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

Lucas do Prado Ferreira Pinto

MACHINE LEARNING
PARA A
DETECÇÃO DE EMPREGADOR COM IRREGULARIDADE TRABALHISTA

Belo Horizonte
2021

Lucas do Prado Ferreira Pinto

MACHINE LEARNING
PARA A
DETECÇÃO DE EMPREGADOR COM IRREGULARIDADE TRABALHISTA

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

No dia 1º de maio de 1943 foi criada e aprovada, pelo então presidente Getúlio Vargas, a Consolidação das Leis do Trabalho - CLT.

Acompanhada pelo Decreto-Lei nº 5.452, a Consolidação estatuiu as normas que regulamentavam as relações individuais e coletivas de trabalho nela previstas.

Unificando toda a legislação trabalhista à época vigente no Brasil, a CLT veio como uma espécie de resposta à luta de trabalhadores e movimentos sindicais por melhores condições de trabalho, seja no ambiente laboral, seja na remuneração dos obreiros e até mesmo no combate ao trabalho de crianças e mulheres.

No governo de Getúlio Vargas também foram criados o Ministério do Trabalho e as Comissões e Juntas de Conciliação para a resolução de conflitos coletivos e individuais respectivamente.

Pouco mais de 50 anos antes, em 1891, a Inspeção do Trabalho era criada no país pelo então chefe de governo provisório, Manoel Deodoro da Fonseca, com o objetivo de fiscalizar estabelecimentos que exploravam mão de obra de crianças e adolescentes.

Ao longo do tempo, diversas mudanças foram acontecendo no mundo do trabalho tendo sido necessárias a criação de novos dispositivos legais voltados à regulamentação das relações de trabalho e emprego e a revisão dos normativos já existentes.

Diante das mudanças, a Inspeção do Trabalho cresceu e passou a ganhar cada vez mais função social tendo, hoje, como pilares norteadores para o cumprimento de sua missão institucional a(o):

- Erradicação do trabalho análogo ao de escravo;
- Erradicação do trabalho infantil e proteção do adolescente trabalhador;

- Combate à informalidade no trabalho assalariado;
- Garantia do cumprimento das cotas legais para admissão de aprendizes e pessoas com deficiência;
- Redução da morbimortalidade por acidentes ou doenças do trabalho;
- Garantia de ambientes e processos de trabalho seguros e saudáveis;
- Aperfeiçoamento contínuo das normas regulamentadoras de segurança e saúde no trabalho;
- Prevenção de acidentes e doenças do trabalho por meio de investigações;
- Combate da inadimplência e da sonegação do Fundo de Garantia do Tempo de Serviço – FGTS.

1.1. Contextualização

Conforme as Diretrizes do Planejamento da Inspeção do Trabalho para o ano de 2021:

"As ações fiscais deverão ser planejadas de forma a direcionar a emissão de Ordens de Serviço (OS) àqueles segmentos econômicos e estabelecimentos com maiores indícios de irregularidades e maiores riscos à integridade do trabalhador, conforme o foco de cada Atividade e Projeto".

Assim,

"considerando o exposto acima e, também, a otimização de recursos financeiros e humanos, é importante destacar que os esforços despendidos com as fiscalizações não devem ser direcionados a estabelecimentos sem irregularidades, uma vez que tais ações implicariam na ineficiência da atuação da Inspeção do Trabalho".

"Sempre que possível as ações planejadas no âmbito das Atividades e Projetos deverão considerar tanto os objetivos de Legislação quanto os de Segurança e Saúde no Trabalho. Desta forma, espera-se uma maior eficácia das inspeções e consequente melhoria dos resultados alcançados".

Desta forma, o presente trabalho de conclusão de curso visa ao desenvolvimento de um algoritmo de *Machine Learning* voltado para a detecção de empregadores que desrespeitam a legislação trabalhista, incluindo as normas de segurança e saúde no trabalho.

Com o modelo criado, pretende-se prever se um determinado empregador, prestes a ser fiscalizado sob as condições impostas pela Inspeção Trabalhista, será flagrado ou não com irregularidade e, a partir de tal informação, ser confirmada ou não o processo de fiscalização.

Todo o processo de tratamento, análise e exploração dos dados, além da criação, treinamento, aplicação e avaliação dos modelos de *Machine Learning* testados foi desenvolvido utilizando a linguagem de programação Python e suas bibliotecas no ambiente de desenvolvimento Jupyter Notebook.

1.2. O Problema Proposto

Why? - Por que esse problema é importante?

O descumprimento, por parte de empregadores, de leis trabalhistas e normas de segurança e saúde no trabalho é real e prejudica milhões de trabalhadores todos os anos no Brasil.

A lesão aos trabalhadores vão desde perdas financeiras, provocadas por pagamento de salários abaixo do piso, ausência de cômputo integral das horas extras prestadas, não pagamento de adicionais de periculosidade, insalubridade e noturno, ausência do recolhimento do fundo de garantia do tempo de serviço – FGTS devido ao obreiro, entre outros, à danos à integridade física e emocional do trabalhador, ocasionados por acidentes e doenças do trabalho que podem levar a amputações de membros e até mesmo à morte.

Desta forma, sendo um assunto extremamente relevante para a sociedade, é importantíssimo que o Estado aja de maneira eficiente na identificação dos empregadores que descumprem as leis trabalhistas e normas de segurança e saúde no trabalho, devendo a

Inspeção do Trabalho planejar e direcionar suas ações fiscais àqueles segmentos econômicos e estabelecimentos que apresentem algum indício de irregularidade, de forma a combater práticas ilegais, abusivas e danosas ao trabalhador brasileiro, parte hipossuficiente na relação de emprego.

Who? - De quem são os dados analisados?

O conjunto de dados a serem analisados será extraído por meio de consulta SQL (Structered Query Language) da base de dados da Subsecretaria de Inspeção do Trabalho, vinculada à Secretaria de Trabalho, da Secretaria Especial de Previdência e Trabalho, do Ministério da Economia. Os dados constam de relatórios referentes a ações fiscais já realizadas pela Inspeção do Trabalho.

Também será necessária a coleta de dados referentes à população dos municípios brasileiros no ano de 2020. Tais dados serão obtidos diretamente da internet, através do site **https://basedosdados.org/dataset/br-ibge-populacao**, e, também, serão extraídos por meio de consulta SQL.

O dataset final será formado com a integração dos 2 (dois) conjuntos de dados mencionados, provenientes de fontes diferentes, e terá mais de 50.000 (cinquenta mil) registros sem o uso de quaisquer ferramentas de balanceamento de dados, como por exemplo, o oversampling.

What? - Quais os objetivos com essa análise?

O objetivo com as análises dos dados coletados é desenvolver um modelo de *Machine Learning* voltado ao aprendizado supervisionado de classificação, que deverá detectar empregadores que desrespeitam a legislação trabalhista, incluindo as normas de segurança e saúde no trabalho.

Com o modelo criado, pretende-se prever se um determinado empregador, prestes a ser fiscalizado sob as condições impostas pela Inspeção Trabalhista, será flagrado ou não

com irregularidade e, a partir de tal informação, ser confirmada ou não o processo de fiscalização.

Where? - Trata dos aspectos geográficos e logísticos de sua análise.

A análise contará com dados de ações fiscais realizadas em todo o território nacional, incluindo municípios pequenos e grandes centros urbanos, como cidades interioranas e capitais estaduais respectivamente.

Uma das variáveis preditoras do modelo será a população do município do empregador a ser fiscalizado. Diretamente ligada ao desenvolvimento regional, especula-se que o tamanho populacional possa ter uma influência significativa na relação entre patrão e empregado, e no cumprimento ou não das normas trabalhistas por parte daquele.

When? - Qual o período está sendo analisado?

Os dados analisados, referentes às ações fiscais já realizadas pela Inspeção do Trabalho, contemplam todo o ano de 2020 e vão até o mês de maio de 2021.

Já a população do município do empregador fiscalizado se refere àquela registrada pelo IBGE em 2020, tendo em vista a grande maioria das ações fiscais ter ocorrida naquele ano e por entender não haver grandes diferenças com os números atuais de 2021.

1.3. Objetivos

O presente trabalho tem como objetivo elaborar um modelo de *Machine Learning* que detecte se um determinado empregador desrespeita, de alguma forma, a legislação brasileira voltada ao tratamento das relações de trabalho e emprego, além das normas regulamentadoras de segurança e saúde no trabalho.

A partir do modelo criado, pretende-se prever se um empregador, prestes a entrar no planejamento do órgão fiscalizador e ser submetido à uma auditoria trabalhista *“in loco”*,

com um quantitativo pré-definido de itens normativos a serem verificados pela Inspeção do Trabalho, será flagrado ou não com irregularidade e, a partir de tal informação, ser confirmada ou não o processo de fiscalização.

O objetivo da predição é selecionar apenas aqueles empregadores com fortes indícios de irregularidades e/ou riscos à integridade do trabalhador, fazendo com que a atuação da Inspeção do Trabalho no Brasil se torne cada vez mais eficiente, eficaz e efetiva.

Pretende-se uma acurácia do modelo preditivo superior a 70%.

2. Coleta de Dados

Para o desenvolvimento do trabalho foram utilizadas 2 (duas) fontes de dados distintas. Para cada uma das fontes foram extraídos conjuntos de dados (datasets) que se integraram em uma única fonte com mais de 50.000 (cinquenta mil) registros, sem utilização de oversampling.

A integração entre os datasets se deu devido a existência, em ambos os conjuntos de dados, da variável "Código do Município". Utilizando-se a função "merge", da biblioteca Pandas do Python, foi possível realizar a junção dos dados em uma única fonte, passando esta a conter todas as variáveis preditoras e target necessárias para a construção do modelo de Machine Learning.

2.1 Fonte 1 – Banco de Dados Relacional da Inspeção do Trabalho

O dataset "Fiscalizações_Diretas" foi obtido diretamente do DataWarehouse da Inspeção do Trabalho, em um Banco de Dados Relacional.

A extração se deu por meio de consulta SQL (Structered Query Language) no Sistema de Gerenciamento de Banco de Dados - SGBD - Microsoft SQL Server.

O banco de dados mencionado armazena dados referentes às ações fiscais realizadas pela Inspeção do Trabalho no Brasil. Para o trabalho em questão, optou-se por coletar informações apenas das fiscalizações diretas, ou seja, aquelas que necessariamente ocorreram *"in loco"*, no estabelecimento do empregador.

O período de coleta compreende as competências de janeiro de 2020 a maio de 2021.

Abaixo apresentamos a Query utilizada para a extração dos dados.

```
; WITH Tabela_Temporaria_1 AS

(

SELECT DISTINCT A.numerori AS Numero_do_Relatorio_de_Inspecao

FROM [DBSFIT].[dbo].[TBSFITWEB_RI_DadosEmpregador] A

INNER JOIN [DBSFIT].[dbo].[TBSFITWEB_RI_OcorrenciaEspecial] B
ON A.numerori = B.numerori

WHERE A.situacao = 'FISC_CONCLUIDA_E_AFERIDA'

AND NOT (B.ocorrenciaespecial = 't' AND B.fiscalizacaorealizada = 'f')

AND A.competencia BETWEEN 202001 AND 202105

GROUP BY A.numerori

),

Tabela_Temporaria_2 AS

(

SELECT A.Numero_do_Relatorio_de_Inspecao AS Numero_do_Relatorio_de_Inspecao,
       B.estado AS UF_do_Empregador_Fiscalizado,
       B.municipio AS Codigo_do_Municipio

FROM Tabela_Temporaria_1 A

LEFT JOIN [DBSFIT].[dbo].[TBSFITWEB_RI_DadosEmpregador] B
ON A.Numero_do_Relatorio_de_Inspecao = B.numerori

),

Tabela_Temporaria_3 AS

(

SELECT numerori AS Numero_do_Relatorio_de_Inspecao,
       tipoos AS Modalidade_de_Fiscalizacao

       FROM [DBSFIT].[dbo].[TBSFITWEB_OS]

      WHERE tipoos = 'DIRIGIDA'

      AND datacadastro >= '20190101'

),
```

```
Tabela_Temporaria_4 AS

(

SELECT numerori AS Numero_do_Relatorio_de_Inspecao,
       (SUM (ISNULL(qtdtrabalhadoresestabh18,0)) +
        SUM (ISNULL(qtdtrabalhadoresestabm18,0)) +
        SUM (ISNULL(qtdtrabalhadoresestabh17,0)) +
        SUM (ISNULL(qtdtrabalhadoresestabm17,0))) AS Quantidade_de_Trabalhadores_do_Empregador

  FROM [DBSFIT].[dbo].[TBSFITWEB_RI_Vinculos]

  GROUP BY numerori

),

Tabela_Temporaria_5 AS

(

SELECT DISTINCT (numerori) AS Numero_do_Relatorio_de_Inspecao,
       COUNT (DISTINCT cif) AS Número_de_AFTs_na_Equipe

FROM [DBSFIT].[dbo].[TBSFITWEB_RI_Equipe]

GROUP BY numerori

),

Tabela_Temporaria_6 AS

(

SELECT DISTINCT (numerori) AS Numero_do_Relatorio_de_Inspecao,
       COUNT (DISTINCT ementa) AS Quantidade_de_Ementas_Fiscalizadas

FROM [DBSFIT].[dbo].[TBSFITWEB_RI_Ementa]

WHERE situacao IN ('REGULAR', 'IRREGULAR')

GROUP BY numerori

),

Tabela_Temporaria_7 AS

(

SELECT DISTINCT (numerori) AS Numero_do_Relatorio_de_Inspecao,
       COUNT (DISTINCT ementa) AS Quantidade_de_Ementas_de_Legislação_Fiscalizadas

FROM [DBSFIT].[dbo].[TBSFITWEB_RI_Ementa]

WHERE (ementa LIKE '0%' AND situacao IN ('REGULAR', 'IRREGULAR'))

GROUP BY numerori

),

Tabela_Temporaria_8 AS

(

SELECT DISTINCT (numerori) AS Numero_do_Relatorio_de_Inspecao,
       COUNT (DISTINCT ementa) AS Quantidade_de_Ementas_de_SST_Fiscalizadas

FROM [DBSFIT].[dbo].[TBSFITWEB_RI_Ementa]

WHERE (ementa NOT LIKE '0%' AND situacao IN ('REGULAR', 'IRREGULAR'))

GROUP BY numerori

),
```

```
Tabela_Temporaria_9 AS

(

SELECT DISTINCT numerori AS Numero_do_Relatorio_de_Inspecao_com_Irregularidade,
       COUNT (DISTINCT ementa) AS Quantidade_de_Ementas_Fiscalizadas,
       1 AS Empregador_com_Ementas_Irregulares

FROM [DBSFIT].[dbo].[TBSFITWEB_RI_Ementa]

WHERE situacao IN ('REGULAR', 'IRREGULAR')

AND numerori IN (

                    SELECT DISTINCT numerori AS Numero_do_Relatorio_de_Inspecao

                    FROM [DBSFIT].[dbo].[TBSFITWEB_RI_Ementa]

                    WHERE situacao = 'IRREGULAR'

                )

GROUP BY numerori

)

SELECT A.Numero_do_Relatorio_de_Inspecao,
       A.UF_do_Empregador_Fiscalizado,
       B.NOMunicipio AS Municipio,
       A.Codigo_do_Município,
       D.Quantidade_de_Trabalhadores_do_Empregador,
       E.Numero_de_AFTs_na_Equipe,
       ISNULL(G.Quantidade_de_Ementas_de_Legislação_Fiscalizadas,0) AS
       Quantidade_de_Ementas_de_Legislação_Fiscalizadas,
       ISNULL(H.Quantidade_de_Ementas_de_SST_Fiscalizadas,0) AS
       Quantidade_de_Ementas_de_SST_Fiscalizadas,
       ISNULL(I.Empregador_com_Ementas_Irregulares,0) AS Empregador_com_Irregularidade

FROM Tabela_Temporaria_2 A

LEFT JOIN [DBIDEB].[dbo].[TBMunicipio] B
ON A.Codigo_do_Municipio = B.CDMunicipio

INNER JOIN Tabela_Temporaria_3 C
ON A.Numero_do_Relatorio_de_Inspecao = C.Numero_do_Relatorio_de_Inspecao

LEFT JOIN Tabela_Temporaria_4 D
ON A.Numero_do_Relatorio_de_Inspecao = D.Numero_do_Relatorio_de_Inspecao

LEFT JOIN Tabela_Temporaria_5 E
ON D.Numero_do_Relatorio_de_Inspecao = E.Numero_do_Relatorio_de_Inspecao

INNER JOIN Tabela_Temporaria_6 F
ON A.Numero_do_Relatorio_de_Inspecao = F.Numero_do_Relatorio_de_Inspecao

LEFT JOIN Tabela_Temporaria_7 G
ON F.Numero_do_Relatorio_de_Inspecao = G.Numero_do_Relatorio_de_Inspecao

LEFT JOIN Tabela_Temporaria_8 H
ON F.Numero_do_Relatorio_de_Inspecao = H.Numero_do_Relatorio_de_Inspecao

LEFT JOIN Tabela_Temporaria_9 I
ON A.Numero_do_Relatorio_de_Inspecao = I.Numero_do_Relatorio_de_Inspecao_com_Irregularidade

ORDER BY A.Numero_do_Relatorio_de_Inspecao
```

Nome da Coluna / Campo	Descrição	Tipo
Numero_do_Relatorio_de_Inspecao	Número do relatório da ação fiscal.	Pandas (Int64)
UF_do_Empregador_Fiscalizado	Unidade da Federação onde o empregador atua e foi fiscalizado.	Pandas (Object)
Municipio	Município onde o empregador atua e foi fiscalizado.	Pandas (Object)
Codigo_do_Municipio	Identificação numérica do município, dada pelo Instituto Brasileiro de Geografia e Estatística – IBGE.	Pandas (Int64)
Quantidade_de_Trabalhadores_do_Empregador	Número de empregados do empregador fiscalizado.	Pandas (Int64)
Numero_de_AFTs_na_Equipe	Número de Auditores-Fiscais do Trabalho que participaram da ação fiscal.	Pandas (Int64)
Quantidade_de_Ementas_de_Legislação_Fiscalizadas	Número de itens da legislação trabalhista auditados na ação fiscal.	Pandas (Int64)
Quantidade_de_Ementas_de_SST_Fiscalizadas	Número de itens normativos de segurança e saúde do trabalho auditados na ação fiscal.	Pandas (Int64)
Empregador_com_Irregularidade	Informa se o empregador fiscalizado se encontrava irregular ou não (S/N).	Pandas (Int64)

2.2 Fonte 2 - Internet

O dataset “Municipio_Populacao_2020” foi obtido diretamente da internet, através do sítio eletrônico **https://basedosdados.org/dataset/br-ibge-populacao** e também foi extraído por meio de consulta SQL (Structured Query Language).

Esses dados se referem à população de todos os municípios brasileiros no ano de 2020, e foram divulgados pelo Instituto Brasileiro de Geografia e Estatística - IBGE.

Abaixo apresentamos a Query utilizada para a extração dos dados.

```
SELECT ano AS Ano,
       id_municipio AS Codigo_Municipio,
       populacao AS Populacao

FROM basedosdados-staging.br_ibge_populacao_staging.municipio

WHERE ano = '2020'

ORDER BY id_municipio
```

Nome da Coluna / Campo	Descrição	Tipo
Ano	Ano a que se refere o total da população de cada município brasileiro divulgado pelo Instituto Brasileiro de Geografia e Estatística – IBGE.	Pandas (Int64)
Codigo_do_Municipio	Identificação numérica do município, dada pelo Instituto Brasileiro de Geografia e Estatística – IBGE.	Pandas (Int64)
Populacao_do_Municipio	Quantidade de indivíduos seres humanos que vivem em um determinado município.	Pandas (Int64)

3. Processamento/Tratamento de Dados

Todo o processamento e tratamento dos dados utilizados foi realizado através da linguagem Python, em sua versão 3.7.6, no ambiente do Jupyter Notebook versão 6.0.3.

Inicialmente foram importadas as bibliotecas Pandas, Matplotlib, Numpy e Seaborn, e, posteriormente, verificadas as versões das ferramentas.

Aluno: Lucas do Prado Ferreira Pinto

Pós Graduação - PUC Minas

Ciência de Dados

Trabalho de Conclusão de Curso - Turma 2020

Objetivo:

Prever se um determinado Empregador possui alguma Irregularidade voltada à Legislação Trabalhista ou à Segurança e Saúde no Trabalho.

Mínimo de Precisão: 70%

In [1]:
```
# Verificando a Versão do Python

from platform import python_version
print('Versão Python:', python_version())
```

```
Versão Python: 3.7.6
```

In [2]:
```
# Importando os Módulos Necessários para a Exploração e Preparação dos Dados

import pandas as pd
import matplotlib as mat
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

# Para Gráficos Dentro do Jupyter Notebook

%matplotlib inline
```

In [3]:
```
# Verificando a Versão do Pandas

pd.__version__
```

Out[3]: '1.0.1'

In [4]:
```
# Verificando a Versão do Matplotlib

mat.__version__
```

Out[4]: '3.1.3'

```
In [5]: # Verificando a Versão do Numpy
        np.__version__
Out[5]: '1.18.1'

In [6]: # Verificando a Versão do Seaborn
        sns.__version__
Out[6]: '0.10.0'
```

3.1 Dataset nº 1 – Fiscalizações Diretas

Em seguida, carregou-se o primeiro dataset, em formato .xlsx, referente às fiscalizações diretas realizadas pela Inspeção do Trabalho entre as competências de janeiro de 2020 e maio de 2021.

Dataset nº 1 - Fiscalizações Realizadas

Explorando e Preparando os Dados

```
In [7]: # Carregando o Dataset nº 1 - Fiscalizações Diretas (Formato xlsx) - FONTE: SGBD Microsoft SQL Server - BD Marfim
        dataset_1 = pd.read_excel("Fiscalizacoes_Diretas.xlsx")
```

Após o carregamento, verificou-se o formato do conjunto de dados, possuindo, este, 55.601 (cinquenta e cinco mil seiscentos e um) registros ou observações (linhas) e 9 (nove) variáveis (colunas).

```
In [8]: # Verificando o Formato dos Dados do Dataset nº 1 - Fiscalizações Diretas
        dataset_1.shape
Out[8]: (55601, 9)

In [9]: # Resumo do Dataset nº 1 - Fiscalizações Diretas
        # 55.601 Observações ou Registros (Linhas)
        # 9 Variáveis (Colunas)
```

Posteriormente, utilizando-se as funções ***"head()"*** e ***"tail()"*** foi possível visualizar as primeiras e últimas linhas do conjunto de dados "Fiscalizacoes_Diretas".

```
In [10]: # Verificando as Primeiras linhas do Dataset nº 1 - Fiscalizações Diretas

         dataset_1.head(5)
```

Out[10]:

	Numero_do_Relatorio_de_Inspecao	UF_do_Empregador_Fiscalizado	Municipio	Codigo_do_Municipio	Quantidade_de_Trabalhadores_do_Empregador	Nume
0	305311247	MG	Contagem	3118601	1569	
1	305312979	PA	Belém	1501402	131	
2	305335146	SP	São Paulo	3550308	1	
3	305337807	SP	Novo Horizonte	3533502	2090	
4	305359150	DF	Brasília	5300108	29	

```
In [11]: # Verificando as Últimas linhas do Dataset nº 1 - Fiscalizações Diretas

         dataset_1.tail(5)
```

Out[11]:

mero_de_AFTs_na_Equipe	Quantidade_de_Ementas_de_Legislação_Fiscalizadas	Quantidade_de_Ementas_de_SST_Fiscalizadas	Empregador_com_Irregularidade
1	0	17	1
1	3	0	0
1	2	0	0
3	0	17	1
3	0	17	1

3.2 Dataset nº 2 – População dos Municípios

Da mesma forma, também foi carregado o segundo dataset, em formato .csv, referente à população dos municípios brasileiros em 2020, divulgada pelo Instituto Brasileiro de Geografia e Estatística - IBGE.

Dataset nº 2 - População dos Municípios

Explorando e Preparando os Dados

```
In [12]: # Carregando o Dataset nº 2 - População dos Municípios (Formato csv) - FONTE: https://basedosdados.org/dataset/br-ibge-popula

         dataset_2 = pd.read_csv("Municipio_Populacao_2020.csv")
```

Também foi verificado o formato do conjunto de dados, possuindo, este, 5.570 (cinco mil quinhentos e setenta) registros ou observações (linhas) e 3 (três) variáveis (colunas).

```
In [13]: # Verificando o Formato dos Dados do Dataset nº 2 - População dos Municípios

         dataset_2.shape
```

Out[13]: (5570, 3)

```
In [14]: # Resumo do Dataset nº 2 - População dos Municípios

         # 5.570 Observações ou Registros (Linhas)

         # 3 Variáveis (Colunas)
```

Por fim, visualizou-se as primeiras e últimas linhas do conjunto de dados "Municipio_Populacao_2020".

```
In [15]: # Verificando as Primeiras linhas do Dataset nº 2 - População dos Municípios

         dataset_2.head(5)
```

Out[15]:

	Ano	Codigo_do_Municipio	Populacao_do_Municipio
0	2020	1100015	22728
1	2020	1100023	109523
2	2020	1100031	5188
3	2020	1100049	85893
4	2020	1100056	16204

```
In [16]: # Verificando as Últimas linhas do Dataset nº 2 - População dos Municípios

         dataset_2.tail(5)
```

Out[16]:

	Ano	Codigo_do_Municipio	Populacao_do_Municipio
5565	2020	5222005	13977
5566	2020	5222054	8873
5567	2020	5222203	6312
5568	2020	5222302	5882
5569	2020	5300108	3055149

3.3 Dataset nº 3 – Integração dos Datasets nº 1 e nº 2

Nesta etapa foi feita a junção dos 2 (dois) conjuntos de dados carregados anteriormente em uma única fonte de dados. A partir desta fonte resultante, novos tratamentos de limpeza e transformação são realizados para que se possa chegar ao dataset final, utilizado no modelo de *Machine Learning* para a detecção de empregador irregular.

Primeiramente, portanto, utilizou-se a função ***"merge()"*** para viabilizar a união dos dados.

Dataset nº 3 - Integração dos Datasets nº 1 e nº 2

Explorando e Preparando os Dados

```
In [17]: # Integrando os Datasets nº 1 e nº 2 e Gerando o Dataset nº 3

         dataset_3 = pd.merge(left=dataset_1,right=dataset_2,how='left',left_on="Codigo_do_Municipio",right_on="Codigo_do_Municipio")
```

Em seguida, foram verificados, novamente, o formato dos dados, além das primeiras e últimas linhas do dataset nº 3.

```
In [18]: # Verificando o Formato dos Dados do Dataset nº 3

         dataset_3.shape

Out[18]: (55601, 11)

In [19]: # Resumo do Dataset nº 3

         # 55.601 Observações ou Registros (Linhas)

         # 11 Variáveis (Colunas)

In [20]: # Verificando as Primeiras linhas do Dataset nº 3

         dataset_3.head(5)
```

Out[20]:

	Numero_do_Relatorio_de_Inspecao	UF_do_Empregador_Fiscalizado	Municipio	Codigo_do_Municipio	Quantidade_de_Trabalhadores_do_Empregador	Nume
0	305311247	MG	Contagem	3118601	1569	
1	305312979	PA	Belém	1501402	131	
2	305335146	SP	São Paulo	3550308	1	
3	305337807	SP	Novo Horizonte	3533502	2090	
4	305359150	DF	Brasília	5300108	29	

```
In [21]: # Verificando as Últimas linhas do Dataset nº 3

         dataset_3.tail(5)
```

Out[21]:

tidade_de_Ementas_de_Legislação_Fiscalizadas	Quantidade_de_Ementas_de_SST_Fiscalizadas	Empregador_com_Irregularidade	Ano	Populacao_do_Municipio
0	17	1	2020	817511
3	0	0	2020	230371
2	0	0	2020	230371
0	17	1	2020	1108975
0	17	1	2020	1108975

Começa-se, então, a alterar o dataset nº 3, excluindo-se as variáveis desnecessárias para o modelo e reordenando as colunas do conjunto de dados.

```
In [22]: # Alterando o Dataset nº 3

         # Excluindo as Colunas Desnecessárias

         dataset_3 = dataset_3.drop(columns=["Numero_do_Relatorio_de_Inspecao", "UF_do_Empregador_Fiscalizado","Municipio", "Codigo_do

In [23]: # Alterando o Dataset nº 3

         # Reordenando as Colunas

         dataset_3 = dataset_3[['Populacao_do_Municipio', 'Numero_de_AFTs_na_Equipe', 'Quantidade_de_Trabalhadores_do_Empregador', 'Qu
```

Em seguida, verifica-se o formato e tipo dos dados, além das primeiras e últimas linhas do dataset nº 3.

```
In [24]: # Verificando algumas Informações do Dataset nº 3, incluisve seus Tipos de Dados

         dataset_3.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 55601 entries, 0 to 55600
Data columns (total 6 columns):
 #   Column                                        Non-Null Count  Dtype
---  ------                                        --------------  -----
 0   Populacao_do_Municipio                        55601 non-null  int64
 1   Numero_de_AFTs_na_Equipe                      55601 non-null  int64
 2   Quantidade_de_Trabalhadores_do_Empregador     55601 non-null  int64
 3   Quantidade_de_Ementas_de_Legislação_Fiscalizadas  55601 non-null  int64
 4   Quantidade_de_Ementas_de_SST_Fiscalizadas     55601 non-null  int64
 5   Empregador_com_Irregularidade                 55601 non-null  int64
dtypes: int64(6)
memory usage: 3.0 MB
```

```
In [25]: # Verificando o Formato dos Dados do Dataset nº 3 Após Alterações

         dataset_3.shape
```

```
Out[25]: (55601, 6)
```

```
In [26]: # Resumo do Dataset nº 3 Após Alterações

         # 55.601 Observações ou Registros (Linhas)

         # 6 Variáveis (Colunas)
```

```
In [27]: # Verificando as Primeiras linhas do Dataset nº 3 Após Alterações

         dataset_3.head(5)
```

Out[27]:

	Populacao_do_Municipio	Numero_de_AFTs_na_Equipe	Quantidade_de_Trabalhadores_do_Empregador	Quantidade_de_Ementas_de_Legislação_Fiscalizadas
0	668949	1	1569	2
1	1499641	2	131	9
2	12325232	1	1	1
3	41414	2	2090	4
4	3055149	2	29	7

```
In [28]: # Verificando as Últimas linhas do Dataset nº 3 Após Alterações

         dataset_3.tail(5)
```

Out[28]:

alhadores_do_Empregador	Quantidade_de_Ementas_de_Legislação_Fiscalizadas	Quantidade_de_Ementas_de_SST_Fiscalizadas	Empregador_com_Irregularidade
8	0	17	1
3	3	0	0
2	2	0	0
8	0	17	1
64	0	17	1

Uma etapa importante de todo este processo é a verificação e, se for o caso, o tratamento dos valores ausentes, ou nulos, eventualmente existentes no conjunto de dados.

Assim, utilizando-se o comando ***"dataset_3.isnull().values.any()"*** foi possível constatar a inexistência de valores nulos, o que foi posteriormente confirmada por meio do uso de outro comando, ***"dataset_3.isnull().sum()"***, para a visualização de forma individual para cada variável.

```
In [29]: # Verificando a Existência de Valores Nulos (NaN) no Dataset nº 3 Após Alterações

         dataset_3.isnull().values.any()

Out[29]: False
```

```
In [30]: # Confirmando a Inexistência de Valores Nulos (NaN) em Cada Variável (Coluna) do Dataset nº 3 Após Alterações

         dataset_3.isnull().sum()

Out[30]: Populacao_do_Municipio                        0
         Numero_de_AFTs_na_Equipe                      0
         Quantidade_de_Trabalhadores_do_Empregador     0
         Quantidade_de_Ementas_de_Legislação_Fiscalizadas  0
         Quantidade_de_Ementas_de_SST_Fiscalizadas     0
         Empregador_com_Irregularidade                 0
         dtype: int64
```

Além da constatação da inexistência de valores nulos, também se confirmou a inexistência de registros duplicados, o que já era esperado, uma vez que a extração do primeiro dataset, em linguagem SQL, contou com a cláusula *SELECT DISTINCT ()*.

```
In [31]: dataset_3.index.duplicated().sum()

Out[31]: 0
```

Por fim, outras informações de interesse, como a quantidade de fiscalizações realizadas em empregadores que não possuíam, à época da inspeção, trabalhadores a eles vinculados, também foram analisadas.

```
In [32]: # Analisando Outras Informações do Dataset nº 3 (Após Alterações) Referentes às Fiscalizações Realizadas

         print("1 - Quantidade de Fiscalizações: {0}".format(len(dataset_3)))
         print("2 - Quantidade de Fiscalizações em Municípios Acima de 500 Mil Habitantes: {0}".format(len(dataset_3.loc[dataset_3['Po
         print("3 - Quantidade de Fiscalizações com Apenas 1 Auditor-Fiscal na Equipe: {0}".format(len(dataset_3.loc[dataset_3['Numero
         print("4 - Quantidade de Fiscalizações em Empregadores sem Trabalhador: {0}".format(len(dataset_3.loc[dataset_3['Quantidade_d
         print("5 - Quantidade de Fiscalizações com Nenhuma Ementa de Legislação Verificada: {0}".format(len(dataset_3.loc[dataset_3['
         print("6 - Quantidade de Fiscalizações com Nenhuma Ementa de Segurança e Saúde Verificada: {0}".format(len(dataset_3.loc[data
         print("7 - Quantidade de Fiscalizações em Empregador Irregular: {0}".format(len(dataset_3.loc[dataset_3['Empregador_com_Irreg
         print("8 - Quantidade de Fiscalizações em Empregador Regular: {0}".format(len(dataset_3.loc[dataset_3['Empregador_com_Irregul

         1 - Quantidade de Fiscalizações: 55601
         2 - Quantidade de Fiscalizações em Municípios Acima de 500 Mil Habitantes: 27456
         3 - Quantidade de Fiscalizações com Apenas 1 Auditor-Fiscal na Equipe: 30458
         4 - Quantidade de Fiscalizações em Empregadores sem Trabalhador: 1645
         5 - Quantidade de Fiscalizações com Nenhuma Ementa de Legislação Verificada: 7312
         6 - Quantidade de Fiscalizações com Nenhuma Ementa de Segurança e Saúde Verificada: 28114
         7 - Quantidade de Fiscalizações em Empregador Irregular: 35835
         8 - Quantidade de Fiscalizações em Empregador Regular: 19766
```

3.4 Dataset Final

A partir das informações levantadas, fez-se necessário remover os registros desnecessários para, então, se chegar ao dataset final.

Assim, todas aquelas observações cujas fiscalizações foram realizadas em empregadores sem trabalhador foram retiradas do conjunto de dados, uma vez que não agregariam nenhum valor para o Modelo de Machine Learning a ser implementado, podendo, inclusive, comprometer o seu desempenho.

Dataset Final

Explorando e Preparando os Dados

In [33]:
```
# Removendo Registros Desnecessários do Dataset nº 3 - Fiscalizações em Empregadores sem Trabalhador
registros_remover = dataset_3.loc[(dataset_3['Quantidade_de_Trabalhadores_do_Empregador'] == 0)]
```

In [34]:
```
# Visualizando os Registros Desnecessários à Remover do Dataset nº 3
registros_remover
```

Out[34]:

	Populacao_do_Municipio	Numero_de_AFTs_na_Equipe	Quantidade_de_Trabalhadores_do_Empregador	Quantidade_de_Ementas_de_Legislação_Fiscaliz
44	12325232	1	0	
45	12325232	1	0	
48	12325232	1	0	
49	1653461	1	0	
52	12325232	4	0	
...	...	...	...	
55466	105809	2	0	
55475	91169	2	0	
55505	235416	3	0	
55510	8347	2	0	
55563	3055149	2	0	

1645 rows × 6 columns

In [35]:
```
# Gerando o Dataset Final
dataset_final = dataset_3.drop(registros_remover.index)
```

O formato do dataset final ficou então com 53.956 (cinquenta e três mil novecentos e cinquenta e seis) registros e 6 (seis) variáveis.

In [35]:
```
# Gerando o Dataset Final
dataset_final = dataset_3.drop(registros_remover.index)
```

In [36]:
```
# Visualizando o Dataset Final Após a Remoção dos Registros Desnecessários
dataset_final
```

Out[36]:

	Populacao_do_Municipio	Numero_de_AFTs_na_Equipe	Quantidade_de_Trabalhadores_do_Empregador	Quantidade_de_Ementas_de_Legislação_Fiscaliz
0	668949	1	1569	
1	1499641	2	131	
2	12325232	1	1	
3	41414	2	2090	
4	3055149	2	29	
...	...	...	...	
55596	817511	1	8	
55597	230371	1	3	
55598	230371	1	2	
55599	1108975	3	8	
55600	1108975	3	64	

53956 rows × 6 columns

```
In [37]: # Confirmando o Formato dos Dados do Dataset Final

         dataset_final.shape

Out[37]: (53956, 6)
```

```
In [38]: # Resumo do Dataset Final

         # 53.956 Observações ou Registros (Linhas)

         # 6 Variáveis (Colunas)
```

4. Análise e Exploração dos Dados

A partir deste ponto começou-se a análise exploratória dos dados do dataset final.

Inicialmente buscou-se identificar, através de 2 (dois) métodos distintos, a correlação entre as variáveis do conjunto de dados.

Em seguida, outras análises foram feitas, como o levantamento das informações estatísticas dos dados e a relação de distribuição entre as classes da variável target do modelo

4.1 Método nº 1 - Correlação de cada Variável com as Demais

O gráfico plotado representa, através de uma escala de cores, a relação existente entre uma variável e todas as outras do dataset final.

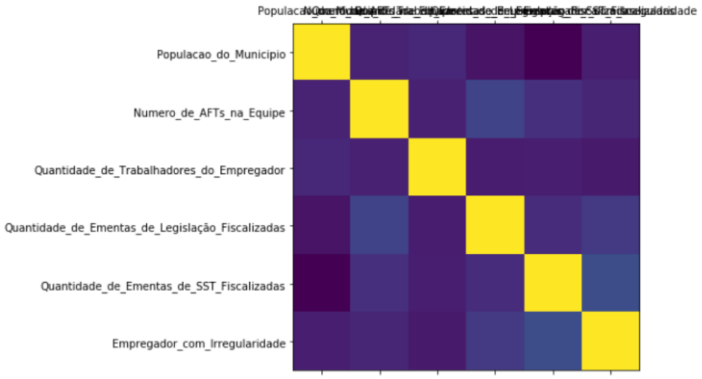
Quanto mais clara a cor maior é a correlação entre as variáveis, sendo a cor amarela a que indica relação máxima.

Identificando a Correlação entre as Variáveis:

Método nº 1 - Com Cada Variável:

```
In [39]: # Identificando a Correlação de Cada Variável com as Demais

         # Permite Identificar Quais as Variáveis são Mais Relevantes para a Construção do Modelo Preditivo

         # Obs: Correlação não Implica, Necessariamente, em Causalidade

         def plot_corr(dataset_final, size=6):
             corr = dataset_final.corr()
             fig, ax = plt.subplots(figsize = (size, size))
             ax.matshow(corr)
             plt.xticks(range(len(corr.columns)), corr.columns)
             plt.yticks(range(len(corr.columns)), corr.columns)
```

```
In [40]:
# Gerando o Gráfico de Correlação
plot_corr(dataset_final)
```



Utilizando-se uma forma diferente de demonstrar esta correlação, optou-se por plotar uma tabela de valores, onde o valor:

- **+1:** Significa uma correlação positiva e forte. Ou seja, se o valor de uma variável aumenta, o da outra também aumenta.

- **0:** Significa ausência de correlação.

- **-1:** Significa uma correlação negativa e forte. Ou seja, se o valor de uma variável aumenta, o da outra diminui.

```
In [41]:
# Identificando a Correlação Através de uma Tabela de Valores
# Coeficiente de Correlação:
# +1  = Correlação Positiva Forte - Se o Valor de uma Variável Aumenta, o da Outra Também Aumenta
# 0   = Ausência de Correlação
# -1  = Correlação Negativa Forte - Se o Valor de uma Variável Aumenta, o da Outra Diminui
dataset_final.corr()
```

Out[41]:

	Populacao_do_Municipio	Numero_de_AFTs_na_Equipe	Quantidade_de_Trabalhadores_do_Empregador	Q
Populacao_do_Municipio	1.000000	0.012232	0.025727	
Numero_de_AFTs_na_Equipe	0.012232	1.000000	0.003387	
Quantidade_de_Trabalhadores_do_Empregador	0.025727	0.003387	1.000000	
Quantidade_de_Ementas_de_Legislação_Fiscalizadas	-0.036730	0.124200	-0.010844	
Quantidade_de_Ementas_de_SST_Fiscalizadas	-0.095588	0.054895	-0.004884	
Empregador_com_Irregularidade	-0.004657	0.023295	-0.018501	

4.2 Método nº 2 – Correlação de cada Variável com a Variável Target (A Ser Prevista)

Com o método nº 2, criou-se um modelo de *Machine Learning* para prever a importância de cada variável preditora em relação à variável target do modelo a ser implementado.

Para isso, importou-se o algoritmo **"Random Forest"** que, após treinado, foi capaz de atribuir pesos às variáveis preditoras no que tange a sua relevância.

Com os pesos de cada variável criou-se um dataset e, em seguida, um gráfico de barras para a demonstração, de maneira fácil e amigável, da relação de importância de cada variável preditora para a variável a ser prevista.

Método nº 2 - Com a Variável Target (A Ser Prevista):

In [42]:
```
# Utilizando o Feature Importance da Scikit Learn

from sklearn.ensemble import RandomForestClassifier
```

In [43]:
```
# Instanciando o Modelo para o Feature Importance (com 10 Árvores)

modelo_fi = RandomForestClassifier (n_estimators=10)
```

In [44]:
```
# Identificando as Variáveis Preditoras (X)

# Identificando a Variável Target (A Ser Prevista) (Y)

X = dataset_final.drop(['Empregador_com_Irregularidade'], axis=1)
Y = dataset_final['Empregador_com_Irregularidade']
```

In [45]:
```
# Treinando o Algoritmo do Modelo para o Feature Importance

modelo_fi.fit(X,Y)
```

Out[45]:
```
RandomForestClassifier(n_estimators=10)
```

In [46]:
```
# Visualizando a Importância de Cada Variável Preditora para a Variável Target (A Ser Prevista)

print(modelo_fi.feature_importances_)
```

```
[0.39347632 0.04316372 0.31979095 0.11031569 0.13325331]
```

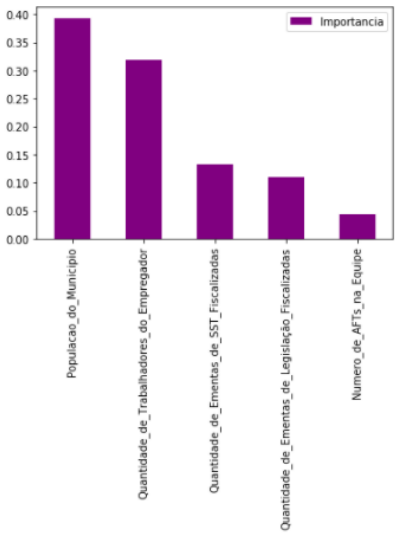
In [47]:
```
# Criando um Dataset com os Pesos de cada Variável Preditora Quanto à sua Importância

var_relevantes = pd.DataFrame(modelo_fi.feature_importances_, index = X.columns, columns = ['Importancia']).sort_values('Impc
```

In [48]:
```
# Visualizando o Dataset

var_relevantes
```

Out[48]:

	Importancia
Populacao_do_Municipio	0.393476
Quantidade_de_Trabalhadores_do_Empregador	0.319791
Quantidade_de_Ementas_de_SST_Fiscalizadas	0.133253
Quantidade_de_Ementas_de_Legislação_Fiscalizadas	0.110316
Numero_de_AFTs_na_Equipe	0.043164

```
In [49]: # Visualizando o Dataset em Forma de Gráfico

         var_relevantes.plot(kind='bar', color='purple')

Out[49]: <matplotlib.axes._subplots.AxesSubplot at 0x20b72d39108>
```



4.3 Outras Transformações e Tratamentos

Nesta etapa foi realizada a conversão da variável target (Empregador com Irregularidade) do tipo numérico para o tipo categórico. Vale lembrar que o modelo de *Machine Learning* a ser criado se enquadra na categoria de aprendizado supervisionado do tipo Classificação.

Convertendo a Variável Target (A Ser Prevista) do Tipo Numérico para Categórico (Classe)

```
In [50]: # Convertendo a Variável (Coluna) "Empregador_com_Irregularidade" do Dataset Final do Tipo Numérico para o Tipo Categórico (C

         dataset_final['Empregador_com_Irregularidade'] = dataset_final.Empregador_com_Irregularidade.astype('category')

In [51]: # Verificando o Tipo de Dado da Variável (Coluna) "Empregador_com_Irregularidade" do Dataset Final

         dataset_final.Empregador_com_Irregularidade.dtypes

Out[51]: CategoricalDtype(categories=[0, 1], ordered=False)

In [52]: # Verificando Todos os Tipos de Dados do Dataset Final

         dataset_final.dtypes

Out[52]: Populacao_do_Municipio                        int64
         Numero_de_AFTs_na_Equipe                      int64
         Quantidade_de_Trabalhadores_do_Empregador     int64
         Quantidade_de_Ementas_de_Legislação_Fiscalizadas    int64
         Quantidade_de_Ementas_de_SST_Fiscalizadas     int64
         Empregador_com_Irregularidade              category
         dtype: object
```

4.4 Outras Análises Exploratórias

Por fim, algumas outras análises foram feitas através da exploração dos dados do dataset final, como:

- As suas principais informações estatísticas:

Realizando Outras Análises dos Dados

In [53]:
```
# Analisando as Principais Informações Estatísticas do Dataset Final

dataset_final.describe()
```

Out[53]:

	Populacao_do_Municipio	Numero_de_AFTs_na_Equipe	Quantidade_de_Trabalhadores_do_Empregador	Quantidade_de_Ementas_de_Legislação_Fiscaliz
count	5.395600e+04	53956.000000	53956.000000	53956.00(
mean	1.438603e+06	1.630588	160.829917	3.80(
std	2.513905e+06	1.228238	2328.467321	4.69(
min	1.118000e+03	1.000000	1.000000	0.00(
25%	1.357830e+05	1.000000	5.000000	1.00(
50%	4.447840e+05	1.000000	13.000000	2.00(
75%	1.488252e+06	2.000000	44.000000	5.00(
max	1.232523e+07	26.000000	288236.000000	72.00(

- A relação de distribuição, em valores absolutos, entre as duas classes da variável target (Empregador com Irregularidade x Empregador sem Irregularidade):

In [54]:
```
# Verificando a Relação de Distribuição entre as Classes da Variável Target (A Ser Prevista)

dataset_final.Empregador_com_Irregularidade.value_counts()
```

Out[54]:
```
1    35123
0    18833
Name: Empregador_com_Irregularidade, dtype: int64
```

Sendo: ***1 para Empregador Irregular; e***

0 para Empregador Regular

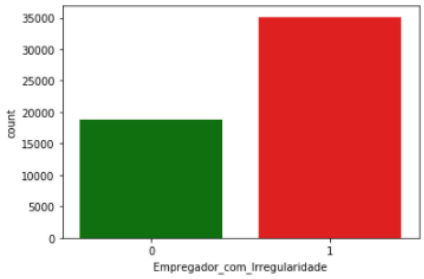
- A relação de distribuição, em valores percentuais, entre as duas classes da variável target (Empregador com Irregularidade x Empregador sem Irregularidade):

In [55]:
```
# Verificando a Proporção Referente à Relação de Distribuição entre as Classes da Variável Target (A Ser Prevista)

num_true = len(dataset_final.loc[dataset_final['Empregador_com_Irregularidade'] == 1])
num_false = len(dataset_final.loc[dataset_final['Empregador_com_Irregularidade'] == 0])

print("Número de Empregadores Irregulares: {0} ({1:2.2f}%)".format(num_true, (num_true/ (num_true + num_false)) * 100))
print("Número de Empregadores Regulares:   {0} ({1:2.2f}%)".format(num_false, (num_false/ (num_true + num_false)) * 100))
```

```
Número de Empregadores Irregulares: 35123 (65.10%)
Número de Empregadores Regulares:   18833 (34.90%)
```

- A relação de distribuição, na forma gráfica, entre as duas classes da variável target (Empregador com Irregularidade x Empregador sem Irregularidade):

```
In [56]: # Visualizando a Proporção Referente à Relação de Distribuição entre as Classes da Variável Target (A Ser Prevista) em Forma
         grafico = sns.countplot(x="Empregador_com_Irregularidade", palette=['g','r'], data=dataset_final)
```



Sendo: ***1 para Empregador Irregular; e***
0 para Empregador Regular

5. Criação de Modelos de *Machine Learning*

Após a coleta, processamento, tratamento, análise e exploração dos dados, é chegada a hora de avançar para a criação e construção dos modelos de *Machine Learning*.

Para este trabalho, foram escolhidos 3 (três) algoritmos para serem testados.

- **Naive Bayes**

  É um classificador probabilístico baseado no “Teorema de Bayes”.

  Possui a vantagem de ser muito simples e rápido, tendendo a possuir um desempenho relativamente maior do que outros classificadores.

  Como desvantagem, o Nayve Bayes possui uma forte suposição de independência condicional entre as variáveis.

- **Random Forest**

  O algoritmo Random Forest pode ser utilizado tanto para problemas de classificação quanto para regressão.

  Uma grande vantagem deste algoritmo é ser considerado fácil e acessível, possuindo hiperparâmetros que, com valores *defaut,* já produzem bons resultados de predição.

  Uma de suas desvantagens é a limitação ao ser setado com uma grande quantidade de árvores. Nestes casos, o algoritmo se torna lento e ineficiente para predições em tempo real. Em geral, este algoritmo é rápido para treinar, mas muito lento para fazer predições depois de treinado. Uma predição com mais acurácia requer mais árvores, o que faz o modelo ficar mais lento.

  Em muitas aplicações do mundo real o Random Forest é rápido o suficiente, mas pode certamente haver situações em que a performance em tempo de execução é importante, sendo outras abordagens mais apropriadas.

- **Regressão Logística**

  A regressão logística é um dos algoritmos de aprendizado de máquina mais simples e comumente usados para classificação de duas classes.

  A vantagem do algoritmo é que ele é fácil de implementar e interpretar, além de poder ser usado como linha de base para qualquer problema de classificação binária.

  Por sua natureza simples e eficiente, não requer alto poder de computação.

  Como desvantagens, a Regressão Logística não é capaz de lidar com muitas variáveis categóricas e é vulnerável ao Overfitting.

A regressão logística não terá um bom desempenho com variáveis independentes que não estão correlacionadas com a variável alvo e muito semelhantes ou correlacionadas entre si.

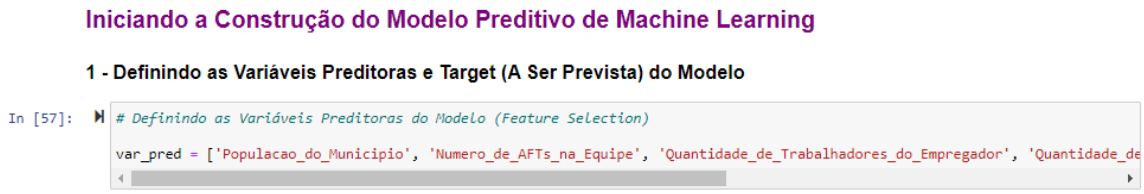
Explicados os modelos escolhidos para testes, antes de criá-los e treiná-los, algumas etapas necessárias são realizadas e detalhadas a seguir.

5.1 Definição das Variáveis Preditoras e Target (A Ser Prevista) do Modelo

Apesar de realizada a análise da importância de cada variável para o modelo, descrita nos itens 4.1 e 4.2 deste relatório, optou-se por manter todas elas, ainda que umas possuam pesos de relevância menores do que outras.

Assim, as seguintes variáveis foram definidas como preditoras:

- População do Município;
- Número de Auditores-Fiscais do Trabalho na Equipe;
- Quantidade de Trabalhadores do Empregador;
- Quantidade de Ementas de Legislação Fiscalizadas; e
- Quantidade de Ementas de Segurança e Saúde do Trabalho Fiscalizadas.



A variável target definida foi:

- Empregador com Irregularidade.

```
In [58]: # Definindo a Variável Target (A ser Prevista) do Modelo (Feature Selection)

         var_prev = ['Empregador_com_Irregularidade']
```

Após a definição das variáveis, como preditoras e target do modelo, foram coletados e visualizados os valores a elas atribuídos no dataset final.

```
In [59]: # Coletando os Valores das Variáveis "Preditoras" (X)

         # Coletando os Valores da Variável "A Ser Prevista (Target)" (Y)

         X = dataset_final[var_pred].values
         Y = dataset_final[var_prev].values
```

```
In [60]: # Visualizando o Conjunto das Variáveis "Preditoras" (X)

         X
```

```
Out[60]: array([[  668949,        1,     1569,        2,        0],
                [ 1499641,        2,      131,        9,        8],
                [12325232,        1,        1,        1,        0],
                ...,
                [  230371,        1,        2,        2,        0],
                [ 1108975,        3,        8,        0,       17],
                [ 1108975,        3,       64,        0,       17]], dtype=int64)
```

```
In [61]: # Visualizando o Conjunto da Variável "A Ser Prevista (Target)" (Y)

         Y
```

```
Out[61]: array([[1],
                [1],
                [1],
                ...,
                [0],
                [1],
                [1]], dtype=int64)
```

5.2 Divisão dos Dados em Treino e Teste

Em seguida, importou-se o módulo Scikit Learn do Python e sua função de divisão dos dados entre treino e teste, ***train_test_split***, do pacote Model Selection.

2 - Dividindo os Dados entre Treino e Teste

Dados de Treino: 70% (Treinamento do Algoritmo e Criação do Modelo de Predição)

Dados de Teste: 30% (Teste e Avaliação do Modelo de Predição)

```
In [62]: # Importando o Módulo Scikit Learn e Verificando sua Versão

         import sklearn as sk
         sk.__version__
```

```
Out[62]: '0.24.2'
```

```
In [63]: # Importando a Função de Divisão Entre Treino e Teste do Pacote Model Selection

         from sklearn.model_selection import train_test_split
```

Posteriormente, foi definida uma taxa de divisão de 70% para dados de treino e 30% para dados de teste.

```
In [64]: # Definindo a Taxa de Divisão (70% Treino / 30% Teste)

taxa_teste = 0.30
```

Com a chamada da função ***train_test_split()***, criou-se os dados de treino e de teste que, posteriormente, foram visualizados.

```
In [65]: # Chamando a Função Train_Test_Split

# Criando os Dados de Treino e de Teste

X_treino, X_teste, Y_treino, Y_teste = train_test_split(X, Y, test_size = taxa_teste, random_state = 42)
```

```
In [66]: # Verificando os Resultados

print("Dados de Treino: {0:0.2f}%".format((len(X_treino)/len(dataset_final.index)) * 100))
print("Dados de Teste: {0:0.2f}%".format((len(X_teste)/len(dataset_final.index)) * 100))
```

```
Dados de Treino: 70.00%
Dados de Teste: 30.00%
```

```
In [67]: # Visualizando o Conjunto de Dados de Treino

X_treino
```

```
Out[67]: array([[ 868075,       1,       6,       1,      20],
                [ 664908,       2,    1276,       8,       0],
                [ 211508,       1,     144,       8,       0],
                ...,
                [ 868075,       3,     545,       0,      14],
                [ 699097,       1,     152,       4,      28],
                [3055149,       1,       1,       1,       0]], dtype=int64)
```

```
In [68]: # Visualizando o Conjunto de Dados de Teste

X_teste
```

```
Out[68]: array([[ 573285,       1,       6,       1,      11],
                [ 365855,       2,    1071,       1,       0],
                [ 619609,       1,       5,      12,       0],
                ...,
                [ 391772,       2,       2,       3,       0],
                [2521564,       2,       7,      10,       7],
                [ 283542,       1,      13,      30,       7]], dtype=int64)
```

5.3 Verificação da Relação de Proporção entre os Dados

Por fim, verificou-se a relação de proporção entre as classes da variável target para os dados originais, os dados de treino e os dados de teste.

Vale ressaltar que a relação proporcional se manteve praticamente idêntica para todos os 3 (três) casos apresentados.

3 - Verificando as Divisões

In [69]:
```
# Verificando a Relação de Proporção Entre os Dados Originais, de Treino e de Teste

print("Dados Originais: Empregadores Irregulares: {0} ({1:0.2f}%)".format(len(dataset_final.loc[dataset_final['Empregador_com
                                                    (len(dataset_final.loc[dataset_final['Empregador_com_Irregularidade'] ==1])

print("Dados Originais: Empregadores Regulares: {0} ({1:0.2f}%)".format(len(dataset_final.loc[dataset_final['Empregador_com_I
                                                   (len(dataset_final.loc[dataset_final['Empregador_com_Irregularidade'] == 0])/
print("")
print("Dados de Treino: Empregadores Irregulares: {0} ({1:0.2f}%)".format(len(Y_treino[Y_treino[:] == 1]),
                                                    (len(Y_treino[Y_treino[:] == 1])/len(Y_treino) * 100)))

print("Dados de Treino: Empregadores Regulares: {0} ({1:0.2f}%)".format(len(Y_treino[Y_treino[:] == 0]),
                                                   (len(Y_treino[Y_treino[:] == 0])/len(Y_treino) * 100)))
print("")
print("Dados de Teste: Empregadores Irregulares: {0} ({1:0.2f}%)".format(len(Y_teste[Y_teste[:] == 1]),
                                                    (len(Y_teste[Y_teste[:] == 1])/len(Y_teste) * 100)))

print("Dados de Teste: Empregadores Regulares: {0} ({1:0.2f}%)".format(len(Y_teste[Y_teste[:] == 0]),
                                                   (len(Y_teste[Y_teste[:] == 0])/len(Y_teste) * 100)))
```

```
Dados Originais: Empregadores Irregulares: 35123 (65.10%)
Dados Originais: Empregadores Regulares: 18833 (34.90%)

Dados de Treino: Empregadores Irregulares: 24539 (64.97%)
Dados de Treino: Empregadores Regulares: 13230 (35.03%)

Dados de Teste: Empregadores Irregulares: 10584 (65.39%)
Dados de Teste: Empregadores Regulares: 5603 (34.61%)
```

5.4 Construindo e Treinando Modelos

Feitas as etapas descritas nos itens 5.1 a 5.3 é chegada a hora de criar, treinar e avaliar os modelos de classificação testados.

5.4.1 Método nº 1 - Sem Balanceamento dos Dados

Primeiramente, optou-se por testar algoritmos de classificação sem que os dados de treino fossem balanceados, ou equilibrados.

Assim, manteve-se a proporção de 64,97% e 35,03% para os dados de treino, sendo a parcela maior referente aos empregadores irregulares.

5.4.1.1 Modelo Preditivo nº 1 - Naive Bayes

Após a importação do algoritmo de classificação Naive Bayes, do módulo Scikit Learn, criou-se o primeiro modelo com o uso do método nº 1 (sem balanceamento dos dados).

Utilizando-se a função ***"fit()"*** treinou-se o modelo preditivo com os dados de treino (variáveis preditoras e variável target).

4 - Construindo e Treinando Modelos com o Método nº 1: Sem Balanceamento dos Dados:

Método nº 1 - Sem Balanceamento dos Dados

4.1 - Modelo Preditivo nº 1 - Naive Bayes

Criando e Treinando

```
In [70]: # Importando o Algoritmo de Classificação (Modelo) Naive Bayes
         from sklearn.naive_bayes import GaussianNB

In [71]: # Criando o Modelo Preditivo nº 1 - Naive Bayes
         modelo_1_metodo_1 = GaussianNB()

In [72]: # Treinando o Modelo Preditivo nº 1 - Naive Bayes
         modelo_1_metodo_1.fit(X_treino, Y_treino.ravel())

Out[72]: GaussianNB()
```

Em seguida, por meio da função ***"predict()"*** foram repassados ao modelo os dados de teste referentes às variáveis preditoras apenas.

O objetivo é testar o modelo fazendo-o prever os valores de saída.

Testando

```
In [73]: from sklearn import metrics

In [74]: # Fazendo as Previsões com o Modelo Passando, como Entrada, os Valores das Variáveis Preditoras
         previsao_modelo_1_metodo_1 = modelo_1_metodo_1.predict(X_teste)
```

Por fim, avaliou-se o desempenho do modelo verificando a sua acurácia e outras métricas através de ferramentas como:

- **Matriz de Confusão:**

Representa a quantidade de valores verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) previstos.

- **Relatório de Classificação:**

Retorna diversas métricas como a acurácia, que indica a performance geral do modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, variando de 0 a 100%; e o recall, que demonstra o número de vezes que uma classe foi

predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).

Vale lembrar que tanto a Matriz de Confusão como o Relatório de Classificação foram aplicados sobre os dados de teste, ou seja, sobre os dados que o modelo ainda não havia conhecido. Desta forma, tem-se um resultado mais real e preciso do algoritmo.

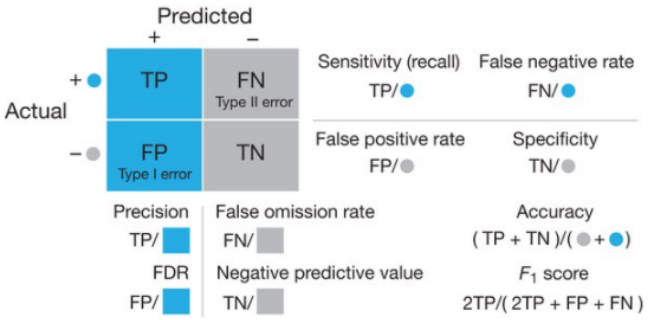
Avaliando

Verificando a Acurácia e Outras Métricas

Matriz de Confusão e Relatório de Classificação

In [75]:
```
# Imprimindo o Diagrama da Matriz de Confusão

from IPython.display import Image

Image('Matriz_de_Confusao.jpg')
```

Out[75]:



In [76]:
```
# Comparando os Valores da Variável Target Esperados (Y_teste) com os Valores da Variável Target Realmente Previstos pelo Mod

print("Exatidão (Acurácia): {0:.4f}".format(metrics.accuracy_score(Y_teste, previsao_modelo_1_metodo_1)))
print()
```

```
Exatidão (Acurácia): 0.6519
```

In [77]:
```
# Verificando a Matriz de Confusão e o Relatório de Classificação

print("Modelo Preditivo nº 1 - Naive Bayes")
print("")

print("Matriz de Confusão:")
print("")
print("{0}".format(metrics.confusion_matrix(Y_teste, previsao_modelo_1_metodo_1, labels = [1, 0])))
print("")

print("Relatório de Classificação:")
print("")
print(metrics.classification_report(Y_teste, previsao_modelo_1_metodo_1, labels = [1, 0]))
```

```
Modelo Preditivo nº 1 - Naive Bayes

Matriz de Confusão:

[[10505    79]
 [ 5555    48]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.65      0.99      0.79     10584
           0       0.38      0.01      0.02      5603

    accuracy                           0.65     16187
   macro avg       0.52      0.50      0.40     16187
weighted avg       0.56      0.65      0.52     16187
```

Observando as métricas, nota-se que embora o **Modelo Nayve Bayes, criado sem o balanceamento dos dados de treino**, tenha obtido uma acurácia de 65%, seu recall para o valor "0" foi extremamente baixo (0,01), indicando uma ineficiência do desempenho do algoritmo para a predizer um empregador sem irregularidade.

5.4.1.2 Modelo Preditivo nº 2 – Randon Forest

Após a importação do algoritmo de classificação Random Forest, do módulo Scikit Learn, criou-se o segundo modelo com o uso do método nº 1 (sem balanceamento dos dados).

Utilizando-se a função ***"fit()"*** treinou-se o modelo preditivo com os dados de treino (variáveis preditoras e variável target).

4.2 - Modelo Preditivo nº 2 - Random Forest

Criando e Treinando

```
In [78]: # Importando o Algoritmo de Classificação (Modelo) Random Forest
         from sklearn.ensemble import RandomForestClassifier

In [79]: # Criando o Modelo Preditivo nº 2 - Random Forest
         modelo_2_metodo_1 = RandomForestClassifier(random_state = 42)

In [80]: # Treinando o Modelo Preditivo nº 2 - Random Forest
         modelo_2_metodo_1.fit(X_treino, Y_treino.ravel())
Out[80]: RandomForestClassifier(random_state=42)
```

Em seguida, por meio da função ***"predict()"*** foram repassados ao modelo os dados de teste referentes às variáveis preditoras apenas.

O objetivo é testar o modelo fazendo-o prever os valores de saída.

Testando

```
In [81]: # Fazendo as Previsões com o Modelo Passando, como Entrada, os Valores das Variáveis Preditoras
         previsao_modelo_2_metodo_1 = modelo_2_metodo_1.predict(X_teste)
```

Por fim, avaliou-se o desempenho do modelo verificando a sua acurácia e outras métricas através de ferramentas como:

- **Matriz de Confusão:**

Representa a quantidade de valores verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) previstos.

- **Relatório de Classificação:**

Retorna diversas métricas como a acurácia, que indica a performance geral do Modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, variando de 0 a 100%; e o recall, que demonstra o número de vezes que uma classe foi predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).

Vale lembrar que tanto a Matriz de Confusão como o Relatório de Classificação foram aplicados sobre os dados de teste, ou seja, sobre os dados que o Modelo ainda não havia conhecido. Desta forma, tem-se um resultado mais real e preciso do algoritmo.

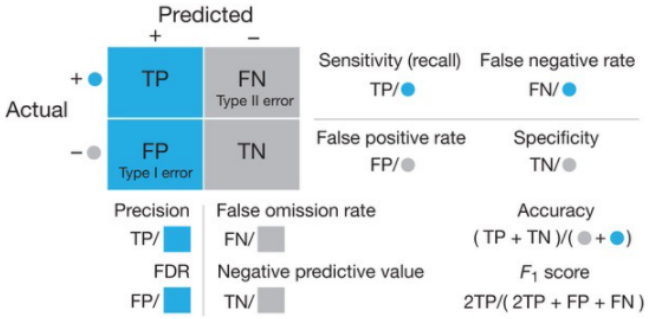
Avaliando

Verificando a Acurácia e Outras Métricas

Matriz de Confusão e Relatório de Classificação

In [82]:
```
# Imprimindo o Diagrama da Matriz de Confusão
from IPython.display import Image
Image('Matriz_de_Confusao.jpg')
```

Out[82]:



In [83]:
```
# Comparando os Valores da Variável Target Esperados (Y_teste) com os Valores da Variável Target Realmente Previstas pelo Mod
print("Exatidão (Acurácia): {0:.4f}".format(metrics.accuracy_score(Y_teste, previsao_modelo_2_metodo_1)))
print()
```

```
Exatidão (Acurácia): 0.7316
```

```
In [84]:
# Verificando a Matriz de Confusão e o Relatório de Classificação

print("Modelo Preditivo nº 2 - Random Forest")
print("")

print("Matriz de Confusão:")
print("")
print("{0}".format(metrics.confusion_matrix(Y_teste, previsao_modelo_2_metodo_1, labels = [1, 0])))
print("")

print("Relatório de Classificação:")
print("")
print(metrics.classification_report(Y_teste, previsao_modelo_2_metodo_1, labels = [1, 0]))
```

```
Modelo Preditivo nº 2 - Random Forest

Matriz de Confusão:

[[8712 1872]
 [2472 3131]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.78      0.82      0.80     10584
           0       0.63      0.56      0.59      5603

    accuracy                           0.73     16187
   macro avg       0.70      0.69      0.70     16187
weighted avg       0.73      0.73      0.73     16187
```

Observando as métricas, nota-se que o **Modelo Random Forest, criado sem o balanceamento dos dados de treino**, obteve uma excelente acurácia de 73%, e seu recall para os valores de saída, “0” e “1”, foi bem satisfatório, indicando uma melhora significativa na eficiência do desempenho do algoritmo de *Machine Learning*.

5.4.1.3 Modelo Preditivo nº 3 – Regressão Logística

Após a importação do algoritmo de classificação Regressão Logística, do módulo Scikit Learn, criou-se o terceiro modelo com o uso do método nº 1 (sem balanceamento dos dados).

Utilizando-se a função ***“fit()”*** treinou-se o modelo preditivo com os dados de treino (variáveis preditoras e variável target).

4.3 - Modelo Preditivo nº 3 - Regressão Logística

Criando e Treinando

```
In [85]:
# Importando o Algoritmo de Classificação (Modelo) Regressão Logística

from sklearn.linear_model import LogisticRegression
```

```
In [86]:
# Criando o Modelo Preditivo nº 3 - Regressão Logística

modelo_3_metodo_1 = LogisticRegression(C = 0.7, random_state = 42, max_iter = 1000)
```

```
In [87]:
# Treinando o Modelo Preditivo nº 3 - Regressão Logística

modelo_3_metodo_1.fit(X_treino, Y_treino.ravel())
```

```
Out[87]: LogisticRegression(C=0.7, max_iter=1000, random_state=42)
```

Em seguida, por meio da função ***"predict()"*** foram repassados ao modelo os dados de teste referentes às variáveis preditoras apenas.

O objetivo é testar o modelo fazendo-o prever os valores de saída.

Testando

```
In [88]: # Fazendo as Previsões com o Modelo Passando, como Entrada, os Valores das Variáveis Preditoras
         previsao_modelo_3_metodo_1 = modelo_3_metodo_1.predict(X_teste)
```

Por fim, avaliou-se o desempenho do modelo verificando a sua acurácia e outras métricas através de ferramentas como:

- **Matriz de Confusão:**

Representa a quantidade de valores verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) previstos.

- **Relatório de Classificação:**

Retorna diversas métricas como a acurácia, que indica a performance geral do Modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, variando de 0 a 100%; e o recall, que demonstra o número de vezes que uma classe foi predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).

Vale lembrar que tanto a Matriz de Confusão como o Relatório de Classificação foram aplicados sobre os dados de teste, ou seja, sobre os dados que o Modelo ainda não havia conhecido. Desta forma, tem-se um resultado mais real e preciso do algoritmo.

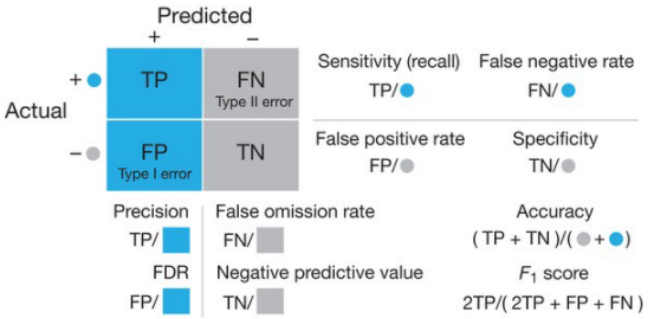
Avaliando

Verificando a Acurácia e Outras Métricas

Matriz de Confusão e Relatório de Classificação

In [89]:
```
# Imprimindo o Diagrama da Matriz de Confusão

from IPython.display import Image

Image('Matriz_de_Confusao.jpg')
```

Out[89]:



In [90]:
```
# Comparando os Valores da Variável Target Esperados (Y_teste) com os Valores da Variável Target Realmente Previstas pelo Mod

print("Exatidão (Acurácia): {0:.4f}".format(metrics.accuracy_score(Y_teste, previsao_modelo_3_metodo_1)))
print()
```

```
Exatidão (Acurácia): 0.6539
```

In [91]:
```
# Verificando a Matriz de Confusão e o Relatório de Classificação

print("Modelo Preditivo nº 3 - Regresão Logística")
print("")

print("Matriz de Confusão:")
print("")
print("{0}".format(metrics.confusion_matrix(Y_teste, previsao_modelo_3_metodo_1, labels = [1, 0])))
print("")

print("Relatório de Classificação:")
print("")
print(metrics.classification_report(Y_teste, previsao_modelo_3_metodo_1, labels = [1, 0]))
```

```
Modelo Preditivo nº 3 - Regresão Logística

Matriz de Confusão:

[[10584     0]
 [ 5603     0]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.65      1.00      0.79     10584
           0       0.00      0.00      0.00      5603

    accuracy                           0.65     16187
   macro avg       0.33      0.50      0.40     16187
weighted avg       0.43      0.65      0.52     16187
```

Observando as métricas, nota-se que embora o **Modelo de Regressão Logística, criado sem o balanceamento dos dados de treino**, tenha obtido uma acurácia de 65%, seu recall para o valor “0” foi péssimo (0,00), indicando uma completa ineficiência do desempenho do algoritmo para predizer um empregador sem irregularidade.

5.4.2 Método nº 2 - Com Balanceamento dos Dados - UNDER SAMPLING

Por meio deste método foram testados algoritmos de classificação com dados de treino balanceados, ou equilibrados.

A distribuição de classes dos dados de treino original estava bastante distorcida. Isso faz com que o algoritmo de *Machine Learning* aprenda mais sobre uma classificação do que sobre outra, gerando um modelo viciado.

Por tal razão, optou-se por equilibrar esses dados utilizando-se de uma técnica conhecida como **Under Sampling**.

O **Under Sampling** permite acabar com o desbalanceamento do conjunto de dados focando na classe majoritária. Desta forma o quantitativo de dados desta classe é reduzido a proporções iguais à classe minoritária.

Após a importação da função ***"RandomUnderSampler()"*** do módulo Imblearn, instanciou-se um objeto que, após treinado, viabilizou a distribuição de 50% dos dados para cada classe da variável target.

5 - Construindo e Treinando Modelos com o Método nº 2: Com Balanceamento dos Dados - Under Sampling:

Método nº 2 - Aplicando o Under Sampling - Random Under Sampling

Equililbrando os Dados de Treino (50% para cada Classe):

In [92]:
```
import warnings
warnings.filterwarnings("ignore",category=DeprecationWarning )
```

In [93]:
```
!pip install imbalanced-learn
```

```
Requirement already satisfied: imbalanced-learn in c:\users\lucas prado\anaconda3\lib\site-packages (0.8.0)
Requirement already satisfied: scipy>=0.19.1 in c:\users\lucas prado\anaconda3\lib\site-packages (from imbalanced-learn) (1.
4.1)
Requirement already satisfied: numpy>=1.13.3 in c:\users\lucas prado\anaconda3\lib\site-packages (from imbalanced-learn) (1.
18.1)
Requirement already satisfied: scikit-learn>=0.24 in c:\users\lucas prado\anaconda3\lib\site-packages (from imbalanced-lear
n) (0.24.2)
Requirement already satisfied: joblib>=0.11 in c:\users\lucas prado\anaconda3\lib\site-packages (from imbalanced-learn) (0.1
4.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\lucas prado\anaconda3\lib\site-packages (from scikit-learn>=
0.24->imbalanced-learn) (2.2.0)
```

In [94]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [95]:
```
from imblearn.under_sampling import RandomUnderSampler
```

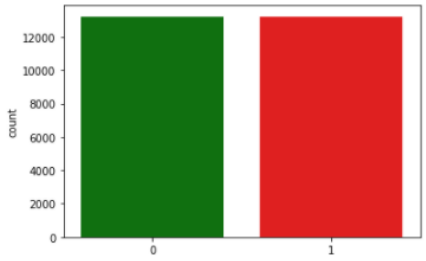
In [96]:
```
us = RandomUnderSampler()
```

In [97]:
```
X_treino_us, Y_treino_us = us.fit_resample(X_treino, Y_treino)
```

```
In [98]: # Checando o Balanceamento e a Quantidade em Cada Classe da Variável Target (A Ser Prevista)

print("Dados de Treino: Empregadores Irregulares: {0} ({1:0.2f}%)".format(len(Y_treino_us[Y_treino_us[:] == 1]),
                                                    (len(Y_treino_us[Y_treino_us[:] == 1])/len(Y_treino_us) * 100)))

print("Dados de Treino: Empregadores Regulares: {0} ({1:0.2f}%)".format(len(Y_treino_us[Y_treino_us[:] == 0]),
                                                   (len(Y_treino_us[Y_treino_us[:] == 0])/len(Y_treino_us) * 100)))
```

```
Dados de Treino: Empregadores Irregulares: 13230 (50.00%)
Dados de Treino: Empregadores Regulares: 13230 (50.00%)
```

```
In [99]: # Verificando o Balanceamento em Forma de Gráfico

ax = sns.countplot(x=Y_treino_us, palette=['g','r'])
```



5.4.2.1 Modelo Preditivo nº 1 - Naive Bayes

Em seguida, criou-se e treinou-se, com os dados de treino balanceados (variáveis preditoras e variável target), o primeiro modelo com o uso do método nº 2 (com balanceamento dos dados – Under Sampling).

5.1 - Modelo Preditivo nº 1 - Naive Bayes

Criando e Treinando

```
In [100]: # Criando o Modelo Preditivo nº 1 - Naive Bayes

modelo_1_metodo_2 = GaussianNB()
```

```
In [101]: # Treinando o Modelo Preditivo nº 1 - Naive Bayes

modelo_1_metodo_2.fit(X_treino_us, Y_treino_us.ravel())
```

```
Out[101]: GaussianNB()
```

Em seguida, por meio da função ***"predict()"*** foram repassados ao modelo os dados de teste referentes às variáveis preditoras apenas.

O objetivo é testar o modelo fazendo-o prever os valores de saída.

Testando

```
In [102]: # Fazendo as Previsões com o Modelo Passando, como Entrada, os Valores das Variáveis Preditoras

previsao_modelo_1_metodo_2 = modelo_1_metodo_2.predict(X_teste)
```

Por fim, avaliou-se o desempenho do Modelo verificando a sua acurácia e outras métricas através de ferramentas como:

- **Matriz de Confusão:**

Representa a quantidade de valores verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) previstos.

- **Relatório de Classificação:**

Retorna diversas métricas como a acurácia, que indica a performance geral do Modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, variando de 0 a 100%; e o recall, que demonstra o número de vezes que uma classe foi predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).

Vale lembrar que tanto a Matriz de Confusão como o Relatório de Classificação foram aplicados sobre os dados de teste, ou seja, sobre os dados que o Modelo ainda não havia conhecido. Desta forma, tem-se um resultado mais real e preciso do algoritmo.

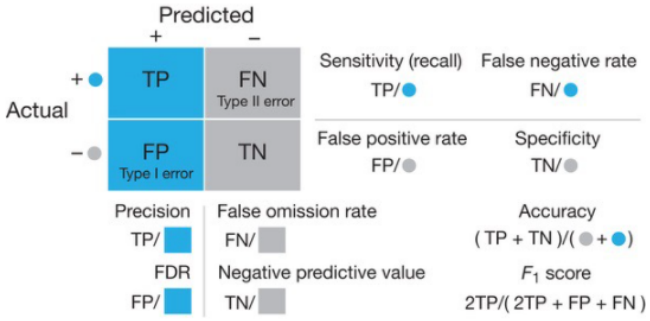
Avaliando

Verificando a Acurácia e Outras Métricas

Matriz de Confusão e Relatório de Classificação

In [103]:
```
# Imprimindo o Diagrama da Matriz de Confusão
from IPython.display import Image
Image('Matriz_de_Confusao.jpg')
```

Out[103]:



In [104]:
```
# Comparando os Valores da Variável Target Esperados (Y_teste) com os Valores da Variável Target Realmente Previstos pelo Moc
print("Exatidão (Acurácia): {0:.4f}".format(metrics.accuracy_score(Y_teste, previsao_modelo_1_metodo_2)))
print()
```

```
Exatidão (Acurácia): 0.6521
```

```
In [105]: # Verificando a Matriz de Confusão e o Relatório de Classificação

print("Modelo Preditivo nº 1 - Naive Bayes")
print("")

print("Matriz de Confusão:")
print("")
print("{0}".format(metrics.confusion_matrix(Y_teste, previsao_modelo_1_metodo_2, labels = [1, 0])))
print("")

print("Relatório de Classificação:")
print("")
print(metrics.classification_report(Y_teste, previsao_modelo_1_metodo_2, labels = [1, 0]))
```

```
Modelo Preditivo nº 1 - Naive Bayes

Matriz de Confusão:

[[10500    84]
 [ 5548    55]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.65      0.99      0.79     10584
           0       0.40      0.01      0.02      5603

    accuracy                           0.65     16187
   macro avg       0.52      0.50      0.40     16187
weighted avg       0.56      0.65      0.52     16187
```

Observando as métricas, nota-se que embora o **Modelo Nayve Bayes, criado com o balanceamento dos dados de treino por meio da técnica Under Sampling**, tenha obtido uma acurácia de 65%, seu recall para o valor "0" foi extremamente baixo (0,01), indicando uma ineficiência do desempenho do algoritmo para predizer um empregador sem irregularidade.

5.4.2.2 Modelo Preditivo nº 2 - Randon Forest

Uma vez já importado o algoritmo de classificação Random Forest, do módulo Scikit Learn, criou-se o segundo modelo com o uso do método nº 2 (com balanceamento dos dados – Under Sampling).

Utilizando-se a função ***"fit()"*** treinou-se o modelo preditivo com os dados de treino (variáveis preditoras e variável target).

5.2 - Modelo Preditivo nº 2 - Random Forest

Criando e Treinando

```
In [106]: # Criando o Modelo Preditivo nº 2 - Random Forest

modelo_2_metodo_2 = RandomForestClassifier(random_state = 42)
```

```
In [107]: # Treinando o Modelo Preditivo nº 2 - Random Forest

modelo_2_metodo_2.fit(X_treino_us, Y_treino_us.ravel())
```

```
Out[107]: RandomForestClassifier(random_state=42)
```

Em seguida, por meio da função ***"predict()"*** foram repassados ao modelo os dados de teste referentes às variáveis preditoras apenas.

O objetivo é testar o modelo fazendo-o prever os valores de saída.

Testando

In [108]:
```
# Fazendo as Previsões com o Modelo Passando, como Entrada, os Valores das Variáveis Preditoras
previsao_modelo_2_metodo_2 = modelo_2_metodo_2.predict(X_teste)
```

Por fim, avaliou-se o desempenho do Modelo verificando a sua acurácia e outras métricas através de ferramentas como:

- **Matriz de Confusão:**

Representa a quantidade de valores verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) previstos.

- **Relatório de Classificação:**

Retorna diversas métricas como a acurácia, que indica a performance geral do Modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, variando de 0 a 100%; e o recall, que demonstra o número de vezes que uma classe foi predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).

Vale lembrar que tanto a Matriz de Confusão como o Relatório de Classificação foram aplicados sobre os dados de teste, ou seja, sobre os dados que o Modelo ainda não havia conhecido. Desta forma, tem-se um resultado mais real e preciso do algoritmo.

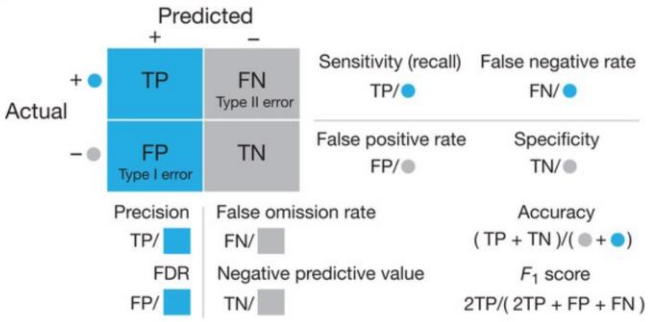
Avaliando

Verificando a Acurácia e Outras Métricas

Matriz de Confusão e Relatório de Classificação

In [109]:
```
# Imprimindo o Diagrama da Matriz de Confusão

from IPython.display import Image

Image('Matriz_de_Confusao.jpg')
```

Out[109]:



In [110]:
```
# Comparando os Valores da Variável Target Esperados (Y_teste) com os Valores da Variável Target Realmente Previstas pelo Mod

print("Exatidão (Acurácia): {0:.4f}".format(metrics.accuracy_score(Y_teste, previsao_modelo_2_metodo_2)))
print()
```

```
Exatidão (Acurácia): 0.6910
```

In [111]:
```
# Verificando a Matriz de Confusão e o Relatório de Classificação

print("Modelo Preditivo nº 2 - Random Forest")
print("")

print("Matriz de Confusão:")
print("")
print("{0}".format(metrics.confusion_matrix(Y_teste, previsao_modelo_2_metodo_2, labels = [1, 0])))
print("")

print("Relatório de Classificação:")
print("")
print(metrics.classification_report(Y_teste, previsao_modelo_2_metodo_2, labels = [1, 0]))
```

```
Modelo Preditivo nº 2 - Random Forest

Matriz de Confusão:

[[7262 3322]
 [1679 3924]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.81      0.69      0.74     10584
           0       0.54      0.70      0.61      5603

    accuracy                           0.69     16187
   macro avg       0.68      0.69      0.68     16187
weighted avg       0.72      0.69      0.70     16187
```

Observando as métricas, nota-se que o **Modelo Random Forest, criado com o balanceamento dos dados de treino por meio da técnica Under Sampling**, obteve uma acurácia de 69%, e seu recall para os valores de saída, “0” e “1”, foi bem satisfatório, indicando uma melhora significativa na eficiência do desempenho do algoritmo de *Machine Learning*.

5.4.2.3 Modelo Preditivo nº 3 - Regressão Logística

Uma vez já importado o algoritmo de classificação de Regressão Logística, do módulo Scikit Learn, criou-se o terceiro modelo com o uso do método nº 2 (com balanceamento dos dados – Under Sampling).

Utilizando-se a função ***"fit()"*** treinou-se o modelo preditivo com os dados de treino (variáveis preditoras e variável target).

5.3 - Modelo Preditivo nº 3 - Regressão Logística

Criando e Treinando

```
In [112]: # Criando o Modelo Preditivo nº 3 - Regressão Logística
          modelo_3_metodo_2 = LogisticRegression(C = 0.7, random_state = 42, max_iter = 1000)

In [113]: # Treinando o Modelo Preditivo nº 3 - Regressão Logística
          modelo_3_metodo_2.fit(X_treino_us, Y_treino_us.ravel())

Out[113]: LogisticRegression(C=0.7, max_iter=1000, random_state=42)
```

Em seguida, por meio da função ***"predict()"*** foram repassados ao Modelo os dados de teste referentes às variáveis preditoras apenas.

O objetivo é testar o modelo fazendo-o prever os valores de saída.

Testando

```
In [114]: # Fazendo as Previsões com o Modelo Passando, como Entrada, os Valores das Variáveis Preditoras
          previsao_modelo_3_metodo_2 = modelo_3_metodo_2.predict(X_teste)
```

Por fim, avaliou-se o desempenho do Modelo verificando a sua acurácia e outras métricas através de ferramentas como:

- **Matriz de Confusão:**

Representa a quantidade de valores verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) previstos.

- **Relatório de Classificação:**

Retorna diversas métricas como a acurácia, que indica a performance geral do Modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, variando de 0 a 100%; e o recall, que demonstra o número de vezes que uma classe foi predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).

Vale lembrar que tanto a Matriz de Confusão como o Relatório de Classificação foram aplicados sobre os dados de teste, ou seja, sobre os dados que o Modelo ainda não havia conhecido. Desta forma, tem-se um resultado mais real e preciso do algoritmo.

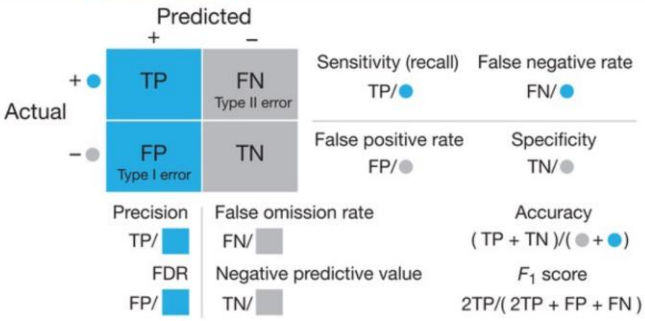
Avaliando

Verificando a Acurácia e Outras Métricas

Matriz de Confusão e Relatório de Classificação

In [115]:
```
# Imprimindo o Diagrama da Matriz de Confusão
from IPython.display import Image
Image('Matriz_de_Confusao.jpg')
```

Out[115]:



In [116]:
```
# Comparando os Valores da Variável Target Esperados (Y_teste) com os Valores da Variável Target Realmente Previstas pelo Mod
print("Exatidão (Acurácia): {0:.4f}".format(metrics.accuracy_score(Y_teste, previsao_modelo_3_metodo_2)))
print()
```

```
Exatidão (Acurácia): 0.3462
```

```
In [117]:
# Verificando a Matriz de Confusão e o Relatório de Classificação

print("Modelo Preditivo nº 3 - Regresão Logística")
print("")

print("Matriz de Confusão:")
print("")
print("{0}".format(metrics.confusion_matrix(Y_teste, previsao_modelo_3_metodo_2, labels = [1, 0])))
print("")

print("Relatório de Classificação:")
print("")
print(metrics.classification_report(Y_teste, previsao_modelo_3_metodo_2, labels = [1, 0]))
```

```
Modelo Preditivo nº 3 - Regresão Logística

Matriz de Confusão:

[[    1 10583]
 [    0  5603]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       1.00      0.00      0.00     10584
           0       0.35      1.00      0.51      5603

    accuracy                           0.35     16187
   macro avg       0.67      0.50      0.26     16187
weighted avg       0.77      0.35      0.18     16187
```

Observando as métricas, nota-se que o **Modelo de Regressão Logística, criado com o balanceamento dos dados de treino por meio da técnica Under Sampling**, obteve resultados muito ruins. Com acurácia de apenas 35% e recall igual a 0,00 para o valor “1”, o modelo se mostrou bastante ineficiente.

5.4.3 Método nº 3 – Com Balanceamento dos Dados – OVER SAMPLING

Por meio deste método, também foram testados algoritmos de classificação com dados de treino balanceados, ou equilibrados.

No entanto, a técnica utilizada para os testes dos Modelos foi o **Over Sampling**.

O **Over Sampling** permite acabar com o desbalanceamento do conjunto de dados focando na classe minoritária. Desta forma, por meio da seleção aleatória de exemplos da classe minoritária, o quantitativo de dados desta classe é elevado a proporções iguais à classe majoritária.

Após a importação da função ***“SMOTE()”*** do módulo Imblearn, instanciou-se um objeto que, após treinado, viabilizou a distribuição de 50% dos dados para cada classe da variável target.

6 - Construindo e Treinando Modelos com o Método nº 3: Com Balanceamento dos Dados - Over Sampling:

Método nº 3 - Aplicando o Over Sampling - Synthetic Minority Oversampling Technique - SMOTE

Equililbrando os Dados de Treino (50% para cada Classe):

In [118]:
```
from imblearn.over_sampling import SMOTE
```

In [119]:
```
os = SMOTE()
```

In [120]:
```
X_treino_os, Y_treino_os = os.fit_resample(X_treino, Y_treino)
```

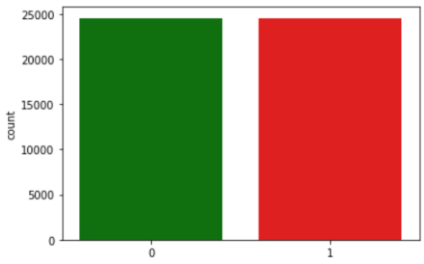
In [121]:
```
np.bincount(Y_treino_os)
```

Out[121]:
```
array([24539, 24539], dtype=int64)
```

In [122]:
```
# Checando o Balanceamento e a Quantidade em Cada Classe da Variável Target (A Ser Prevista)

print("Dados de Treino: Empregadores Irregulares: {0} ({1:0.2f}%)".format(len(Y_treino_os[Y_treino_os[:] == 1]),
                                                    (len(Y_treino_os[Y_treino_os[:] == 1])/len(Y_treino_os) * 100)))

print("Dados de Treino: Empregadores Regulares: {0} ({1:0.2f}%)".format(len(Y_treino_os[Y_treino_os[:] == 0]),
                                                   (len(Y_treino_os[Y_treino_os[:] == 0])/len(Y_treino_os) * 100)))
```

```
Dados de Treino: Empregadores Irregulares: 24539 (50.00%)
Dados de Treino: Empregadores Regulares: 24539 (50.00%)
```

In [123]:
```
# Verificando o Balanceamento em Forma de Gráfico

ax = sns.countplot(x=Y_treino_os, palette=['g','r'])
```



5.4.3.1 Modelo Preditivo nº 1 - Naive Bayes

Em seguida, criou-se e treinou-se, com os dados de treino balanceados (variáveis preditoras e variável target), o primeiro modelo com o uso do método nº 3 (com balanceamento dos dados – Over Sampling).

6.1 - Modelo Preditivo nº 1 - Naive Bayes

Criando e Treinando

In [124]:
```
# Criando o Modelo Preditivo nº 1 - Naive Bayes

modelo_1_metodo_3 = GaussianNB()
```

In [125]:
```
# Treinando o Modelo Preditivo nº 1 - Naive Bayes

modelo_1_metodo_3.fit(X_treino_os, Y_treino_os.ravel())
```

Out[125]:
```
GaussianNB()
```

Após, por meio da função **“predict()”** foram repassados ao modelo os dados de teste referentes às variáveis preditoras apenas.

O objetivo é testar o modelo fazendo-o prever os valores de saída.

Testando

```
In [126]: # Fazendo as Previsões com o Modelo Passando, como Entrada, os Valores das Variáveis Preditoras
          previsao_modelo_1_metodo_3 = modelo_1_metodo_3.predict(X_teste)
```

Por fim, avaliou-se o desempenho do Modelo verificando a sua acurácia e outras métricas através de ferramentas como:

- **Matriz de Confusão:**

Representa a quantidade de valores verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) previstos.

- **Relatório de Classificação:**

Retorna diversas métricas como a acurácia, que indica a performance geral do Modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, variando de 0 a 100%; e o recall, que demonstra o número de vezes que uma classe foi predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).

Vale lembrar que tanto a Matriz de Confusão como o Relatório de Classificação foram aplicados sobre os dados de teste, ou seja, sobre os dados que o Modelo ainda não havia conhecido. Desta forma, tem-se um resultado mais real e preciso do algoritmo.

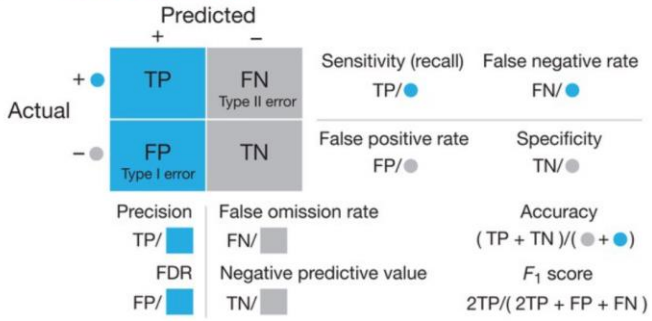
Avaliando

Verificando a Acurácia e Outras Métricas

Matriz de Confusão e Relatório de Classificação

In [127]:
```
# Imprimindo o Diagrama da Matriz de Confusão

from IPython.display import Image

Image('Matriz_de_Confusao.jpg')
```

Out[127]:



In [128]:
```
# Comparando os Valores da Variável Target Esperados (Y_teste) com os Valores da Variável Target Realmente Previstas pelo Moc

print("Exatidão (Acurácia): {0:.4f}".format(metrics.accuracy_score(Y_teste, previsao_modelo_1_metodo_3)))
print()
```

```
Exatidão (Acurácia): 0.6512
```

In [129]:
```
# Verificando a Matriz de Confusão e o Relatório de Classificação

print("Modelo Preditivo nº 1 - Naive Bayes")
print("")

print("Matriz de Confusão:")
print("")
print("{0}".format(metrics.confusion_matrix(Y_teste, previsao_modelo_1_metodo_3, labels = [1, 0])))
print("")

# print (pd.crosstab(Y_teste, teste_previsao_nb, rownames=['Real'], colnames=['Predito'], margins=True))

print("Relatório de Classificação:")
print("")
print(metrics.classification_report(Y_teste, previsao_modelo_1_metodo_3, labels = [1, 0]))
```

```
Modelo Preditivo nº 1 - Naive Bayes

Matriz de Confusão:

[[10482   102]
 [ 5544    59]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.65      0.99      0.79     10584
           0       0.37      0.01      0.02      5603

    accuracy                           0.65     16187
   macro avg       0.51      0.50      0.40     16187
weighted avg       0.55      0.65      0.52     16187
```

Observando as métricas, nota-se que embora o **Modelo Nayve Bayes, criado com o balanceamento dos dados de treino por meio da técnica Over Sampling**, tenha obtido uma acurácia de 65%, seu recall para o valor “0” foi extremamente baixo (0,01), indicando uma ineficiência do desempenho do algoritmo para predizer um empregador sem irregularidade.

5.4.3.2 Modelo Preditivo nº 2 - Randon Forest

Uma vez já importado o algoritmo de classificação Random Forest, do módulo Scikit Learn, criou-se o segundo modelo com o uso do método nº 3 (com balanceamento dos dados – Over Sampling).

Utilizando-se a função ***"fit()"*** treinou-se o modelo preditivo com os dados de treino (variáveis preditoras e variável target).

6.2 - Modelo Preditivo nº 2 - Random Forest

Criando e Treinando

```
In [130]: # Criando o Modelo Preditivo nº 2 - Random Forest
          modelo_2_metodo_3 = RandomForestClassifier(random_state = 42)

In [131]: # Treinando o Modelo Preditivo nº 2 - Random Forest
          modelo_2_metodo_3.fit(X_treino_os, Y_treino_os.ravel())

Out[131]: RandomForestClassifier(random_state=42)
```

Em seguida, por meio da função ***"predict()"*** foram repassados ao modelo os dados de teste referentes às variáveis preditoras apenas.

O objetivo é testar o modelo fazendo-o prever os valores de saída.

Testando

```
In [132]: # Fazendo as Previsões com o Modelo Passando, como Entrada, os Valores das Variáveis Preditoras
          previsao_modelo_2_metodo_3 = modelo_2_metodo_3.predict(X_teste)
```

Por fim, avaliou-se o desempenho do Modelo verificando a sua acurácia e outras métricas através de ferramentas como:

- **Matriz de Confusão:**

Representa a quantidade de valores verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) previstos.

- **Relatório de Classificação:**

Retorna diversas métricas como a acurácia, que indica a performance geral do Modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, variando de 0 a 100%; e o recall, que demonstra o número de vezes que uma classe foi predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).

Vale lembrar que tanto a Matriz de Confusão como o Relatório de Classificação foram aplicados sobre os dados de teste, ou seja, sobre os dados que o Modelo ainda não havia conhecido. Desta forma, tem-se um resultado mais real e preciso do algoritmo.

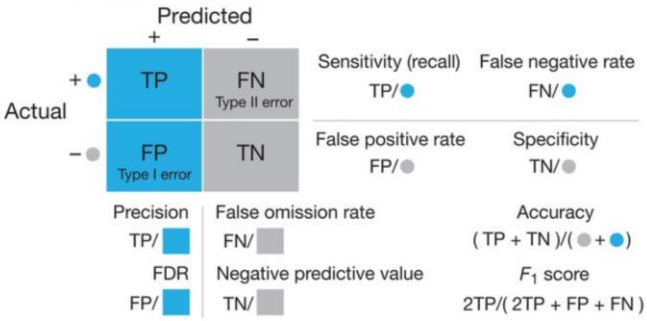
Avaliando

Verificando a Acurácia e Outras Métricas

Matriz de Confusão e Relatório de Classificação

In [133]:
```
# Imprimindo o Diagrama da Matriz de Confusão
from IPython.display import Image
Image('Matriz_de_Confusao.jpg')
```

Out[133]:



In [134]:
```
# Comparando os Valores da Variável Target Esperados (Y_teste) com os Valores da Variável Target Realmente Previstas pelo Moc
print("Exatidão (Acurácia): {0:.4f}".format(metrics.accuracy_score(Y_teste, previsao_modelo_2_metodo_3)))
print()
```

```
Exatidão (Acurácia): 0.7066
```

```
In [135]:
# Verificando a Matriz de Confusão e o Relatório de Classificação

print("Modelo Preditivo nº 2")
print("")

print("Matriz de Confusão:")
print("")
print("{0}".format(metrics.confusion_matrix(Y_teste, previsao_modelo_2_metodo_3, labels = [1, 0])))
print("")

print("Relatório de Classificação:")
print("")
print(metrics.classification_report(Y_teste, previsao_modelo_2_metodo_3, labels = [1, 0]))
```

```
Modelo Preditivo nº 2

Matriz de Confusão:

[[7755 2829]
 [1921 3682]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.80      0.73      0.77     10584
           0       0.57      0.66      0.61      5603

    accuracy                           0.71     16187
   macro avg       0.68      0.69      0.69     16187
weighted avg       0.72      0.71      0.71     16187
```

Observando as métricas, nota-se que o **Modelo Random Forest, criado com o balanceamento dos dados de treino por meio da técnica Over Sampling**, obteve uma acurácia de 71%, e seu recall para os valores de saída, “0” e “1”, foi bem satisfatório, indicando uma melhora significativa na eficiência do desempenho do algoritmo de *Machine Learning*.

5.4.3.3 Modelo Preditivo nº 3 - Regressão Logística

Uma vez já importado o algoritmo de classificação de Regressão Logística, do módulo Scikit Learn, criou-se o terceiro modelo com o uso do método nº 3 (com balanceamento dos dados – Over Sampling).

Utilizando-se a função fit() treinou-se o modelo preditivo com os dados de treino (variáveis preditoras e variável target).

6.3 - Modelo Preditivo nº 3 - Regressão Logística

Criando e Treinando

```
In [136]:
# Criando o Modelo Preditivo nº 3 - Regressão Logística

modelo_3_metodo_3 = LogisticRegression(C = 0.7, random_state = 42, max_iter = 1000)
```

```
In [137]:
# Treinando o Modelo Preditivo nº 3 - Regressão Logística

modelo_3_metodo_3.fit(X_treino_os, Y_treino_os.ravel())
```

```
Out[137]: LogisticRegression(C=0.7, max_iter=1000, random_state=42)
```

Em seguida, por meio da função ***"predict()"*** foram repassados ao Modelo os dados de teste referentes às variáveis preditoras apenas.

O objetivo é testar o modelo fazendo-o prever os valores de saída.

Testando

```
In [138]: # Fazendo as Previsões com o Modelo Passando, como Entrada, os Valores das Variáveis Preditoras
          previsao_modelo_3_metodo_3 = modelo_3_metodo_3.predict(X_teste)
```

Por fim, avaliou-se o desempenho do Modelo verificando a sua acurácia e outras métricas através de ferramentas como:

- **Matriz de Confusão:**

Representa a quantidade de valores verdadeiros positivos (TP), falsos positivos (FP), verdadeiros negativos (TN) e falsos negativos (FN) previstos.

- **Relatório de Classificação:**

Retorna diversas métricas como a acurácia, que indica a performance geral do Modelo, ou seja, dentre todas as classificações, quantas o modelo classificou corretamente, variando de 0 a 100%; e o recall, que demonstra o número de vezes que uma classe foi predita corretamente (TP) dividido pelo número de vezes que a classe aparece no dado de teste (FN).

Vale lembrar que tanto a Matriz de Confusão como o Relatório de Classificação foram aplicados sobre os dados de teste, ou seja, sobre os dados que o Modelo ainda não havia conhecido. Desta forma, tem-se um resultado mais real e preciso do algoritmo.

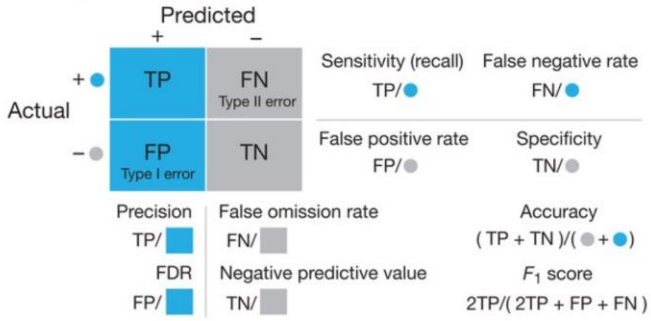
Avaliando

Verificando a Acurácia e Outras Métricas

Matriz de Confusão e Relatório de Classificação

In [139]:
```
# Imprimindo o Diagrama da Matriz de Confusão

from IPython.display import Image

Image('Matriz_de_Confusao.jpg')
```

Out[139]:



In [140]:
```
# Comparando os Valores da Variável Target Esperados (Y_teste) com os Valores da Variável Target Realmente Previstas pelo Moc

print("Exatidão (Acurácia): {0:.4f}".format(metrics.accuracy_score(Y_teste, previsao_modelo_3_metodo_3)))
print()
```

```
Exatidão (Acurácia): 0.3464
```

In [141]:
```
# Verificando a Matriz de Confusão e o Relatório de Classificação

print("Modelo Preditivo nº 3 - Regressão Logística")
print("")

print("Matriz de Confusão:")
print("")
print("{0}".format(metrics.confusion_matrix(Y_teste, previsao_modelo_3_metodo_3, labels = [1, 0])))
print("")

print("Relatório de Classificação:")
print("")
print(metrics.classification_report(Y_teste, previsao_modelo_3_metodo_3, labels = [1, 0]))
```

```
Modelo Preditivo nº 3 - Regressão Logística

Matriz de Confusão:

[[    4 10580]
 [    0  5603]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       1.00      0.00      0.00     10584
           0       0.35      1.00      0.51      5603

    accuracy                           0.35     16187
   macro avg       0.67      0.50      0.26     16187
weighted avg       0.77      0.35      0.18     16187
```

Observando as métricas, nota-se que o **Modelo de Regressão Logística, criado com o balanceamento dos dados de treino por meio da técnica Over Sampling**, também obteve resultados ruins. Com acurácia de apenas 35% e recall igual a 0,00 para o valor “1”, o modelo se mostrou bastante ineficiente.

6. Interpretação dos Resultados

6.1 Resultados da Análise e Exploração dos Dados

Durante a análise e exploração dos dados foi possível:

- **Verificar a inexistência de valores nulos, ou ausentes, no conjunto de dados**

Utilizando-se o comando ***"dataset_3.isnull().values.any()"*** foi possível constatar a inexistência de valores nulos, o que foi posteriormente confirmada por meio do uso de outro comando, ***"dataset_3.isnull().sum()"***, para a visualização de forma individual para cada variável.

Algoritmos de *Machine Learnig* não são capazes de lidar com valores ausentes. Sendo assim, é importantíssimo, caso o conjunto de dados possua valores ausentes, limpar esses dados (Data Clearing).

Para o presente caso, não houve a necessidade de realizar o Data Clearing, uma vez que não havia nenhum valor nulo no conjunto de dados analisado.

- **Verificar a inexistência de valores duplicados no conjunto de dados**

Utilizando-se o comando ***"dataset_3.index.duplicated().sum()*** foi possível constatar, também, a inexistência de valores em duplicidade.

Registros duplicados são desnecessários e só tendem a poluir o conjunto de dados. Por tal razão, é importante excluí-los.

No entanto, conforme observado, o conjunto de dados analisado não possuía registros duplicados, não havendo necessidade de tratamento.

- **Verificar a existência de fiscalizados sem trabalhador**

Através da análise de outras informações, foi possível constatar a existência de 1.645 (mil seiscentas e quarenta e cinco) auditorias ocorridas em fiscalizados que não mais dispunham de trabalhadores a eles vinculados.

Tais registros não agregariam nenhum valor ao modelo criado, podendo, conforma já mencionado, comprometer o seu desempenho.

Diante da informação adquirida através da análise, e do quantitativo extremamente baixo de registros nessas condições, decidiu-se por exclui-los.

- **Verificar a correlação entre as variáveis do conjunto de dados**

Embora a correlação não implique, necessariamente, em causalidade, por meio desta análise foi possível identificar quais variáveis seriam mais relevantes para a construção do modelo preditivo.

Apesar disso, optou-se por não excluir nenhuma delas, tendo, portanto, todas as variáveis do dataset dinal sido consideradas para a criação do algoritmo.

- **Verificar informações estatísticas de cada variável do conjunto de dados**

Com o comando ***"dataset_final.describe()"*** diversas informações estatísticas referente ao conjunto de dados pôde ser analisada, como a média e o desvio padrão, além do primeiro, segundo e terceiro quartil, valor mínimo e valor máximo de cada variável.

- **Verificar a relação de distribuição entre as duas classes da variável target**

Esta análise possibilitou verificar, em valores absolutos, percentuais, e de forma gráfica, a relação de distribuição entre as duas classes da variável target, "Empregador sem Irregularidade" e "Empregador com Irregularidade".

6.2 Resultados da Aplicação dos Algoritmos de *Machine Learning*

6.2.1 Sem Balanceamento dos Dados de Treino

6.2.1.1 Modelo Preditivo nº 1 – Naive Bayes

```
Modelo Preditivo nº 1 - Naive Bayes

Matriz de Confusão:

[[10505    79]
 [ 5555    48]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.65      0.99      0.79     10584
           0       0.38      0.01      0.02      5603

    accuracy                           0.65     16187
   macro avg       0.52      0.50      0.40     16187
weighted avg       0.56      0.65      0.52     16187
```

O modelo Nayve Bayes, criado a partir de dados de treino não balanceados, apresentou uma acurácia de 65%, ou seja, insatisfatória para objetivo do projeto, que pretende implementar um modelo com no mínimo 70% de acurácia.

Ademais, com um recall de 0,01 para a saída “0”, o modelo demonstrou predizer a classe “Empregador sem Irregularidade” com uma baixíssima frequência.

Assim, o algoritmo criado, apesar de possuir uma acurácia de 65%, é bastante ineficiente para predizer, corretamente, um empregador sem irregularidade.

6.2.1.2 Modelo Preditivo nº 2 – Random Forest

```
Modelo Preditivo nº 2 - Random Forest

Matriz de Confusão:

[[8712 1872]
 [2472 3131]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.78      0.82      0.80     10584
           0       0.63      0.56      0.59      5603

    accuracy                           0.73     16187
   macro avg       0.70      0.69      0.70     16187
weighted avg       0.73      0.73      0.73     16187
```

O modelo Random Forest, criado a partir de dados de treino não balanceados, apresentou uma acurácia de 73%, ou seja, satisfatória para objetivo do projeto, que pretendia implementar um modelo com no mínimo 70% de acurácia.

Ademais, com um recall de 0,82 para a saída “1” e 0,56 para a saída “0”, o modelo demonstrou predizer ambas as classes, “Empregador sem Irregularidade” e “Empregador com Irregularidade”, com uma frequência também considerada satisfatória para o projeto.

6.2.1.3 Modelo Preditivo nº 3 – Regressão Logística

```
Modelo Preditivo nº 3 - Regresão Logística

Matriz de Confusão:

[[10584     0]
 [ 5603     0]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.65      1.00      0.79     10584
           0       0.00      0.00      0.00      5603

    accuracy                           0.65     16187
   macro avg       0.33      0.50      0.40     16187
weighted avg       0.43      0.65      0.52     16187
```

O modelo de Regressão Logística, criado a partir de dados de treino não balanceados, apresentou uma acurácia de 65%, ou seja, insatisfatória para objetivo do projeto, que pretende implementar um modelo com no mínimo 70% de acurácia.

Ademais, com um recall de 0,00 para a saída “0”, o modelo demonstrou não predizer a classe “Empregador sem Irregularidade”.

Assim, o algoritmo criado, apesar de possuir uma acurácia de 65%, é completamente ineficiente para predizer um empregador sem irregularidade.

6.2.2 Com Balanceamento dos Dados de Treino – Under Sampling

6.2.2.1 Modelo Preditivo nº 1 – Naive Bayes

```
Modelo Preditivo nº 1 - Naive Bayes

Matriz de Confusão:

[[10500    84]
 [ 5548    55]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.65      0.99      0.79     10584
           0       0.40      0.01      0.02      5603

    accuracy                           0.65     16187
   macro avg       0.52      0.50      0.40     16187
weighted avg       0.56      0.65      0.52     16187
```

O modelo Nayve Bayes, criado a partir de dados de treino com o balanceamento realizado por meio da técnica do Under Sampling, apresentou uma acurácia de 65%, ou seja, insatisfatória para objetivo do projeto, que pretende implementar um modelo com no mínimo 70% de acurácia.

Ademais, com um recall de 0,01 para a saída “0”, o modelo demonstrou predizer a classe “Empregador sem Irregularidade” com uma baixíssima frequência.

Assim, o algoritmo criado, apesar de possuir uma acurácia de 65%, é bastante ineficiente para predizer, corretamente, um empregador sem irregularidade.

6.2.2.2 Modelo Preditivo nº 2 – Random Forest

```
Modelo Preditivo nº 2 - Random Forest

Matriz de Confusão:

[[7262 3322]
 [1679 3924]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.81      0.69      0.74     10584
           0       0.54      0.70      0.61      5603

    accuracy                           0.69     16187
   macro avg       0.68      0.69      0.68     16187
weighted avg       0.72      0.69      0.70     16187
```

O modelo Random Forest, criado a partir de dados de treino com o balanceamento realizado por meio da técnica do Under Sampling, apresentou uma acurácia de 69%, ou seja, insatisfatória para objetivo do projeto, que pretende implementar um modelo com no mínimo 70% de acurácia.

No entanto, com um recall de 0,69 para a saída “1” e 0,70 para a saída “0”, o modelo demonstrou predizer ambas as classes, “Empregador sem Irregularidade” e “Empregador com Irregularidade”, com frequências semelhantes e satisfatórias.

6.2.2.3 Modelo Preditivo nº 3 – Regressão Logística

```
Modelo Preditivo nº 3 - Regresão Logística

Matriz de Confusão:

[[    1 10583]
 [    0  5603]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       1.00      0.00      0.00     10584
           0       0.35      1.00      0.51      5603

    accuracy                           0.35     16187
   macro avg       0.67      0.50      0.26     16187
weighted avg       0.77      0.35      0.18     16187
```

O modelo de Regressão Logística, criado a partir de dados de treino com o balanceamento realizado por meio da técnica do Under Sampling, apresentou uma acurácia de apenas 35%, ou seja, bastante distante do objetivo do projeto, que pretende implementar um modelo com no mínimo 70% de acurácia.

Ademais, com um recall de 0,00 para a saída “1”, o modelo demonstrou não predizer a classe “Empregador com Irregularidade”.

6.2.3 Com Balanceamento dos Dados de Treino – Over Sampling

6.2.3.1 Modelo Preditivo nº 1 – Naive Bayes

```
Modelo Preditivo nº 1 - Naive Bayes

Matriz de Confusão:

[[10482   102]
 [ 5544    59]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.65      0.99      0.79     10584
           0       0.37      0.01      0.02      5603

    accuracy                           0.65     16187
   macro avg       0.51      0.50      0.40     16187
weighted avg       0.55      0.65      0.52     16187
```

O modelo Nayve Bayes, criado a partir de dados de treino com o balanceamento realizado por meio da técnica do Over Sampling, apresentou uma acurácia de 65%, ou seja, insatisfatória para objetivo do projeto, que pretende implementar um modelo com no mínimo 70% de acurácia.

Ademais, com um recall de 0,01 para a saída “0”, o modelo demonstrou predizer a classe “Empregador sem Irregularidade” com uma baixíssima frequência.

Assim, o algoritmo criado, apesar de possuir uma acurácia de 65%, é bastante ineficiente para predizer, corretamente, um empregador sem irregularidade.

6.2.3.2 Modelo Preditivo nº 2 – Random Forest

```
Modelo Preditivo nº 2

Matriz de Confusão:

[[7755 2829]
 [1921 3682]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       0.80      0.73      0.77     10584
           0       0.57      0.66      0.61      5603

    accuracy                           0.71     16187
   macro avg       0.68      0.69      0.69     16187
weighted avg       0.72      0.71      0.71     16187
```

O modelo Random Forest, criado a partir de dados de treino com o balanceamento realizado por meio da técnica do Over Sampling, apresentou uma acurácia de 71%, ou seja, satisfatória para objetivo do projeto, que pretendia implementar um modelo com no mínimo 70% de acurácia.

Ademais, com um recall de 0,73 para a saída “1” e 0,66 para a saída “0”, o modelo demonstrou predizer ambas as classes, “Empregador sem Irregularidade” e “Empregador com Irregularidade”, com frequências semelhantes e satisfatórias.

6.2.3.3 Modelo Preditivo nº 3 – Regressão Logística

```
Modelo Preditivo nº 3 - Regressão Logística

Matriz de Confusão:

[[    4 10580]
 [    0  5603]]

Relatório de Classificação:

              precision    recall  f1-score   support

           1       1.00      0.00      0.00     10584
           0       0.35      1.00      0.51      5603

    accuracy                           0.35     16187
   macro avg       0.67      0.50      0.26     16187
weighted avg       0.77      0.35      0.18     16187
```

O modelo de Regressão Logística, criado a partir de dados de treino com o balanceamento realizado por meio da técnica do Over Sampling, apresentou uma acurácia de apenas 35%, ou seja, bastante distante do objetivo do projeto, que pretende implementar um modelo com no mínimo 70% de acurácia.

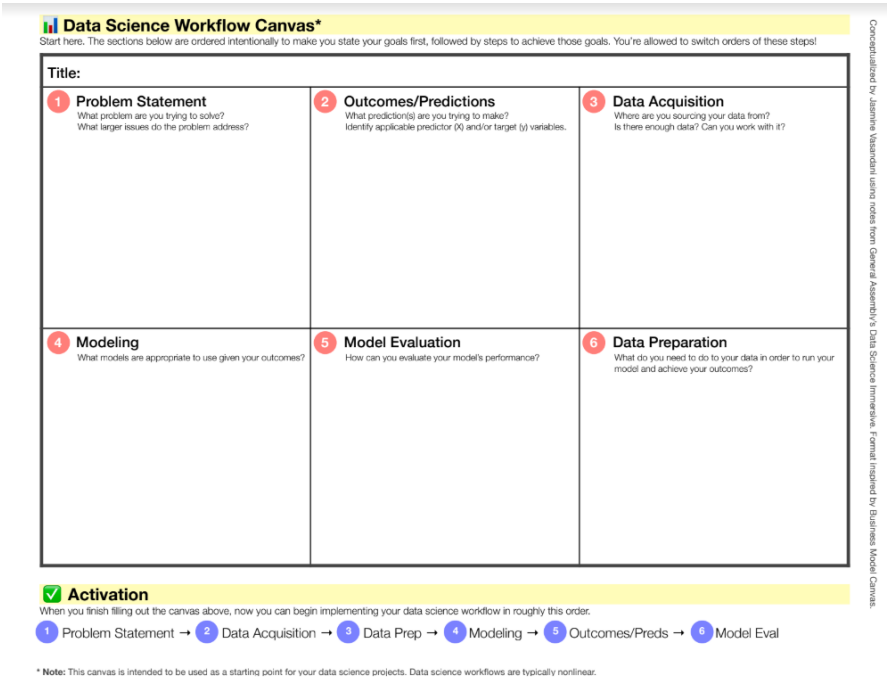
Ademais, com um recall de 0,00 para a saída “1”, o modelo demonstrou não predizer a classe “Empregador com Irregularidade”.

7. Apresentação dos Resultados

A apresentação dos resultados levou em consideração o modelo Canvas proposto por Jasmine Vasandani.

Aprendendo com seus próprios erros Vasandani projetou um Data Sciene Workflow dividido por seções e etapas intencionalmente ordenadas para ajudá-la com a organização e distribuição de ideias para o processo de construção de um modelo de *Machine Learning*.

O modelo é compartilhado a quem possa se interessar e o objetivo é ajudar outras pessoas a realizarem seus próprios projetos de ciência de dados.



1 Problem Statement

What problem are you trying to solve?
What larger issues do the problem address?

Que problema você está tentando resolver? E quais questões maiores esse problema aborda? Esta seção ajuda a abordar o "porquê" do seu projeto.

O descumprimento, por parte de empregadores, de leis trabalhistas e normas de segurança e saúde no trabalho prejudica milhões de trabalhadores todos os anos no Brasil.

Assim, a Inspeção do Trabalho deve planejar e direcionar suas ações fiscais àqueles segmentos econômicos e estabelecimentos que apresentem algum indício de irregularidade, de forma a tornar a sua atuação ainda mais efetiva.

Deste modo, como detectar um empregador que desrespeita a legislação trabalhista, incluindo as normas de Segurança e Saúde no Trabalho? Como detectar um empregador irregular?

2 Outcomes/Predictions

What prediction(s) are you trying to make?
Identify applicable predictor (X) and/or target (y) variables.

Sim, você não saberá quais são seus resultados até depois de concluir seu projeto, mas você deve pelo menos ter uma ideia de como você acha que eles deveriam ser. Identifique as variáveis preditoras (X) e / ou alvo (y) potenciais.

- Variáveis Preditoras:

  - ✓ População do Município do Empregador a ser Fiscalizado;
  - ✓ Número de Auditores-Fiscais do Trabalho na Equipe da Ação Fiscal;
  - ✓ Quantidade de Trabalhadores do Empregador a ser Fiscalizado;
  - ✓ Quantidade de Itens da Legislação Trabalhista a ser Fiscalizada; e
  - ✓ Quantidade de Itens de Segurança e Saúde no Trabalho a ser Fiscalizada.

- Variável Target:

Empregador com Irregularidade (1) ou sem Irregularidade (0)

Deseja-se prever se um empregador possui irregularidade(s) trabalhista(s) ou não.

De onde você está obtendo seus dados? Existem dados suficientes? E você pode realmente trabalhar com isso? Às vezes, você pode ter acesso a conjuntos de dados prontos ou pode precisar raspar seus dados.

- Fonte 1:

Dados serão obtidos diretamente do DataWarehouse da Inspeção do Trabalho, extraídos por meio de consulta SQL (Structered Query Language) no Sistema de Gerenciamento de Banco de Bados - SGBD - Microsoft SQL Server.

Esses dados se referem às ações fiscais já realizadas pela Inspeção do Trabalho.

- Fonte 2:

Dados serão obtidos diretamente por meio da internet, através do site **https://basedosdados.org/dataset/br-ibge-populacao**, e também serão extraídos por meio de consulta SQL (Structured Query Language).

Esses dados se referem à população dos municípios brasileiros no ano de 2020.

O conjunto de dados final terá registros suficientes para a implementação do modelo de *Machine Learning*.

Escolha seu (s) modelo (s) dependendo de suas respostas a estas perguntas: seus resultados são discretos ou contínuos? Você tem conjuntos de dados rotulados ou não? Você está preocupado com outliers? Você deseja interpretar bem seus resultados? A lista de perguntas pode variar dependendo do seu projeto.

As previsões do modelo deverão trazer resultados discretos. “Empregador com Irregularidade” (1) ou “Empregador sem Irregularidade” (0).

Os dados são rotulados e outliers não serão motivo de preocupação por se entender que eles não provocarão distorções nas análises e não prejudicarão os resultados previstos.

Todas as variáveis preditoras serão numéricas, sendo apenas a variável target do tipo categórica (classe).

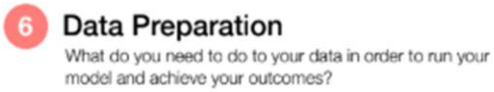
Assim, serão testados 3 (três) modelos de *Machine Learning* baseados em classificação:

- Nayve Bayes;
- Random Forest; e
- Regressão Logística.

Identifique as métricas de avaliação do modelo correspondentes para interpretar seus resultados. Cada modelo terá seu próprio conjunto de métricas de avaliação.

Para a avaliação do desempenho dos modelos de *Machine Learning* serão utilizadas a Matriz de Confusão e o Relatório de Classificação.

Tais ferramentas informam a acurácia ou precisão dos modelos, além de outras métricas relevantes.



O que você precisa fazer com seus dados para executar seu modelo e alcançar seus resultados? A preparação de dados inclui limpeza de dados, seleção de recursos, engenharia de recursos, análise exploratória de dados e assim por diante.

Os dados serão tratados na linguagem de programação Python.

A preparação dos dados contemplará a verificação de eventuais valores nulos (*NaN*) e/ou zeros (“0”) existentes, eventuais duplicidades de registros, eliminação de registros considerados inconsistentes, desnecessários e prejudiciais à confiabilidade dos dados, conversão de tipos de dados (numérico para categórico), identificação da correlação entre as variáveis preditoras e sua importância perante a variável target, além da análise da relação de distribuição entre classes e outras informações relevantes.

Por fim, todos os 3 (três) modelos serão testados e treinados através de 3 (três) métodos:

- Sem o Balanceamento de Dados;
- Com o Balanceamento de Dados utilizando-se o Under Sampling; e
- Com o Balanceamento de Dados utilizando-se o Over Sampling.

8. O Modelo Escolhido

O **Modelo Random Forest, criado a partir de dados de treino com o balanceamento realizado por meio da técnica do Over Sampling**, apresentou uma acurácia de 71% e recall de 0,73 para a saída “1” e 0,66 para a saída “0”.

Assim, entendeu-se que este seria o mais adequado para os objetivos do projeto, tendo sido escolhido como o melhor modelo dentre todos os outros testados.

Diante da escolha, após a importação do pacote Picke foi salva a versão final do modelo.

Modelo Final Selecionado:

Modelo Preditivo nº 2 - Random Forest - Com Over Sampling

Fazendo Previsões Com o Modelo

In [142]:
```
# Importando o Pacote Pickle

import pickle
```

In [143]:
```
# Salvando a Versão Final do Modelo Selecionado - Modelo Preditivo nº 2 - Random Forest - Over Sampling

modelo = 'modelo_final.sav'
pickle.dump(modelo_2_metodo_3, open(modelo, 'wb'))
```

Em seguida, imprimiu-se os dados de teste, carregou-se o modelo salvo e, utilizando estes dados, foram realizadas algumas predições.

In [144]:
```
# Imprimindo os Dados de Teste

X_teste
```

Out[144]:
```
array([[ 573285,       1,       6,       1,      11],
       [ 365855,       2,    1071,       1,       0],
       [ 619609,       1,       5,      12,       0],
       ...,
       [ 391772,       2,       2,       3,       0],
       [2521564,       2,       7,      10,       7],
       [ 283542,       1,      13,      30,       7]], dtype=int64)
```

In [145]:
```
# Carregando o Modelo Final Salvo

modelo_carregado = pickle.load(open(modelo, 'rb'))
```

In [146]:
```
# Fazendo Algumas Previsões Utilizando os Dados de Teste

previsao_1 = modelo_carregado.predict(X_teste[2].reshape(1, -1))
previsao_2 = modelo_carregado.predict(X_teste[6].reshape(1, -1))
previsao_3 = modelo_carregado.predict(X_teste[9].reshape(1, -1))
previsao_4 = modelo_carregado.predict(X_teste[15].reshape(1, -1))

print("Previsão 1:", previsao_1)
print("")
print("Previsão 2:", previsao_2)
print("")
print("Previsão 3:", previsao_3)
print("")
print("Previsão 4:", previsao_4)
print("")
```

```
Previsão 1: [1]

Previsão 2: [0]

Previsão 3: [0]

Previsão 4: [1]
```

Porf fim, alguns dados de entrada foram inseridos manualmente e novas predições foram realizadas.

In [147]:
```
# Fazendo Algumas Previsões Utilizando Dados Quaisquer Inseridos Manualmente

Dados_1 = np.array([500000, 1, 5, 0, 2])
Dados_2 = np.array([30000, 2, 15, 0, 0])
Dados_3 = np.array([15000, 1, 5, 2, 0])
Dados_4 = np.array([15000, 1, 5, 20, 2])
Dados_5 = np.array([1000000, 2, 15, 3, 7])
Dados_6 = np.array([600000, 5, 62, 10, 15])


previsao_5 = modelo_carregado.predict(Dados_1.reshape(1, -1))
previsao_6 = modelo_carregado.predict(Dados_2.reshape(1, -1))
previsao_7 = modelo_carregado.predict(Dados_3.reshape(1, -1))
previsao_8 = modelo_carregado.predict(Dados_4.reshape(1, -1))
previsao_9 = modelo_carregado.predict(Dados_5.reshape(1, -1))
previsao_10 = modelo_carregado.predict(Dados_6.reshape(1, -1))

print("Previsão 5:", previsao_5)
print("")
print("Previsão 6:", previsao_6)
print("")
print("Previsão 7:", previsao_7)
print("")
print("Previsão 8:", previsao_8)
print("")
print("Previsão 9:", previsao_9)
print("")
print("Previsão 10:", previsao_10)
```

```
Previsão 5: [1]

Previsão 6: [0]

Previsão 7: [0]

Previsão 8: [1]

Previsão 9: [1]

Previsão 10: [1]
```

Fim

9. Links

A seguir, estão os links do vídeo de apresentação do Trabalho de Conclusão do Curso e do repositório contendo os dados utilizados no projeto.

Link para o vídeo:

https://youtu.be/1pTVVOHbG_Y

Link para o repositório:

https://github.com/LucasdoPrado/TCC-PUC-MG-Ciencia-de-Dados-e-Big-Data-Lucas-do-Prado