

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**  
**NÚCLEO DE EDUCAÇÃO A DISTÂNCIA**  
**Pós-graduação *Lato Sensu* em Inteligência Artificial e Aprendizado de Máquina**

**Lennon Fernandes Oliveira**

**Predição de Valor Líquido dos Cooperados Filiados a FENCOM**

Contagem  
Junho/2022

**Lennon Fernandes Oliveira**

**Predição de Valor Líquido dos Cooperados Filiados a FENCOM**

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Inteligência Artificial e Aprendizado de Máquina, como requisito parcial à obtenção do título de *Especialista*.

Contagem

Junho/2022

## SUMÁRIO

1. Introdução.....	4
2. Descrição do Problema e da Solução Proposta .....	4
3. Canvas Analítico .....	7
Após o modelo treinado será realizada a predição da competência 06/2022 depois que o modelo for treinado e testado.....	7
Caso a predição esteja fora da margem de erro, considerar incluir outros campos.....	8
4. Coleta de Dados .....	9
5. Processamento/Tratamento de Dados .....	11
6. Análise e Exploração dos Dados .....	12
7. Preparação dos Dados para os Modelos de Aprendizado de Máquina .....	25
8. Aplicação de Modelos de Aprendizado de Máquina .....	32
Abaixo segue o pipeline de todo o processo de implementação, teste, deploy, e monitoramento dos modelos de machine learning para predição dos valores mensais dos cooperados em uma determinada competência.....	39
9. Discussão dos Resultados.....	40
10. Conclusão .....	40
11. Links .....	42

## **1. Introdução**

A FENCOM, Federação Nacional das cooperativas Médicas é uma federação criada em 1994 com objetivo de trazer soluções em vários campos para cooperativas médicas e seus cooperados. As cooperativas médicas utilizam o sistema SASC com intuito de realizar a folha de pagamento dos cooperados. O prontuário médico é digitado no sistema, possuindo uma gama de status como digitação, glosa, faturamento até chegar ao repasse do valor ao médico.

## **2. Descrição do Problema e da Solução Proposta**

A FENCOM possui uma parceria com o banco CREDICOM e surgiu a necessidade de realizar uma previsão do quanto o médico possa receber em um determinado mês. Uma vez que a predição alcance um resultado satisfatório, o sistema poderá ser usado, servindo de base de informação a CREDICOM, para que o banco possa realizar uma espécie de antecipação de crédito para o cooperado.

A solução proposta é a utilização da base de dados do sistema SASC. Quando uma guia médica de consulta ou cirurgia é digitada no software, ela fica com status de digitado. O convênio pode não concordar com o valor que foi informado no procedimento, no ato da digitação, gerando uma glosa parcial ou total. O procedimento fica com status de glosado (Em negociação). Quando não possui nenhuma restrição do convênio quanto ao futuro pagamento, esse procedimento passa para faturado e assim estará disponível para ser repassado ao médico. Também ocorre o caso de o convênio realizar uma glosa e depois informar que vai pagar aquele procedimento. Nesse caso o procedimento passa para o status de faturado. Na maioria das vezes, a negociação para saber se a guia será paga ou não, demora por volta de três meses.

Para realizar a predição, serão utilizados dois algoritmos de Machine Learning. O primeiro algoritmo classificará se o médico receberá ou não algum valor. O segundo algoritmo vai realizar a previsão do valor líquido daqueles cooperados que no algoritmo anterior vão receber algum valor em um determinado mês. Devido a quantidade de registros, as competências de 01/2021 à 05/2022 serão usadas para treinamento e teste do algoritmo. Cada linha do dataset corresponde a procedimentos digitados antes da competência. O campo valor\_predicao\_liquido se

referem as guias que foram repassados na competência, porém não foram digitados naquela mesma competência.

. Importante salientar que um prontuário digitado, não necessariamente será repassado no mesmo mês que foi digitado no sistema SASC.

No ato do repasse, acontecem algumas incidências referente ao valor bruto. É correspondente a um contra cheque, constando o salário do médico e descontos como INSS, imposto de renda, PIS, COFINS e outros impostos. A Figura 1 é um exemplo das incidências que ocorrem no repasse. O alvo deste trabalho é o valor após as incidências. Estamos trabalhando com guias que constam dentro do espelho, fatura ou que o pagamento da fatura foi realizado. Atendimentos digitados normalmente mudam seus status rapidamente para espelho. Vide Figura 2, que constam todas as situações possíveis de uma guia no sistema SASC e seus significados. Atendimentos com status de glosado, demoram muito para serem negociados, portanto, não sabemos quando o convênio vai liberar o pagamento.

**Figura 1 - Demonstrativo de Pagamento**

Demonstrativo de pagamentos		Período:	Emissão em:	Página: 1	
Nome:				Matrícula:	
Num. inscrição INSS:		CPF/CNPJ:			
Data de crédito:	23/05/2022	Número do Repasse:			
<b>Lançamento</b>		<b>Proventos</b>	<b>Descontos</b>	<b>Valor Recebido</b>	
Repasse de Honorários	PMMG/IPSM	R\$ 93,52			
Repasse de Honorários	PREMIUM	R\$ 60,00			
Repasse de Honorários	FBG	R\$ 251,56			
Repasse SUS INCENTIVO	FBG	R\$ 1.469,02			
Repasse SUS INCENTIVO	SUS(SANTA	R\$ 204,57			
Taxa Custeio PIS			R\$ 13,51		
Taxa Custeio/Cofins			R\$ 62,36		
Taxa de Administracao Entidade			R\$ 83,15		
Pis retenção P.Jurídica - art.30º -Lei 10.833/2003			R\$ 13,51		
Cofins retenção P.Jurídica - art. 30º - Lei 10.833			R\$ 62,36		
CSLL retenção P.Jurídica - art. 30º - Lei 10.833/2			R\$ 20,79		
Imposto de Renda Pessoa Jurídica			R\$ 31,18		
Liquido				R\$ 1.791,81	
Total.....		R\$ 2.078,67	R\$ 286,86	R\$ 1.791,81	

Demonstrativo de Pagamento para exemplificar as incidências repasse dentro do sistema SASC.

Fonte: Imagem retirada do relatório demonstrativo de pagamento do sistema SASC.

## Figura 2 -Situações de Guias no contexto do sistema SASC

**Digitado:** atendimento que foi digitado.

**Atendimento Inconsistente:** é aquele atendimento/procedimento gerado no sistema novo, que possui algum tipo de inconsistência e é identificado com o alerta de inconsistência (exemplos: cooperado não vinculado ao hospital, valor de procedimento zerado, quantidade de dígitos de matrícula inválido etc.).

Para atendimento/procedimento migrado do Web, é considerado inconsistentes se no Web também estiver na situação Inconsistente.

**Atendimento Indefinido:** é o atendimento gerado no sistema novo e que não possui procedimento. Para atendimentos/procedimentos importados do Web, são considerados como "Indefinido" os atendimentos zerados, atendimentos que não possuem registro de pagamento (valor) em qualquer situação, atendimentos glosados que não possuem registro de glosa ou situação de glosa (será analisada, devida etc.) e se o atendimento não possuir dados obrigatórios do procedimento.

**Consistido:** atendimento digitado que não possui inconsistências e está pronto para entrar na geração de espelho.

**Espelhado:** atendimento que está dentro de espelho.

**Espelho Pago:** atendimento que será pago através de espelho (informativo).

**Faturado:** atendimento que possui procedimentos faturados (dentro de uma Fatura).

**Pago:** atendimento que está dentro de fatura e possui pagamento.

**Glosado:** atendimento que possui glosa(s).

**Repassado:** atendimento que possui repasse.

**Com pendência:** atendimento digitado e que foi incluída pendência. Esse atendimento não entra na geração de espelho até que sua pendência seja retirada.

**Excluído:** atendimento que foi excluído do banco de dados. Esse atendimento nunca retornará para ser reaproveitado para digitação.





**Pendência:** atendimento que está digitado aguardando a liberação para poder ser incluso no espelho.



**Atenção:** a cada virada de ano, a contagem de números de atendimentos do SASC reinicia a partir do número 1. Assim, para acessar as informações de um determinado atendimento, é necessário fazer a pesquisa informando o ano desejado.

Fonte: Imagem criada pelo autor.

### 3. Canvas Analítico

Projeto: Predição de Valor Líquidos dos Cooperados Filiados a FENCOM.

 <b>1. Question</b>  <p>Utilizando a base de dados extraída do sistema SASC, é possível realizar uma previsão de quanto um cooperado possa receber em um mês específico ?</p>	 <b>2. Data Sources</b>  <p>DataSet privado extraído da base de dados do Sistema SASC utilizando uma consulta SQL padrão ANSI no SGBD SQL SERVER da FENCOM. Principais features que podem compor o dataset são os valores brutos de procedimento, que correspondem as consultas e cirurgias executadas pelos médicos e lançamentos eventuais. Estes se referem a acertos manuais.</p>	 <b>3. Heuristics</b>  <p>Com as features do dataset, é possível um modelo prever o valor líquido que um cooperado possa receber em um determinado mês?</p> <p>Mesmo não possuindo informação dos valores de incidência que o médico vai ter em um determinado repasse e sem procedimentos digitados na competência que queremos prever quanto o médico vai receber.</p> <p>Avaliar se existe algum padrão referente às features de especialidade, nacionalidade, estado civil, Hospital de atendimento e grau de instrução.</p>	 <b>4. Validation</b>  <p>Os dados que não são da competência de Junho de 2022 serão divididos em Treino e Validação. A validação será realizada pegando os dados que não são da competência de junho de 2022, comparando com o valor já repassado dessas competências. Será avaliada a acurácia do modelo de classificação e regressão. Serão contados os dados que ficaram dentro da margem de erro de 10%. O resultado será mostrado para a FENCOM e a instituição vai analisar se é satisfatório.</p> <p>Após o modelo treinado será realizada a predição da competência 06/2022 depois que o modelo for treinado e testado.</p>
--	--	---	---

<p><b>5. Implementation</b></p> <p>Transformação das variáveis categóricas em variáveis numéricas.</p> <p>Tratamento de atributos faltantes.</p> <p>Um modelo de classificação(Árvore de decisão) para saber se o médico vai receber algum valor ou não. Para os casos que, no modelo anterior, foi definido que o médico vai receber, será usado um modelo de regressão buscando a margem de erro de 10%.</p>	<p> <b>6. Results</b></p> <p>Predição de todos os cooperados estejam dentro da margem de erro de 10%.</p> <p>Predição de determinados tipos de cooperados estejam dentro da margem de 10% (Ex:Cooperado pessoa física ou cooperado que não esteja dentro de uma regra de conversão).</p>	<p> <b>7. Next Steps</b></p> <p>Caso a predição esteja fora da margem de erro, considerar incluir outros campos.</p> <p>Se o resultado da previsão for satisfatório, implementar a automatização das tarefas como extração, transformação, treinamento e validação para colocar o modelo em produção. Ou seja, ser efetivamente utilizado na FENCOM.</p> <p>Verificar tendencias com padrões entre as features. Necessidade de mercado como cooperativas de especialidades específicas.</p>
--	---	--



#### 4. Coleta de Dados

Os dados foram coletados da base de dados do sistema SASC através de uma consulta SQL em um servidor de banco de dados SQL SERVER. Se trata de um software de folha de pagamento que as cooperativas vão recebendo o pagamento dos convênios e o SASC realiza o repasse do valor para os médicos. O médico pode ter vários repasses no mesmo mês. Os atributos selecionados via script SQL são importantes porque se referem a base para o processo do repasse, principalmente os valores de procedimentos e de lançamentos eventuais.

Será realizada uma predição do valor líquido que cada cooperado receberá na competência de Junho de 2022. O dataset coletado se refere a uma cooperativa de Belo Horizonte com faixa de datas entre Janeiro de 2021 e Junho de 2022.

**Descrição: Dados com valores de procedimentos e lançamentos eventuais agrupados por cooperado e competência.**

**Link:** <https://github.com/lennonfernandesoliveira/TCC/blob/main/bdPredicao.xlsx>

Nome do Atributo	Descrição	Tipo
pf_ou_pj	Pf para pessoa física PJ para pessoa jurídica	Texto
especialidade	Especialidades Médicas	Texto
competência	Mês e Ano que o repasse ocorreu.	Texto
nacionalidade	Nacionalidade do médico	Texto
situacao_epoca_do_repasse	Situação quando o repasse ocorreu ou situação atual para o caso da competência de 06/2022	Texto
hospital	Hospital ou consultório que o médico atua.	Texto
estado_civil	Estado civil do médico	Texto
grau_instucao	Superior, Mestrado ou Doutorado	Texto
cooperado_origem_de_conversao	1 - Se o cooperado participa de alguma regra de conversão como cooperado origem. 0 – Não participa	Inteiro
cooperado_destino_de_conversao	1 - Se o cooperado participa de alguma regra de conversão como	Inteiro

	cooperado destino. 0 – Não participa	
participa_servico_especial	1 - Se o cooperado participa de algum serviço especial. 0 – Não participa	Inteiro
valor_procedimento_a_repassar	Valor bruto dos procedimentos realizados pelos médicos. Se refere a todos os procedimentos que entraram no repasse.	Real
valor_lancamento_eventual_a_repassar	Valores dos lançamentos eventuais(acertos manuais.	Real
valor_predicao_liquido	Se refere aos valores que cada cooperado recebeu na competência, após as incidências do processo do repasse serem aplicadas.	Real

## 5. Processamento/Tratamento de Dados

Para implementar o código de processamento e tratamento de dados do dataset, foi utilizado o Google COLAB na linguagem Python versão 3.7.13. O link do código, assim como o dataset, está disponibilizado na parte reservada a links do relatório técnico.

As colunas pf\_ou\_pj, competência, situacao\_epoca\_do\_repasso não possuem dados nulos.

Os registros com dado de nacionalidade nulo, tiveram a moda “Brasileira” informada na coluna.

```
df['nacionalidade'].fillna(('Brasileira'), inplace=True)
```

Nas colunas especialidade, hospital e estado civil existem dados ausentes. Para esses casos, foram setados uma descrição de ausente. Foi utilizado essa estratégia devido a essas features serem difíceis de uma definição. Também evita que os dados cheguem nulo para os modelos. Esses registros possuem valores de procedimentos e lançamentos eventuais que não podem ser ignorados

```
df['especialidade'].fillna(('Sem Especialidade'), inplace=True)
```

```
df['hospital'].fillna(('Sem Hospital'), inplace=True)
```

```
df['estado_civil'].fillna(('Sem'), inplace=True)
```

Registros sem grau de instrução se referem a cooperados do tipo pessoa jurídica. Será adotado uma opção pj nesse caso.

```
df['grau_instrucao'].fillna(('pj'), inplace=True)
```

As colunas estado civil, nacionalidade e situação\_epoca\_do\_repasso como se tratam de atributos categóricos não ordinais, foram convertidos utilizando one hot encoding. O algoritmo one hot encoding foi escolhido para que o modelo não interprete uma relação de ordem.

```
df = pd.get_dummies(df, columns = ['estado_civil'])
df = pd.get_dummies(df, columns = ['nacionalidade'])
df = pd.get_dummies(df, columns = ['situacao_epoca_do_repassse'])
df
```

Para as colunas categóricas pf\_ou\_pj, grau\_instrucao, competência que possuem a relação de ordem explícita ou se tratam de domínios binários, foi utilizado o método label encoder mostrado a seguir. Com relação as colunas hospital e especialidade, devido ao tamanho do domínio de ambas features, serão transformadas usando o labelEncoder.

```
def generate_labelencoder(atts):
    for attr in atts:
        df[attr] = le.fit_transform(df[attr])
    return df
```

```
df = generate_labelencoder(['pf_ou_pj', 'grau_instrucao', 'competencia', 'hospital', 'especialidade'])
```

Coluna id\_cooperado foi excluída. O ID nunca deve ser considerado, pois não serve para descrever o cooperado visto que, normalmente ele é gerado aleatoriamente.

As colunas valor\_lancamento\_eventual\_a\_repassar e valor\_procedimento\_a\_repassar serão unificadas gerando uma coluna valor\_total. Para que seja possível um repasse para o cooperado, é obrigatório que umas das colunas sejam diferente de zero. A cooperativa pode repassar somente o valor de procedimento ou de lançamento eventual. Se tratam de colunas numéricas reais e o modelo de regressão linear comporta esse tipo de dado. Pode aparecer a necessidade de colocar os valores em uma escala menor, para que não tenham um peso maior que o necessário. Um exemplo seria coloca-los entre 0 e 1.

As outras features possuem domínio binário(0,1), portanto não serão modificadas.

## 6. Análise e Exploração dos Dados

A análise exploratória auxilia na obtenção de conhecimento sobre o Dataset e na escolha da melhor estratégia para o processamento do conjunto de dados, impactando diretamente na escolha do algoritmo de machine learning e no resultado do projeto. Uma etapa muito comum antes de implementar o modelo de inteligência artificial, sendo uma das partes mais demoradas de todo o processo. A mesma

consiste na sumarização e visualização dos dados para estabelecer inferências sobre o DataSet, auxiliando na tomada de decisão referente a como os dados deverão ser processados e quais algoritmos se encaixam melhor para realizar a inferência.

Para a etapa de análise exploratória, foi utilizado o Google COLAB na linguagem Python versão 3.7.13. O acesso ao colab foi colocado na parte de links deste trabalho.

## 6.1 Estatística descritiva.

A estatística descritiva é uma área da estatística que tem por objetivo sumarizar e resumir um conjunto de dados. Alguns indícios de como os dados estão distribuídos ou como eles devem ser tratados aparecem já na estatística descritiva.

Agora será realizada uma estatística descritiva de cada feature que compõe o conjunto de dados. Portanto, para cada feature do dataset, vamos plotar seus dados em gráficos e discriminar suas características estatísticas.

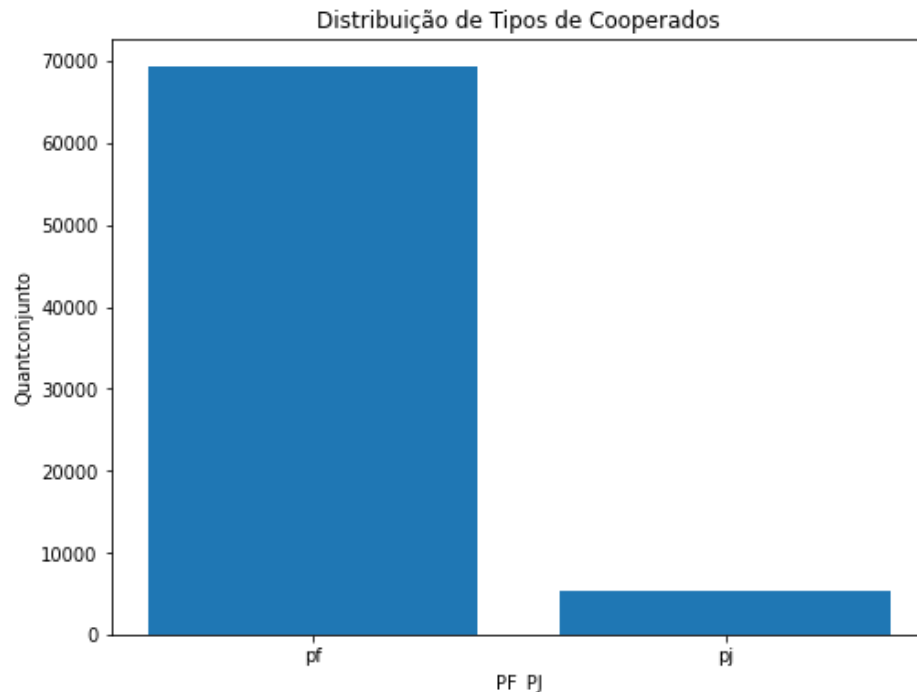
A primeira coluna do nosso dataset é pf\_pj. Refere-se a um cooperado médico ou uma pessoa jurídica. É possível ver na figura 3, o quanto essa feature está desbalanceada. Através do comando a seguir, podemos ver que a frequência de um cooperado pessoa física é mais dominante no conjunto de dados.

Cooperado Pessoa Física corresponde a quase 93% dos dados no dataset.

```
print('A Moda é ',df['pf_ou_pj'].mode())  
df['pf_ou_pj'].describe()
```

```
A Moda é 0    pf  
dtype: object  
count    74484  
unique      2  
top        pf  
freq     69264  
Name: pf_ou_pj, dtype: object
```

**Figura 3 – Gráfico de Barras. Tipos de cooperado.**



Fonte: Imagem criada pelo autor.

A segunda coluna a ser analisada é a especialidade do cooperado.

Como podemos ver na figura a seguir, as especialidades possuem um domínio com 72 itens e a moda dessa coluna é médico clínico com 14.850 registros. Corresponde a quase 20% dos dados no DataSet.

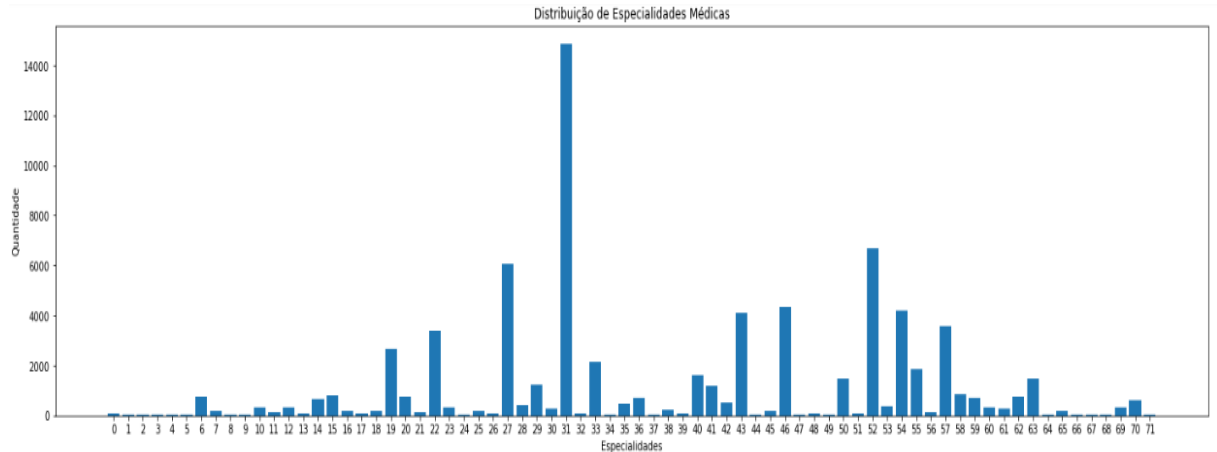
```
print('A Moda é ',df['especialidade'].mode())
df['especialidade'].describe()
```

```
A Moda é 0 Médico clínico
dtype: object
count          74484
unique           72
top      Médico clínico
freq           14850
Name: especialidade, dtype: object
```

Para que fosse possível realizar uma visualização da frequência dos dados de especialidades e seja um gráfico que não ocupasse tanto espaço, foi transformado em uma variável numérica conforme figura 4. O intuito é analisar que, se somar a

frequência das especialidades que estão em segundo e terceiro lugar, ainda ficariam abaixo da frequência da moda.

**Figura 4 – Gráfico de Barras. Frequência especialidades médicas.**

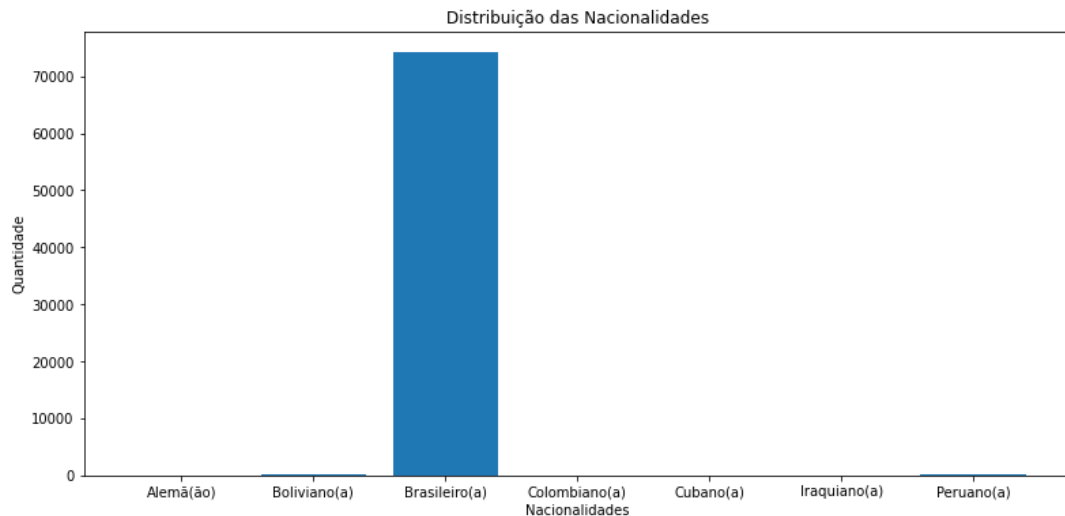


Fonte: Imagem criada pelo autor.

A próxima feature a ser analisada do conjunto de dados é nacionalidade dos cooperados. Brasileira é nacionalidade que corresponde a 99% do conjunto de dados conforme podemos ver a seguir.

```
print('A Moda é ',df_grafico['nacionalidade'].mode())
df_grafico['nacionalidade'].describe()
especialidade = df['nacionalidade']
especialidade_sum = especialidade.value_counts()
especialidade_sum = especialidade_sum.sort_index()
print (especialidade_sum)
```

```
A Moda é 0    2
dtype: int64
Alemã(ão)      18
Boliviano(a)   108
Brasileiro(a)  74178
Colombiano(a)   54
Cubano(a)      18
Iraquiano(a)   18
Peruano(a)     90
Name: nacionalidade, dtype: int64
```

**Figura 5 – Gráfico de Barras. Frequência das nacionalidades dos cooperados.**

Fonte: Imagem criada pelo autor.

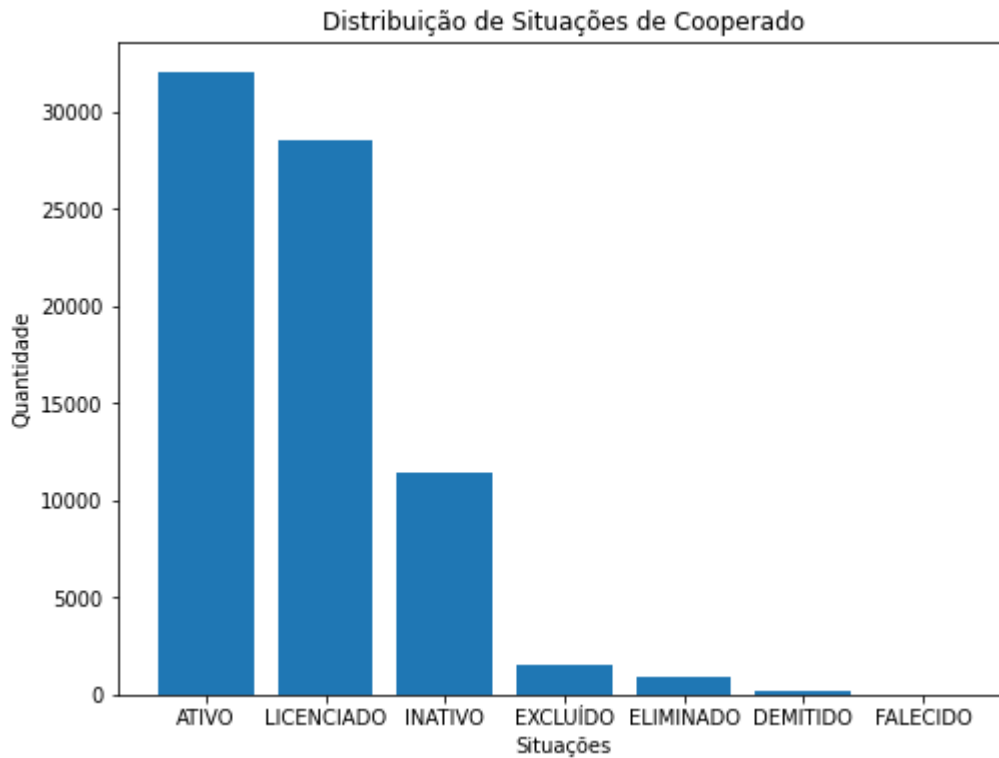
Dentro da FENCOM é unanimidade que a coluna `situacao_epoca_do_repassse`, tenha grande influência para que o cooperado receba ou não, um respectivo valor dentro de um mês. Isso ocorre porque a cooperativa, que realiza a gestão de todo o fluxo dentro do sistema SASC, pode parametrizar em quais situações um cooperado passa a estar habilitado para um eventual recebimento. Conforme podemos ver a seguir, as situações ativo, licenciado e inativo são as situações de cooperado com as maiores frequências dentro do dataset.

```
print('A Moda é ',df['situacao_epoca_do_repassse'].mode())
df_grafico['situacao_epoca_do_repassse'].describe()
data = df['situacao_epoca_do_repassse']
data_sum = data.value_counts()
data_sum = data_sum.sort_index()
print (data_sum)
```

```
A Moda é  0    ATIVO
dtype: object
ATIVO      31995
DEMITIDO    161
ELIMINADO   890
EXCLUÍDO   1487
FALECIDO    18
INATIVO    11439
LICENCIADO  28494
Name: situacao_epoca_do_repassse, dtype: int64
```



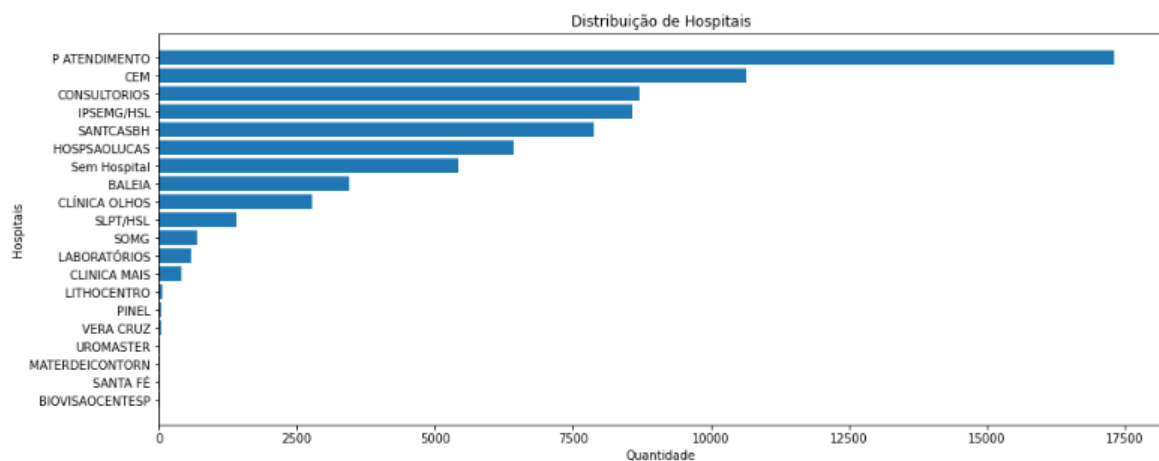
**Figura 6 – Gráfico de Barras. Frequência da situação dos cooperados.**



Fonte: Imagem criada pelo autor.

Na figura 7, segue, a frequência dos hospitais plotados em um gráfico de barras horizontais.

**Figura 7 – Gráfico de Barras Horizontais. Frequência dos Hospitais.**



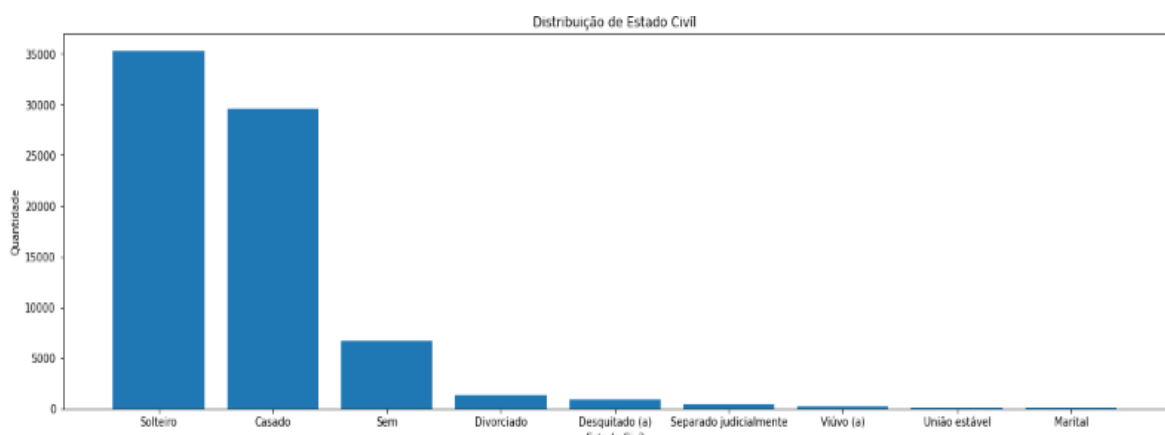
Fonte: Imagem criada pelo autor.

O próximo atributo a ser discriminado é estado civil e na figura 8, consta a frequência de registros na coluna. Alguns questionamentos nos remetem ao estado civil do médico. Um cooperado pessoa física casado, tem a média de procedimentos em aberto diferente de um cooperado solteiro? Tal questionamento é realizado, para saber se deveríamos fazer alguma campanha em específico para um determinado nicho dentro das cooperativas médicas.

```
print('A Moda é ',df['estado_civil'].mode())
df_grafico['estado_civil'].describe()
data = df['estado_civil']
data_sum = data.value_counts()
data_sum = data_sum.sort_index()
print (data_sum)
```

```
A Moda é 0      Solteiro
dtype: object
Casado          29556
Desquitado (a)    918
Divorciado       1386
Marital          36
Sem             6660
Separado judicialmente 396
Solteiro        35244
União estável    126
Viúvo (a)       162
Name: estado_civil, dtype: int64
```

**Figura 8 – Gráfico de Barras. Frequência do estado civil dos médicos.**

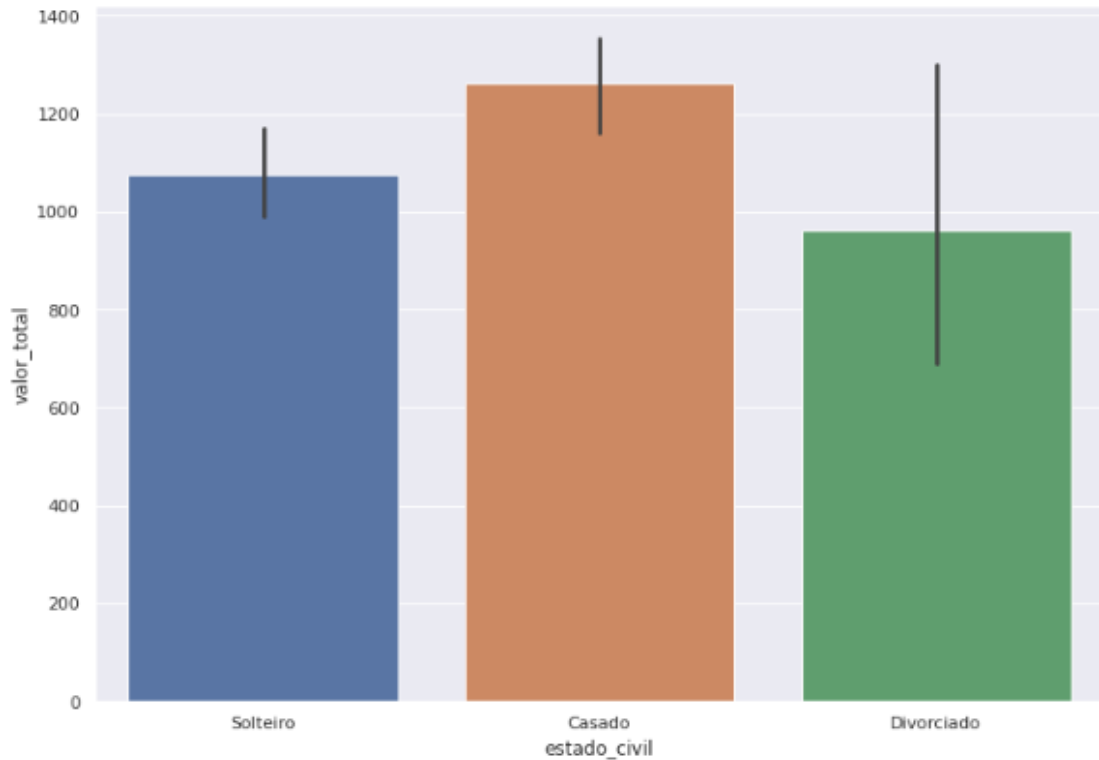


Fonte: Imagem criada pelo autor.

Na figura 9, podemos observar que ao considerarmos todos os registros do dataset, desconsiderando o fato do cooperado possuir ou não valores em aberto, é possível

ver que a média dos cooperados casados é ligeiramente maior que os cooperados solteiros e cooperados divorciados.

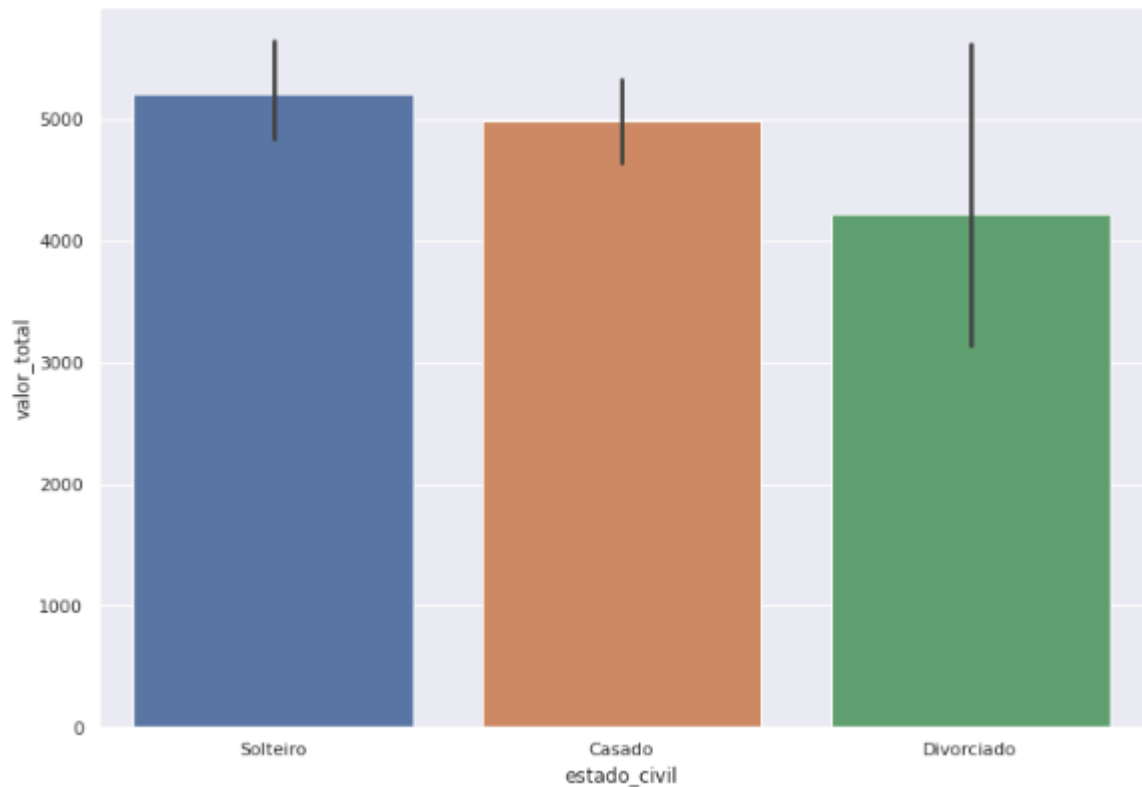
**Figura 9 – Gráfico com média dos procedimentos em aberto por estado civil.**



Fonte: Imagem criada pelo autor.

Já na figura 10, considerando, somente, cooperados que possuem nas colunas valor\_procedimento\_a\_repassar ou valor\_lancamento\_eventual\_a\_repassar valores maiores que zero, os cooperados solteiros possuem um média quase equivalente aos médicos casados.

**Figura 10 – Gráfico com média dos procedimentos em aberto por estado civil.**



Fonte: Imagem criada pelo autor.

Realizaremos um teste de hipótese a seguir para validar se as médias são iguais entre os grupos de médicos solteiros e casados.

O teste de hipótese, fornece ferramentas para rejeitarmos ou não uma hipótese estatística através de evidências fornecidas na amostra. Para tal, pegaremos uma amostra de 10% de forma aleatória do conjunto de dados.

Será utilizado um teste de hipótese, onde a hipótese nula, vai verificar se a média dos procedimentos realizados por casados, não possui uma diferença considerável para os cooperados solteiros.

A hipótese alternativa, considera as médias entre as duas populações como diferentes.

$$H_0 : \underline{\mu}_1 = \underline{\mu}_2 \text{ contra } H_1 : \underline{\mu}_1 \neq \underline{\mu}_2$$

então a estatística se simplifica, sob  $H_0$ , em

$$T^2 = (\bar{X}_1 - \bar{X}_2)^T \frac{S_{pooled}^{-1}}{n_1 + n_2 - 2} (\bar{X}_1 - \bar{X}_2) \sim \frac{(n_1 + n_2 - 2)p}{n_1 + n_2 - p - 1} F_{p, n_1 + n_2 - p - 1}.$$

Para implementarmos a estatística de teste, utilizaremos o método a seguir no google colab. Será utilizado um nível de significância de 5% e um intervalo de confiança de 95%.

```
from scipy.stats import f
def T2Hotelling_duas_amostras(df1, df2, delta0):
    n1 = len(df1)
    n2 = len(df2)
    p = len(df1.columns)
    Xbarra1=df1.mean()
    Xbarra2=df2.mean()
    S1 = df1.cov()
    S2 = df2.cov()
    S_pooled = ((n1-1)*S1 + (n2-1)*S2)/(n1+n2-2)
    S_pooled_inv = np.linalg.inv(S_pooled)

    T2Hotelling_duas_amostras = np.array(Xbarra1-Xbarra2-delta0).T.dot(S_pooled_inv).dot(np.array(Xbarra1-Xbarra2-delta0)) / [(n1+n2-2)]
    qf = f.ppf(0.95, p, (n1+n2-2), loc=0, scale=1)
    teste = T2Hotelling_duas_amostras > (n1+n2-2) * p / (n1+n2-p-1) * qf
    pvalor = 1-f.cdf(T2Hotelling_duas_amostras/((n1+n2-2) * p / (n1+n2-p-1)), p, (n1+n2-2))
    print('Rejeitamos H0') if teste else print('Não rejeitamos H0')
    print('Valor da estatística', T2Hotelling_duas_amostras)
    print('valor p', pvalor)
```

```
T2Hotelling_duas_amostras(df_amostra_casado,df_amostra_solteiro,[0])
```

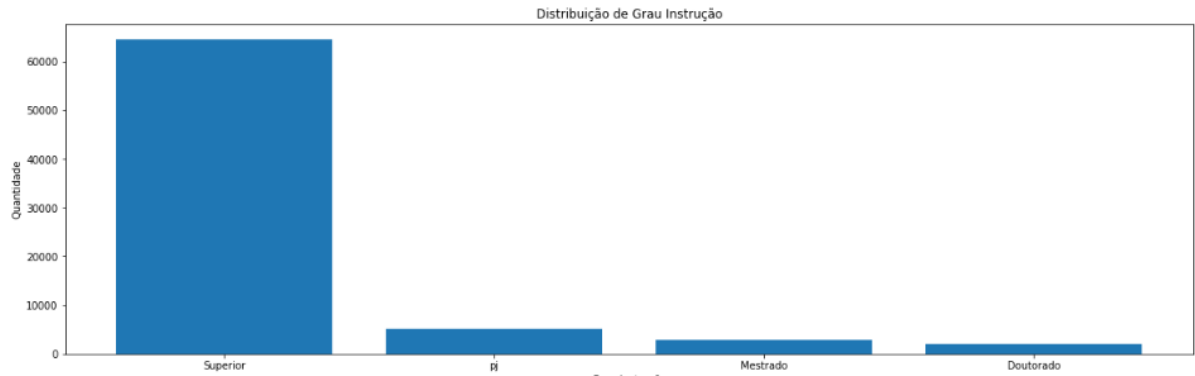
```
Não rejeitamos H0
Valor da estatística 4.67942676794176e-06
valor p 0.9982743119535746
```

Conforme podemos ver o resultado, não rejeitamos a hipótese de que as médias dos procedimentos dos médicos solteiros e casados são iguais.

A mesma análise bivariada que fizemos com a feature estado civil, será realizada com o atributo grau\_instrucao. Primeiramente, vamos plotar na figura 11, a frequência dos dados de grau de instrução. Já na figura 12, o objetivo é demonstrar se existe relação do crescimento do grau de instrução, com a média dos procedimentos realizados, concluindo que também não há qualquer relação proporcional ou desproporcional das duas features.

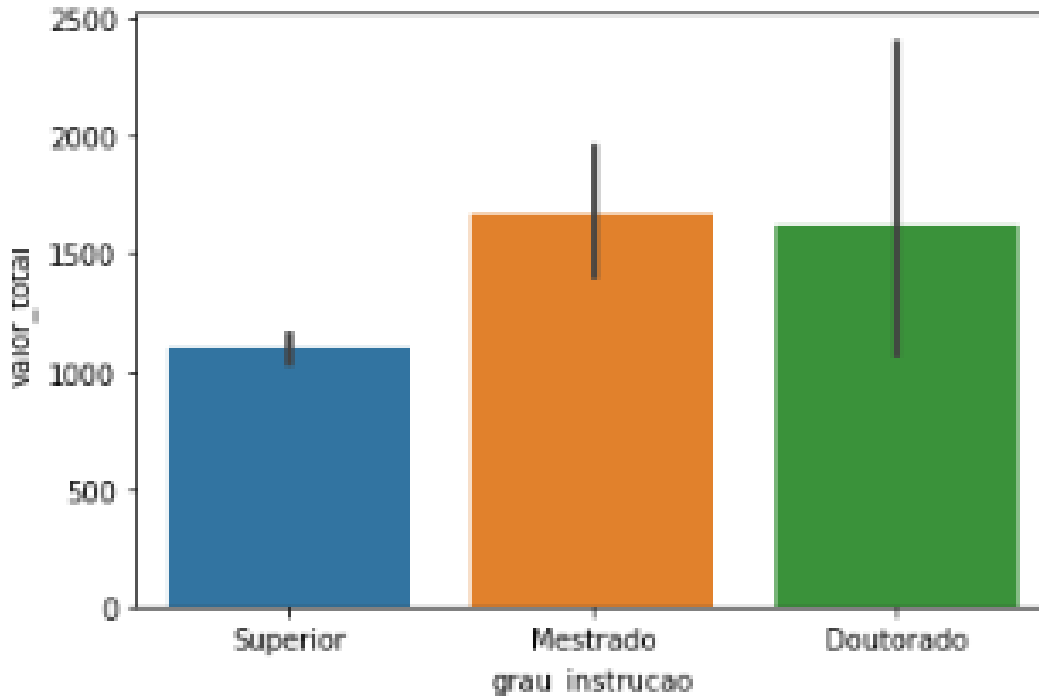
A maioria dos médicos que não possuem procedimentos a serem repassados, constam grau de instrução igual a Superior.

**Figura 11 – Gráfico com frequência da feature estado civil.**



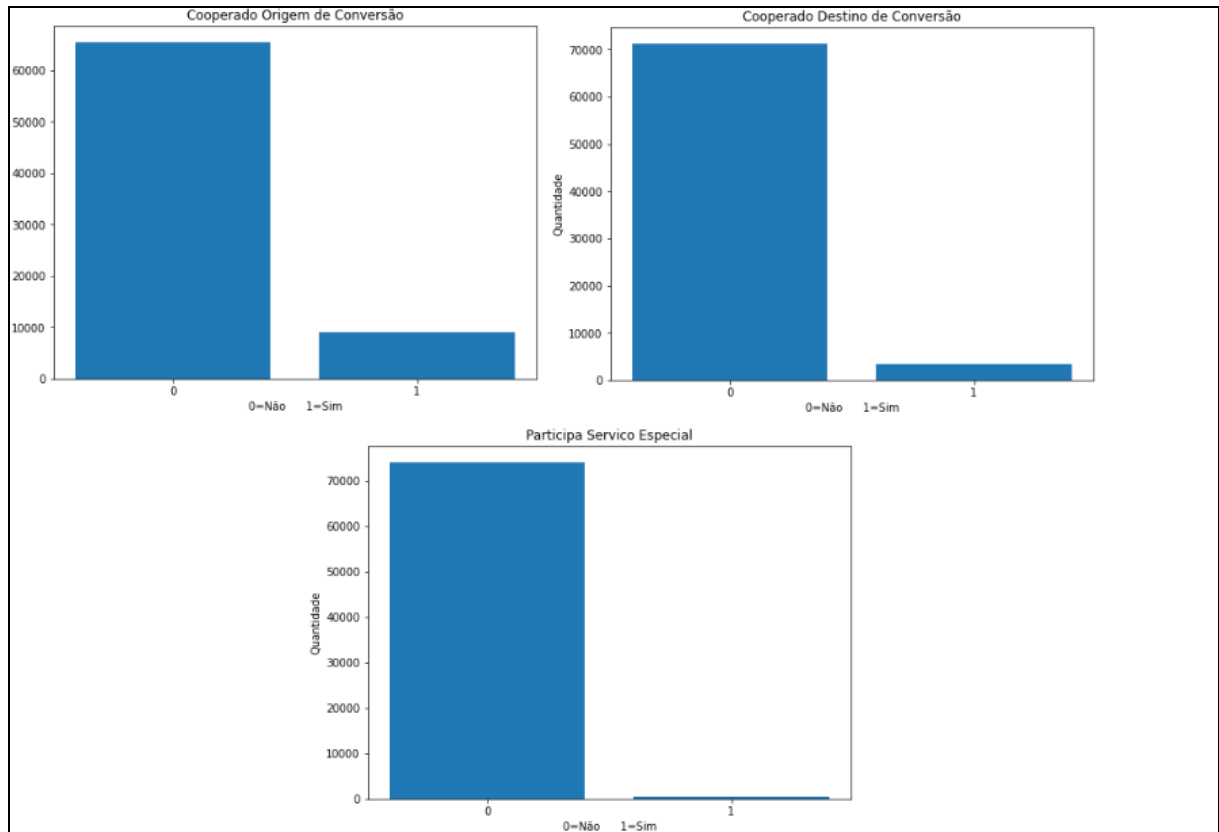
Fonte: Imagem criada pelo autor.

**Figura 12 – Gráfico com média dos procedimentos por grau instrução.**



Fonte: Imagem criada pelo autor.

**Figura 13 – Gráfico com frequência das features origem\_conversao, destino\_conversao e participa\_servico\_especial no conjunto de dados.**

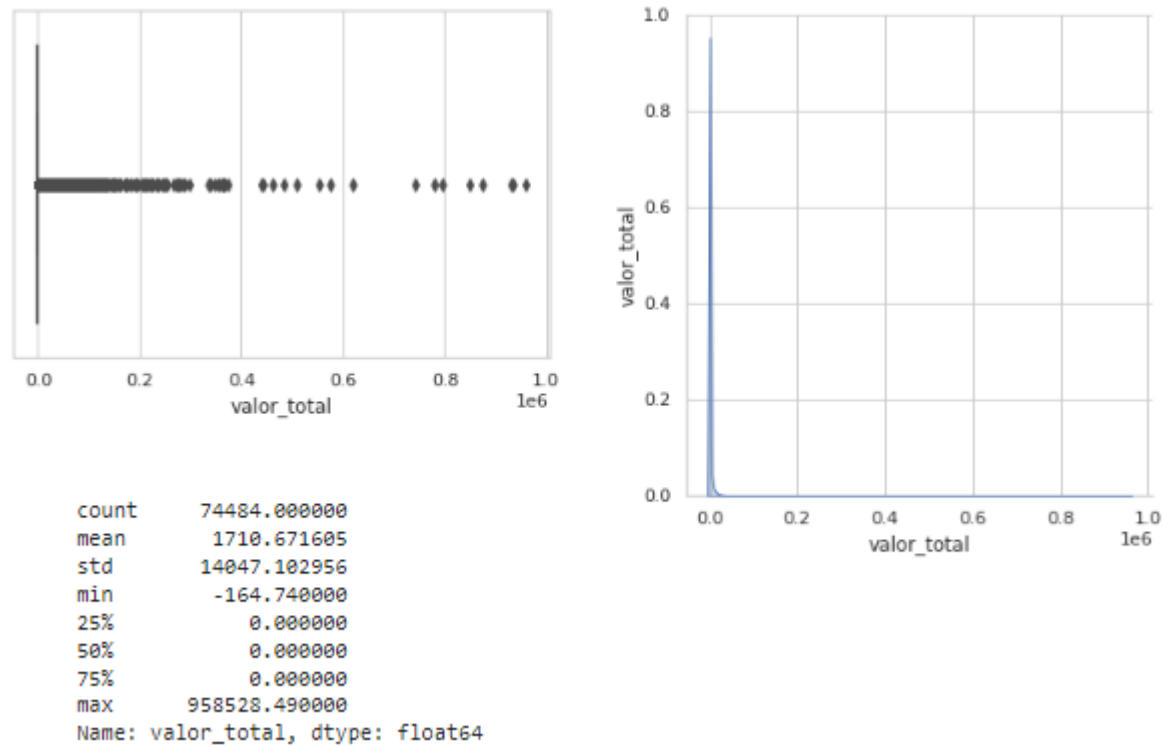


Fonte: Imagem criada pelo autor.

Conforme podemos ver na figura 13, constam frequência das features origem\_conversao, destino\_conversao e participa\_servico\_especial no conjunto de dados. Como se tratam de colunas com o tipo de dado booleano, constam poucas frequências setadas como Sim. Pode ser um indício de que essas colunas terão de ser removidas antes do dataset ser repassado para os algoritmos de machine learning. Porém essa análise determinística será realizada na etapa 7 com a feature engineering.

A última feature a discriminarmos suas características estatísticas será valor\_total. Ela compõe o somatório da coluna valor\_procedimento\_a\_repassar e valor\_lancamento\_eventual\_a\_repassar. Conforme gráficos plotados na figura 14, é possível que a coluna valor\_total, não se trate de uma distribuição normalizada.

**Figura 14 – Gráficos com distribuição da coluna valor\_total.**



Fonte: Imagem criada pelo autor.

Para certificarmos que valor\_total não possui uma distribuição normalizada, vamos realizar um teste de hipótese onde a hipótese nula se trata de uma distribuição normal e a hipótese alternativa de não possuir distribuição normal.

```
from scipy.stats import normaltest
```

```
normaltest(df_grafico['valor_total'])
```

```
NormaltestResult(statistic=206324.4986078155, pvalue=0.0)
```

```
stat_test, p_valor = normaltest(df_grafico['valor_total'])
print(stat_test)
print(p_valor)
```

```
206324.4986078155
0.0
```

```
significancia = 0.05
print('Rejeitamos H0') if p_valor <= significancia else print('Não rejeitamos H0')
```

```
Rejeitamos H0
```

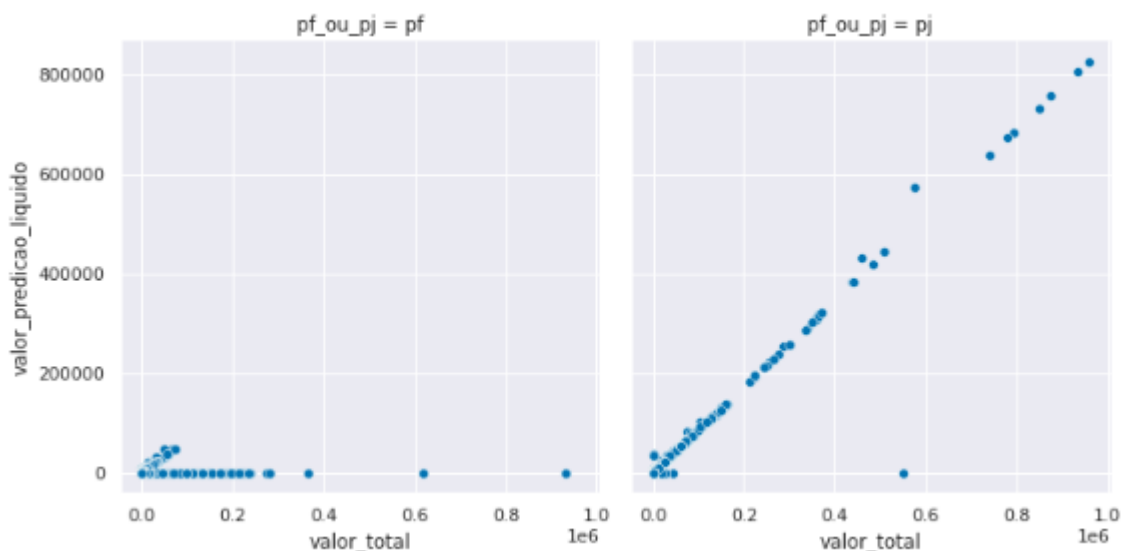


Conforme mostrado no teste de hipótese acima, temos evidências estatísticas para rejeitarmos que a feature `valor_total`, possui uma distribuição normalizada.

Ao realizarmos uma análise da figura 15, através do gráfico de dispersão entre as três variáveis `pf_ou_pj`, `valor_total` e o target `valor_predicao_liquido`, chegamos à conclusão que grande parte dos valores outliers, se encontram no grupo dos cooperados pessoa jurídica. Já analisamos e verificamos que não se trata de erros do software ou falhas na extração e processamento dos dados. Alguns cooperados pessoa jurídica possuem valores a receber atípicos.

Uma boa parte dos registros do conjunto de dados, possuem as colunas `valor_total` e `valor_predicao_liquido` próximo de zeros. Por isso trabalharemos com dois algoritmos de machine learning. O primeiro classificará se o cooperado vai receber ou não. Por isso a criação da variável `vai_receber`. Para cooperados que foram classificados com a feature `vai_receber` igual a 1, terão seus dados processados por um algoritmo de regressão na tentativa de prever o `valor_predicao_liquido`. Já os cooperados que foram classificados na coluna `vai_receber` igual a zero, indicaram que aquele cooperado não vai receber nenhum valor para a competência.

**Figura 15 – Gráfico de dispersão para análise multivariada.**



Fonte: Imagem criada pelo autor.

## 7. Preparação dos Dados para os Modelos de Aprendizado de Máquina

Para realizarmos as previsões do conjunto de dados no dataset, dividiremos o processo em duas etapas. A primeira se refere a uma classificação do dataset para saber se cada linha do conjunto de dados, vai possuir um `valor_predicao_liquido`

igual a zero ou maior. Para isso criamos uma nova feature chamada vai receber. A segunda etapa pegará cada linha do conjunto de dados, que na etapa 1 foi classificado como vai\_receber, e realizará uma regressão para prever o valor que o médico receberá em um determinado mês. O colab preparacao\_dados\_para\_predicao\_machine\_learning, que pode ser acessado na declaração de links deste trabalho, contém as linhas de código para realizar o processamento e deixar o dataset preparado para ser usado nos dois modelos de machine learning.

A maioria dos tratamentos já foram realizados na etapa 5 do trabalho, então aqui vamos incluir apenas aquelas que não constavam em etapas anteriores.

Abaixo o comando para criação da coluna vai\_receber. Para linhas cujo a competência é igual a junho de 2022, serão setados como 2. Isso para que possamos separar futuramente do conjunto de treinamento e teste. Não usaremos a coluna competência para fazermos a separação porque ela sofrerá uma transformação se tornando uma feature numérica.

```
df['vai_receber'] = np.where(df['competencia']=='2022-06-01',2,
                             (np.where(df['valor_predicao_liquido'] > 0.000000001, 1, 0)))
```

Outra mudança é a criação da feature valor\_total que será uma composição das colunas valor\_procedimento\_a\_repassar e valor\_lancamento\_eventual\_a\_repassar.

```
df_classificador['valor_total'] = df_classificador['valor_procedimento_a_repassar']+df_classificador['valor_lancamento_eventual_a_repassar']
```

Para que possamos realizar a classificação, primeiro precisamos realizar a feature selection. Utilizaremos o algoritmo de machine learning Floresta aleatória ou RandomForest. Ao final do treinamento do modelo, é possível acessar o atributo feature\_importances que contém os pesos e importância de cada coluna para o alvo vai\_receber. Podemos analisar através do gráfico da figura 16, quais são as colunas mais importantes para a inferência.

```

from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
import seaborn as sns
# Carregando dataset
X = df_classificador[['pf_ou_pj', 'especialidade', 'competencia', 'nacionalidade', 'situacao_epoca_do_repasse']]
y = df_classificador[['vai_receber']]
# Criando conjunto de treino e teste
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
# Treinando modelo
model = RandomForestClassifier()
model.fit(X_train, y_train)
# Mostrando importância de cada feature

RandomForestClassifier()

importances = pd.Series(data=model.feature_importances_, index=X.columns)
sns.barplot(x=importances, y=importances.index, orient='h').set_title('Importância de cada feature')

```

**Figura 16 – Gráfico contendo a importância de cada feature.**



Fonte: Imagem criada pelo autor.

Para o dataset para o modelo de classificação, vamos manter as colunas competência, situação\_epoca\_do\_repasse e valor\_total. Abaixo segue comando para excluirmos as outras colunas.

```
df_classificador = df_classificador.drop(['valor_lancamento_eventual_a_repassar'], axis=1)
df_classificador = df_classificador.drop(['pf_ou_pj'], axis=1)
df_classificador = df_classificador.drop(['valor_procedimento_a_repassar'], axis=1)
df_classificador = df_classificador.drop(['especialidade'], axis=1)
df_classificador = df_classificador.drop(['nacionalidade'], axis=1)
df_classificador = df_classificador.drop(['hospital'], axis=1)
df_classificador = df_classificador.drop(['estado_civil'], axis=1)
df_classificador = df_classificador.drop(['grau_instrucao'], axis=1)
df_classificador = df_classificador.drop(['cooperado_origem_de_conversao'], axis=1)
df_classificador = df_classificador.drop(['cooperado_destino_de_conversao'], axis=1)
df_classificador = df_classificador.drop(['participa_servico_especial'], axis=1)
```

Para evitarmos que alguma coluna tenha uma importância maior que a outra para o algoritmo devido a sua escala, iremos utilizar o MinMaxScaler. A distribuição da coluna valor\_total, agora ficará entre -1 e 1 por possuir valores negativos.

E para as colunas competência e situação\_epoca\_do\_repasse seus valores ficaram distribuídos entre 0 e 1.

```
X_Resto = X.loc[:, X.columns != 'valor_total']
X_Total = X.loc[:, X.columns == 'valor_total']

np.seterr(divide = 'ignore')

|
# Normalização Min-Max dos dados.

min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-1,1))
X_Total['valor_total'] = min_max_scaler.fit_transform(X_Total)
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(0,1))
X_Resto[['competencia', 'situacao_epoca_do_repasse']] = min_max_scaler.fit_transform(X_Resto)

#Y = min_max_scaler.fit_transform(Y)
```

Para que possamos generalizar um modelo e prepara-lo para dados que o algoritmo não conhece, dividiremos o dataset em dados de treino e teste. O percentual de teste será 20% do conjunto, e o restante será usado para treinar o modelo. A divisão será implementada usando o método train\_test\_split do sklearn que separa de forma randômica a base de teste da base de treinamento.

```
x_train, x_test, y_train, y_test = train_test_split( X, Y.vai_receber, test_size=0.2)
```

O problema que temos referente a quantidade de registros com a coluna zerada será resolvido pelo algoritmo de classificação árvore de decisão. Um algoritmo simples mas muito usado devido a sua transparência na tomada de decisão do modelo. Após a utilização do algoritmo, é possível gerar uma árvore de decisão para visualizarmos como as regras foram implementadas.

Para a escolha dos melhores hiper parâmetros para o modelo de árvore de decisão, será utilizado o módulo do sklearn GridSearchCV. Nele é possível introduzir vários hiperparâmetros diferentes para o modelo de machine learning e encontrar quais entregam os melhores resultados.

```
%%time

params = {'min_samples_split': [20,30,40,50,100,200],
          'max_depth': [10,15,20,30,40,50],
          'criterion': ['gini','entropy']}

decision_tree = GridSearchCV(DecisionTreeClassifier(random_state=0), param_grid=params, n_jobs=-1, cv=5, verbose=5)
decision_tree.fit(x_train,y_train)
```

Após o treinamento e teste do modelo de árvore de decisão, os registros que são da competência de 06/2022, serão processados também pelo algoritmo. No próximo passo, será realizada uma regressão com dados que foram classificados como 1 pelo algoritmo de árvore de decisão.

Após o processamento abaixo, a coluna vai\_receber se encontrará atualizada com a predição realizada pela árvore de decisão e o dataset poderá ser tratado para a regressão.

```
df['vai_receber']=df_apos_classificacao['vai_receber']
```

O alvo para o algoritmo de regressão é a coluna valor\_predicao\_liquido. Faremos a tratativa das features novamente uma vez que se trata de outro target. As colunas importantes para o algoritmo de classificação podem ser diferentes para o algoritmo de regressão.

Também será realizada uma feature selection para o dataset usado na regressão.

Foi implementado o método RFE (Recursive Feature Elimination), que analisa de forma recursiva quais são as melhores features baseado na quantidade de colunas do dataset. A cada iteração do RFE, o dataset passará pelo modelo de regressão RandomForestRegressor que apesar de não possuir os melhores resultados, será de grande valia quanto a escolha das features. O número de colunas é decrementado até que realize a predição com apenas uma coluna.

```

ort train_test_split
modelo = rf()
lista=np.arange(1,12)
melhor_score=0
nf=0
lista_score=[]
for i in range(len(lista)):

    X_treino, X_teste, Y_treino, Y_teste = train_test_split(X_feature_selection, Y_feature_selection, test_size = 0.2, random_state = 0)

    modelo = rf()
    rfe = RFE(modelo,n_features_to_select=lista[i])
    X_treino_rfe = rfe.fit_transform(X_treino,Y_treino)
    X_teste_rfe = rfe.transform(X_teste)
    modelo.fit(X_treino_rfe,Y_treino)
    score = modelo.score(X_teste_rfe,Y_teste)
    lista_score.append(score)
    if(score > melhor_score):
        melhor_score = score
        nf = lista[i]
print("Numero ideal de features: %d" %nf)
print("Score com %d features: %f" % (nf, melhor_score))

Numero ideal de features: 6
Score com 6 features: 0.996023

```

```

colunas = list(X_feature_selection.columns)

model = rf()

rfe = RFE(modelo, n_features_to_select=nf)

Previsores_rfe = rfe.fit_transform(X_feature_selection,Y_feature_selection)

modelo.fit(X_feature_selection,Y_feature_selection)

temp = pd.Series(rfe.support_,index=colunas)

features_selecionadas_rfe = temp[temp==True].index
print(features_selecionadas_rfe)

Index(['pf_ou_pj', 'especialidade', 'competencia', 'hospital',
       'grau_instrucao', 'valor_total'],
      dtype='object')

```

Abaixo segue transformação das variáveis do dataset que serão usados para treino e teste do algoritmo de regressão. As features categóricas ficarão entre zero e um e a coluna valor\_total será modificada para uma faixa de -1 e 1.

```

X_Resto = X_regressao.loc[:, X_regressao.columns != 'valor_total']
X_Total = X_regressao.loc[:, X_regressao.columns == 'valor_total']

np.seterr(divide = 'ignore')

# Normalização Min-Max dos dados.

min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-1,1))
valor_total = min_max_scaler.fit_transform(X_Total)
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(0,1))
resto = min_max_scaler.fit_transform(X_Resto)

```

Para os registros cujo a competência é junho de 2022, sofrerá a mesma transformação. A diferença é que os registros ficarão no data\_set `X_predicao_regressao`.

```

X_Resto_predicao_regressao = X_predicao_da_regressao.loc[:, X_predicao_da_regressao.columns != 'valor_total']
X_Total_predicao_regressao = X_predicao_da_regressao.loc[:, X_predicao_da_regressao.columns == 'valor_total']

np.seterr(divide = 'ignore')

min_max_scaler = preprocessing.MinMaxScaler(feature_range=(-1,1))
valor_total_predicao_regressao = min_max_scaler.fit_transform(X_Total_predicao_regressao)
min_max_scaler = preprocessing.MinMaxScaler(feature_range=(0,1))
resto_predicao_regressao = min_max_scaler.fit_transform(X_Resto_predicao_regressao)

```

```

X = np.concatenate((resto, valor_total), axis=1)

```

```

X_predicao_regressao = np.concatenate((resto_predicao_regressao, valor_total_predicao_regressao), axis=1)

```

O percentual dos dados de teste serão 20% do conjunto, e o restante será usado para treinar o modelo.

```

x_train, x_test, y_train, y_test = train_test_split(X, Y_regressao, test_size=0.2)

```

O modelo de regressão a ser utilizado será uma rede neural. Utilizaremos novamente o `GridSearchCV` para a escolha dos melhores hiper parâmetros.

```

params = {'activation': ['relu', 'tanh', 'logistic', 'identity'],
          'solver': ['adam', 'sgd', 'lbfgs'],
          'hidden_layer_sizes': [(1,),(10,),(20,),(30,)] ,
          'learning_rate': ['constant', 'adaptive', 'invscaling']}

mlp_regressor_grid = GridSearchCV(MLPRegressor(random_state=123, alpha=0.001, ), param_grid=params, n_jobs=-1, cv=5, verbose=5)
mlp_regressor_grid.fit(x_train,y_train)

print('Train R^2 Score : %.3f'%mlp_regressor_grid.best_estimator_.score(x_train, y_train))
print('Test R^2 Score : %.3f'%mlp_regressor_grid.best_estimator_.score(x_test, y_test))
print('Best R^2 Score Through Grid Search : %.3f'%mlp_regressor_grid.best_score_)
print('Best Parameters : ',mlp_regressor_grid.best_params_)

```

No próximo vamos detalhar quais hiperparâmetros foram escolhidos para o modelo realizar a regressão.

## 8. Aplicação de Modelos de Aprendizado de Máquina

Como o objetivo do trabalho é realizar uma regressão para se determinar quanto um médico receberá em uma certa competência, e possuem no dataset muitos dados com target zerado, se realizarmos uma regressão com todos os dados as linhas com o alvo zerado influenciaram negativamente no resultado da predição. Também não podemos eliminar tais registros pois a informação do não recebimento pelo cooperado é de suma importância. A estratégia então é separar o processamento da predição em dois momentos. O primeiro será uma classificação dos dados para saber se o médico vai receber algum valor ou não. O segundo passo se refere a pegar os registros que foram classificados no primeiro algoritmo com o target vai\_receber igual a 1 e realizar uma regressão dos valores alcançando a melhor acurácia possível.

Conforme pode ser visto no colab comparação\_modelo.ipynb, para realizar a classificação dos registros comparamos os modelos SVM(Support Vector Machine), Naive Bayes e a Árvore de decisão. O SVM é um algoritmo linear que busca traçar uma linha servindo como uma fronteira entre as duas classes. Já o Naive Bayes, consiste em considerar todas as features do dataset como independentes na predição do alvo. A árvore de decisão é uma representação da tabela de decisão em forma de árvore. A estratégia por trás da tomada de decisão do modelo se torna muito explícita com a visualização da árvore de decisão.

Abaixo veremos que a árvore de decisão apresentou um melhor resultado dentre os modelos de classificação utilizados.

```
gnb = GaussianNB()
y_pred = gnb.fit(x_train, y_train).predict(x_test)

print("Número de erros de classificação do modelo Naive Bayes {0} de {1}"
      .format((y_test != y_pred).sum(), y_pred.shape))

ac = gnb.score(x_test, y_test)

print("\nAcurácia do modelo: {0:.2f}%\n".format(100*ac))
```

Número de erros de classificação do modelo Naive Bayes 359 de (14070,)

Acurácia do modelo: 97.45%



```
svm = SVC(kernel='linear')
svm = svm.fit(x_train,y_train)
yprevisao = svm.predict(x_test)
```

```
| from sklearn import metrics
| metrics.accuracy_score(y_test, yprevisao)
```

```
0.832906894100924
```

```
| print("Número de erros de classificação do modelo SVM {0} de {1}"
|       .format((y_test != yprevisao).sum(), yprevisao.shape))
```

```
Número de erros de classificação do modelo SVM 2351 de (14070,)
```

```
tree_recebe_nao_recebe = DecisionTreeClassifier(random_state=0, criterion='entropy',min_samples_split =100,max_depth=10)
tree_recebe_nao_recebe = tree_recebe_nao_recebe.fit(x_train, y_train)
print("Acurácia (base de treinamento):", tree_recebe_nao_recebe.score(x_train, y_train))
```

```
Acurácia (base de treinamento): 0.9906176700547302
```

```
print("Número de erros de classificação {0} de {1}"
      .format((y_test != y_pred).sum(), y_pred.shape))
```

```
Número de erros de classificação 131 de (14070,)
```

Em um cenário de 14.070 registros, a árvore de decisão só errou em 131 linhas. Para a escolha dos melhores hiper parâmetros para o modelo de árvore de decisão, foi utilizado o módulo do sklearn GridSearchCV. Conforme mostrado nos comandos abaixo, os hiper parâmetros testados no gridsearch foram criterion, min\_samples\_split e max\_deph.

Criterion é a estratégia utilizada pela árvore para separação dos dados. Possui as opções entropia e a impureza de gini. Os dois critérios atuam na separação dos dados, porém o critério gini é computacionalmente mais eficiente. Ele tem de a isolar em um ramo os registros que possuem a classe mais frequente. Já a entropia, tende a balancear melhor os registros nos ramos. Foram testados ambos os critérios e os melhores resultado foram alcançados com a entropia.

O hiper parâmetro min\_samples\_split, se refere ao mínimo de exemplos para se criar um ramo na árvore. Para não correr o risco de criar uma árvore com poucos registros por ramo, o que traria uma menor generalização do problema proposto, foi colocado um mínimo de 20 registros. Com gridSearch o parâmetro definido foram 100 registros por ramo.

O hiper parâmetro max\_depth se refere a profundidade da árvore. Para evitarmos criar uma árvore muito complexa o limite para max\_depth foi 50. Assim o melhor resultado encontrado com max\_depth igual a 10.

```
import itertools
import warnings
warnings.filterwarnings('ignore')
```

```
| %%time

params = {'min_samples_split': [20,30,40,50,100],
          'max_depth': [10,15,20,30,40,50],
          'criterion': ['gini','entropy']}
decision_tree = GridSearchCV(DecisionTreeClassifier(random_state=0), param_grid=params, n_jobs=-1, cv=5, verbose=5)
decision_tree.fit(x_train,y_train)

print('Train R^2 Score : %.3f'%decision_tree.best_estimator_.score(x_train, y_train))
print('Test R^2 Score : %.3f'%decision_tree.best_estimator_.score(x_test, y_test))
print('Best R^2 Score Through Grid Search : %.3f'%decision_tree.best_score_)
print('Best Parameters : ',decision_tree.best_params_)

Fitting 5 folds for each of 60 candidates, totalling 300 fits
Train R^2 Score : 0.991
Test R^2 Score : 0.991
Best R^2 Score Through Grid Search : 0.991
Best Parameters : {'criterion': 'entropy', 'max_depth': 10, 'min_samples_split': 100}
CPU times: user 533 ms, sys: 147 ms, total: 681 ms
Wall time: 6.4 s
```

```
y_pred = tree_recebe_nao_recebe.predict(x_test)
print("Acurácia de previsão:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred, target_names=['NAO','SIM']))
```

```
Acurácia de previsão: 0.9906894100923952
```

	precision	recall	f1-score	support
NAO	0.99	1.00	0.99	10931
SIM	1.00	0.96	0.98	3139
accuracy			0.99	14070
macro avg	0.99	0.98	0.99	14070
weighted avg	0.99	0.99	0.99	14070

Para o problema de regressão serão comparados a regressão ridge, regressão lasso e o modelo de rede neural artificial da sklearn.

Novamente para a escolha dos melhores hiper parâmetros para os modelos, será utilizado a biblioteca GridSeach da Sklearn.

Conforme possa ser visto abaixo, a comparação entre os modelos de regressão começará com a regressão lasso. Se trata de uma regularização do modelo de regressão linear. Para o caso em que existam features altamente correlacionadas, o algoritmo utiliza apenas uma das features e as outras tenham seus coeficientes zerados.

```
Lasso alpha=1
Erro quadrático Médio = 1149160.5777263104
R2 = 0.9967071527441639
Número de atributos usados: 6
Lasso alpha=.01
Erro quadrático Médio = 1158415.4117383966
R2 = 0.9966806336002161
Número de atributos usados: 6
Lasso alpha=.0001
Erro quadrático Médio = 1158512.1438790841
R2 = 0.9966803564203596
Número de atributos usados: 6
```

```
lasso = Lasso().fit(x_train, y_train)
pred = lasso.predict(x_test)
print("Lasso alpha=1")
print("Erro quadrático Médio = ", mean_squared_error(y_test, pred)) #Erro quadrático médio
print("R2 = ", r2_score(y_test, pred)) #Coeficiente de determinação
print("Número de atributos usados: {}".format(np.sum(lasso.coef_ != 0)))
```

```
Lasso alpha=1
Erro quadrático Médio = 1149160.5777263104
R2 = 0.9967071527441639
Número de atributos usados: 6
```

```
y_test['valor_predito']=pred
```

```
#Com margem de erro de 10%
len(y_test[(10*abs(y_test['valor_predicao_liquido']/100))>=abs(y_test['valor_predicao_liquido']-y_test['valor_predito'])])
```

724

Em um cenário de 3192 registros para teste, apenas 724 registros se encontram dentro da margem de erro de 10%. Isso ocorre porque conforme podemos ver no colab `comparacao_modelo.ipynb`, o dataset já passou pelo modelo de classificação e 15958 registros tiveram o target `vai_receber` classificado como 1. Destas, 12766 linhas foram separadas para treino e 3192 registros foram utilizadas para teste.

O próximo modelo a ser analisado será a regressão ridge.

```
%%time
from sklearn.linear_model import Ridge

params = {'alpha': [0.01, 0.1, 1, 10]}
ridgeReg = GridSearchCV(Ridge(random_state=0), param_grid=params, n_jobs=-1, cv=5, verbose=5)
ridgeReg.fit(x_train, y_train)

print('Train R^2 Score : %.3f'%ridgeReg.best_estimator_.score(x_train, y_train))
print('Test R^2 Score : %.3f'%ridgeReg.best_estimator_.score(x_test, y_test))
print('Best R^2 Score Through Grid Search : %.3f'%ridgeReg.best_score_)
print('Best Parameters : ', ridgeReg.best_params_)
```

```
ridgeReg = Ridge(alpha=0.1).fit(x_train,y_train)
```

```
pred = ridgeReg.predict(x_test)
```

```
mse = np.mean((pred - y_test)**2)
```

```
mse
```

```
valor_predicao_liquido    3.694201e+06
dtype: float64
```

```
print('acuracia na base de treinamento ' + str(ridgeReg.score(x_train,y_train)))
print('acuracia na base de teste ' + str(ridgeReg.score(x_test,y_test)))
```

```
acuracia na base de treinamento 0.9973603971125162
acuracia na base de teste 0.9940041264597358
```

```
print("Erro quadrático Médio = ",mean_squared_error(y_test, pred)) #Erro quadrático médio
```

```
Erro quadrático Médio = 3694201.398536824
```

```
y_test['valor_predito']=pred
```

```
#Ridge com com margem de erro de 10%
```

```
len(y_test[(10*abs(y_test['valor_predicao_liquido']/100))>abs(y_test['valor_predicao_liquido']-y_test['valor_predito'])])
```

```
670
```

Podemos concluir que o resultado da regressão ridge foi abaixo da regressão Lasso. Ao compararmos o MSE, a regressão lasso obteve um resultado melhor. Em um cenário de 3192 registros, apenas 670 estão dentro da margem de erro de 10%.

Agora iremos analisar a rede neural da Sklearn MLPRegressor. É uma rede neural multicamada perceptron, ou seja, uma rede neural artificial. Para a escolha dos melhores hiper parâmetros no cenário do dataset do trabalho, novamente vamos usar o gridsearch.

```
%%time
```

```
params = {'activation': ['relu', 'tanh', 'logistic', 'identity'],
          'solver': ['adam','sgd', 'lbfgs'],
          'hidden_layer_sizes': [(1,),(10,),(20,),(30,)] ,
          'learning_rate' : ['constant', 'adaptive', 'invscaling']
        }
```

```
mlp_regressor_grid = GridSearchCV(MLPRegressor(random_state=123, alpha=0.001, ), param_grid=params, n_jobs=-1, cv=5, verbose=1)
mlp_regressor_grid.fit(x_train,y_train)
```

```
print('Train R^2 Score : %.3f'%mlp_regressor_grid.best_estimator_.score(x_train, y_train))
print('Test R^2 Score : %.3f'%mlp_regressor_grid.best_estimator_.score(x_test, y_test))
print('Best R^2 Score Through Grid Search : %.3f'%mlp_regressor_grid.best_score_)
print('Best Parameters : ',mlp_regressor_grid.best_params_)
```

```
Fitting 5 folds for each of 144 candidates, totalling 720 fits
```

```
Train R^2 Score : 0.998
```

```
Test R^2 Score : 0.998
```

```
Best R^2 Score Through Grid Search : 0.997
```

```
Best Parameters : {'activation': 'relu', 'hidden_layer_sizes': (20,), 'learning_rate': 'constant', 'solver': 'lbfgs'}
```

```
CPU times: user 16.2 s, sys: 4.08 s, total: 20.3 s
```

```
Wall time: 23min 17s
```

Devido ao processamento do gridSearch, a arquitetura da rede neural ficou com 1 camada oculta com 20 neurônios cada. Uma camada oculta é o suficiente por se tratar de uma regressão linear. A função de ativação da camada oculta escolhida foi a relu que retorna  $f(x) = \max(0, x)$ . O alfa, que se trata do termo de regularização L2, ficou com 0.001. O hiper parâmetro learning\_rate ficou com uma taxa de aprendizado constante. E o solver ficou como lbfgs que converge mais rapidamente e possui um melhor desempenho. Solver é quem define uma estratégia para a otimização dos pesos. Devido a convergência rápida do modelo, o máximo de iterações da rede neural ficou com 5000 iterações.

```
mlp = MLPRegressor(random_state=123,activation='relu', alpha=0.001, max_iter=5000,hidden_layer_sizes= (20),epsilon=1e-08, verbose=
mlp.fit(x_train, y_train)

MLPRegressor(alpha=0.001, hidden_layer_sizes=20, max_iter=5000,
              random_state=123, solver='lbfgs', verbose=True)

pred = mlp.predict(x_test)

from sklearn.metrics import r2_score

print('acuracia na base de Teste ' + str(metrics.r2_score(y_test.to_numpy(), pred)))
acuracia na base de Teste 0.9961175247267366

from sklearn.metrics import mean_squared_error, r2_score #Métricas para avaliar a regressão

print("Erro quadrático Médio = ",mean_squared_error(y_test, pred)) #Erro quadrático médio
print("R2 = ", r2_score(y_test, pred)) #Coeficiente de determinação

Erro quadrático Médio = 3056147.318657154
R2 = 0.9961175247267366

y_test['valor_predito']=pred

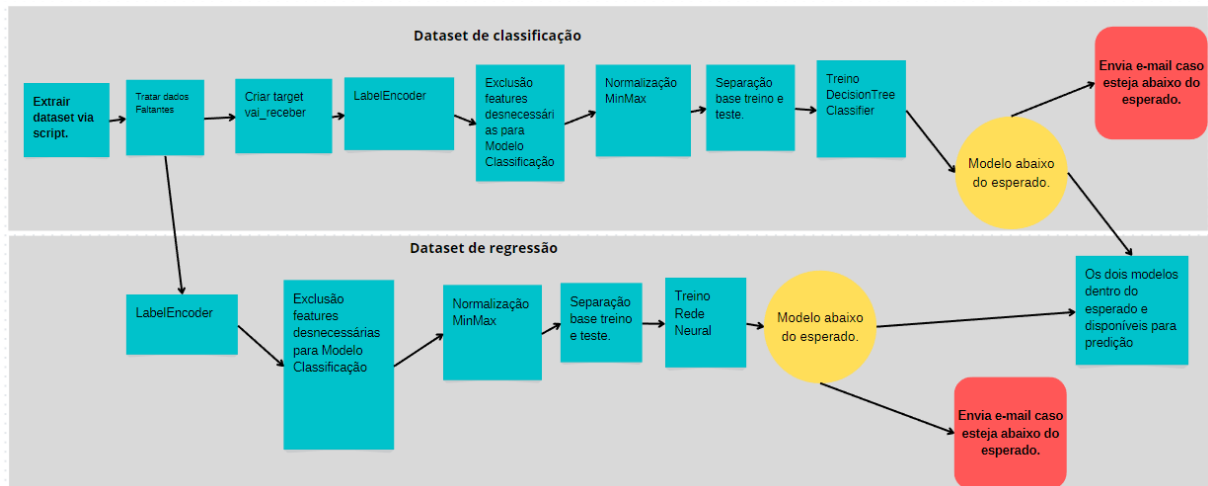
#Com margem de erro de 10%
len(y_test[(10*abs(y_test['valor_predicao_liquido']/100))>=abs(y_test['valor_predicao_liquido']-y_test['valor_predito'])])
1221
```

A rede neural da Sklearn alcançou o melhor resultado. Em um cenário de 3192 registros, 1221 estão dentro da margem de erro de 10%. Com os dados extraídos no trabalho, para o problema de regressão, foi o melhor resultado encontrado.

Ao aumentarmos a margem de erro para 20%, o resultado melhora significativamente chegando a 2002 registros dentro de uma margem de erro de 20%. Isso se refere a 62% dos registros. Se considerarmos o resultado do modelo de classificação usado na etapa anterior e somarmos com o resultado da predição, com uma margem de 20%, o resultado alcançado é de 90%.

Abaixo segue pipeline para treinamento e validação dos modelos. Caso algum dos modelos tenham uma performance abaixo do esperado será enviado um e-mail com as métricas de cada modelo. Caso os dois modelos entreguem um resultado satisfatório os modelos estarão disponíveis para predição.

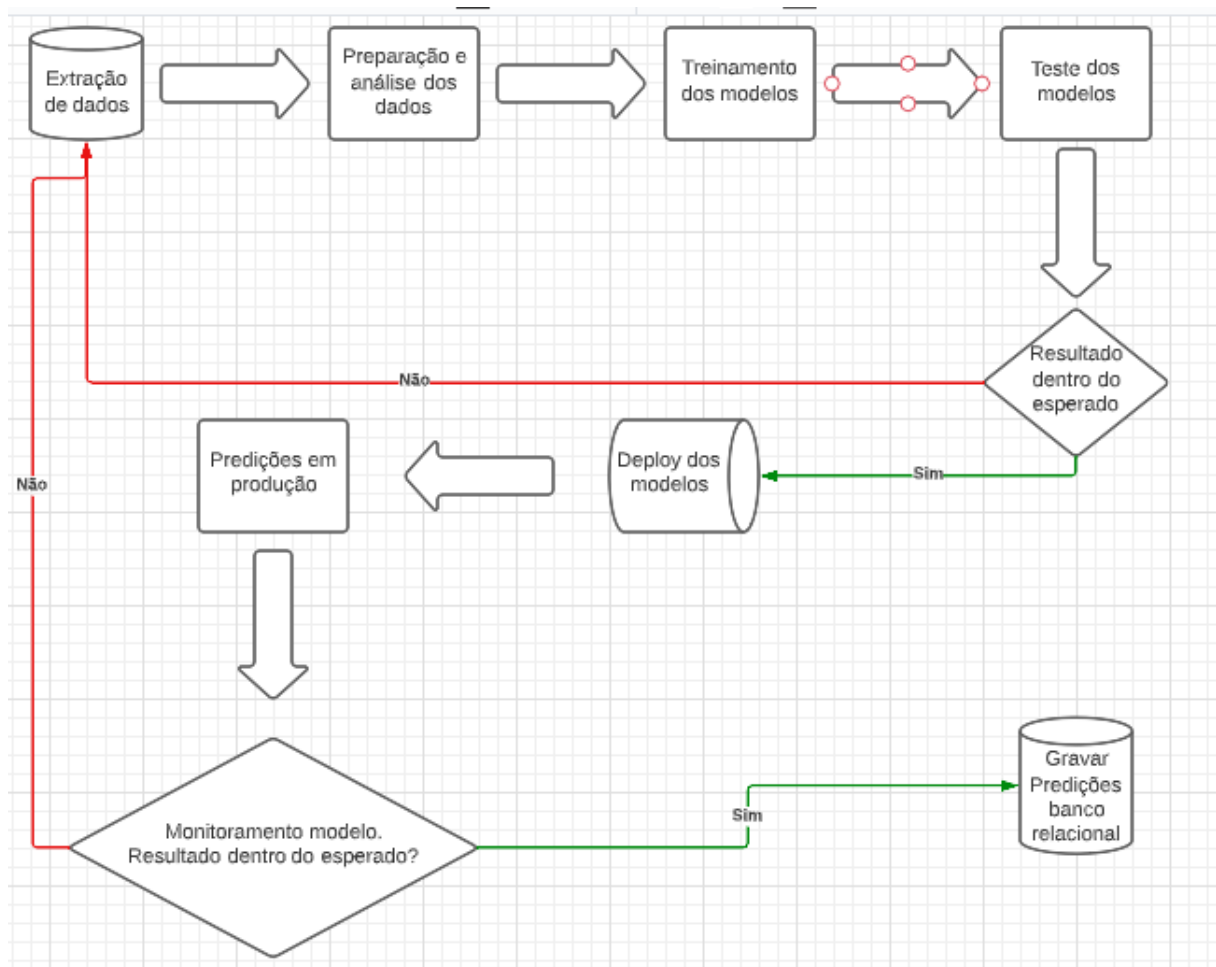
**Figura 17 – Pipeline de execução de trechos de código para criação do modelo.**



Fonte: Imagem criada pelo autor.

Abaixo segue o pipeline de todo o processo de implementação, teste, deploy, e monitoramento dos modelos de machine learning para predição dos valores mensais dos cooperados em uma determinada competência.

**Figura 18 – Pipeline de todo o processo de machine learning.**



Fonte: Imagem criada pelo autor.

## 9. Discussão dos Resultados

Inicialmente foi utilizado uma regressão em todos os registros do dataset e foi observado que os dados com target zerado influenciavam no resultado da regressão. A regressão não trabalha bem com o alvo zerado portanto separamos o problema em duas etapas. O professor e doutor em criptografia pela Universidade Bochun Robert Kubler, chama esse cenário de “The zero-inflated regressor”, ou seja, a regressão inflada a zero. Para a primeira etapa foi criado uma nova coluna vai\_receber para o modelo árvore de decisão classificar quais linhas teriam o vai\_receber setado como 1 ou 0. O resultado nesta etapa foi surpreendentemente positivo já que apenas 398 dos 14070 registros não foram classificados corretamente.

Já a segunda etapa, utilizando um modelo de rede neural, se considerarmos 10% de margem de erro, o resultado não foi satisfatório. 38 por cento dos registros de teste tiveram uma predição dentro da margem de erro de 10%.

Ao elevarmos a margem de erro para 20%, o resultado da segunda etapa melhora consideravelmente. 62 % dos registros do dataset de validação estão dentro de uma margem de erro de 20%.

Analisando as duas etapas em conjunto e considerando uma margem de erro de 10%, alcançamos um resultado de mais de 80 % dos dados testados. E se elevarmos a margem de erro para 20% o resultado alcançado é de 90%.

## 10. Conclusão

Surgiu a necessidade de realizar uma previsão do quanto o médico possa receber em um determinado mês utilizando a base de dados o sistema SASC. Para extração dos dados foi utilizado um script no padrão ANSI do SQL SERVER e a cooperativa escolhida não pode ser mencionada por questão da Lei Geral de Proteção de Dados Pessoais (LGPD), Lei nº 13.709/2018. Nenhum dado pessoal do cooperado também é mencionado no trabalho. A cooperativa se encontra em Belo Horizonte e foi escolhida aleatoriamente.

Devido ao dataset possuir a coluna alvo com muitos registros zerados, o resultado de apenas um modelo de regressão não estava sendo satisfatório. Após uma pesquisa para entender melhor do que se tratava o problema, foi encontrado um artigo escrito pelo doutor em criptografia da Universidade de Bochun, Doutor Robert



Kubler, denominado Zero-Inflated Regression. É salientado pelo autor do artigo que algoritmos de regressão não trabalham bem como dataset inflado a zero, ou seja, dataset com muitos target zerado. Também é mencionado no artigo que a melhor estratégia seria separar o problema em duas etapas. A primeira etapa utiliza um modelo de classificação separando os dados zerados e os não zerados. Na segunda etapa, para os dados cujo o modelo de classificação foram setados como não zerados, serão processados por um modelo de regressão. Os algoritmos escolhidos foram a árvore de decisão e a rede neural da sklearn através de uma comparação entre os modelos árvore de decisão, SVM, Naive Bayes, Regressão Lasso, Regressão Ridge e Rede neural.

Foi encontrado um resultado positivo no trabalho já que foi utilizada uma abordagem pouco ortodoxa para predição, trazendo aplicação de muitos dos conteúdos vistos durante o curso de Especialização em Inteligência Artificial e Aprendizado de Máquina da PUC-Minas.

O objetivo do trabalho era alcançar uma margem de erro de 10% utilizando as duas etapas. Na segunda etapa o resultado não foi satisfatório. Deixando para um próximo trabalho, com mais cooperativas e mais dados, tentar melhorar o resultado da regressão. Se considerarmos uma margem de erro de 20%, o trabalho mostra possuir um resultado melhor alcançando um resultado de 62%. O modelo classificação demonstrou ser de suma importância já que ela quem melhora bastante o resultado deste trabalho. Se considerarmos as duas etapas com uma métrica de 20% dentro da margem de erro, o resultado alcançado é de 90% dos dados testados. Um resultado muito bom servindo mais que um gatilho para implementação de um sistema de predição na FENCOM.

## 11. Links

Termo de autorização de coleta de dados da empresa FENCOM.

<https://github.com/lennonfernandesoliveira/TCC/blob/main/Termo%20libera%C3%A7%C3%A3o%20coleta%20de%20dados.pdf>

Para implementar o código de processamento e tratamento de dados do dataset, foi utilizado o Google COLAB na linguagem Python versão 3.7.13. O código foi disponibilizado no link

<https://github.com/lennonfernandesoliveira/TCC/blob/main/tcc.ipynb>

Para a etapa 6 de análise exploratória, foi utilizado o Google COLAB na linguagem Python versão 3.7.13. O código foi disponibilizado no link

[https://github.com/lennonfernandesoliveira/TCC/blob/main/An%C3%A1lise\\_Exploratoria\\_TCC.ipynb](https://github.com/lennonfernandesoliveira/TCC/blob/main/An%C3%A1lise_Exploratoria_TCC.ipynb)

Para a etapa 7 de preparação dos dados para o modelo de machine learning, foi utilizado o Google COLAB na linguagem Python versão 3.7.13. O código foi disponibilizado no link

[https://github.com/lennonfernandesoliveira/TCC/blob/main/preparacao\\_dados\\_para\\_predicao\\_machine\\_learning.ipynb](https://github.com/lennonfernandesoliveira/TCC/blob/main/preparacao_dados_para_predicao_machine_learning.ipynb)

Para a etapa 8, aplicação de modelos, foi criado um colab para comparação de modelos de classificação e modelos de regressão.

[https://github.com/lennonfernandesoliveira/TCC/blob/main/comparacao\\_modelo.ipynb](https://github.com/lennonfernandesoliveira/TCC/blob/main/comparacao_modelo.ipynb)

Link da dataml contendo implementação do RFE.

<https://dataml.com.br/como-escolher-as-melhores-features-para-um-modelo-de-aprendizado-de-maquina/>

O dataset está disponibilizado no link

<https://github.com/lennonfernandesoliveira/TCC/blob/main/bdPredicao.xlsx>

GitGub contendo implementação de teste de hipótese para médias em duas amostras independentes.

<https://github.com/cibelerusso/AnaliseMultivariadaEAprendizadoNaoSupervisionado/blob/master/Comandos%20em%20Python/5b%20Testes%20de%20Hip%C3%B3teses.ipynb>

Link do artigo do Dr. Robert Kubler

<https://towardsdatascience.com/zero-inflated-regression-c7dfc656d8af>