Tech Challenge - Fase 2

Matheus Muzy Talpe

Rm: 356507

Introdução

Este documento foi criado como entregável do projeto “Tech Challenge – Fase 02” do curso de Pós-graduação, IA para Devs ministrado pela instituição FIAP.

O mesmo descreve o problema selecionado para resolução, objetivos, critérios de sucesso e detalhes de implementação do algoritmo.

[Descrição 3](#_Toc292203047)

[Objetivos 4](#_Toc2024556879)

[Critérios de Sucesso 4](#_Toc956659890)

[Resultados iniciais 4](#_Toc914479949)

[Execução do algoritmo: 5](#_Toc1482285810)

[Resultado final 5](#_Toc1916234594)

[Implementação do algoritmo genético 7](#_Toc1938531475)

[Fluxo de Execução: 7](#_Toc934613847)

[Parâmetros: 7](#_Toc358819335)

[Comportamento esperado do algoritmo: 8](#_Toc761335183)

[Passo 1: 8](#_Toc588902152)

[Passo 2: 9](#_Toc624610420)

[Passo 3: 9](#_Toc1967152791)

[Passo 4: 9](#_Toc1580102161)

[Passo 5: 10](#_Toc1821708775)

[Passo 6: 10](#_Toc242379819)

[Passo 7: 10](#_Toc824063758)

[Passo 8: 11](#_Toc340146932)

[Repositório 11](#_Toc420305725)

[Vídeo 11](#_Toc172037189)

Definição do Problema

# Descrição

Nesta seção, iremos apresentar o problema proposto para a solução do desafio.

Para tal, iremos utilizar os resultados obtidos na resolução do Tech Challenge da fase 1. Onde, selecionamos o melhor algoritmo de machine learning para realizar a predição de gastos médicos baseado em características de um público contido em uma base.

Para a seleção do melhor candidato, realizamos a validação cruzada entre diversos algoritmos de machine learning. O algoritmo selecionado foi o Random Forest, utilizando uma base normalizada sem outliers. Já que o mesmo, apresentou a maior pontuação R2 (R-Squared).

Porém, este resultado não poderia ser otimizado a fim de obtermos uma precisão ainda maior?

Este é o problema proposto para este novo desafio, resolver um problema de otimização de parâmetros em um algoritmo de machine learning.

# Objetivos

Utilizar técnicas de otimização genética, com o objetivo de encontramos os melhores hiper parâmetros para o algoritmo de Random Forest. Para que possamos obter predições ainda mais precisas.

# Critérios de Sucesso

Adotamos a pontuação R2 como medida de sucesso para otimização do algoritmo através de otimização genética.

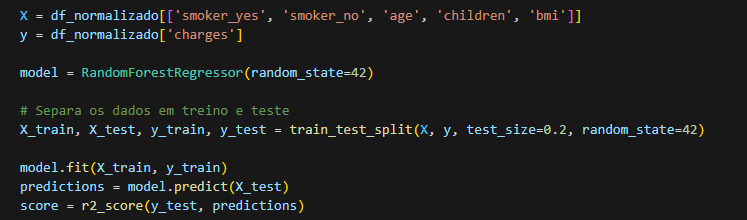
Se, utilizando destas técnicas, obtivermos um resultado melhor do que uma execução sem a preocupação de estipular os melhores hiper parâmetros. Significa que obtivemos sucesso.

Testes e Resultados

# Resultados iniciais

Abaixo, seguem os resultados da execução do algoritmo de Random Forest sem calibração dos hiper parâmetros, utilizando uma base normalizada de custos médicos.

### Execução do algoritmo:

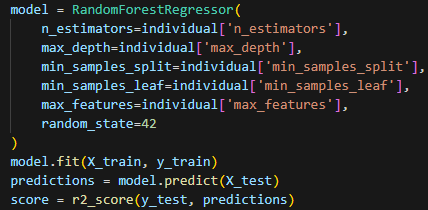
Na imagem acima, podemos observar o algoritmo que será executado sem a definição de hiper parâmetros.



Como resultado obtivemos um score R2 de 0.83046.

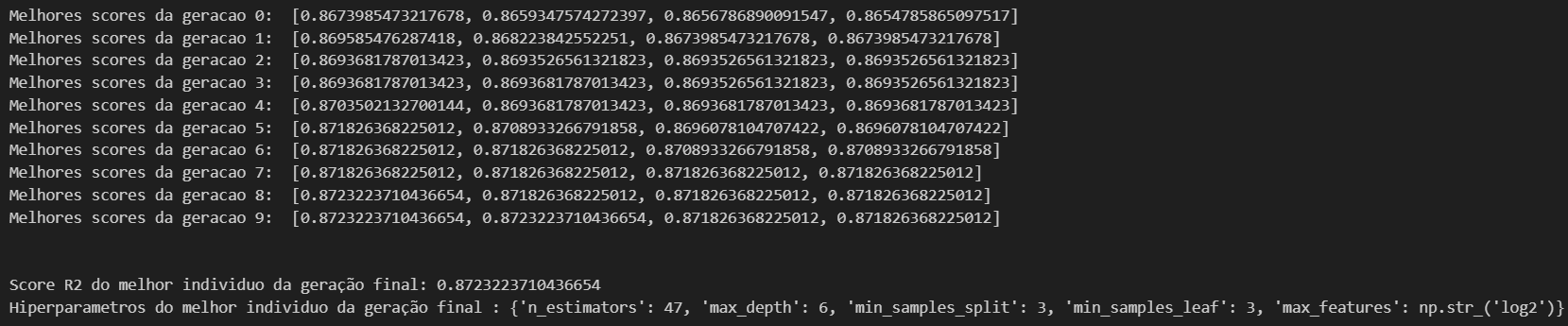
# Resultado final

Abaixo, iremos apresentar os resultados da execução com a utilização de algoritmo genético para calibração dos hiper parâmetros. A descrição mais detalhada das técnicas utilizadas, se encontra na seção de Documentação.



Na imagem acima, podemos observar que a execução do algoritmo com otimização genética se utilizou dos hiper parâmetros apresentados.

Abaixo, podemos observar o resultado das execuções. Onde nosso algoritmo de otimização selecionou os melhores indivíduos de cada geração até chegar no resultado de um score R2 de 0.87232. Onde podemos melhorar uma sensível melhora na qualidade das predições:



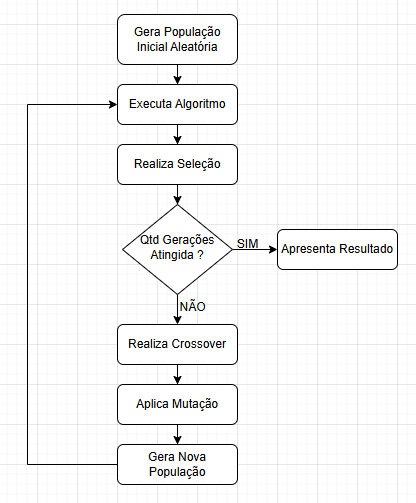
Documentação

# Implementação do algoritmo genético

Implementamos o nosso algoritmo de otimização genética através da função “random\_forest\_genetic\_algorithm”.

Abaixo, segue o descritivo dos parâmetros que a função recebe para direcionar a sua execução, permitindo calibração até mesmo do nosso algoritmo de otimização genética.

### Fluxo de Execução:



### Parâmetros:

* **X**: variáveis utilizadas para predição
* **Y**: variável target
* **generations**: quantidade de gerações que terão o algoritmo executado
* **population\_size**: quantidade de indivíduos que irão compor a população
* **elitism**: quantidade dos indivíduos com melhores resultados, que serão selecionados para criação da nova geração
* **percent\_childs**: percentual da população de cada nova geração, que será populado por filhos dos melhores indivíduos da geração anterior

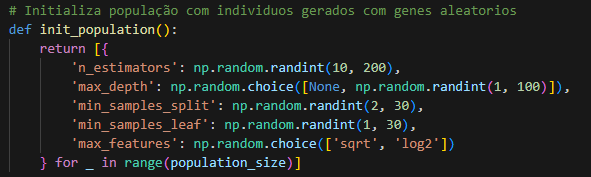
Em seguida, iremos descrever o comportamento esperado do algoritmo, a fim de esclarecer a abordagem utilizada para composição do mesmo.

### Comportamento esperado do algoritmo:

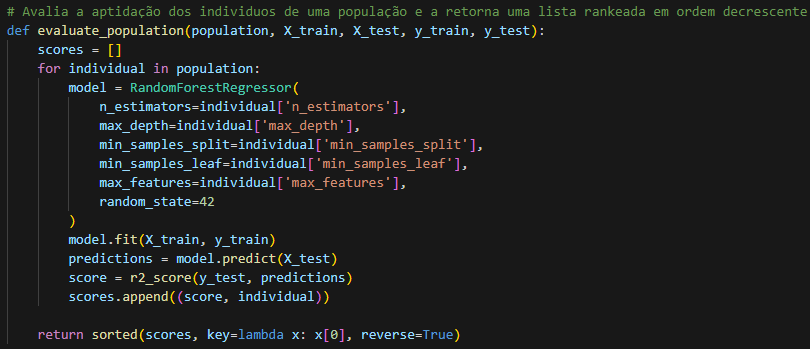
1. Gerar uma população aleatória inicial, com o tamanho determinado pelo parâmetro **population\_size**. Assim poderemos ter uma população diversificada.
2. Executar o algoritmo de Random Forest para cada indivíduo que compõe a população.
3. Selecionar os indivíduos com as maiores pontuações R2. De acordo com a quantidade especificada pelo parâmetro **elitism**. Garantindo que as melhores características sejam preservadas nas novas gerações.
4. Realizar o crossover, utilizando o método de Crossover Uniforme, onde cada gene dos pais, tem a mesma possibilidade (50%) de serem reproduzidos no filho.
5. Aplica mutação utilizando a técnica de “Random Replacement” onde há uma probabilidade de 30% de um gene aleatório (parâmetro) ter seu valor substituído aleatoriamente. Assim iremos reduzir a probabilidade de convergência de resultados logo no início da execução.
6. Gera uma nova população, com o percentual definido pelo parâmetro **percent\_childs** de membros que serão filhos dos melhores indivíduos da geração anterior, e os demais gerados aleatoriamente. Garantindo diversidade nas novas populações.
7. Repete os passos 2, 3, 4, 5 e 6, até que a quantidade de gerações definida pelo parâmetro **generations** seja atingida.
8. Exibe o resultado, selecionando o melhor individuo da geração final, e exibindo seu score R2 e hiper parâmetros.

Abaixo, segue o trecho de código equivalente a cada passo:

### Passo 1:

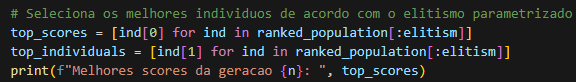
Na imagem acima, podemos observar a função que inicializa uma população com tamanho definido pelo parâmetro **population\_size.** Note que podemos observar cada hiper parâmetro (gene) utilizado para execução do algoritmo de Random Forest.

### Passo 2:

Esta função recebe a população que irá ser avaliada e os dados já separados em treino e teste. Através do laço for cada gene/hiper parâmetro do indivíduo será atribuído ao algoritmo de Random Forest. O modelo será treinado, executado e terá seu score R2 armazenado.

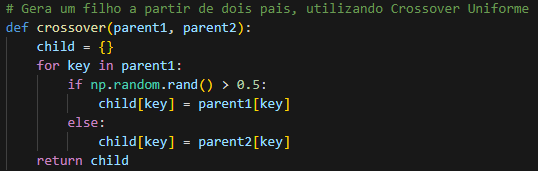
Por fim, será retornada uma lista ordenada decrescente, do melhor para o pior indivíduo.

### Passo 3:



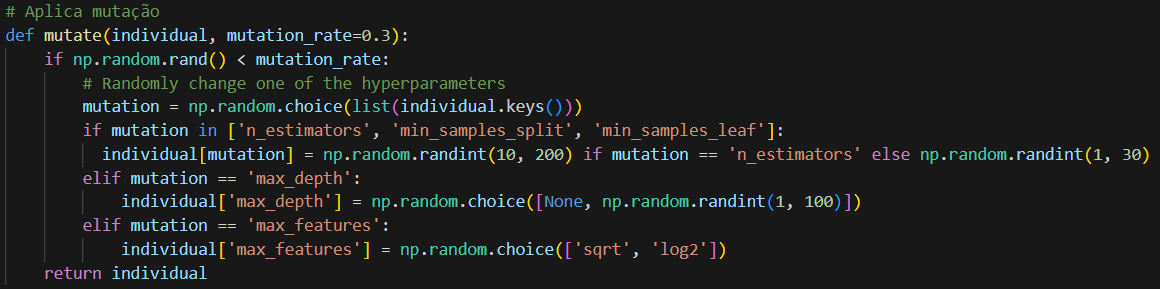
Neste trecho, após a execução de cada geração. Selecionamos os melhores indivíduos (que possuem os maiores scores R2) limitando a quantidade pelo parâmetro **elitism**. E os melhores scores da geração são exibidos no terminal.

### Passo 4:



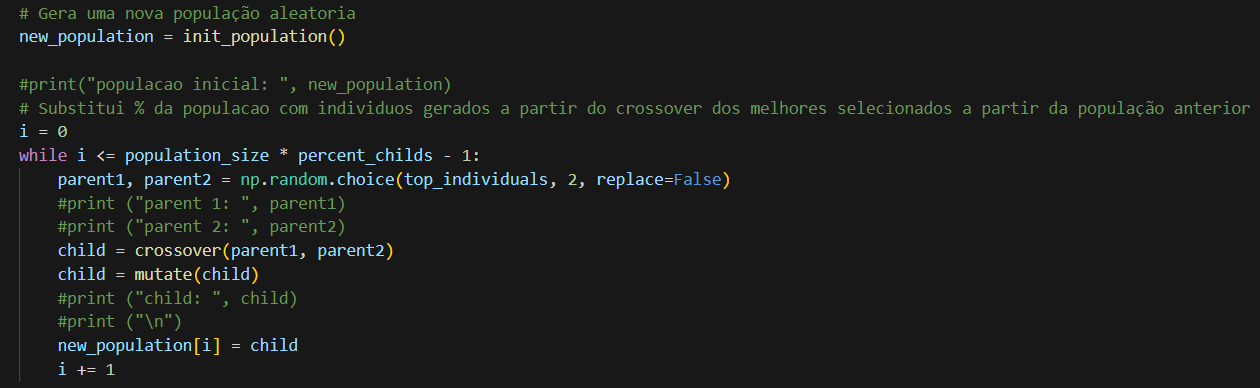
Esta função realiza os procedimentos de crossover. Onde, um filho é gerado a partir de dois pais. Utilizando a técnica de crossover uniforme, onde cada gene tem probabilidade igual de ser originado de um pai ou outro.

### Passo 5:

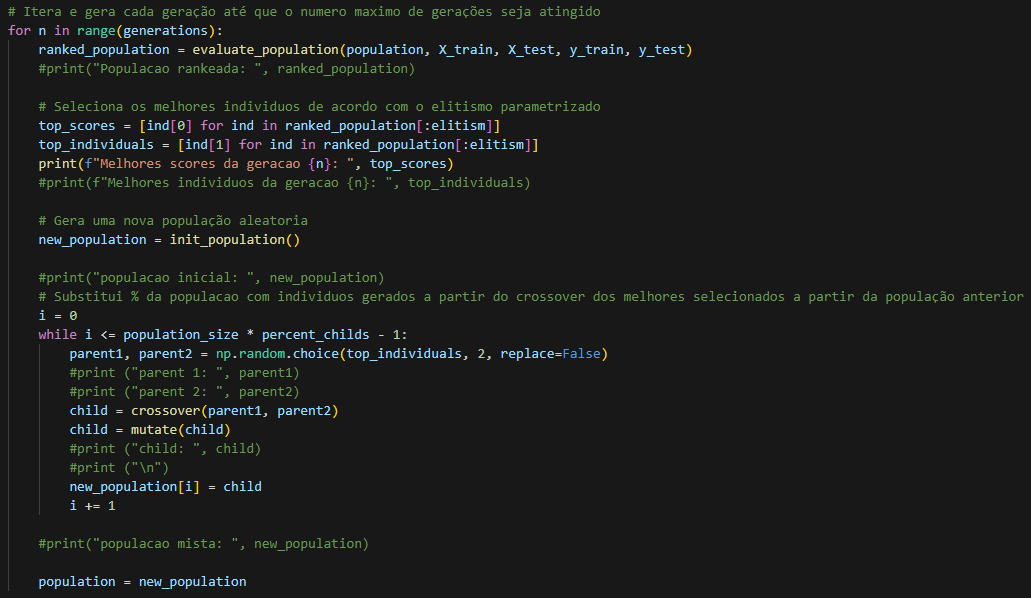
Esta função implementa a mutação de indivíduos, aplicando a técnica de Random Replacement com uma probabilidade de 30% de mutação.

### Passo 6:

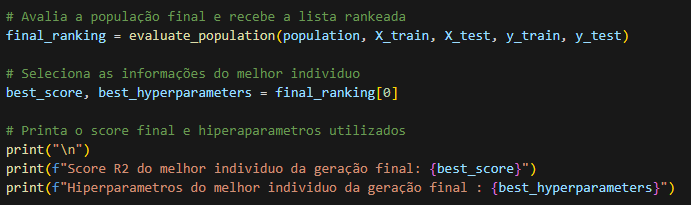
Neste trecho, é gerada uma nova população aleatória, que terá o % de indivíduos definido pelo parâmetro **percent\_childs**, substituído por filhos dos melhores indivíduos da geração anterior.



### Passo 7:

Este laço representa a execução dos passos anteriores até que o número de gerações pré-definido seja atingido.

### Passo 8:

Nesta seção, podemos observar a seleção do melhor indivíduo da última geração e a exibição de suas informações de score e hiper parâmetros.

# Repositório

Segue link contendo o repositório onde está a implementação em código do projeto:

***https://github.com/MatheusTalpe/TechChallengeFase02Fiap.git***

# Vídeo

Segue link do vídeo explicativo do projeto:

***https://youtu.be/rG7JBCkiGJo***

Conclusão

A partir da análise dos resultados apresentados na seção “Testes e Resultados”, podemos concluir que a otimização via algoritmo genético traz resultados significativos.

Durante a execução do algoritmo genético, pudemos realizar algumas observações:

* A execução do algoritmo genético exige um poder de processamento muito maior. Já que a execução do Random Forest sem otimização genética durou apenas 0.3s, enquanto a otimização dos hiper parâmetros via otimização genética, para apenas 10 gerações, durou 49.2s.
* A otimização via algoritmo genético, trouxe ganhos expressivos. Pois passamos de uma pontuação R2 de ~0.83 para ~0.87.
* Com a abordagem escolhida, surgiram novos parâmetros (elitismo, quantidade de gerações, tamanho de população, possível mutação e etc) que abrem inúmeras possibilidades de calibração dos mesmos e implementação de novos recursos para otimização.
* Como passamos a lidar com um certo índice de aleatoriedade, também notamos que nem sempre os resultados das execuções são iguais. O que indica a presença de resultados ‘sub ótimos’.

Novamente, em nosso segundo Tech Challenge, podemos chegar à conclusão de que não existe solução absoluta que possui somente vantagens para um problema.

A implementação de otimização genética, incorpora esta característica aceitando resultados sub ótimos que possuem seus prós e contras. E cabe avaliação de cada cenário, para definir quais “tradeoffs” estamos dispostos a acatar para obter os resultados desejados.

Por exemplo, há situações em que os gastos financeiros com processamento computacional, podem não compensar os ganhos apresentados pela otimização.

Porém, com a tendência de redução de custos de poder computacional, melhora das bibliotecas disponibilizadas e maior acessibilidade à tecnologia. Tendem à aumentar cada vez mais, a utilização deste tipo de solução para problemas heurísticos, onde não há apenas ‘uma maneira correta’ de solução.