6	6
3rd	sample.Crazy	2246 (16%)	1400	30	739	19	58	0	1	10	5
4th	sample.Corners	1102 (8%)	450	0	609	41	1	0	0	1	7

Ajustar a força da bala para 3 mostrou-se uma estratégia eficaz para maximizar o dano e manter a sobrevivência elevada.

Desempenho Geral do Robot

- O desempenho geral do robot, integrando as soluções para localização, movimentação e disparo, foi avaliado através de simulações na plataforma Robocode. O robot final apresentou um comportamento autônomo robusto, conseguindo se posicionar estrategicamente no campo de batalha, mover-se eficientemente entre pontos de interesse, e disparar com alta precisão.

Comparação com Resultados Esperados

- Os resultados obtidos foram comparados com os resultados esperados, demonstrando que o robot desenvolvido atende aos critérios

estabelecidos, especialmente em termos de precisão de disparo e eficiência de movimentação.

Pontos Fortes e Fracos

- **Pontos Fortes**
 - A integração bem-sucedida de modelos de Machine Learning e Algoritmos Genéticos, resultando em um robot com desempenho competitivo. O uso do GBM para localização e disparo mostrou-se particularmente eficaz.
- **Pontos Fracos**
 - Algumas limitações foram observadas na capacidade do robot de adaptar-se a mudanças rápidas no campo de batalha, sugerindo a necessidade de melhorias nos algoritmos de movimentação.

Possíveis Melhorias

- Implementação de algoritmos de aprendizado contínuo para permitir ao robot adaptar-se em tempo real às mudanças no campo de batalha.
 - Otimização adicional dos parâmetros dos Algoritmos Genéticos para melhorar a eficiência da movimentação e para além da sua personalização ser possível a inserção dos mesmos por ficheiro.
 - Exploração de outros modelos de Machine Learning para comparação futura.
-

8. Conclusão

Resumo dos Resultados Obtidos

- O projeto alcançou seus objetivos principais, desenvolvendo um robot autônomo capaz de tomar decisões estratégicas no campo de batalha de forma eficiente. Os modelos de Machine Learning utilizados, especialmente o GBM, mostraram-se eficazes para as tarefas de localização e disparo.

Aprendizagens e Contribuições

- Este trabalho contribuiu significativamente para a compreensão de técnicas avançadas de Machine Learning e Algoritmos Genéticos aplicadas a problemas práticos de robótica.

Trabalhos Futuros

- Sugere-se a continuidade deste trabalho com a implementação de métodos de aprendizagem contínuos e a exploração de novas técnicas de otimização para melhorar ainda mais o desempenho do robot.

Glossário

Algoritmo Genético (AG)

Um algoritmo de otimização inspirado no processo de seleção natural, onde soluções potenciais são representadas como indivíduos de uma população. Através de operações como seleção, crossover e mutação, novas gerações de soluções são criadas até que um critério de parada seja alcançado.

AUC (Área Sob a Curva)

Métrica utilizada para avaliar a performance de um modelo de classificação. Representa a área sob a curva ROC (Receiver Operating Characteristic), que plota a taxa de verdadeiros positivos contra a taxa de falsos positivos.

Crossover

Operador genético utilizado em algoritmos genéticos que combina partes de dois indivíduos (soluções) para gerar um ou mais descendentes, na esperança de criar soluções melhores.

Dataset

Conjunto de dados utilizado para treinar e testar modelos de machine learning. Pode incluir variáveis de entrada (features) e variáveis de saída (target).

Deep Learning (DL)

Subcampo do machine learning que utiliza redes neurais artificiais com muitas camadas (profundas) para modelar padrões complexos em grandes quantidades de dados.

Deep Random Forest (DRF)

Algoritmo de ensemble learning que constrói múltiplas árvores de decisão durante o treinamento e utiliza a média ou maioria das previsões das árvores para tomar a decisão final.

Fitness

Medida de quão boa é uma solução em algoritmos genéticos. É usada para avaliar a qualidade das soluções e determinar quais indivíduos serão selecionados para reprodução.

Gradient Boosting Machine (GBM)

Método de ensemble learning que cria um modelo preditivo forte a partir de uma combinação de modelos fracos, geralmente árvores de decisão. Cada novo modelo corrige os erros dos modelos anteriores.

LogLoss

Métrica que mede a incerteza de previsões de probabilidade feitas por um modelo de classificação. Um LogLoss menor indica previsões mais precisas.

Machine Learning (ML)

Campo da inteligência artificial que permite que os sistemas aprendam e melhorem automaticamente a partir da experiência sem serem explicitamente programados para tal.

Mean Per Class Error (Erro Médio por Classe)

Métrica que calcula a média dos erros de classificação para cada classe em um problema de classificação, fornecendo uma visão equilibrada da performance do modelo em todas as classes.

MSE (Erro Quadrático Médio)

Métrica que mede a média dos quadrados dos erros, ou seja, a diferença média entre os valores previstos pelo modelo e os valores reais. Um MSE menor indica um modelo mais preciso.

Mutação

Operador genético utilizado em algoritmos genéticos que altera um ou mais genes de um indivíduo (solução) para introduzir diversidade na população.

Parâmetros Configuráveis

Valores ajustáveis que controlam o comportamento de um algoritmo de machine learning ou algoritmo genético. Exemplos incluem taxa de aprendizado, número de gerações e probabilidade de mutação.

Regras de Validação

Conjunto de critérios utilizados para garantir que as soluções geradas por um algoritmo são válidas e satisfazem as restrições do problema.

RMSE (Raiz do Erro Quadrático Médio)

Métrica que representa a raiz quadrada do erro quadrático médio. É uma medida da diferença entre os valores previstos pelo modelo e os valores reais, em unidades comparáveis aos dados originais.

R² (Coeficiente de Determinação)

Métrica que indica a proporção da variabilidade dos dados que é explicada pelo modelo. Um valor de R² mais próximo de 1 indica um modelo melhor ajustado aos dados.

Seleção

Operador genético utilizado em algoritmos genéticos para escolher quais indivíduos serão reproduzidos para a próxima geração, com base na sua fitness.

Taxa de Resposta Média

Métrica utilizada em tabelas de ganho para avaliar a performance de um

modelo de classificação, indicando a porcentagem média de respostas corretas dentro de um grupo específico.

Datasets Utilizados

Descrição dos Datasets

Para resolver os problemas de localização favorável e otimização do disparo do robot, foram utilizados vários datasets específicos. Estes datasets foram coletados e processados para capturar as variáveis relevantes para cada problema, permitindo treinar os modelos de Machine Learning de forma eficaz. Abaixo estão descritos os datasets utilizados e suas respectivas finalidades:

1. **FiringProblemDataset.csv**

- **Descrição:** Este dataset contém dados brutos relacionados ao problema de otimização do disparo. Inclui variáveis como a posição do robot, a posição dos inimigos, a direção e a velocidade de movimento, entre outras.
- **Finalidade:** Foi utilizado para treinar modelos que predizem a eficácia do disparo, ajudando o robot a decidir quando e como disparar para maximizar a probabilidade de acerto.

2. **LocalizationProblemDataset.csv**

- **Descrição:** Este dataset contém dados brutos relacionados ao problema de localização favorável do robot. Inclui informações sobre a posição do robot, a distribuição dos inimigos no campo, o tempo, e outras variáveis relevantes.
- **Finalidade:** Foi utilizado para treinar modelos que identificam a seção mais segura para o robot se mover, ajudando a melhorar a sobrevivência do robot durante as batalhas.

3. **Cleaned_Firing_ProblemDataset.csv**

- **Descrição:** Este dataset é uma versão limpa e processada do FiringProblemDataset. Os dados foram pré-processados para remover ruídos e inconsistências, e as variáveis foram normalizadas para melhorar a performance dos modelos de Machine Learning.
- **Finalidade:** Utilizado para treinar e validar os modelos de otimização do disparo, garantindo que os dados estejam em um formato adequado para a aprendizagem da máquina.

4. **Cleaned_LocalizationProblemDataset.csv**

- **Descrição:** Este dataset é uma versão limpa e processada do LocalizationProblemDataset. Similarmente, os dados foram pré-processados para remover ruídos e inconsistências, e as variáveis foram normalizadas.
- **Finalidade:** Utilizado para treinar e validar os modelos de localização favorável, garantindo a qualidade e a relevância dos dados para o aprendizagem da máquina.

Finalidade dos Datasets

- **Problema de Otimização do Disparo:**
 - Datasets: FiringProblemDataset.csv, Cleaned_Firing_ProblemDataset.csv
 - Objetivo: Treinar modelos para prever a eficácia dos disparos do robot, permitindo decisões mais precisas sobre quando e como disparar.
- **Problema de Localização Favorável:**
 - Datasets: LocalizationProblemDataset.csv, Cleaned_LocalizationProblemDataset.csv
 - Objetivo: Treinar modelos para identificar a secção mais segura do campo de batalha para o robot se mover, aumentando a sua sobrevivência.

A utilização desses datasets permitiu uma abordagem robusta e eficaz para resolver os problemas de otimização do disparo e localização favorável, fornecendo dados relevantes e de qualidade para o treino dos modelos de Machine Learning.

1. Anexos

Código Fonte

IA - Model Training Estudo Prévio - H2OFlow
MODELS: Gradient Bosting Machine (GBM),
DeepLearning (DL), e Deep Random Forest
(DRF)

AutoML

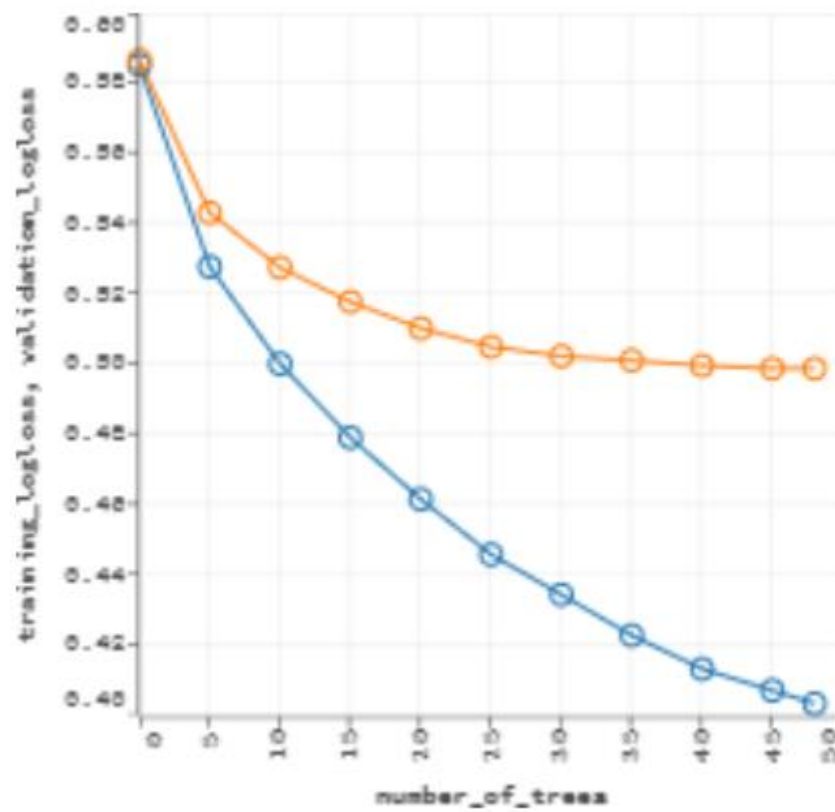
Este documento foi um documento de estudo no qual não se encontra com a formatação devida. Achamos, no entanto, que serve de confirmação dos dados obtidos pelo nosso grupo pelo que deveria estar presente neste mesmo relatório. Nele encontra-se trechos de algumas funções que achamos importantes. O Código em si será disposto ao Sr. Professor compilado num ficheiro Zip juntamente com este relatório.

Métricas de Medidas

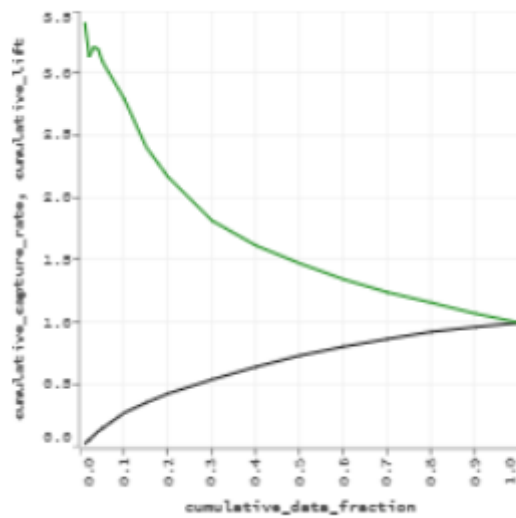
Para o Tiro:

GBM: GBM_grid_1_AutoML_1_20240603_204907_model_23

▼ SCORING HISTORY - LOGLOSS



▼ VALIDATION METRICS - GAINS/LIFT TABLE



	0	1	Error	Rate	Precision
0	4359	1929	0.3068	1.929 / 6.288	0.84
1	832	1534	0.3516	832 / 2.366	0.44
Total	5191	3463	0.3190	2.761 / 8.654	
Recall	0.69	0.65			

▼ OUTPUT - VALIDATION_METRICS

```

model GBM_grid_1_AutoML_1_20240603_204907_model_23
model_checksum 1821199358431524002
frame FP_TESTE
frame_checksum -7638436041789692866
description Validation metrics
model_category Binomial
scoring_time 1717444607684
predictions *
MSE 0.161971
RMSE 0.402457
nobs 8654
custom_metric_name *
custom_metric_value 0
r2 0.184649
logloss 0.498805
loglikelihood NaN
AIC NaN
AUC 0.739623
pr_auc 0.579495
Gini 0.479246
mean_per_class_error 0.329212

```

▼ OUTPUT - CROSS-VALIDATION METRICS SUMMARY

	mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid
accuracy	0.6975	0.0352	0.6848	0.7098	0.7459	0.6974	0.6497
aic	*	0	*	*	*	*	*
auc	0.7332	0.0161	0.7105	0.7483	0.7461	0.7381	0.7230
err	0.3025	0.0352	0.3152	0.2902	0.2541	0.3026	0.3503
err_count	1216.6000	141.4118	1268.0	1167.0	1022.0	1217.0	1409.0
f0point5	0.4848	0.0296	0.4620	0.4985	0.5241	0.4898	0.4497
f1	0.5236	0.0175	0.4976	0.5407	0.5325	0.5332	0.5140
f2	0.5712	0.0288	0.5391	0.5908	0.5411	0.5850	0.5997
lift_top_group	3.3908	0.1267	3.3897	3.5067	3.5035	3.3553	3.1988
loglikelihood	*	0	*	*	*	*	*
logloss	0.4989	0.0110	0.5137	0.4919	0.4861	0.4970	0.5059
max_per_class_error	0.3973	0.0411	0.4291	0.3703	0.4530	0.3744	0.3598
mcc	0.3171	0.0356	0.2783	0.3420	0.3584	0.3242	0.2828
mean_per_class_accuracy	0.6698	0.0156	0.6493	0.6847	0.6822	0.6752	0.6575
mean_per_class_error	0.3302	0.0156	0.3507	0.3153	0.3178	0.3248	0.3425
mse	0.1618	0.0042	0.1677	0.1592	0.1570	0.1608	0.1643
pr_auc	0.5786	0.0170	0.5524	0.5900	0.5837	0.5949	0.5721
precision	0.4626	0.0387	0.4410	0.4738	0.5187	0.4646	0.4150
r2	0.1828	0.0173	0.1558	0.1946	0.1929	0.1957	0.1751
recall	0.6096	0.0508	0.5709	0.6297	0.5470	0.6256	0.6748
rmae	0.4022	0.0052	0.4095	0.3990	0.3963	0.4010	0.4053
specificity	0.7300	0.0629	0.7277	0.7397	0.6174	0.7248	0.6402

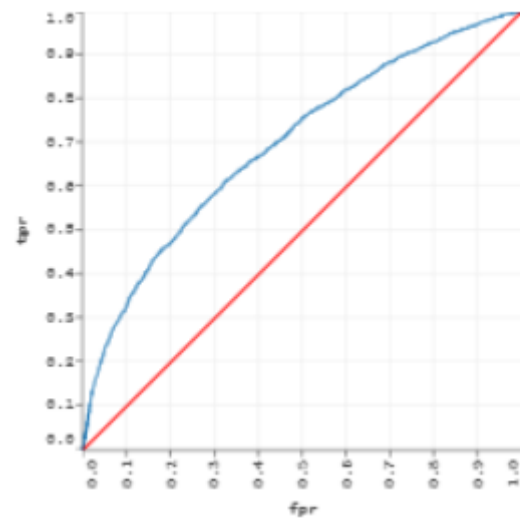
▼ OUTPUT - VARIABLE IMPORTANCES

variable	relative_importance	scaled_importance	percentage
gunAngleDeviation	1749.3524	1.0	0.2379
enemyVelocity	1088.4006	0.6222	0.1480
distance	1049.5148	0.5999	0.1427
enemyHeading	550.0308	0.3144	0.0748
enemyX	455.8361	0.2606	0.0620
enemyY	453.8407	0.2594	0.0617
myGunHeading	405.7349	0.2319	0.0552
time	391.6264	0.2239	0.0533
myX	367.9993	0.2104	0.0500
myY	293.0162	0.1675	0.0398
bearing	244.4987	0.1398	0.0332
myHeading	237.2557	0.1356	0.0323
myVelocity	66.5082	0.0380	0.0090

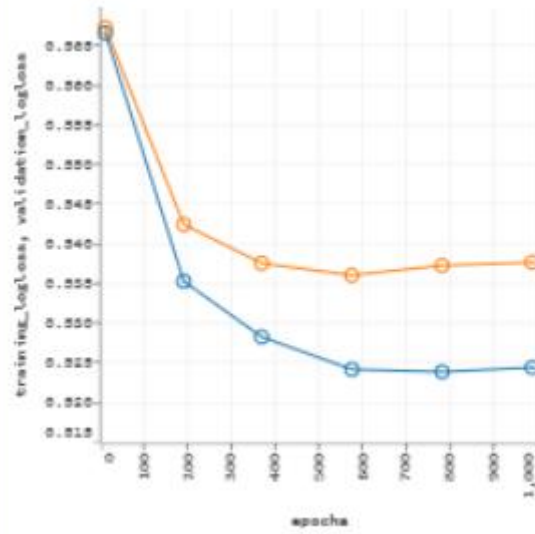
DL:

Model ID: DeepLearning_grid_1_AutoML_1_20240603_204907_model_2

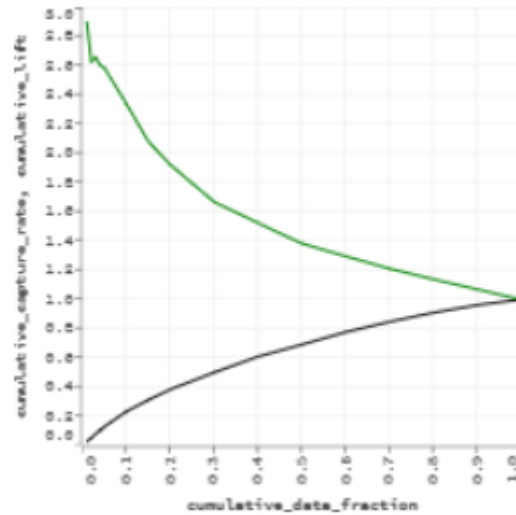
▼ ROC CURVE - VALIDATION METRICS , AUC = 0.698229



▼ SCORING HISTORY - LOGLOSS



▼ VALIDATION METRICS - GAINS/LIFT TABLE



layer	units	type	dropout	l1	l2	mean_rate	rate_rms	momentum	mean_weight	weight_rms	mean_bias	bias_rms
1	13	Input	10.0									
2	50	RectifierDropout	50.0	0	0	0.0124	0.0275	0	-0.1021	1.1380	-4.7720	6.0155
3	2	Softmax		0	0	0.0008	0.0008	0	-0.1456	0.8152	0.3271	0.3722

▼ OUTPUT - VALIDATION_METRICS

<i>model</i>	DeepLearning_grid_1_AutoML_1_20240603_204907_model_2
<i>model_checksum</i>	-7165899146919186945
<i>frame</i>	FP_TESTE
<i>frame_checksum</i>	-7638436041789692866
<i>description</i>	Metrics reported on full validation frame
<i>model_category</i>	Binomial
<i>scoring_time</i>	1717445219554
<i>predictions</i>	.
<i>MSE</i>	0.176525
<i>RMSE</i>	0.420148
<i>nobs</i>	8654
<i>custom_metric_name</i>	.
<i>custom_metric_value</i>	0
<i>r2</i>	0.111388
<i>logloss</i>	0.534078
<i>loglikelihood</i>	NaN
<i>AIC</i>	NaN
<i>AUC</i>	0.698229
<i>pr_auc</i>	0.492295
<i>Gini</i>	0.396459
<i>mean_per_class_error</i>	0.355974

<i>model</i>	DeepLearning_grid_1_AutoML_1_20240603_204907_model_2
<i>model_checksum</i>	-7165899146919186945
<i>frame</i>	AutoML_1_20240603_204907_training_FP_TREINO
<i>frame_checksum</i>	-4221187600069418558
<i>description</i>	5-fold cross-validation on training data (Metrics computed for combined holdout predictions)
<i>model_category</i>	Binomial
<i>scoring_time</i>	1717445219613
<i>predictions</i>	.
<i>MSE</i>	0.174828
<i>RMSE</i>	0.418124
<i>nobs</i>	20111
<i>custom_metric_name</i>	.
<i>custom_metric_value</i>	0
<i>r2</i>	0.117082
<i>logloss</i>	0.529835
<i>loglikelihood</i>	NaN
<i>AIC</i>	NaN
<i>AUC</i>	0.709866
<i>pr_auc</i>	0.519503
<i>Gini</i>	0.419733
<i>mean_per_class_error</i>	0.346643

	mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid
accuracy	0.6610	0.0537	0.5899	0.7300	0.6937	0.6551	0.6365
aic	.	0
auc	0.7105	0.0110	0.6995	0.7168	0.7227	0.7154	0.6984
err	0.3390	0.0537	0.4101	0.2700	0.3063	0.3449	0.3635
err_count	1363.4000	216.0065	1650.0	1086.0	1232.0	1387.0	1462.0
f0point5	0.4535	0.0353	0.4132	0.5051	0.4666	0.4519	0.4308
f1	0.5026	0.0098	0.4939	0.5095	0.5092	0.5104	0.4902
f2	0.5686	0.0367	0.6136	0.5139	0.5603	0.5863	0.5688
lift_top_group	3.1573	0.1462	2.9437	3.3269	3.2269	3.0904	3.1988
loglikelihood	.	0
logloss	0.5298	0.0079	0.5332	0.5288	0.5177	0.5303	0.5392
max_per_class_error	0.4118	0.0595	0.4636	0.4630	0.3994	0.3492	0.3636
mcc	0.2776	0.0357	0.2398	0.3233	0.3013	0.2779	0.2457
mean_per_class_accuracy	0.6503	0.0142	0.6341	0.6631	0.6639	0.6538	0.6366
mean_per_class_error	0.3497	0.0142	0.3659	0.3369	0.3361	0.3462	0.3634
mse	0.1748	0.0027	0.1764	0.1740	0.1707	0.1750	0.1780
pr_auc	0.5208	0.0107	0.5078	0.5328	0.5228	0.5287	0.5117
precision	0.4270	0.0492	0.3727	0.5022	0.4419	0.4199	0.3985
r2	0.1170	0.0077	0.1121	0.1196	0.1226	0.1246	0.1062
recall	0.6274	0.0782	0.7318	0.5170	0.6006	0.6508	0.6368
rmse	0.4181	0.0033	0.4200	0.4172	0.4132	0.4183	0.4219
specificity	0.6732	0.1022	0.5364	0.8093	0.7272	0.6568	0.6364

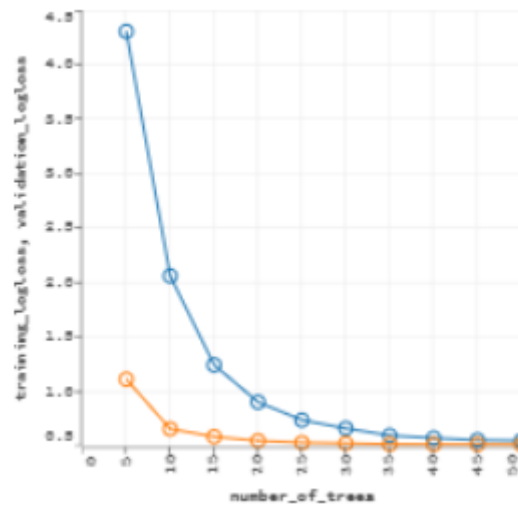
▼ OUTPUT - VARIABLE IMPORTANCES

variable	relative_importance	scaled_importance	percentage
distance	1.0	1.0	0.1715
enemyY	0.8369	0.8369	0.1435
enemyVelocity	0.7787	0.7787	0.1335
gunAngleDeviation	0.7739	0.7739	0.1327
enemyX	0.6594	0.6594	0.1131
enemyHeading	0.3564	0.3564	0.0611
myX	0.2566	0.2566	0.0440
myGunHeading	0.2552	0.2552	0.0438
time	0.2282	0.2282	0.0391
myY	0.1970	0.1970	0.0338
myHeading	0.1782	0.1782	0.0306
bearing	0.1772	0.1772	0.0304
myVelocity	0.1334	0.1334	0.0229

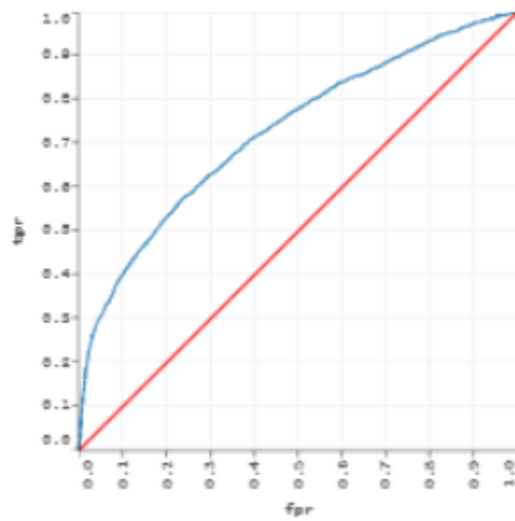
DRF:

Model ID: DRF_1_AutoML_1_20240603_204907

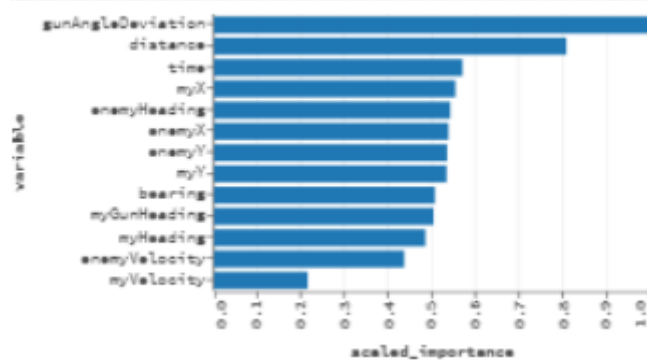
▼ SCORING HISTORY - LOGLOSS



▼ ROC CURVE - VALIDATION METRICS , AUC = 0.727551



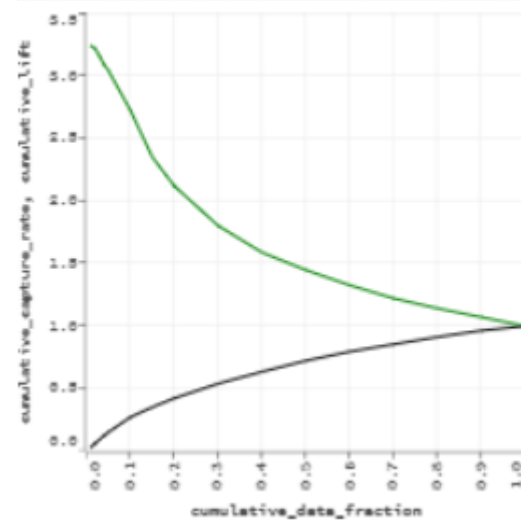
▼ VARIABLE IMPORTANCES



▼ VALIDATION METRICS - CONFUSION MATRIX ROW LABELS: ACTUAL CLASS; COLUMN LABELS: PREDICTED CLASS

	0	1	Error	Rate	Precision
0	4818	1478	0.2351	1.478 / 6.288	0.83
1	1011	1355	0.4273	1.011 / 2.366	0.48
Total	5821	2833	0.2876	2.489 / 8.654	
Recall	0.76	0.57			

▼ VALIDATION METRICS - GAINS/LIFT TABLE



▼ OUTPUT - VALIDATION_METRICS

```

model DRF_1_AutoML_1_20240603_204907
model_checksum 167409450751513610
frame FP_TESTE
frame_checksum -7638436041789692866
description Validation metrics
model_category Binomial
scoring_time 1717444203816
predictions *
MSE 0.165626
RMSE 0.406972
nobs 8654
custom_metric_name *
custom_metric_value 0
r2 0.166251
logloss 0.516470
loglikelihood NaN
AIC NaN
AUC 0.727551
pr_auc 0.563632
Gini 0.455102
mean_per_class_error 0.331177

```

▼ OUTPUT - CROSS-VALIDATION METRICS SUMMARY

	mean	sd	cv_1_valid	cv_2_valid	cv_3_valid	cv_4_valid	cv_5_valid
accuracy	0.6630	0.0472	0.6152	0.6447	0.7297	0.6932	0.6323
aic	.	0
auc	0.7264	0.0113	0.7139	0.7334	0.7343	0.7361	0.7140
err	0.3370	0.0472	0.3848	0.3553	0.2703	0.3068	0.3677
err_count	1355.4000	190.1087	1548.0	1429.0	1087.0	1234.0	1479.0
f0point5	0.4612	0.0317	0.4276	0.4523	0.5022	0.4858	0.4383
f1	0.5183	0.0124	0.5029	0.5282	0.5209	0.5312	0.5081
f2	0.5953	0.0350	0.6104	0.6347	0.5411	0.5859	0.6045
lift_top_group	3.3384	0.2381	3.2113	3.5966	3.5957	3.1787	3.1100
loglikelihood	.	0
logloss	0.5087	0.0065	0.5139	0.5013	0.5020	0.5118	0.5145
max_per_class_error	0.4030	0.0294	0.4211	0.3883	0.4445	0.3708	0.3903
mcc	0.2981	0.0325	0.2591	0.3070	0.3350	0.3198	0.2697
mean_per_class_accuracy	0.6632	0.0139	0.6453	0.6725	0.6739	0.6734	0.6508
mean_per_class_error	0.3368	0.0139	0.3547	0.3275	0.3261	0.3266	0.3492
mse	0.1644	0.0033	0.1680	0.1629	0.1598	0.1639	0.1672
pr_auc	0.5696	0.0127	0.5506	0.5782	0.5776	0.5792	0.5621
precision	0.4306	0.0428	0.3888	0.4128	0.4905	0.4596	0.4015
r2	0.1699	0.0117	0.1544	0.1758	0.1787	0.1801	0.1603
recall	0.6643	0.0722	0.7118	0.7333	0.5555	0.6292	0.6920
rmse	0.4054	0.0041	0.4099	0.4036	0.3997	0.4049	0.4089
specificity	0.6621	0.0898	0.5789	0.6117	0.7924	0.7176	0.6097

▼ OUTPUT - VARIABLE IMPORTANCES

variable	relative_importance	scaled_importance	percentage
gunAngleDeviation	14204.2783	1.0	0.1387
distance	11455.3545	0.8065	0.1118
time	8077.1973	0.5686	0.0789
myX	7848.4658	0.5525	0.0766
enemyHeading	7678.0840	0.5405	0.0750
enemyX	7619.0684	0.5364	0.0744
enemyY	7577.2510	0.5334	0.0740
myY	7560.9663	0.5323	0.0738
bearing	7177.9141	0.5053	0.0701
myGunHeading	7135.1182	0.5023	0.0697
myHeading	6871.8252	0.4838	0.0671
enemyVelocity	6181.9492	0.4352	0.0604
myVelocity	3042.3528	0.2142	0.0297

Parte 2

Localização:

Para o problema da Localização:

Métricas e Confusion Matrix

GBM:

▼ CROSS VALIDATION METRICS - CONFUSION MATRIX ROW LABELS: ACTUAL CLASS; COLUMN LABELS: PREDICTED CLASS

	0	1	Error	Rate	Precision
0	5936	3417	0.3653	3 417 / 9 353	0.83
1	1245	4191	0.2290	1 245 / 5 436	0.55
Total	7181	7608	0.3152	4 662 / 14 789	
Recall	0.63	0.77			

▼ OUTPUT - VALIDATION_METRICS

model	GBM_grid_1_AutoML_1_20240603_204925_model_35
model_checksum	-5426342026583720257
frame	LP_TESTE
frame_checksum	7755677054496332849
description	Validation metrics
model_category	Binomial
scoring_time	1717444420596
predictions	•
MSE	0.184113
RMSE	0.429083
nobs	6445
custom_metric_name	•
custom_metric_value	0
r2	0.215374
logloss	0.545796
loglikelihood	NaN
AIC	NaN
AUC	0.773653
pr_auc	0.676916
Gini	0.547306
mean_per_class_error	0.297170

DL:

▼ PREDICTION - CONFUSION MATRIX ROW LABELS: ACTUAL CLASS; COLUMN LABELS: PREDICTED CLASS

0	4896	4480	0.4778	4.480 / 9.376	0.81
1	1145	4301	0.2102	1.145 / 5.446	0.49
Total	6041	8781	0.3795	5.625 / 14.822	
Recall	0.52	0.79			

▼ OUTPUT - VALIDATION METRICS

model	DeepLearning_grid_1_AutoML_1_20240604_172929_model_6
model_checksum	7987279933031550892
frame	TESTE_LP
frame_checksum	-124323367477842974
description	Metrics reported on full validation frame
model_category	Binomial
scoring_time	1717522009868
predictions	*
MSE	0.205945
RMSE	0.453812
nobs	14822
custom_metric_name	*
custom_metric_value	0
r2	0.113927
logloss	0.609037
loglikelihood	NaN
AIC	NaN
AUC	0.723188
pr_auc	0.578829
Gini	0.446375
mean_per_class_error	0.344031

DRF:

▼ CROSS VALIDATION METRICS - CONFUSION MATRIX ROW LABELS: ACTUAL CLASS; COLUMN LABELS: PREDICTED CLASS

	0	1	Error	Rate	Precision
0	5393	3960	0.4234	3 960 / 9 353	0.84
1	1021	4415	0.1878	1 021 / 5 436	0.53
Total	6414	8375	0.3368	4 981 / 14 789	
Recall	0.58	0.81			

▼ OUTPUT - VALIDATION_METRICS

<i>model</i>	DRF_1_AutoML_1_20240603_204925
<i>model_checksum</i>	6832398984850153167
<i>frame</i>	LP_TESTE
<i>frame_checksum</i>	7755677054496332849
<i>description</i>	Validation metrics
<i>model_category</i>	Binomial
<i>scoring_time</i>	1717444191057
<i>predictions</i>	•
<i>MSE</i>	0.185084
<i>RMSE</i>	0.430213
<i>nobs</i>	6445
<i>custom_metric_name</i>	•
<i>custom_metric_value</i>	0
<i>r2</i>	0.211236
<i>logloss</i>	0.547829
<i>loglikelihood</i>	NaN
<i>AIC</i>	NaN
<i>AUC</i>	0.773203
<i>pr_auc</i>	0.673130
<i>Gini</i>	0.546405
<i>mean_per_class_error</i>	0.310242

```
PathSolution.java ×
public static void readPathConfigs(File fileObject) {
    //RobocodeFileOutputStream fw = new RobocodeFileOutputStream(fileObject.getAbsolutePath());
    //fw.write(new byte[10]);

    //Temp default values
    pathConf = new PathConfiguration(
        minNumberOfPointsPerPath: 1,
        maxNumberOfPointsPerPath: 5,
        numberOfSolutionsPerGeneration: 500,
        xNumberOfBestSolutionsForSelectionFunction: 125,
        xPopulationSizeGoalForMutationFunction: 500,
        yNumberOfEliteNonMutatedSolutions: 5,
        chanceOfSolutionMutation: 0.60,
        chanceOfGeneMutation: 0.20,
        minPixelGeneMutation: 10,
        maxPixelGeneMutation: 30,
        xPopulationSizeGoalForCrossoverFunction: -1,
        yNumberOfEliteNonCrossedSolutions: -1,
        maxNumberOfGenerations: 20,
        maxRunEvolutionDuration: 500,
        useWeightsInDistance: true,
        useStrickArenaBoundsValidation: true
    );
}
```