PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
NÚCLEO DE EDUCAÇÃO A DISTÂNCIA
Pós-graduação *Lato Sensu* em Ciência de Dados e Big Data

Fabrício Almeida da Silva

ANÁLISE E PREVISÃO DE AUTORIZAÇÕES UTILIZANDO MODELOS PREDITIVOS

Belo Horizonte
2021

Fabrício Almeida da Silva

ANÁLISE E PREVISÃO DE AUTORIZAÇÕES UTILIZANDO MODELOS PREDITIVOS

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Belo Horizonte
2021

SUMÁRIO

1. Introdução

1.1. Contextualização

A autorização é um dos diversos mecanismos utilizados pelas operadoras de planos de saúde visando controlar o uso ou a demanda de serviços prestados aos beneficiários. Nesse cenário, uma autorização prévia para realizar exames e processos de alta complexidade é um dos mecanismos das operadoras para gerenciar a utilização dos seus serviços.

Outras maneiras de controle incluem o direcionamento, (que encaminha exames, internações e consultas para a rede credenciada) e a coparticipação (em que o cliente paga uma parcela da despesa do seu procedimento). Há ainda a franquia, na qual um valor máximo para procedimentos é definido em contrato. Se o procedimento extrapolar esse teto, o plano de saúde não tem responsabilidade sobre a cobertura.

Ao controlar melhor as demandas dos serviços prestados aos seus beneficiários, as operadoras conseguem promover um uso mais consciente de seus benefícios. Dessa forma, é possível evitar o desperdício de recursos e a sobrecarga no sistema.

É importante lembrar, ainda que, a autorização no plano de saúde e outros mecanismos de regulação devem ser primeiramente analisados, a Agência Nacional de Saúde Suplementar (ANS), que é a agência reguladora vinculada ao Ministério da Saúde do Brasil, que regula o mercado de planos privados de saúde por determinação da Lei nº 9.656 de 3 de junho de 1998. A condição do órgão para a liberação desses mecanismos é que eles não impeçam ou dificultem o acesso dos usuários aos procedimentos que estão presentes no contrato do plano.

O processo para solicitar a autorização costuma variar de acordo com o plano e o tipo do procedimento. A autorização para exames, internações e serviços específicos costuma envolver a ida até a rede credenciada mais próxima com a guia

ou pedido médico do exame/procedimento, o cartão de identificação do plano e um documento pessoal (como o RG). Algumas operadoras também trabalham com solicitações via internet.

1.2. O problema proposto

O projeto tem como objetivo propor um modelo preditivo para previsão das situações das autorizações solicitas a operadora de plano de saúde, utilizando informações das autorizações já analisadas pelos auditores.

- Justificativa (Por quê?)

O processo atual de autorização de procedimentos ocorre da seguinte forma. Alguns procedimentos realizados pelos beneficiários sejam eles consultas, exames ou cirurgias, geram pedidos de autorização junto a operadora. Esse pedido é chamado guia de autorização, essas guias são recebidas de forma eletrônica e na forma física. Alguns desses procedimentos são autorizados de forma automática pois no sistema de gestão da operadora existem regras configuradas que determinam se o procedimento vai ser autorizado automaticamente ou se será necessário a validação de um auditor.

O auditor é o médico responsável por analisar a guia e decidir se será autorizada ou negada. Para tomar essa decisão o médico usa de várias informações como, histórico médico do paciente, informações sobre os tipos de serviços, os valores dos itens solicitados e resoluções da ANS.

Todo o processo de auditoria exige precisão e agilidade, pois são procedimentos que podem primeiramente salvam vidas, e que também afetam diretamente a saúde financeira da operadora caso a análise não seja feita da forma correta.

Como o volume de solicitações é elevado e carecem de uma análise profunda de auditoria, o processo geralmente é caro e pouco assertivo. Diante disso propõe-se um modelo preditivo baseado no histórico de auditoria capaz prever se a solicitação será autorizada ou negada.

- Fontes de dados (Quem?)

Os dados utilizados são privados e vieram de uma fonte principal que é o data warehouse (DW) da operadora, onde se encontra centralizado informações de diversos sistemas que são utilizados pela empresa.

- Objetivo da análise (O que?)

O objetivo é analisar as características das autorizações auditadas e propor um modelo preditivo usando seus principais atributos para classificá-las em autorizada ou negada.

- Aspectos geográficos (Onde?)

Para análise foram consideradas as solicitações dos beneficiários de todo Brasil.

- Período analisado (Quando?)

O período das solicitações consideradas na análise é definido pela data do atendimento do beneficiário, e foi extraído o período que vai de janeiro a junho de 2021.

2. Coleta de Dados

Os dados utilizados vieram do data warehouse (DW) da operadora, onde se encontra centralizado informações de diversos sistemas utilizados pela empresa. Foram extraídos os datasets guias_autorizadas_01-2021_06-2021.csv e localizacao_operadoras.csv, todos no formato csv.

Para a construção dos indicadores e modelos, foram utilizadas os datasets com a estrutura inicial conforme o quadro abaixo:

Dataset: **guias_autorizadas_01-2021_06-2021.csv**

Coluna	Descrição	Tipo
codigo_servico	Código do serviço solicitado na guia	String
situacao	Situação da autorização se está autorizada ou negada	String
tipo	Identifica se a autorização foi solicitada eletronica-mente ou fisicamente na operadora	String
origem	Identifica se é local ou intercambio	String
faturado	Identifica se o serviço foi faturado para pagamento ou cobrança	String
codigo	Código da transação no sistema	Inteiro
auditoria	Identifica se houve auditoria da autorização	String
origem_beneficiario	Código da operadora do beneficiário	String
codigo_executante	Código do prestador que irá realizar o procedimento	Inteiro
codigo_requisitante	Código do prestador que solicitou o procedimento	Inteiro
especialidade_executante	Código da especialidade do prestador que vai realizar o serviço	Inteiro
especialidade_requisi-tante	Código da especialidade do prestador que solicitou o serviço	Inteiro
plano_beneficiario	Código do plano do beneficiário	Inteiro
carater_atendimento	Identifica se é um atendimento de urgência ou eletivo	String
tipo_consulta	Identifica o tipo de serviço a ser autorizado	String

Dataset: **localizacao_operadoras.csv**

Coluna	Descrição	Tipo
codigo	Código do prestador que solicitou o procedimento	Inteiro
municipio	Nome do município	String
estado	Nome do estado	String
pais	Nome do país	String

3. Processamento/Tratamento de Dados

O trabalho foi realizado utilizando a linguagem Python em sua versão 3.9.5, que é uma linguagem de programação interpretada, orientada a objetos de alto nível. O ambiente de desenvolvimento utilizado foi o Jupyter Notebook, em sua versão 6.2.0 (figura 1). O Jupyter Notebook é um aplicativo web de código aberto que permite criar e compartilhar documentos que contêm código ativo, equações, visualizações e texto narrativo.

Figura 1

Foram utilizadas diversas bibliotecas em todas as etapas do trabalho (figura 2).

```
#importando bibliotecas necessarias
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from imblearn.over_sampling import SMOTE
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from yellowbrick.classifier import ConfusionMatrix
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Figura 2

O dataset **guias_autorizadas_01-2021_06-2021.csv** originalmente tem 11951 linhas e 15 colunas (figura 3). Já o dataset **localização_operadoras.csv** tem 11951 linhas e 4 colunas (figura 4).

```
#importando o dataset para analise
df_guias = pd.read_csv('guias_autorizadas_01-2021_06-2021.csv', sep= ';', encoding='UTF-8')

#visualizando a quantidade de linhas e colunas do dataset
print("Numero de Linhas =",df_guias.shape[0])
print("Numero de Colunas =",df_guias.shape[1])

Numero de Linhas = 11951
Numero de Colunas = 15
```

Figura 3

```
#importando o dataset para analise
df_local = pd.read_csv('localizacao_operadoras.csv', sep= ';', encoding='UTF-8')

#visualizando a quantidade de linhas e colunas do dataset
print("Numero de Linhas =",df_local.shape[0])
print("Numero de Colunas =",df_local.shape[1])

Numero de Linhas = 11951
Numero de Colunas = 4
```

Figura 4

Inicialmente foi verificado se haveria valores nulos nos datasets e não foi encontrado nenhum valor nulo (figura 5 e 6). Como os dados foram extraídos de um DW, onde são transformados no processo de ETL (Extract Transform Load) antes de serem inseridos no banco, é esperado uma melhor qualidade dos dados.

```
#verificando se existem campos nulos
df_guias.isnull().sum()
```

```
codigo_servico               0
situacao                     0
tipo                         0
origem                       0
faturado                     0
codigo                       0
auditoria                    0
origem_beneficiario          0
codigo_executante            0
codigo_requisitante          0
especialidade_executante     0
especialidade_requisitante   0
plano_beneficiario           0
carater_atendimento          0
tipo_consulta                0
dtype: int64
```

Figura 5

```
#verificando se existem campos nulos
df_local.isnull().sum()
```

```
codigo       0
municipio    0
estado       0
pais         0
dtype: int64
```

Figura 6

Foram removidas as colunas, **faturado** e **auditoria** do dataframe (figura 7) **df_guias** (variável que recebeu o dataset guias_autorizadas_01-2021_06-2021.csv), a coluna faturado apenas mostra se a guia auditada foi enviada ou não faturamento o que não tem peso algum na auditoria da guia, já a coluna auditoria mostra se nessa guia houve a auditoria do médico, como todas as guias extraídas foram auditadas, essa informação é irrelevante para as análises. Foram renomeadas algumas colunas para melhor análise (figura 7). Na coluna **cd_servico** foi retirado os pontos e traços e foi feita a sua conversão do tipo String para o tipo inteiro (figura 8). Foi feita uma adequação na coluna **tipo_atendimento** para que os campos tivessem um melhor entendimento (figura 9).

```
#remoção de algumas colunas
df_guias = df_guias.drop(columns=['faturado','auditoria'])

#visualizando a quantidade de linhas e colunas do dataset apos a exclusão das colunas
print("Numero de Linhas =",df_guias.shape[0])
print("Numero de Colunas =",df_guias.shape[1])

Numero de Linhas = 11951
Numero de Colunas = 13

#renomeando algumas colunas
df_guias = df_guias.rename(columns={'codigo_servico': 'cd_servico','codigo':'cd_guia','codigo_executante': 'cd_executante','codig
```

Figura 7

```
#removendo pontos e traços da coluna
df_guias['cd_servico'] = df_guias['cd_servico'].apply(lambda caracter: caracter.replace('.','').replace('-',''))

#convertendo a coluna para o tipo inteiro
df_guias['cd_servico'] = df_guias['cd_servico'].astype(int)
```

Figura 8

```
#visualizando informações
df_guias['tipo_atendimento'].value_counts()

ELETIVA/PRIMEIRA CONSULTA    10317
URGENCIA/EMERGENCIA           1634
Name: tipo_atendimento, dtype: int64

#alterando os conteudos da coluna
df_guias['tipo_atendimento'] = df_guias['tipo_atendimento'].apply(lambda caracter: caracter.replace('ELETIVA/PRIMEIRA CONSULTA','

#conteudo apos a alteração
df_guias['tipo_atendimento'].value_counts()

ELETIVA     10317
URGENCIA     1634
Name: tipo_atendimento, dtype: int64
```

Figura 9

Na coluna **situacao** existiam seis situações: **efetivada**, **cancelada**, **negada**, **parcialmente autorizada** e **em auditoria**. Dessa forma foi necessário adequar algumas linhas e excluir outras que não eram interessantes para o projeto. Foi feita a exclusão da das situações **cancelada**, **parcialmente autorizada** e **em auditoria**, porque para criação do modelo foi pensado em utilizar apenas as guias em que sua situação tem status de autorizada ou negada. Desse modo a situação **efetivada** foi transformada em **autorizada** devida a mesma ter esse significado no sistema de gestão da operadora (figura 10).

```
#visualizando informações
df_guias['situacao'].value_counts()
```

```
EFETIVADA                 8034
CANCELADA                 1782
NEGADA                    1361
PARCIALMENTE AUTORIZADA    690
AUTORIZADA                  67
EM AUDITORIA                17
Name: situacao, dtype: int64
```

```
#nessa coluna temos 6 tipos de situacao, precisamos das linhas que tenham apenas as situações AUTORIZADA E NEGADA.
#iremos manter a situacao EFETIVADA pois a mesma equivale a AUTORIZADA no dataset original

#removendo as linhas com as situações: CANCELADA,PARCIALMENTE AUTORIZADA e EM AUDITORIA
df_guias = df_guias.drop(df_guias.loc[df_guias['situacao'] == 'CANCELADA'].index)
df_guias = df_guias.drop(df_guias.loc[df_guias['situacao'] == 'PARCIALMENTE AUTORIZADA'].index)
df_guias = df_guias.drop(df_guias.loc[df_guias['situacao'] == 'EM AUDITORIA'].index)
```

```
#visualizando informações
df_guias['situacao'].value_counts()
```

```
EFETIVADA    8034
NEGADA       1361
AUTORIZADA     67
Name: situacao, dtype: int64
```

```
#renomeando a situacao EFETIVADA para AUTORIZADA
df_guias['situacao'] = df_guias['situacao'].apply(lambda x: x.replace('EFETIVADA','AUTORIZADA'))
```

```
#visualizando informações
df_guias['situacao'].value_counts()
```

```
AUTORIZADA    8101
NEGADA        1361
Name: situacao, dtype: int64
```

Figura 10

No dataframe **df_local** (variável que recebeu o dataset localizacao_operadoras.csv) foi renomeado a coluna **código** para **cd_operadora** (figura 11).

```
#renomeando coluna
df_local = df_local.rename(columns={'codigo': 'cd_operadora'})
```

```
#visualizando a quantidade de linhas e colunas do dataset apos a exclusão das colunas
print("Numero de Linhas =",df_local.shape[0])
print("Numero de Colunas =",df_local.shape[1])
```

```
Numero de Linhas = 11951
Numero de Colunas = 4
```

```
#visualizando as 5 primeiras linhas do dataframe apos alterações
df_local.head()
```

	cd_operadora	municipio	estado	pais
0	178	NOVA OLINDA DO NORTE	AMAZONAS	BRASIL
1	173	MANAUS	AMAZONAS	BRASIL
2	202	FERREIRA GOMES	AMAPA	BRASIL
3	202	FERREIRA GOMES	AMAPA	BRASIL
4	179	NOVO AIRAO	AMAZONAS	BRASIL

Figura 11

Após a etapa de adequação dos dois dataframes, foi efetuado o processo de união dos dois em apenas um, o novo dataframe criado foi chamado de **df_autorizacao** (figura 12)

Unindo os dois datasets

```
#visualizando a quantidade de linhas e colunas do dataset antes da junção
print("Numero de Linhas =",df_guias.shape[0])
print("Numero de Colunas =",df_guias.shape[1])
```

```
Numero de Linhas = 9462
Numero de Colunas = 13
```

```
#visualizando a quantidade de linhas e colunas do dataset antes da junção
print("Numero de Linhas =",df_local.shape[0])
print("Numero de Colunas =",df_local.shape[1])
```

```
Numero de Linhas = 11951
Numero de Colunas = 4
```

```
#unindo os dois datasets usando join
df_autorizacao = df_guias.join(df_local,lsuffix='cd_requisitante',rsuffix='cd_operadora',how='left')
df_autorizacao.shape
```

```
(9462, 17)
```

```
#visualizando a quantidade de linhas e colunas do dataset apos a junção
print("Numero de Linhas =",df_autorizacao.shape[0])
print("Numero de Colunas =",df_autorizacao.shape[1])
```

```
Numero de Linhas = 9462
Numero de Colunas = 17
```

Figura 12

4. Análise e Exploração dos Dados

Foi iniciada a exploração dos dados visualizando algumas informações estatísticas do dataframe df_autorizacao (figura 13).

```
#verificando algumas estatisticas dos atributos numericos
df_autorizacao.describe().transpose()
```

	count	mean	std	min	25%	50%	75%	max
cd_servico	9462.0	4.284577e+07	1.988035e+07	1244.0	40101010.00	40316548.0	50000470.0	99999943.0
cd_guia	9462.0	6.211932e+06	6.703088e+06	5313592.0	5429088.00	5573452.0	5740870.0	77473972.0
cd_executante	9462.0	3.375014e+02	2.579790e+02	6.0	198.00	229.0	286.0	988.0
cd_requisitante	9462.0	3.381822e+02	2.584196e+02	6.0	199.25	229.0	286.0	988.0
esp_executante	9462.0	2.780397e+02	2.733118e+00	278.0	278.00	278.0	278.0	466.0
esp_requisitante	9462.0	2.780397e+02	2.733118e+00	278.0	278.00	278.0	278.0	466.0
plano_beneficiario	9462.0	5.725736e+02	6.983521e+01	2.0	538.00	569.0	622.0	627.0
cd_operadora	9462.0	3.381822e+02	2.584196e+02	6.0	199.25	229.0	286.0	988.0

Figura 13

Foram feitas análises para tentar responder algumas perguntas como:

- Quantidade de autorizações solicitadas por estado (figura 14)
- Quais os 10 municípios que mais solicitaram autorizações (figura 15)
- Quais as 10 operadoras de planos de saúde que mais solicitaram autorizações (figura 16)
- Quais os 10 planos que mais solicitaram autorizações (figura 17)
- Quais os 10 serviços mais requisitados, qual a quantidade de autorizações por tipo de atendimento e situação. (figura 18,19 e 20)

Como pode ser visto todas as solicitações do dataset tem a origem de seus requisitantes em apenas 5 estados do Brasil, Bahia, Amazonas, Amapá, Alagoas e Acre.

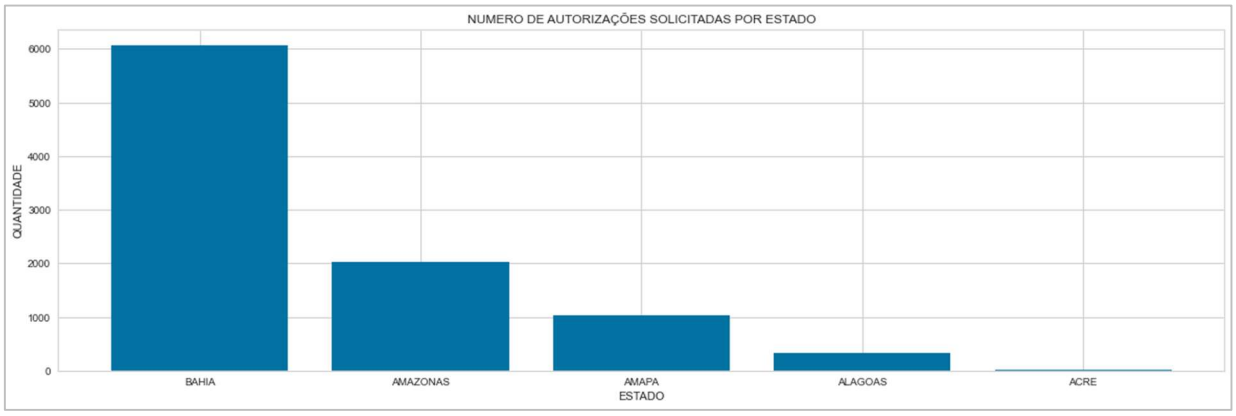

Figura 14

Pode se observar os 10 municípios onde os requisitantes mais fizeram solicitações. Destaque para a cidade de Ipiaú localizada no estado da Bahia.

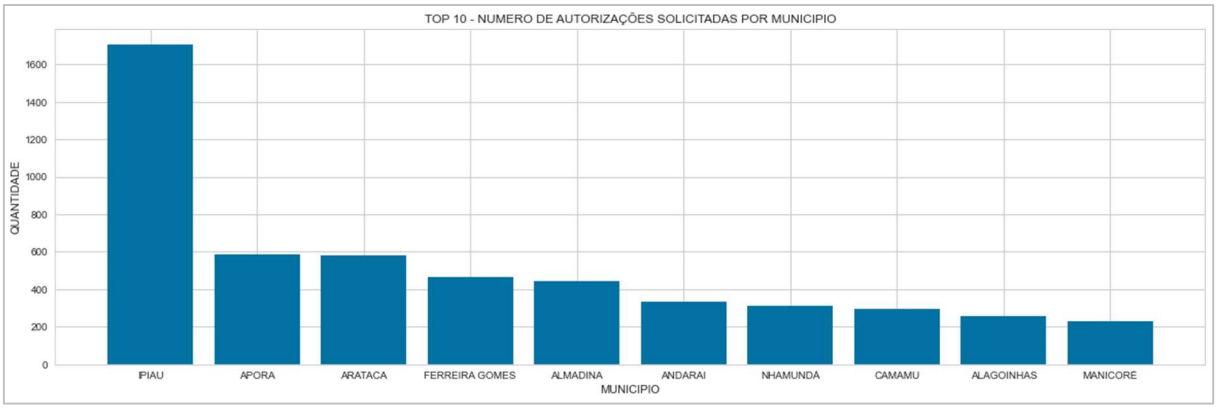

Figura 15

Aqui pode ser ver os números de solicitações de autorização requisitados, com o destaque para as 10 operadoras de planos de saúde que mais fizeram requisições.

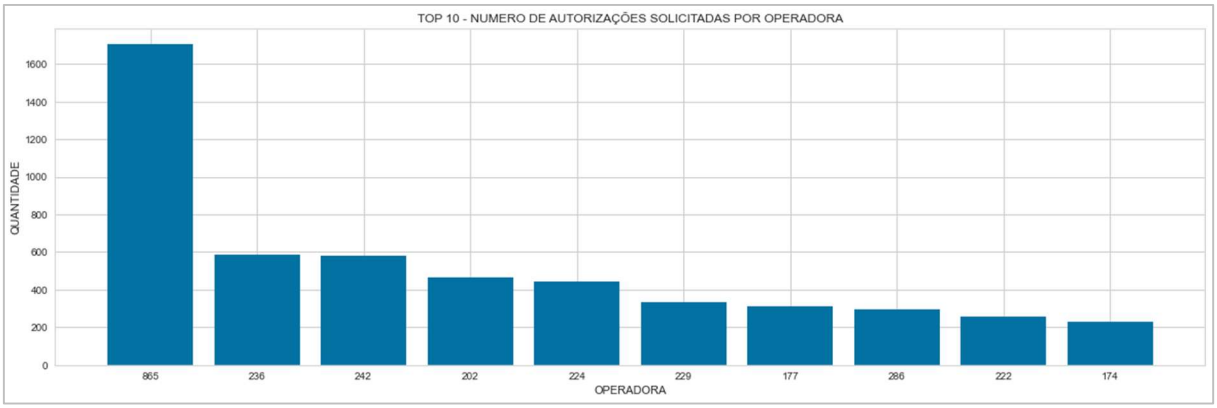

Figura 16

Esse gráfico mostra quais são os 10 planos de saúde dos beneficiários que mais fizeram solicitações de autorização.

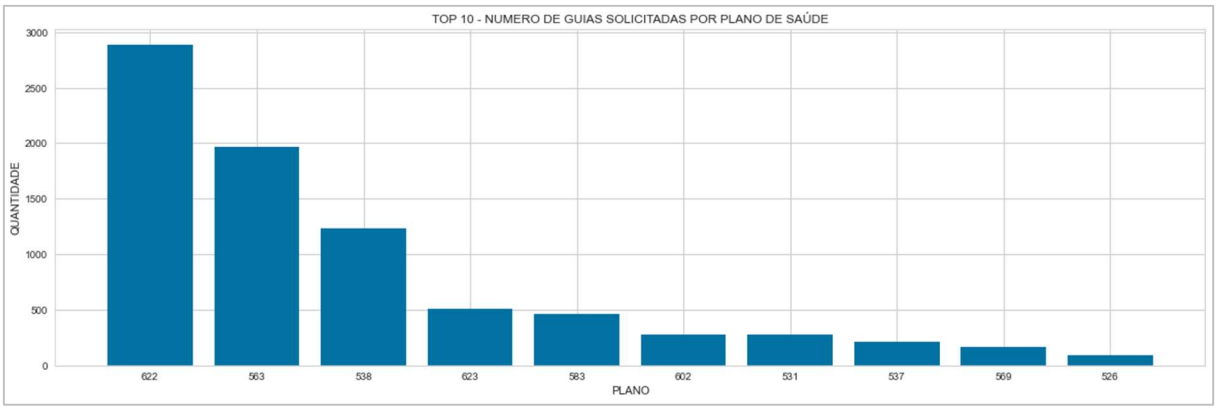

Figura 17

O gráfico abaixo mostra quais foram os 10 serviços mais solicitados em guias de autorização, em destaque o serviço 50000470 que tem uma requisição muito superior aos demais.

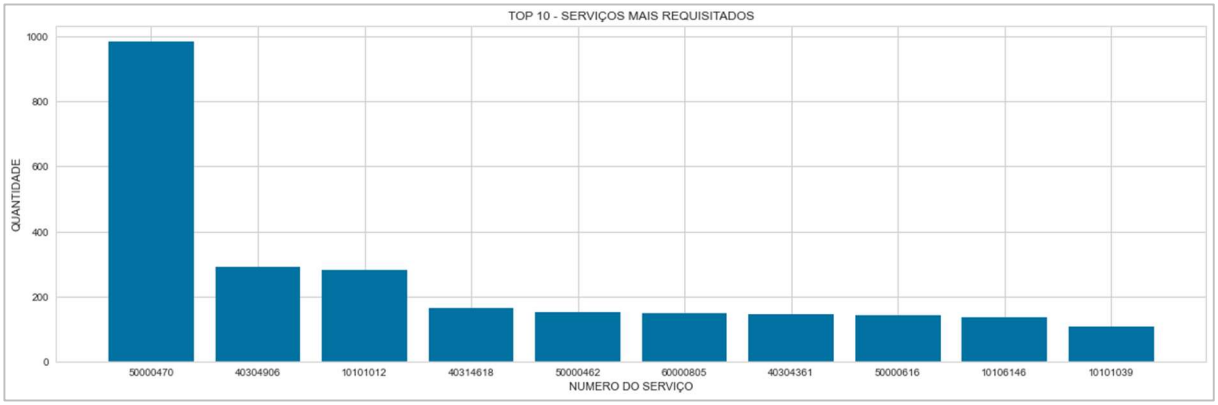

Figura 18

O gráfico mostra quais os tipos de atendimentos que mais foram solicitados através das autorizações. Fica bem claro que os atendimentos de caráter eletivos, que são atendimentos que são planejados com antecedência são a maioria das solicitações, já casos de urgência que são atendimentos de realização imediata tiveram uma parcela significativamente menor.

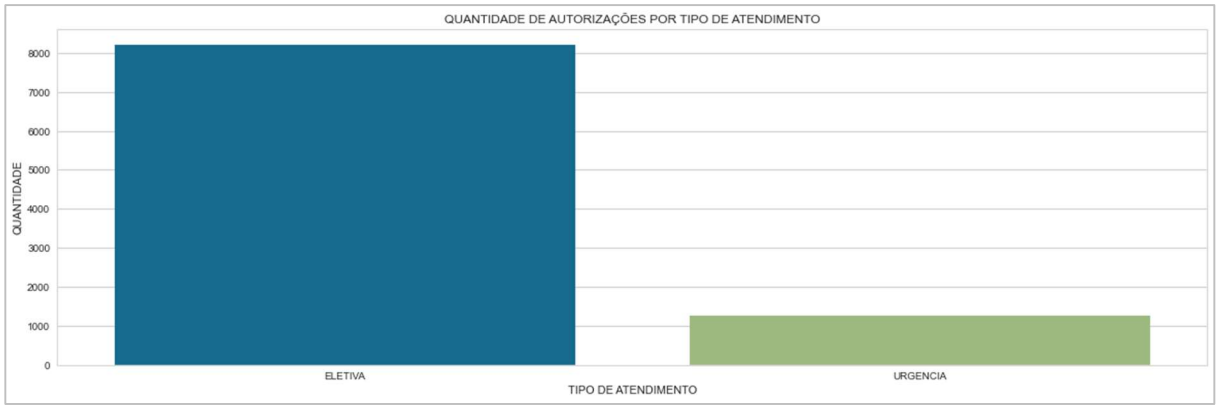

Figura 19

Esse último gráfico mostra que a maior parte das solicitações de autorização foram autorizadas, e como esse campo é a classe que vai tentar se prevista pelo modelo podemos sofrer algum tipo de desbalanceamento mais tarde.

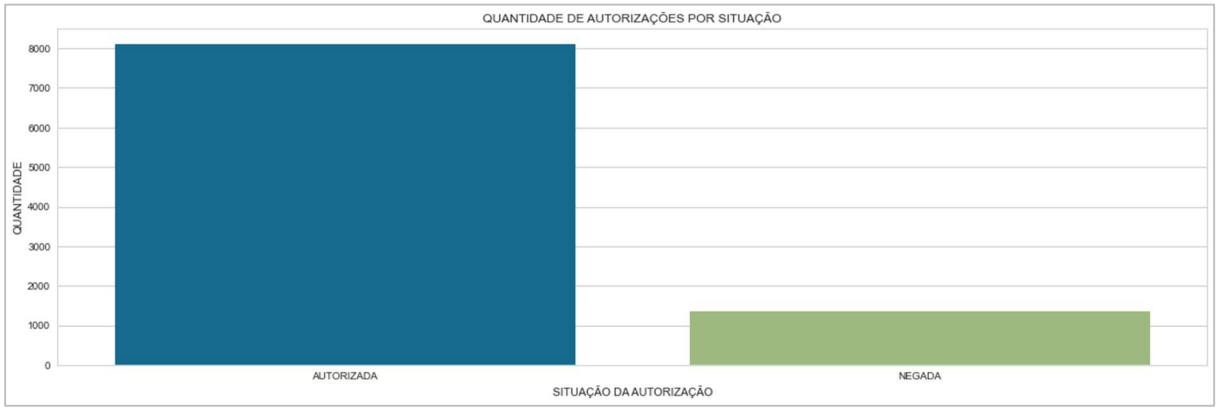

Figura 20

Abaixo pode ser visto a matriz de correlação dos atributos do dataframe df_autorização (figura 21).

Matriz de correlação dos atributos

```
#verificando a correlação dos atributos
corr = df_autorizacao.corr(method='pearson')
corr.style.background_gradient().set_precision(4).set_properties(**{'font-size': '10pt'})
```

	cd_guia	cd_servico	cd_executante	cd_requisitante	esp_executante	esp_requisitante	plano_beneficiario	cd_operadora	situacao
cd_guia	1.0000	0.0653	-0.1010	-0.1011	0.1545	0.1545	-0.0030	-0.1011	-0.0130
cd_servico	0.0653	1.0000	-0.0312	-0.0314	0.0167	0.0167	0.0108	-0.0314	0.0855
cd_executante	-0.1010	-0.0312	1.0000	0.9974	-0.0187	-0.0187	0.1662	0.9974	-0.0425
cd_requisitante	-0.1011	-0.0314	0.9974	1.0000	-0.0187	-0.0187	0.1644	1.0000	-0.0427
esp_executante	0.1545	0.0167	-0.0187	-0.0187	1.0000	1.0000	-0.0522	-0.0187	0.0060
esp_requisitante	0.1545	0.0167	-0.0187	-0.0187	1.0000	1.0000	-0.0522	-0.0187	0.0060
plano_beneficiario	-0.0030	0.0108	0.1662	0.1644	-0.0522	-0.0522	1.0000	0.1644	0.0498
cd_operadora	-0.1011	-0.0314	0.9974	1.0000	-0.0187	-0.0187	0.1644	1.0000	-0.0427
situacao	-0.0130	0.0855	-0.0425	-0.0427	0.0060	0.0060	0.0498	-0.0427	1.0000

Figura 21

5. Criação de Modelos de Machine Learning

Para estre trabalho foram escolhidos três modelos de Machine Learning para treinamento e comparação de seus resultados:

- **Naive Bayes**
- **Decision Tree Classifier**: Árvore de Decisão
- **Random Forest Classifier**: Floresta Aleatória

Naive Bayes é um classificador probabilístico muito utilizado em Machine Learning. Baseado no “Teorema de Bayes”, o modelo foi criado por um matemático inglês, e ministro presbiteriano, chamado Thomas Bayes (1701 – 1761) para tentar provar a existência de Deus.

Hoje é também utilizado na área de Aprendizado de Máquina (Machine Learning) para categorizar textos com base na frequência das palavras usadas.

Entre as possibilidades de aplicações está a classificação de um e-mail como SPAM ou Não-SPAM e a identificação de um assunto com base em seu conteúdo.

Ele recebe o nome de “Naive” (ingênuo) porque desconsidera a correlação entre as variáveis (features). Ou seja, se determinada fruta é rotulada como “Limão”, caso ela também seja descrita como “Verde” e “Redonda”, o algoritmo não vai levar em consideração a correlação entre esses fatores. Isso porque trata cada um de forma independente.

Árvores de Decisão é uma tabela de decisão sob a forma de árvore com nós e folhas sequenciais e interligados que classificam as instâncias, ordenando-as com base nos valores dos recursos. Cada nó em uma árvore de decisão representa uma característica em uma instância a ser classificada, e cada ramo representa um valor que o nó pode assumir. As instâncias são classificadas começando no nó raiz com

base em seus valores de recursos. Trata-se de um dos modelos mais práticos e mais utilizados em inferência por indução.

Random Forest Classifier, ou Floresta Aleatória, é um ensemble, um conjunto de Árvores de Decisão que, apesar da sua simplicidade, é um dos mais poderosos algoritmos de aprendizado de máquina disponíveis atualmente. Baseado na chamada “sabedoria das multidões”, em que a resposta agregada de milhares de pessoas aleatórias a uma pergunta complexa é melhor do que a resposta de um especialista.

Na prática, significa treinar um conjunto de classificadores de árvores de decisão, cada um em um subconjunto aleatório diferente do conjunto de treinamento, e fazer as previsões, obtendo-as de todas as árvores individuais, e, então, prever a classe que obtém a maioria dos votos.

Na preparação para deixar o dataframe da melhor forma para o treinamento dos modelos, foi feita as seguintes alterações e procedimentos:

Reordenação das colunas para melhor organização dos campos e para facilitar posteriormente a separação entreve previsores e a classe (figura 22).

Reordenando as colunas

```
#reordenando as colunas
df_autorizacao = df_autorizacao[['cd_guia','cd_servico','tipo','origem','origem_beneficiario','cd_executante','cd_requisitante',
df_autorizacao.head()
```

	cd_guia	cd_servico	tipo	origem	origem_beneficiario	cd_executante	cd_requisitante	esp_executante	esp_requisitante	plano_beneficiario	t
0	5672237	1244	TRANSITO INTERNACAO	TRANSACAO	LOCAL	178	178	278	278	563	
2	5332998	2500	TRANSITO INTERNACAO	TRANSACAO	LOCAL	202	202	278	278	538	
3	5332998	9350	TRANSITO INTERNACAO	TRANSACAO	LOCAL	202	202	278	278	538	
7	5404770	22403	TRANSITO INTERNACAO	TRANSACAO	LOCAL	282	282	278	278	583	
8	5654489	24554	TRANSITO SERVICO	TRANSACAO	LOCAL	236	236	278	278	537	

Figura 22

Como modelos tendem a trabalhar melhor com números ao invés de strings, foi realizado alguns tratamentos nas strings do dataframe:

Na coluna situacao foram convertidas as linhas com situação **Autorizada** para o número **1** e **Negada** para o número **0** (figura 23).

Transformando a atributo situacao em codificação numerica

```
df_autorizacao['situacao'] = df_autorizacao['situacao'].map({'AUTORIZADA':1, 'NEGADA':0})
```

Figura 23

Nas demais strings do dataframe foi efetuado o tratamento utilizando a função **LabelEncoder** que irá converter os valores que são categóricos em valores numéricos (figura 24).

Metodo Label Enconder

```
from sklearn.preprocessing import LabelEncoder
label_enconder = LabelEncoder()
```

```
label_encoder_tipo = LabelEncoder()
label_encoder_origem = LabelEncoder()
label_encoder_origem_beneficiario = LabelEncoder()
label_encoder_tipo_atendimento = LabelEncoder()
label_encoder_tipo_consulta = LabelEncoder()
label_encoder_municipio = LabelEncoder()
label_encoder_estado = LabelEncoder()
label_encoder_pais = LabelEncoder()
```

```
X_autorizacao[:,2] = label_encoder_tipo.fit_transform(X_autorizacao[:,2])
X_autorizacao[:,3] = label_encoder_origem.fit_transform(X_autorizacao[:,3])
X_autorizacao[:,4] = label_encoder_origem_beneficiario.fit_transform(X_autorizacao[:,4])
X_autorizacao[:,10] = label_encoder_tipo_atendimento.fit_transform(X_autorizacao[:,10])
X_autorizacao[:,11] = label_encoder_tipo_consulta.fit_transform(X_autorizacao[:,11])
X_autorizacao[:,13] = label_encoder_municipio.fit_transform(X_autorizacao[:,13])
X_autorizacao[:,14] = label_encoder_estado.fit_transform(X_autorizacao[:,14])
X_autorizacao[:,15] = label_encoder_pais.fit_transform(X_autorizacao[:,15])
```

```
X_autorizacao
```

```
array([[5672237, 1244, 4, ..., 115, 3, 0],
       [5332998, 2500, 4, ..., 84, 2, 0],
       [5332998, 9350, 4, ..., 84, 2, 0],
       ...,
       [5750255, 99999943, 4, ..., 93, 4, 0],
       [5790953, 99999943, 4, ..., 93, 4, 0],
       [77419622, 99999943, 2, ..., 75, 0, 0]], dtype=object)
```

Figura 24

A próxima etapa foi dividir os dados em treinamento de teste utilizando a função **train_test_split** da biblioteca **sklearn,** foi utilizado 75% para treino e 25% para teste (figura 25).

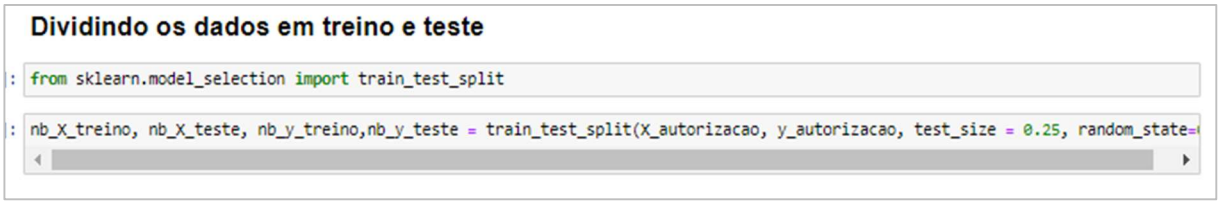

Figura 25

Depois de toda a preparação foi iniciado treinamento dos modelos, em ordem Naive Bayes, Decision Tree e Random Forest, todos instanciados com configurações padrão.

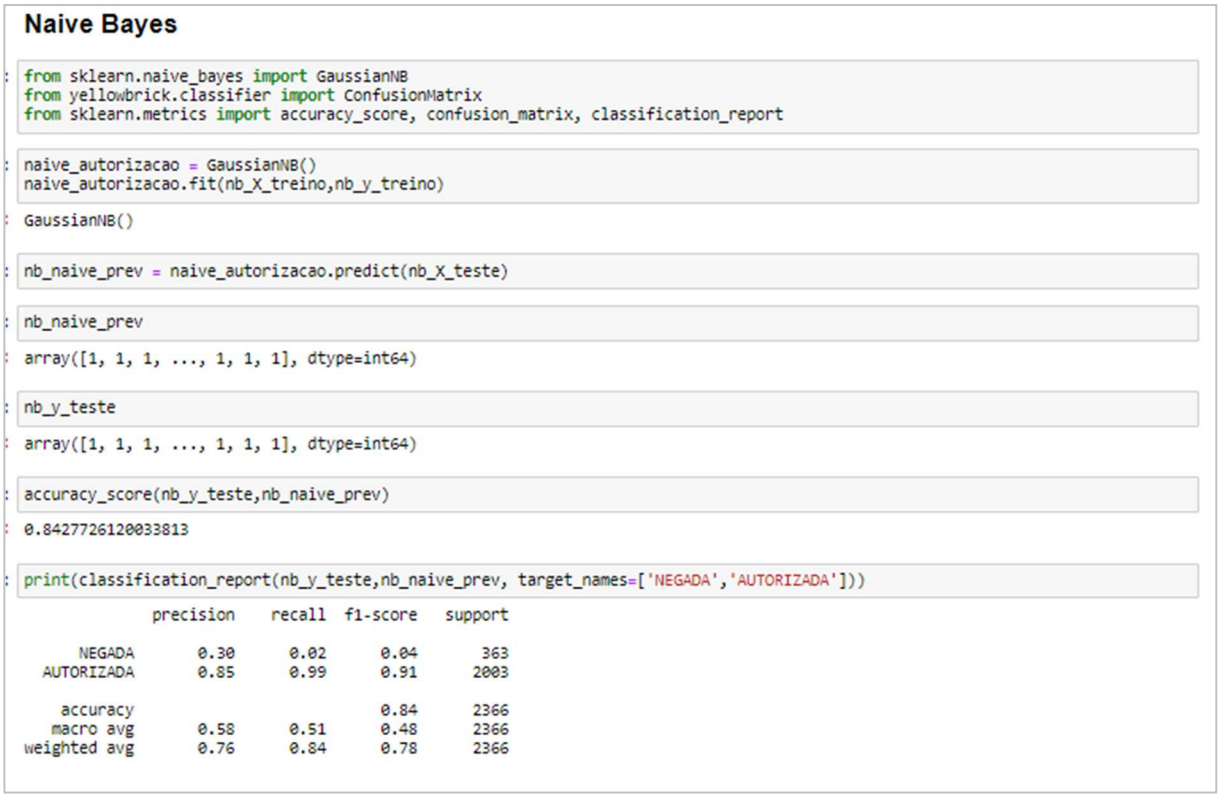

Figura 26

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

```
arvore_autorizacao = DecisionTreeClassifier(random_state=0)
```

```
arvore_autorizacao.fit(nb_X_treino,nb_y_treino)
```

```
DecisionTreeClassifier(random_state=0)
```

```
nb_arvore_prev = arvore_autorizacao.predict(nb_X_teste)
```

```
nb_arvore_prev
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

```
nb_y_teste
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

```
accuracy_score(nb_y_teste,nb_arvore_prev)
```

```
0.8829247675401521
```

```
print('\n',classification_report(nb_y_teste,nb_arvore_prev, target_names=['NEGADA','AUTORIZADA']))
```

```
              precision    recall  f1-score   support

      NEGADA       0.62      0.60      0.61       363
  AUTORIZADA       0.93      0.93      0.93      2003

    accuracy                           0.88      2366
   macro avg       0.78      0.77      0.77      2366
weighted avg       0.88      0.88      0.88      2366
```

Figura 27

Random Forest

```
from sklearn.ensemble import RandomForestClassifier
```

```
random_forest_autorizacao = RandomForestClassifier(random_state=0)
```

```
random_forest_autorizacao.fit(nb_X_treino,nb_y_treino)
```

```
RandomForestClassifier(random_state=0)
```

```
nb_random_forest_prev = random_forest_autorizacao.predict(nb_X_teste)
```

```
nb_random_forest_prev
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

```
nb_y_teste
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int64)
```

```
accuracy_score(nb_y_teste,nb_random_forest_prev)
```

```
0.908284023668639
```

```
print('\n',classification_report(nb_y_teste,nb_random_forest_prev, target_names=['NEGADA','AUTORIZADA']))
```

```
              precision    recall  f1-score   support

      NEGADA       0.78      0.56      0.65       363
  AUTORIZADA       0.92      0.97      0.95      2003

    accuracy                           0.91      2366
   macro avg       0.85      0.77      0.80      2366
weighted avg       0.90      0.91      0.90      2366
```

Figura 28

Pode ser visto que todos os modelos tiveram resultados satisfatórios se levarmos em consideração apenas a acurácia. Entretanto quando analisados os demais indicadores como **precision**, **recall, f1-score** e **support**, vemos que todos os modelos foram muito bem em prever quando a situação das guias era autorizada, porém no caso de situação negada vemos que o desempenho ficou muito ruim.

Isso ocorreu pois temos nesse caso uma classe desbalanceada, pois temos uma desproporcionalidade entre as classes autorizada e negada (figura 29), o que faz com que o modelo fique muito especializado em prever guias autorizadas e quase não acertar as com situação negada.

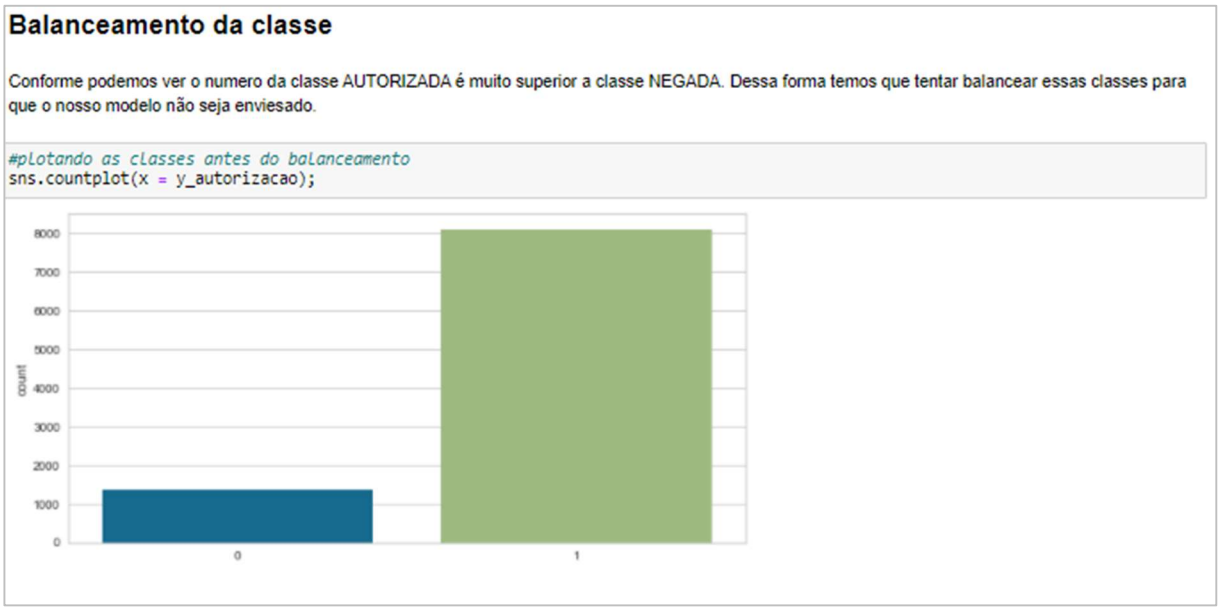

Figura 29

Para resolver esse problema foi feito o rebalanceamento das classes usando a função **SMOTE** (Synthetic Minority Over-sampling Technique), que realiza um oversampling gerando dados sintéticos (não duplicados) da classe minoritária a partir de vizinhos. Como pode ser visto após o oversampling a classe ficou balanceada (figura 30).

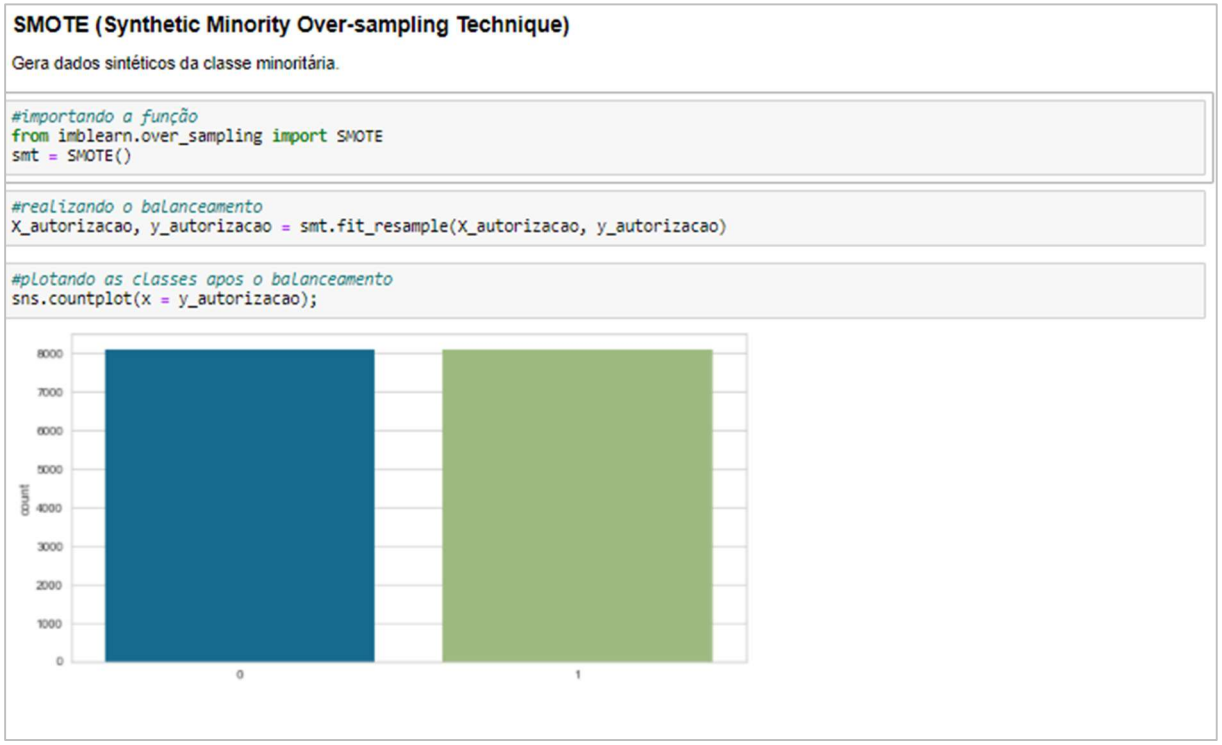

Figura 30

Após o balanceamento da classe foi efetuado novamente um treino dos modelos e foram obtidos resultados diferentes. Houve uma queda de acurácia vista no modelo Naive Bayes, por outro lado os resultados para previsão de situação negada ficaram melhores (figura 31). Já nos modelos Decision Tree e Random Forest a acurácia foi aumentada e obteve-se um resultado muito melhor para previsão da situação negada (figura 32 e 33).

Naive Bayes

```
print("##### Não Balanceado #####")
print(classification_report(nb_y_teste,nb_naive_prev, target_names=['NEGADA','AUTORIZADA']))
print("\n\n##### Balanceado #####")
print('\n',classification_report(y_teste,naive_prev, target_names=['NEGADA','AUTORIZADA']))
```

```
##### Não Balanceado #####
              precision    recall  f1-score   support

      NEGADA       0.30      0.02      0.04       363
  AUTORIZADA       0.85      0.99      0.91      2003

    accuracy                           0.84      2366
   macro avg       0.58      0.51      0.48      2366
weighted avg       0.76      0.84      0.78      2366


##### Balanceado #####

              precision    recall  f1-score   support

      NEGADA       0.56      0.09      0.16      2073
  AUTORIZADA       0.49      0.92      0.64      1978

    accuracy                           0.50      4051
   macro avg       0.53      0.51      0.40      4051
weighted avg       0.53      0.50      0.40      4051
```

Figura 31

Arvore de decisão

```
print("##### Não Balanceado #####")
print(classification_report(nb_y_teste,nb_arvore_prev, target_names=['NEGADA','AUTORIZADA']))
print("\n\n##### Balanceado #####")
print('\n',classification_report(y_teste,arvore_prev, target_names=['NEGADA','AUTORIZADA']))
```

```
##### Não Balanceado #####
              precision    recall  f1-score   support

      NEGADA       0.62      0.60      0.61       363
  AUTORIZADA       0.93      0.93      0.93      2003

    accuracy                           0.88      2366
   macro avg       0.78      0.77      0.77      2366
weighted avg       0.88      0.88      0.88      2366


##### Balanceado #####

              precision    recall  f1-score   support

      NEGADA       0.93      0.93      0.93      2073
  AUTORIZADA       0.93      0.93      0.93      1978

    accuracy                           0.93      4051
   macro avg       0.93      0.93      0.93      4051
weighted avg       0.93      0.93      0.93      4051
```

Figura 32

Random Forest

```
print("##### Não Balanceado #####")
print(classification_report(nb_y_teste,nb_random_forest_prev, target_names=['NEGADA','AUTORIZADA']))
print("\n\n##### Balanceado #####")
print('\n',classification_report(y_teste,random_forest_prev, target_names=['NEGADA','AUTORIZADA']))
```

```
##### Não Balanceado #####
              precision    recall  f1-score   support

      NEGADA       0.78      0.56      0.65       363
  AUTORIZADA       0.92      0.97      0.95      2003

    accuracy                           0.91      2366
   macro avg       0.85      0.77      0.80      2366
weighted avg       0.90      0.91      0.90      2366


##### Balanceado #####

              precision    recall  f1-score   support

      NEGADA       0.97      0.94      0.95      2073
  AUTORIZADA       0.94      0.97      0.95      1978

    accuracy                           0.95      4051
   macro avg       0.95      0.95      0.95      4051
weighted avg       0.95      0.95      0.95      4051
```

Figura 33

Aqui foi utilizado apenas a acurácia dos modelos para realizar uma comparação de antes do balanceamento e depois do balanceamento (figura 34).

Comparando a acurácia

```
print("##### Não Balanceado #####")
print('Naive Bayes:',accuracy_score(nb_y_teste,nb_naive_prev))
print('Arvore de decisão:',accuracy_score(nb_y_teste,nb_arvore_prev))
print('Random forest:',accuracy_score(nb_y_teste,nb_random_forest_prev))

print("\n##### Balanceado #####")
print('Naive Bayes:',accuracy_score(y_teste,naive_prev))
print('Arvore de decisão:',accuracy_score(y_teste,arvore_prev))
print('Random forest:',accuracy_score(y_teste,random_forest_prev))
```

```
##### Não Balanceado #####
Naive Bayes: 0.8427726120033813
Arvore de decisão: 0.8829247675401521
Random forest: 0.908284023668639

##### Balanceado #####
Naive Bayes: 0.49913601579856826
Arvore de decisão: 0.9296470007405578
Random forest: 0.9545791162675883
```

Figura 34

6. Apresentação dos Resultados e Trabalhos Futuros

Como pode ser visto no capítulo anterior obteve-se um bom desempenho com o treinamento dos modelos inicialmente, mas foi necessário realizar o balanceamento para que os acertos das guias com situação negada fossem melhores. Após o acerto todos os modelos melhoraram significativamente e se tornaram mais generalistas, o que os torna melhores em realizar previsões.

Podemos concluir que o modelo com melhor resultado geral foi o **Radom Forest**, seguido do **Decicion Tree** e por último o **Naive Bayes**.

Penso que este estudo de previsão de autorização das guias, é de uma importância muito grande pois envolve não só lucro ou prejuízo para operadoras de planos de saúde, mas é um processo que pode salvar vidas. Com a implementação de modelos como esse, pode-se agilizar as análises, tomando decisões de forma mais rápida, diminuindo a mão de obra na auditoria e tornando o processo mais confiável.

Como pontos de melhoria do estudo pode ser observado que todos os modelos foram usados com suas configurações padrão, o que abre margem para diversos testes de configuração para uma provável melhoria dos resultados.

7. Links

Link para o vídeo:
https://youtu.be/DmLg580f9ow

Link para o repositório:
https://1drv.ms/u/s!AgWyPnU0Li-WsMdHHeatwcNPA_bILw?e=QyA725