Roteiro para trabalho

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Introdução

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Olá, eu sou o Samuel Rodrigues e faço parte do grupo 16.

Este vídeo é o trabalho feito em grupo de Tech Challenge da fase 1 do curso de pós graduação de Inteligência Artificial para DEVs.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Tech Challenge - descrição

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

De forma resumida, para o Tech Challenge, o grupo deve realizar tarefas de geração de massa de dados para o desafio, análise exploratória dos dados, o pré-processamento dos dados, definir e realizar treinamento do modelo preditivo, com análise e apresentação dos resultados. A idéia é aplicar os conceitos aprendidos durante as aulas.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Tech Challenge - inicio

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Vamos para o nosso projeto!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Tech Challenge - Estrutura da Base de dados

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Utilizamos a mesma estrutura proposta na tech challenge para a base de dados, portanto, a base de dados conterá informações dos custos do indivíduo. Temos as colunas:

- A idade, com informação numérica da idade da pessoa;

- O gênero, com campo alfanumérico informando se é masculino ou feminino;

- O índice de massa corporal (imc), com informação numérica do IMC da pessoa;

- A quantidade de filhos que posssui, lembrando que são valores inteiros;

- O indicativo se é fumante, com valores 'sim' ou 'não';

- A região onde reside no Brasil: norte, nordeste, sudeste, entre outros;

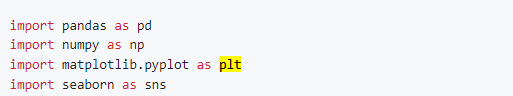
- Os valores de encargos cobrados pelo plano de saúde

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Tech Challenge - Código - Massa de dados

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Nós também realizamos o projeto no Google Colab e, para iniciarmos a criação da massa de dados, utilizamos a linguagem Python e as bibliotecas que nos apoiarão no desenvolvimento:

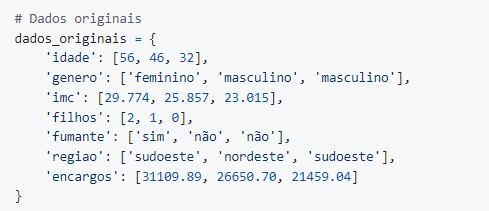


- pandas, para nos apoiar a trabalhar com os dados, através dos DataFrames

- numpy, que possui inúmeras funções matemáticas e é excelente para processamento de grandes volumes de dados

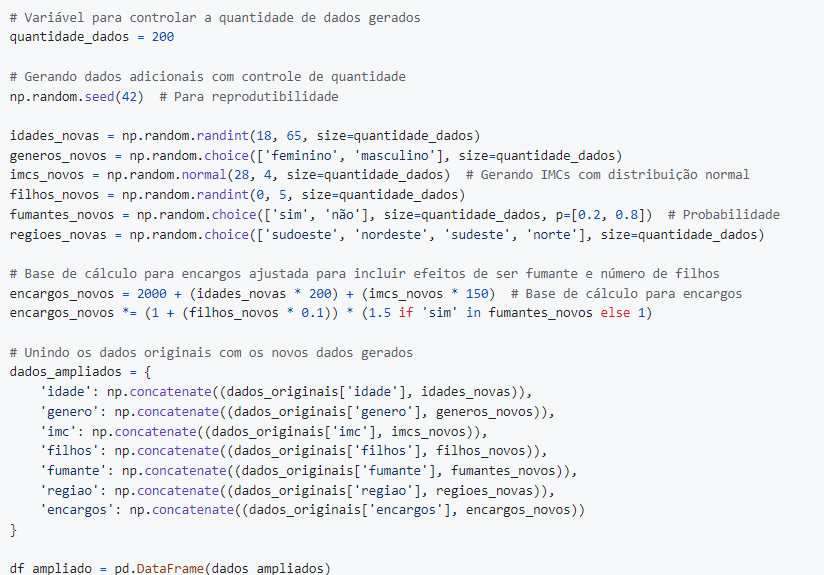
- matplotlib e seaborn, para nos apoiar na geração dos gráficos para as análises visuais

**Agora, vamos ao código**



No arquivo eda.py, iniciamos pegando os dados e estrutura de exemplo da própria Tech Challenge.

Depois, precisávamos gerar mais massa de dados, para poder ter uma população maior que nos permitisse utilizar neste trabalho.



Criamos então a variável 'quantidade\_dados', para nos apoiar e facilitar a definir essa quantidade adicional de dados para trabalhar posteriormente, e vamos umas o 'random' do numpy, para a geração de forma aleatória dos novos dados para cada campo.

Como é uma massa de exemplo, definimos no grupo para essa aleatoriedade:

- Idades: decidimos gerar idades entre 18 e 65 anos;

- Gêneros: com valores aceitáveis sendo 'feminino' ou 'masculino';

- IMC's: gerar os valores de imc's com distribuição normal, considerando o centro de distribuição o valor 28, e o desvio padrão com valor 4;

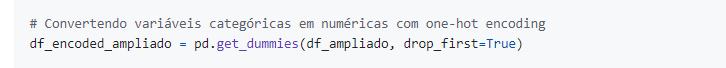
- Filhos: gerar valores entre 0 (ou seja, sem filhos) até 5 filhos;

- Fumantes: com valores aceitáveis sendo 'sim' ou 'não'. Porém, aqui, colocamos "pesos" na geração da aleatoriedade, com probabilidade de 20% para o valor 'sim', e 80% pro valor 'não';

- Regiões: adicionamos mais algumas regiões para gerar de forma aleatória, sendo os valores possíveis: 'sudoeste', 'nordeste', 'sudeste' e 'norte';

- Encargos: para os encargos, para o exemplo, tentamos considerar um cálculo ajustado para ter efeito proposital, principalmente, relacionado ao número de filhos e ser fumante.

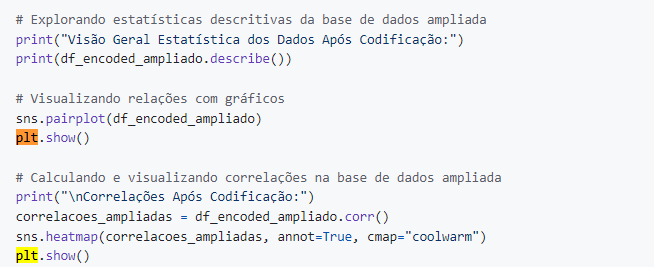
Depois, criamos um Dataframe contendo tanto os dados originais quanto os novos, atribuindo para uma variável chamada 'dados\_ampliados'



Já sabendo que temos variáveis categórias na nossa base de dados, ou seja, campos com dados em forma de texto, precisamos transformar esses dados em numéricos, já que os algoritmos de treinamento dos modelos de ML não trabalham com dados com texto (só números).

decidimos usar a função 'get\_dummies' do próprio pandas, que permite fazer essa transformação das variáveis categórias através de um processo chamado "one-hot". Essa codificação pega as categorias de valores e transforma em colunas. Os valores possíveis dessas colunas seriam 0, caso a característica relacionada não esteja presente, ou 1 caso esteja;

Então, a variável 'df\_encoded\_ampliado' será a variável de DataFrame com os dados transformados



**Análise exploratória inicial:**

O pandas tem uma função bem legal, que é o ‘describe’, e ele já nos traz algumas informações estatíticas com base no dataframe. Então, realizamos o describe da variável 'df\_encoded\_ampliado' e printamos para analisar.

Se perceber, ele nos trouxe nas colunas os campos idade, imc, filhos e encargnos, e na legenda das linhas:

- count: que faz a contagem dos registros para cada campo (ou coluna);

- mean: que calcula o valor da média dos valores numéricos por campo;

- std: que calcula o desvio padrão;

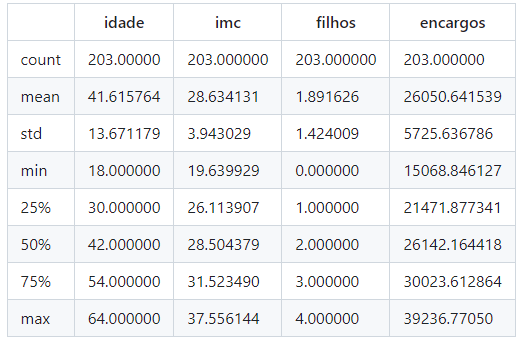
- mín: valor mínimo encontrado;

- 25%: o percentil-25 (P25)

- 50%: o percentil-50 (P50)

- 75%: o percentil-75 (P75)

- max: valor máximo encontrado.



Vamos explicar como podemos analisar os dados:

Geral:

- O Dataframe possui 203 registros, que são os 3 dados originais mais a massa de teste com 200 registros aleatórios que geramos

Sobre a idade vs amostra:

- as idades mínima e máxima foram de 18 e 64 anos

- a média da idade ficou entre 41 e 42 anos;

- o desvio padrão acusou 13.67 pontos. O desvio padrão ele traz o quão disperso estão os dados. Ou seja, quanto mais próximo de 0, mais homogêneos são os dados (valores muito próximos).

- o P25 (percentil 25) foi de 30. Ou seja, se colocarmos todos os valores de idades de forma crescente, indica que 25% dos valores estão abaixo do valor 30 anos. Também indica que, 75% da população está acima dos 30 anos;

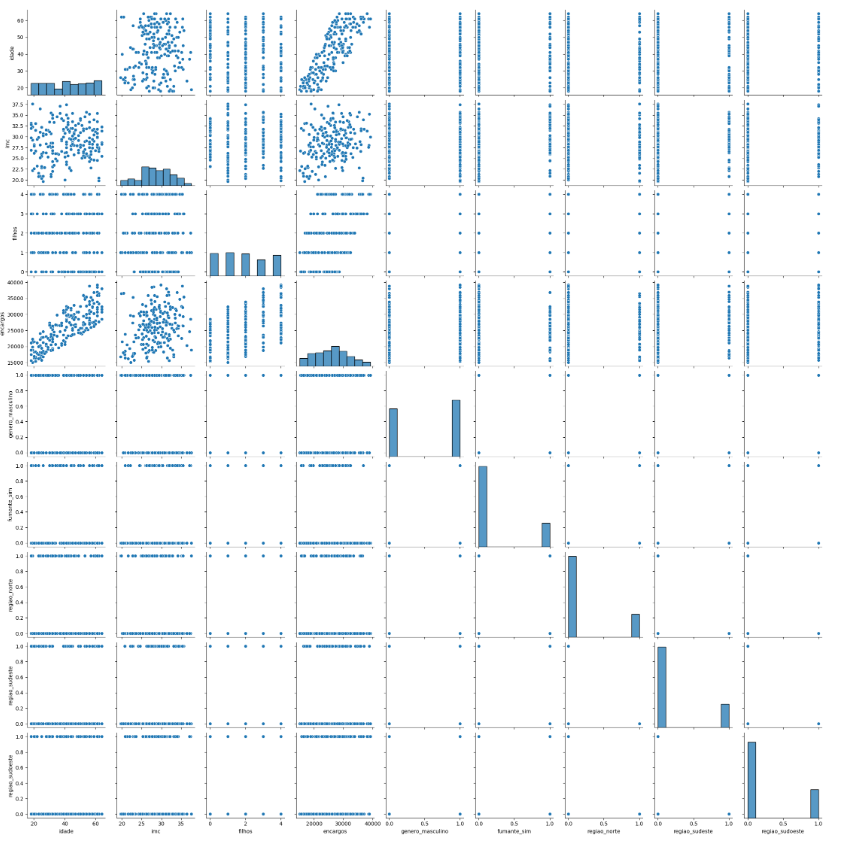
- o P50 (percentil 50) foi de 42. P50, ou mediana, corresponde ao valor que esteja bem no meio, quando a gente ordena todos os valores de idade. Então o valor 42 anos é a mediana;

- o P75 (percentil 75) foi de 54. Ou seja, indica que 75% dos valores estão abaixo do valor 54 anos. Então, 25% da amostra possui mais de 54 anos.

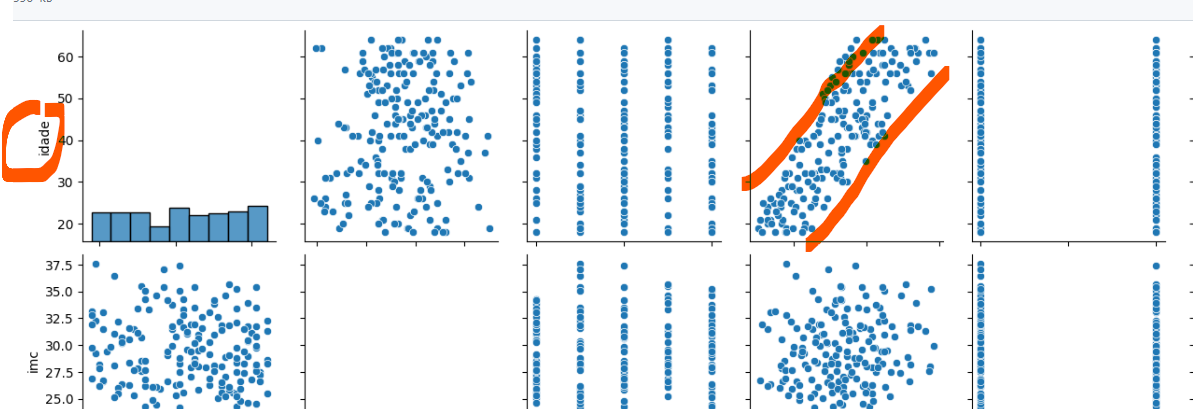
Percebemos que, apesar de trazer somente uma tabela com números, é possível tirar muitos insights dos dados usando o método “describe” do pandas. Podemos analisar as demais colunas, e revisitar essas informações sempre que for interessante para as nossas análises.

Vamos continuar com mais análises sobre os dados.

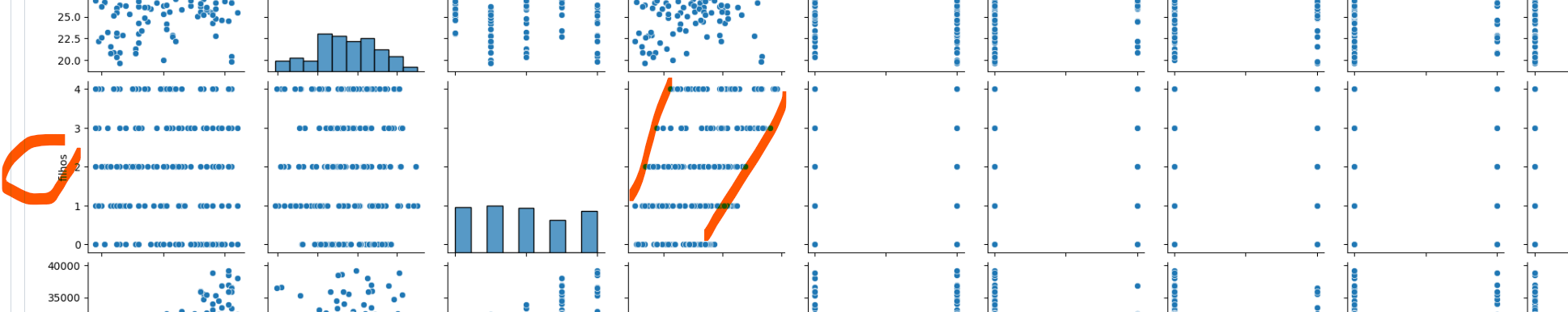
Uma forma interessante de tentar identificar relações entre os dados que aprendemos foi através da visualização de gráficos, um exemplo é a geração de gráficos que relacionam campos.



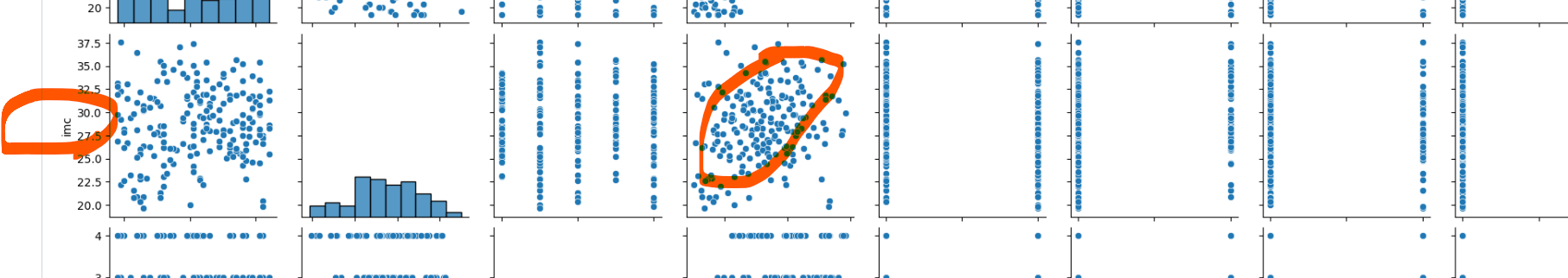
Um exemplo para isso é o uso do ‘pairplot’ do seaborn. Ele é bem simples de usar, pois ele trazer gráficos cruzando os campos. Vamos analisar os gráficos gerados:

Logo de cara, percebemos que o gráfico de relação entre a idade e os encargos, os pontos de valores para a idade vão subindo conforme vai aumentando o valor do encargos

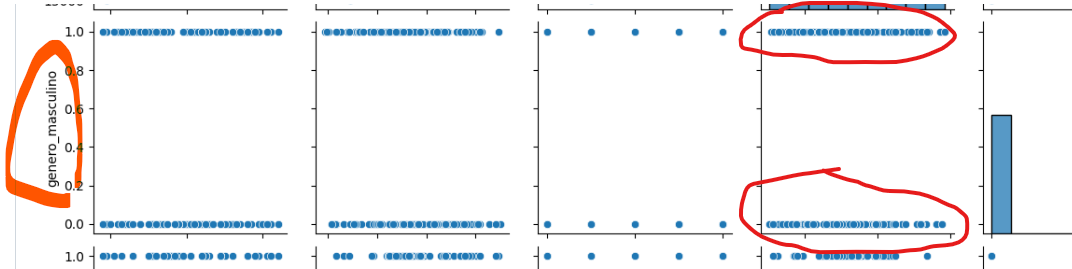
Se olharmos a de filhos vs encargos também.



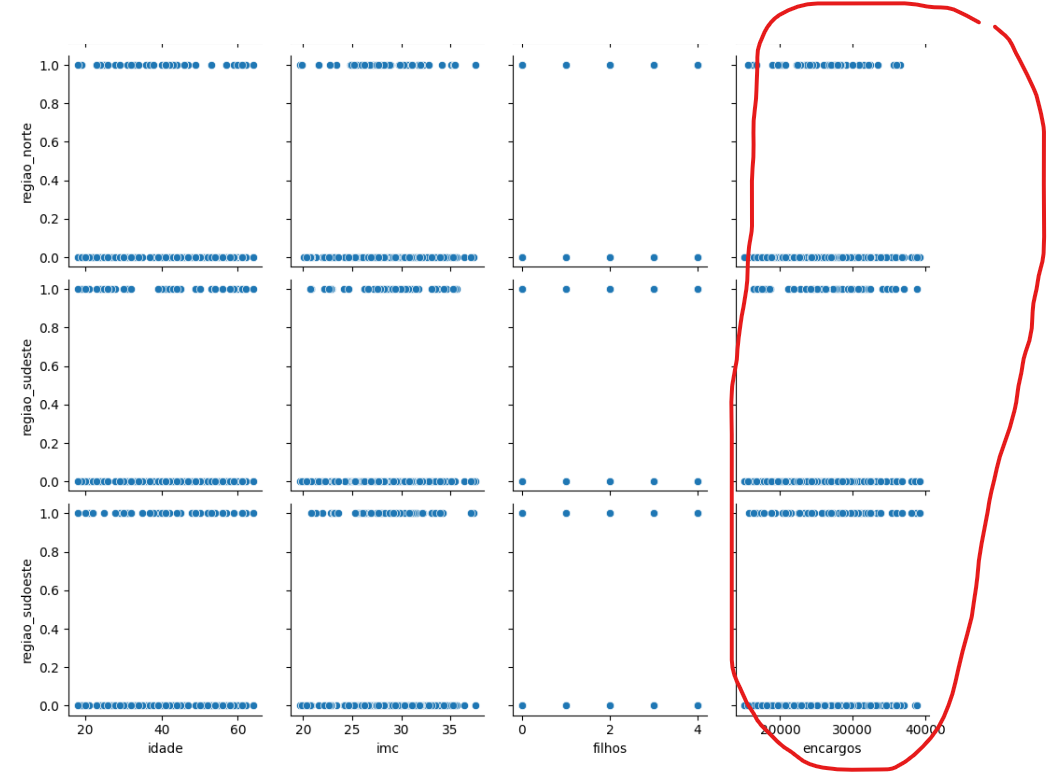
Se olharmos o imc vs encargos, por exemplo, apesar de alguns outliers e ter dados um pouco mais dispersos, ainda aparenta que pode seguir uma certa relação



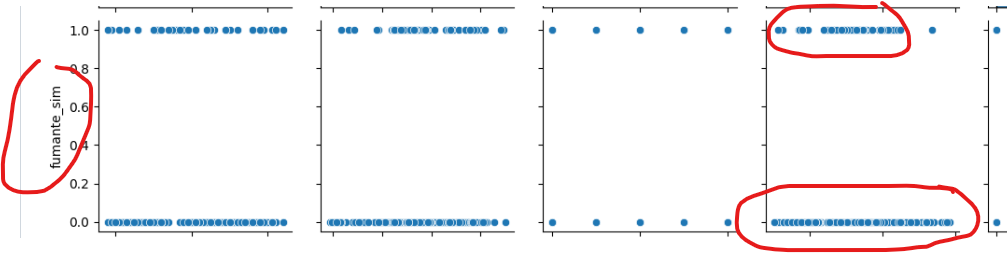
Já os dados de gênero vs encargos parece não ter muita relação,, já que tanto os valores codificados 0 e 1 aparentemente estão distribuídos de forma mais igual independente do valor de encargo.



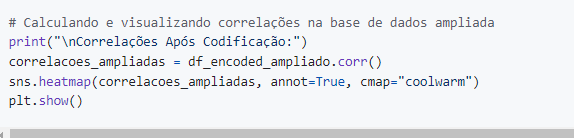
O mesmo coorre para as regiões



Um ponto interessante aqui para o campo de fumante. Lembra que tao tentarmos gerar a massa de dados, para o cálculo dos encargos também definimos um certo ‘peso’. Porém, talvez não tenha sido tão refletido, já que a distribuição dos encargos foi muito próxima, aparentemente com pouca diferença, o que gerou certa dúvida. Talvez para este campo tenhamos que avaliar melhor. Por exemplo, ao analisar o próximo gráfico, talvez ela nos apoie melhor a decidir sobre esse campo



Um outa forma bem rápida e que axilia a visualização de correlações através de gráficos é o gráfico do tipo “heatmap”, que também pode nos apoiar muito nessas relações, ou melhor, correlações entre os dados

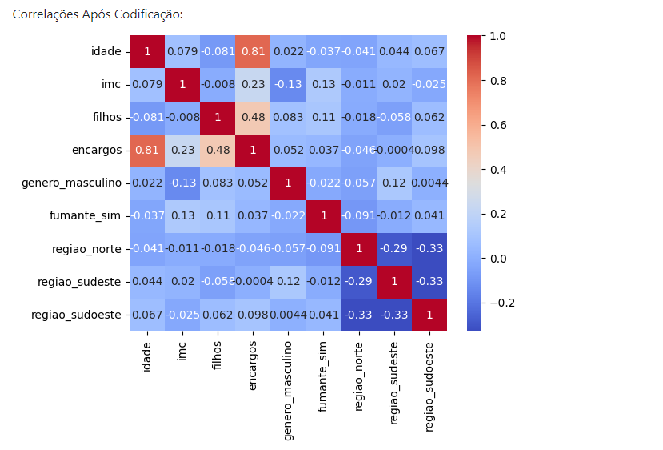


Para isso, primeriro vamos usar o método “corr()”, que faz exatamente a correlação entre as colunas com valores numéricos do DataFrame, atribuindo a resultante para uma variável chamada “correlacoes\_ampliadas” e, em seguida, podemos usar essa variável para montarmos nosso gráfico de mapa de calor, ou heatmap.

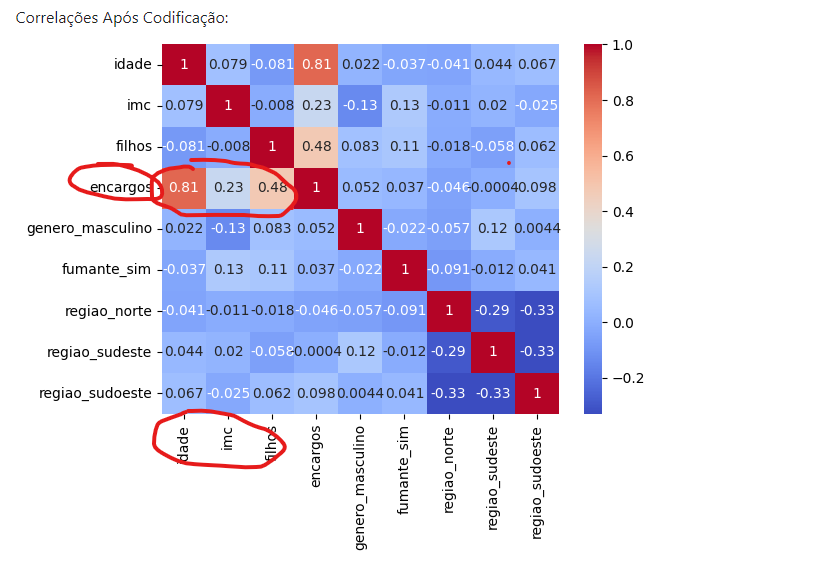
O Seaborn nos auxiliar a gerar esse gráfico de heatmap, onde configuramos par que a escala de cores seja do tipo “coolwam”, ou seja, quanto menor o valor, vai ficando mais da cor azul, e quanto maior o valor, mais vermelho fica a cor.

Para este gráfico e dados de correlação, quando maior o valor, maior a correlação entre os campos, o que facilita identificarmos quando uma coluna possivelmente poderia influenciar em outra(s).

Lembrando que, neste caso, qureemos analisar os campos que possam influenciar nos encargos

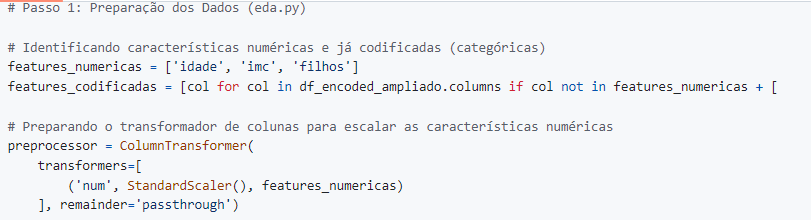


Ao analisar o gráfico, já percebemos uma maior correlação dos campos idade, imc e filhos, com encargos. E o legal aqui é que, ao analisar esse gráfico, percebemos que o campo de fumantes não tem muita correlação. Isso então pode nos apoiar nas etapas seguintes para a preparação dos dados para usar nos modelos.



Acho que já conseguimos fazer uma boa análise exploratória dos dados, então vamos agora para a preparação dos dados! Vamos ao código!

Passo1 – Preparando os dados

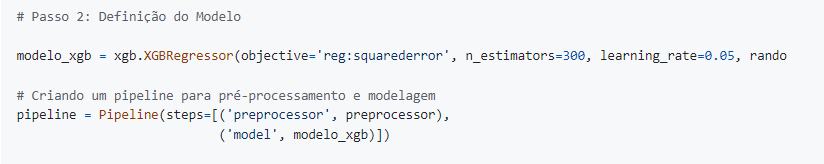


Os algoritmos de ML podem ser mais sensíveis com relação à escala de valores contidos nos campo. Então, vamos precisar “normalizar” a escala dos dados, ou seja, precisamos garantir que os dados que vamos trabalhar estejam em uma escala comum para facilitar quando formos trabalhar com os modelos de ML

Para isso, vamos usar o ColumnTransformer com o StandardScaler do sklearn para isso, de forma a transformar as colunas numéricas em escalas padronizadas com relação à distrbuição normal

Vamos usar bastante aqui algumas bibliotecas, como o ‘sklearn’, que nos apoiará no pré-processamento dos dados e na modelagem da ML. É uma biblioteca opensource que auxilia muito os profissionais a trabalhar com aprendizado de máquina no Python.

Passo 2: Definição do modelo



Vamos usar a biblioteca ‘xgboot’, que também é uma biblioteca opensource para trabalhar com ML de forma otimizada e eficiente. Porém, esta é baseada em árvores de “gradient booting”. De forma simples, ela combina as estimativas de um conjunto de modelos que são mais simples e fracos, ou seja, que geram poucas suposições sobre os dados, para auxiliar na geração de predições mais eficientes.

Definimos que vamos trabalhar com um modelo baseado em Regressão linear, então vamos usar o modelo XGBRegressor para isso.

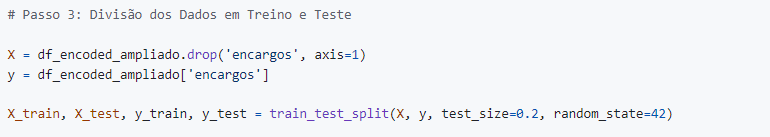
Decidimos que a forma de aprendizagem será do tipo ‘squarederror’, ou melhor, “erro quadrático médio”. Ela serve para calcular o quão próximo os dados estão de uma linha de regressão (ou seja, regressão linear), ajudando a validar com qualidade o modelo de previsão.

Aqui definimos um valor de 300 para a quantidade de estimativas, ou seja, a quantidade de árvores do conjunto.

Também definimos a taxa de aprendizado par 0.05, com a intenção de aplicar um “peso” para as correções de cada iteração (árvore), onde, quanto menor o valor (menor que 1), menor seria essa “taxa” de correções para cada árvore adicionada ao modelo, resultando na necessidade de uma quantidade maior de árvores (iterações/treinos) ao modelo, e evitando assim um “overfit” no treino do modelo, por exemplo.

Para realizar a predição, vamos criar uma pipeline para o pré-processamento e modelagem

Passo 3: dados de treino e teste

Outro passo importante para poder treinar e avaliar o modelo é gerarmos os dados para treinos e testes. Para isso, preciamos realizar os passos:

- Vamos remover somente o campo de encargos e atribuir, , já que é o nosso campo”alvo”, o Dataframe para uma variável ‘X’

- E vamos atribuir para uma variável ‘Y’ somente o Dataframe contendo o nosso campo de encargos

Vamos agora utilizar um método chamado ‘train\_test\_split‘, para ter os dados de treino e teste para X e Y

Definimos o parâmetro ‘test\_size’ para 0.2, que sifnifica que queremos que 20% do dataset informado seja incluso nos testes.

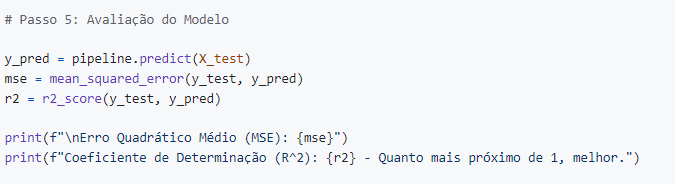
Com isso, agora temos as variáveis X\_train, X\_test, y\_train e y\_test

Passo 4: Treinar o modelo



Feito isso, já podemos treinar o nosso modelo. Para isso, usamos o método “fit” da pipeline que criamos anteriormente, que vai executar o treinamento do modelo.

Passo 5: Avaliação do modelo



Após o treino do modelo, precisamos avaliar se o modelo conseguiria prever, com base em novos dados de teste, de forma mais assertiva, ou melhor, com boa qualidade de predição.

Usando o método ‘predict’ da pipeline, vamos gerar uma variável ‘y\_pred’, que conterá a resultante dessa predição do modelo com base nos dados de testes da variável ‘X\_test’

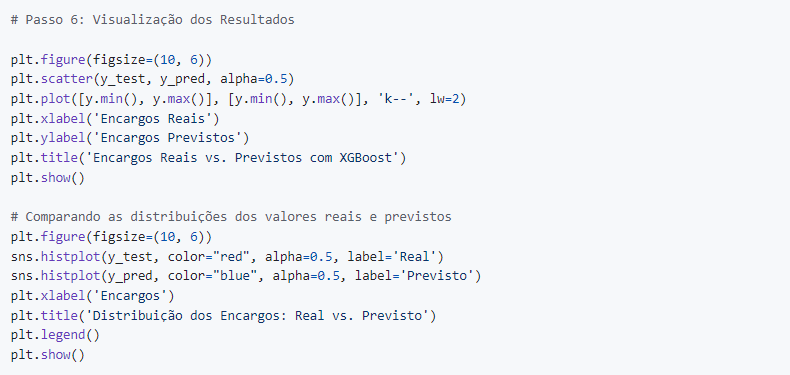
Agora precisamos analisar se a previsão tem qualidade. Para isso, geramos a o erro quadrático médio e também o score, ou melhor, o percentual do coeficiente de determinação.

Quanto mais próximo de 1, o coeficiente de determinação, significa que maior é a previsão que o modelo gerou.

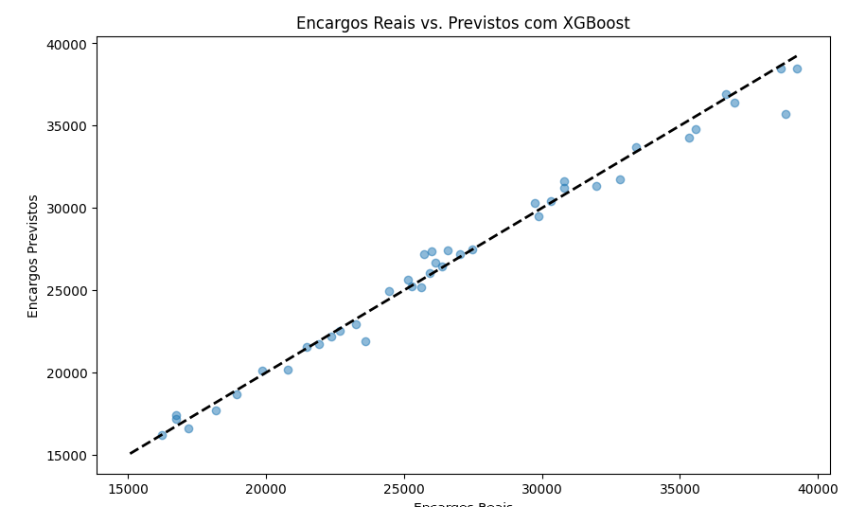
No nosso caso, deu para ver que o valor foi de 0.984, ou seja, foi uma excelente pedição.

Vamos também visualizar os dados através de gráficos, para facilitar ainda mais o entendimento

Passo 6: Visualizando os resultados

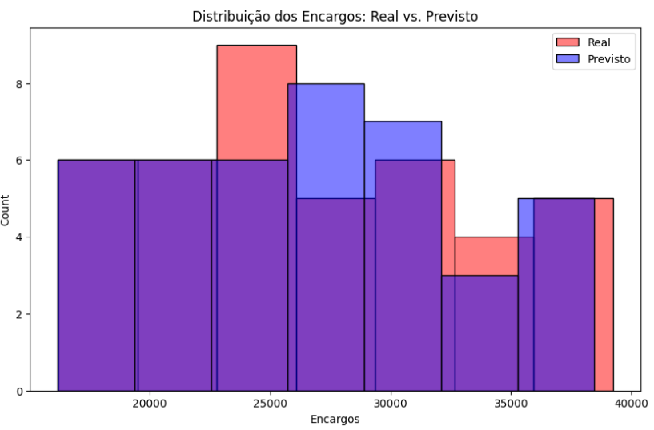


Primeiro vamos gerasr um gráfico que nos mostre a linha de regreção, ou seja, previsão usando o XGBoost, e olhando os dados de encargos reais



Dá para perceber que os valores reais, que são os pontos em azul, estão muito próximos da linha de previsão, mostrando que o modelo está com excelente qualidade de previsão. Só reforçando que o erro quadrático seria, explicando de forma bem simples, essa “distância” entre um ponto com a linha.

Também podemos ver a distribuição em forma de coluna, onde em vermelho seriam os dados reais, e em azul os dados da previsão. Ou seja, quanto mais próximos nas alturas, mais eficaz foi a previsão.



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* Tech Challenge - Conclusão

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Portanto, podemos concluir que, conforme demonstrado nos resultados visuais e coeficiente de determinação, podemos considerar que o modelo XGBoost demonstrou ter sido bastnte eficaz para prever os custos médicos com base nas características fornecidas.

E assim finalizamos o nosso trabalho em grupo dessa Tech Challenge, muito obrigado!