

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

NÚCLEO DE EDUCAÇÃO À DISTÂNCIA

Pós-Graduação Latu Sensu em Ciência de Dados e Big Data

Katia Vieira Pinheiro

**Machine Learn na interpretação e classificação dos
investimentos do Tesouro Nacional**

Rio de Janeiro 2022

Katia Vieira Pinheiro

**Machine Learn na interpretação e classificação dos
investimentos do Tesouro Nacional**

Trabalho de conclusão apresentado ao curso de especialização em Ciência de Dados e Big Data como requisito parcial à obtenção do título de especialista.

Rio de Janeiro 2022

SUMÁRIO

1. Introdução.....	4
1.1 Contextualização.....	5
1.2 O Problema Proposto.....	5
2. Coleta de Dados.....	7
3.Processamento/Tratamento de Dados.....	11
4.Análise e Exploração de Dados.....	16
5. Criação de Modelos de Machine Learn.....	22
6. Apresentação dos resultados.....	42
7. Links.....	44
8. Referências.....	44

1. Introdução

Impulsionados pela crise sanitária imposta pela pandemia da Covid-19, órgãos como a Receita Federal do Brasil reformularam os seus processos de trabalho e a sua estrutura organizacional, com o objetivo de automatizar procedimentos antes realizados manualmente e capacitar os servidores em novas tecnologias, como *ciência de dados* e *big data*, para fazer frente às alterações no cenário externo. Para os servidores públicos, o cenário interno é extremamente desafiador, com a necessidade imperiosa de se reinventar, adquirir novos conhecimentos e habilidades compatíveis com o mundo virtualizado.

Com a modernização acelerada, mergulhamos na indústria 4.0, chamada de a quarta revolução industrial, cujo desafio é integrar as novas tecnologias – *artificial intelligence* (AI, inteligência artificial), *Internet of things* (IOT, *Internet das coisas*), *bots*, *big data*, *business intelligence* – aos processos tradicionais, transformando-os, modernizando-os, tornando-se imperiosa a utilização de tecnologias como *machine learning*, resultando em um enorme salto de qualidade nos serviços públicos prestados à sociedade.

Ao longo dos anos os órgãos públicos tornaram transparentes dados para consulta de várias esferas de governo. O Ministério da Economia assim como todos os outros disponibilizam dados a serem analisados e possibilita uma iteração com as empresas e os cidadãos.

No caso do Tesouro Nacional, são investimentos de compras de títulos. Os títulos são Tesouro IPCA+, Tesouro IPCA com Juros Semestrais, Tesouro Préfixado com Juros Semestrais, Tesouro SELIC, Tesouro IGPM com Juros Semestrais e Tesouro Prefixado. Cada um com suas características de investimentos e indexação. Porém os títulos que tem como base a taxa SELIC sofreram alta nos resgates dos mesmos.

Para demonstrar, vamos trabalhar com um conjunto de arquivos que representam os valores das operações.

Este conjunto de dados apresenta o movimento de operações do volume financeiro ocorridos no Tesouro Direto em determinado mês. Devemos observar que durante a pandemia, houve alta de preços e despesas. O simples fato de comprar documentos do governo como se fosse uma poupança para os cidadãos brasileiros diminuíram bastante. Ainda por cima, a redução dos juros como ferramenta para a contenção da inflação não permitia deixar atraente ao brasileiro o investimento.

1.1 Contextuação

Durante o ano de 2019, as operações foram mínimas, o que reflete a política de juros baixos. Porém, os juros começaram a aumentar a partir do segundo semestre de 2020. Os investidores começaram a fazer mais operações. Como amostra, escolhi somente o mês de fevereiro do ano de 2021. Pois o dataset ficaria muito grande, inviabilizando a minha pesquisa já que ela se dá num ambiente limitado e não num ambiente de desenvolvimento ou de homologação de grandes empresas.

1.2 O Problema proposto

“Um projeto de Data Science começa com uma necessidade ou ideia. Nesta etapa inicial, deve-se primeiro ter em mente o problema que se deseja resolver, e, em seguida, os objetivos devem ser definidos ...” (Escovedo, Tatiana)¹

Verificar e expor as operações realizadas no período de fevereiro de 2021 e fazer uma previsão de quais seriam os investidores de baixa, médio baixo, médio alto e alto investimento nos títulos existentes no Tesouro Direto. O aumento dos investimentos são atrelados a alta dos juros básicos e da taxa SELIC.

Para melhor visão do problema e da solução optou-se por utilizar a técnica dos 5-Ws², respondendo-se as perguntas a que se refere cada W.

(Why) Por que esse problema é importante?

O investimento dentro do Tesouro Direto chama a atenção, pela rentabilidade e a segurança.

O Tesouro Direto é, basicamente, um programa criado em 2002 pelo Tesouro Nacional – órgão responsável pela gestão da dívida pública. Permitir que as pessoas físicas comprem papéis do governo federal pela internet. Ao comprar um título do Tesouro Direto, o investidor está emprestando dinheiro ao governo.

Permite fazer aplicações com valores muito baixos (a partir de R\$ 30) e oferece liquidez diária para todos os papéis.

1-Escovedo, Tatiana (2020-02-27T22:58:59). Introdução a Data Science . Casa do Código. Edição do Kindle.

2- <https://its.unl.edu/bestpractices/remember-5-ws>

Não é restrito a poucas instituições financeiras, pois os investidores podem aplicar por meio de diversos bancos e corretoras de valores. Há várias opções de títulos públicos à venda para perfis diferentes de investidor.

Assim, é possível escolher diferentes indexadores, prazos de vencimento e fluxos de remuneração.

(Who?) De quem são os dados analisados?

São dados públicos e estão disponíveis no site [Dados.gov.br](https://dados.gov.br).

(What) Quais os objetivos com essas análises? O que iremos analisar?

Análise dos títulos como impacto da redução das taxas de juros pelo Banco Central e ainda quais seriam os tipos de investidores existentes e futuros.

(Where) Quais os aspectos geográficos e logísticos de sua análise?

A delimitação geográfica foi em nível Brasil.

(When) Qual o período analisado?

Mês de fevereiro de 2021.

2. Coleta de Dados

O projeto foi desenvolvido utilizando-se Pythonn 3.8.3, uma linguagem de programação interpretada, orientada a objetos e de alto nível com semântica dinâmica³. Foi utilizado o Jupyter versão 6.0.3 aplicativo cliente-servidor de código aberto usado para criar e executar principalmente projetos de Ciência de Dados, porque também fornece ferramentas para visualização, simulação numérica e limpeza de dados entre outras ferramentas⁴. É uma sigla originada de Julia, Python e R porque foram as primeiras linguagens de programação para esse editor. Atualmente oferece suporte a várias outras linguagens.

Os dados analisados são públicos, arquivos publicados no site governamental. Os dados são disponibilizados para *download* no *site*⁵ específico do arquivo analisado.

Como esses dados são de valores, o arquivo original CVS era bastante grande. A data inicial é de janeiro de 2014, de periodicidade diária, limitei o dataset no mês de fevereiro de 2021.

São as transações que ocorrem no Tesouro Direto. As operações consideram o ponto de vista do Tesouro Nacional, enquanto emissor dos títulos. Podem ser de quatro tipos: venda: quando o título é vendido para o investidor; compra: quando o Tesouro Nacional recompra os títulos do investidor antes do vencimento. Essa operação ocorre quando o investidor quer resgatar seu investimento antes do prazo de vencimento. Também pode ser chamada de Compra Antecipada ou Recompra; retirada: quando o investidor retira o título do ambiente do Tesouro Direto e coloca sob custódia direta da própria instituição financeira. Desde maio de 2015 não é mais permitida essa operação. Entretanto, alguns eventos podem ocorrer, por ordem exclusivamente da BM&FBovespa, para correções de lançamentos de depósitos realizados por instituições financeiras; depósito: quando o investidor decide recolocar o título no ambiente do Tesouro Direto. É a operação inversa da retirada.

Inicialmente foi criado um dataset com o ano de 2019. Porem, ficou inviável pois o número de linhas era muito grande. Então fiz alguns estudos iniciais e decidi somente o mês de julho de 2021. Nessa pesquisa, pude detectar que o ano de 2020 foi o ano de menor investimento já que os juros estavam muito pequenos o que desestimulou os investimentos.

3- <https://www.python.org/doc/essays/blurbl/>

4 - <https://learnpython.com/blog/jupyter-notebook-python-ide-installation-tips/>

5 - <https://dados.gov.br/dataset/operacoes-do-tesouro-direto>

Foram importadas várias bibliotecas Python, figura 1, além das bibliotecas nativas do Python, para as análises, processamento, tratamento de dados e criação dos modelos de *machine learning*.

Basicamente foram utilizadas as bibliotecas: “Pandas” e “Numpy” para operacionalização e cálculos no *dataset*;

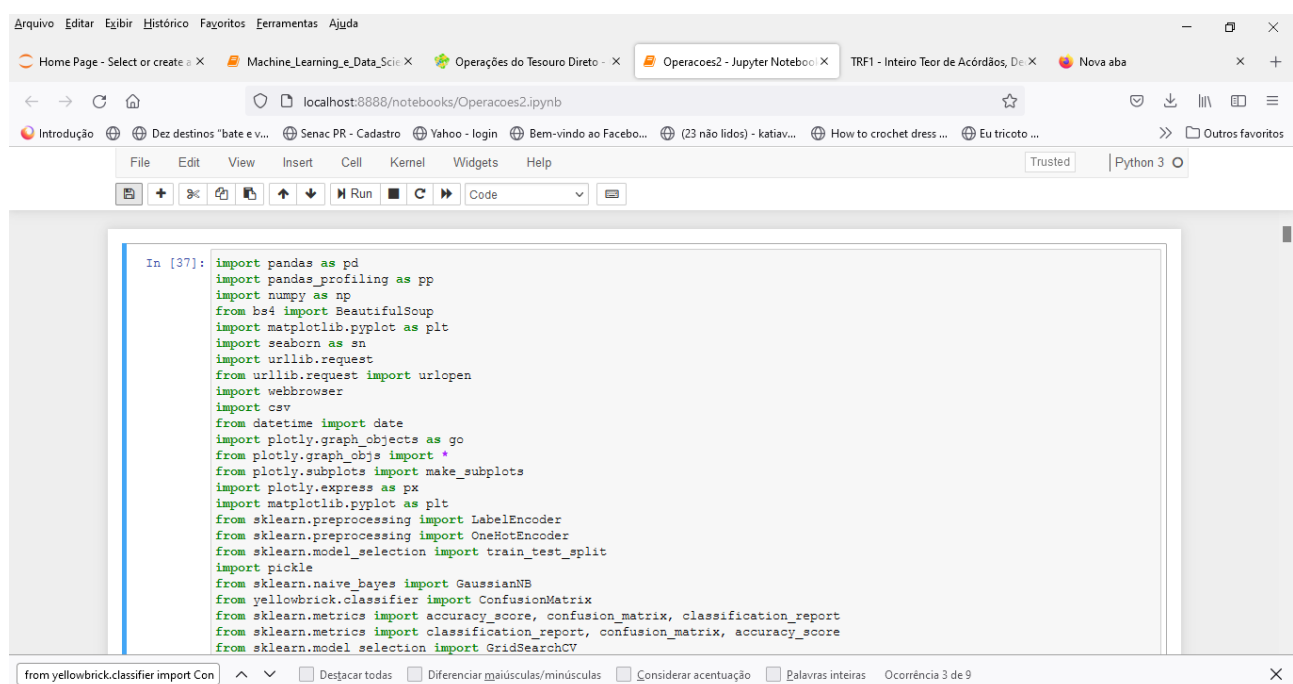
“pandas_profiling” para estatística descrita e outras análises;

“BeautifulSoup” e “request” para *web scraping* ou *web data extraction*;

“scipy” para cálculo de medidas estatísticas; e

“sklearn” para criação do modelo de *machine learning*.

Figura 1 – Bibliotecas utilizadas.



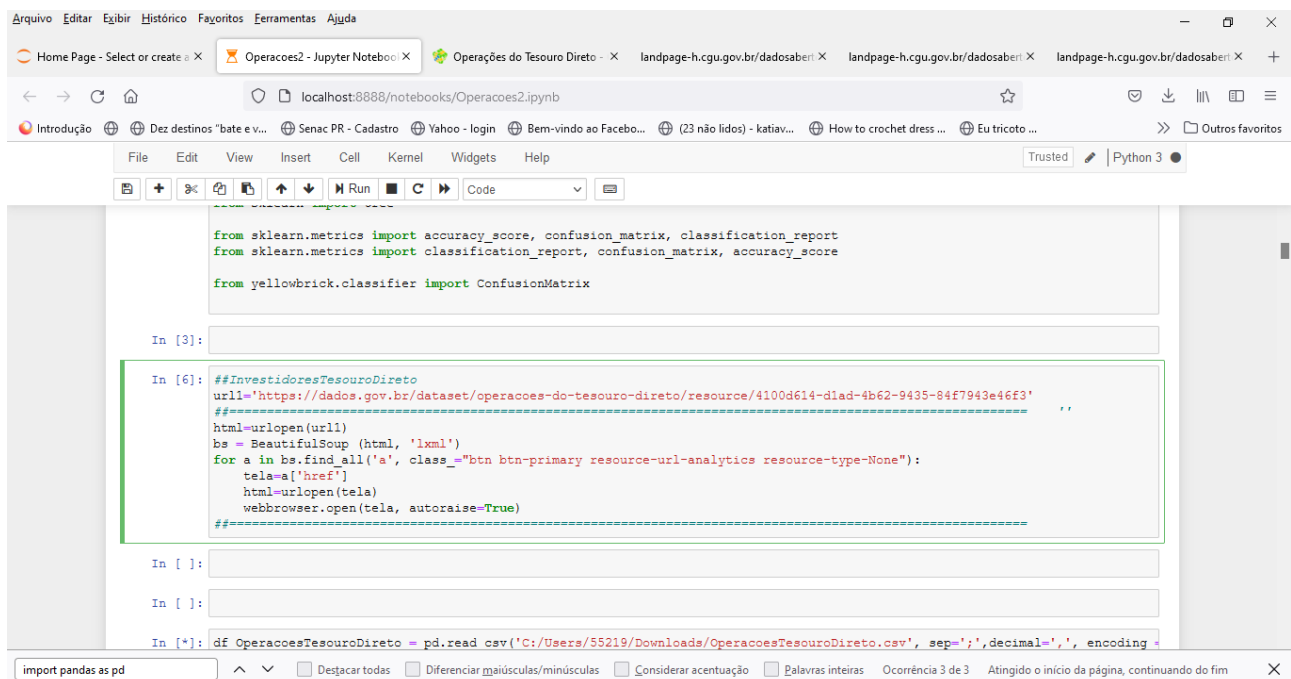
The image shows a Jupyter Notebook interface with a browser window. The notebook is titled "Operacoes2 - Jupyter Notebook" and is running on a local host. The code cell shows the following imports:

```
In [37]: import pandas as pd
import pandas_profiling as pp
import numpy as np
from bs4 import BeautifulSoup
import matplotlib.pyplot as plt
import seaborn as sn
import urllib.request
from urllib.request import urlopen
import webbrowser
import csv
from datetime import date
import plotly.graph_objects as go
from plotly.graph_objs import *
from plotly.subplots import make_subplots
import plotly.express as px
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
import pickle
from sklearn.naive_bayes import GaussianNB
from yellowbrick.classifier import ConfusionMatrix
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from sklearn.model_selection import GridSearchCV
```

The interface includes a menu bar (Arquivo, Editar, Exibir, Histórico, Favoritos, Ferramentas, Ajuda) and a toolbar with icons for file operations, running code, and viewing output. The status bar at the bottom indicates the current cell is "from yellowbrick.classifier import Con" and shows search options.

O código para a extração do arquivo CSV do site do governo referente ao Tesouro Nacional e demonstrado na figura 2.

Figura 2 – Extraíndo o arquivo CSV do site <https://dados.gov.br>.



```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
from yellowbrick.classifier import ConfusionMatrix

In [3]:

In [6]: ##InvestidoresTesouroDireto
url1='https://dados.gov.br/dataset/operacoes-do-tesouro-direto/resource/4100d614-dlad-4b62-9435-84f7943e46f3'
#####
html=urlopen(url1)
bs = BeautifulSoup(html, 'lxml')
for a in bs.find_all('a', class_='btn btn-primary resource-url-analytics resource-type-None'):
    tela=a['href']
    html=urlopen(tela)
    webbrowser.open(tela, autoraise=True)
#####

In [ ]:

In [ ]:

In [*]: df OperacoesTesouroDireto = pd.read_csv('C:/Users/55219/Downloads/OperacoesTesouroDireto.csv', sep=';', decimal=',', encoding =
import pandas as pd
```

O arquivo CSV, demonstra as operações mensais com o código do investidor único, a data da operação, o Tipo do Título, a data do vencimento do título, a quantidade de operações, valor do título, valor da operação, tipo da operação (se é compra, venda, compra, retirada ou depósito) e o canal da operação (se é S= site, H= Homebroker).

O dataset não tem dados de colunas string que tenham necessidade de serem convertidos em números. O que foi resolvido no momento da leitura do arquivo e criação do dataframe para um melhor tratamento na linguagem de máquina. Porém tem tipo data a ser tratado mais tarde.

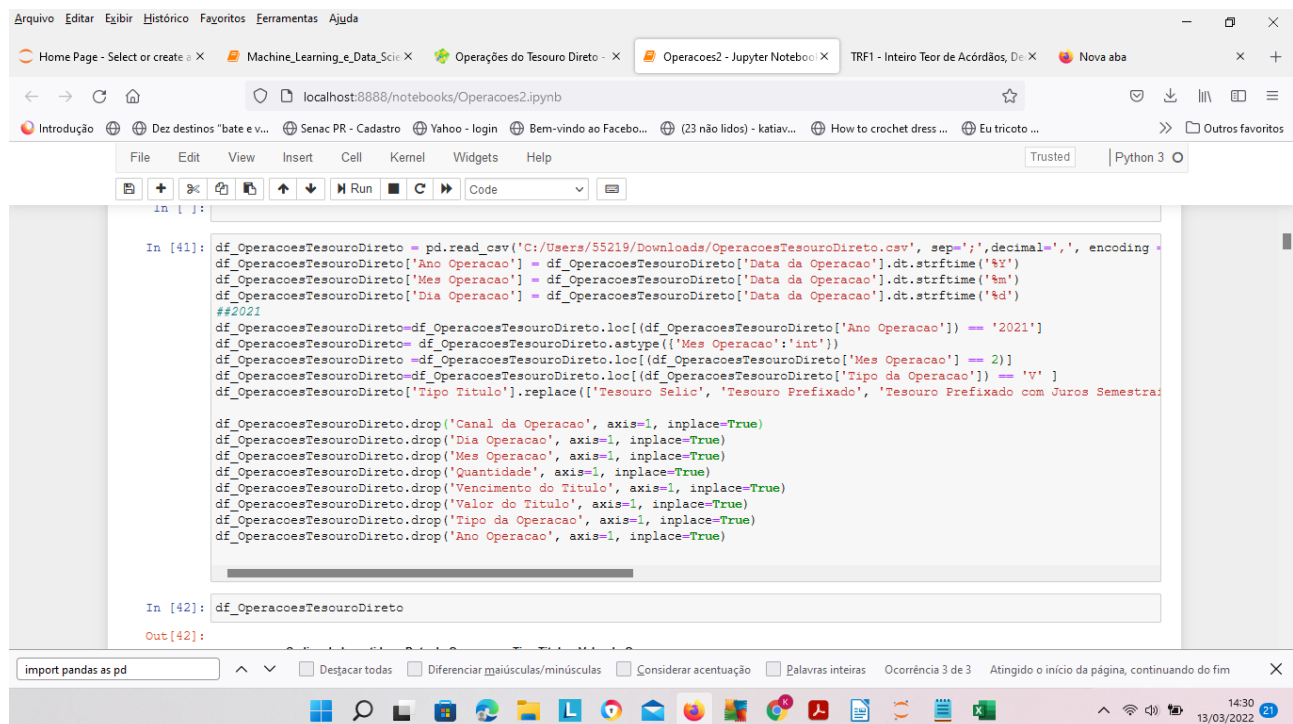
Os tipos do Título são: Tesouro Selic, Tesouro Prefixado, Tesouro Prefixado com Juros Semestrais, Tesouro IPCA+, Tesouro IPCA+ com Juros Semestrais, Tesouro IGP-M + com Juros Semestrais e atribui, nessa ordem, os valores 1,2,3,4,5 e 6.

Foram eliminadas algumas colunas que não estavam acrescentando o estudo no momento da criação do dataframe. A saber: Canal da Operacao, Dia, Mes e Ano da operação (que criei para fazer a seleção do mês e ano escolhido), Quantidade, Vencimento do Titulo (quando iria vencer), Valor do Titulo(unitário no momento da venda), Tipo da Operacao(escolhi só a venda) e Ano Operacao (que foi o de 2021).

As colunas do dataframe gerado são:
Código do investidor
Data da Operação
Tipo Título
Valor

Tanto a criação do dataframe quando a eliminação das colunas estão demonstradas na figura 3.

Figura 3 – Criando o Dataframe com os dados das Operações do Investimento do Tesouro Direto.



```
In [41]: df_OperacoesTesouroDireto = pd.read_csv('C:/Users/55219/Downloads/OperacoesTesouroDireto.csv', sep=';', decimal=',', encoding='utf-8')
df_OperacoesTesouroDireto['Ano Operacao'] = df_OperacoesTesouroDireto['Data da Operacao'].dt.strftime('%Y')
df_OperacoesTesouroDireto['Mes Operacao'] = df_OperacoesTesouroDireto['Data da Operacao'].dt.strftime('%m')
df_OperacoesTesouroDireto['Dia Operacao'] = df_OperacoesTesouroDireto['Data da Operacao'].dt.strftime('%d')
##2021
df_OperacoesTesouroDireto=df_OperacoesTesouroDireto.loc[(df_OperacoesTesouroDireto['Ano Operacao']) == '2021']
df_OperacoesTesouroDireto= df_OperacoesTesouroDireto.astype({'Mes Operacao':'int'})
df_OperacoesTesouroDireto =df_OperacoesTesouroDireto.loc[(df_OperacoesTesouroDireto['Mes Operacao'] == 2)]
df_OperacoesTesouroDireto=df_OperacoesTesouroDireto.loc[(df_OperacoesTesouroDireto['Tipo da Operacao']) == 'V' ]
df_OperacoesTesouroDireto['Tipo Titulo'].replace(['Tesouro Selic', 'Tesouro Prefixado', 'Tesouro Prefixado com Juros Semestrais'], 'V', inplace=True)

df_OperacoesTesouroDireto.drop('Canal da Operacao', axis=1, inplace=True)
df_OperacoesTesouroDireto.drop('Dia Operacao', axis=1, inplace=True)
df_OperacoesTesouroDireto.drop('Mes Operacao', axis=1, inplace=True)
df_OperacoesTesouroDireto.drop('Quantidade', axis=1, inplace=True)
df_OperacoesTesouroDireto.drop('Vencimento do Titulo', axis=1, inplace=True)
df_OperacoesTesouroDireto.drop('Valor do Titulo', axis=1, inplace=True)
df_OperacoesTesouroDireto.drop('Tipo da Operacao', axis=1, inplace=True)
df_OperacoesTesouroDireto.drop('Ano Operacao', axis=1, inplace=True)

In [42]: df_OperacoesTesouroDireto

Out[42]:
```

3. Processamento/Tratamento de Dados

As operações de ETL (extraction, transformation, loading) dos dados, atividades de pré-processamento, representam a etapa mais demorada e trabalhosa de um projeto de data science, consumindo pelo menos 70% do tempo total do projeto, segundo Escovedo:

“Nesta etapa, pode ser necessário remover ou complementar dados faltantes; corrigir ou amenizar dados discrepantes (outliers) e desbalanceamento entre classes, e selecionar as variáveis e instâncias mais adequadas para compor o(s) modelo(s) que serão construídos na etapa seguinte.”⁶

No presente projeto foram necessários inúmeros procedimentos de data cleaning, assim como um extenso pipeline de transformação dos dados brutos das decisões baixados da web e importados via planilha:

1. transformação e conversão das colunas do dataframe;
2. remoção de linhas que não possuem os dados necessários;
3. remoção de dados duplicados;
4. rotulagem dos atributos Tipo Título e Investidor e;
5. remoção de partes desnecessárias dos documentos;

Foi feito um estudo dos dados antes de chegarmos a esta conclusão. Para o ano de 2021, fiz 9 dataframes para pesquisar o mais atraente. Conclui que o mês de fevereiro é o melhor a ser estudado já que o volume de dados dos demais maiores.

Mês de janeiro = 13481 rows × 3 columns

Mês de fevereiro=13388 rows × 3 columns

Mês de março = 119935 rows × 3 columns

Mês de abril = 135383 rows × 3 columns

Mês de maio = 210283 rows × 3 columns

Mês de junho = 24951 rows × 3 columns

Mês de julho = 21013 rows × 3 columns

Mês de agosto = 14349 rows × 3 columns

Mês de setembro = 19135 rows × 3 columns

Mês de outubro = 20726 rows × 3 columns

Nesse mês de fevereiro, foram 13388 investidores movimentando o Tesouro Nacional.

Nesse estudo, determinamos uma faixa de valor de investimentos dando graduações para as operações dos investidores.

⁶ - Escovedo, Tatiana (2020-02-27T22:58:59). Introdução a Data Science . Casa do Código. Edição do Kindle.

Cada investidor recebeu um indicador.

1 para baixo investimento,

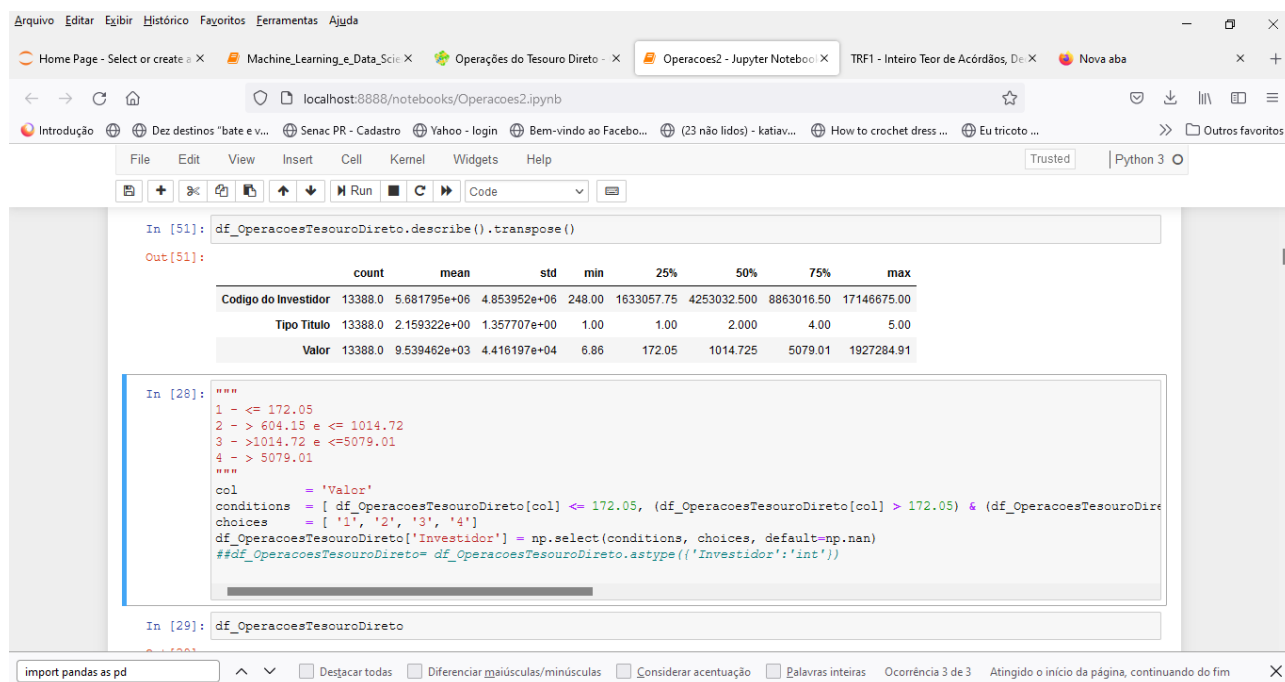
2 para média baixo,

3 para médio alto e

4 para investimentos com valores maiores que 75% da média.

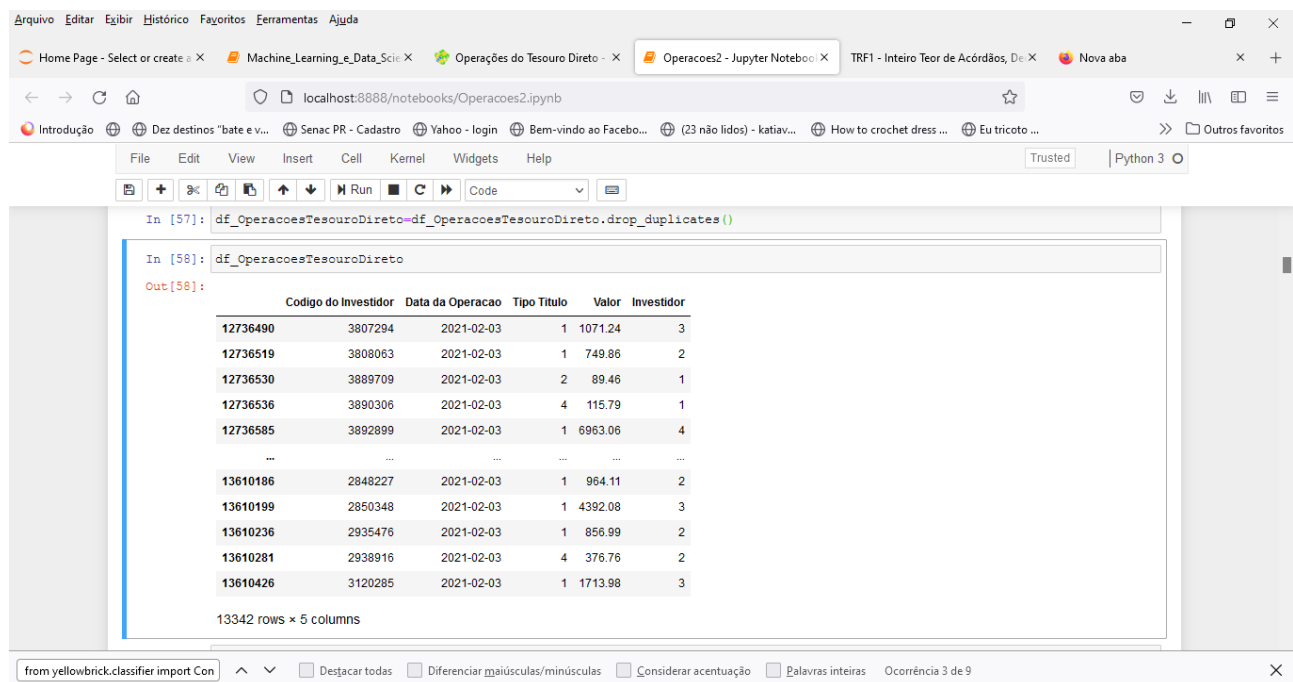
Para isso, verifiquei os valores de acordo com as funções `describe()` e `transpose()` que nos dá essa visão. Demonstrado na figura 4.

Figura 4 – Classificação dos investidores.



Para remoção dos registros duplicados foi utilizado o método “drop_duplicates”, resultando, ao final, no *dataframe* “df_OperacoesTesouroDireto” sem duplicidade de registros. Figura 5.

Figura 5 – Eliminando as duplicatas.



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [57]: df_OperacoesTesouroDireto=df_OperacoesTesouroDireto.drop_duplicates()
```

```
In [58]: df_OperacoesTesouroDireto
```

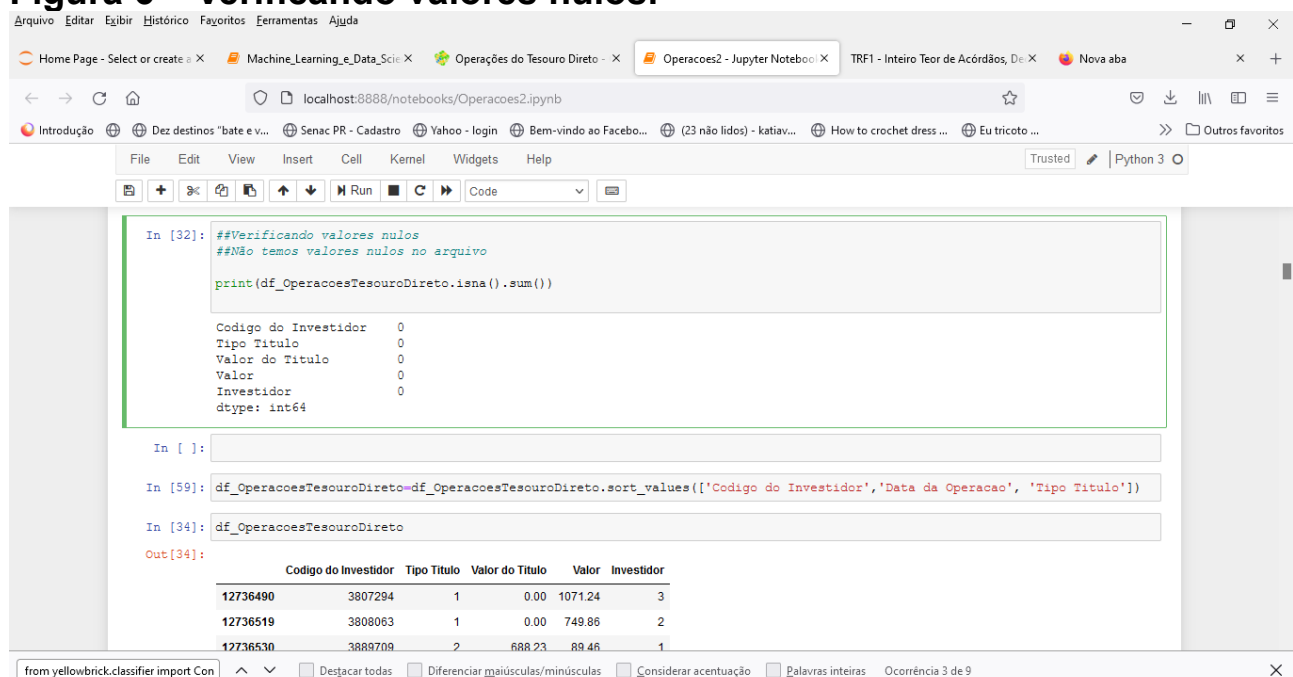
Out[58]:

	Codigo do Investidor	Data da Operacao	Tipo Titulo	Valor	Investidor
12736490	3807294	2021-02-03	1	1071.24	3
12736519	3808063	2021-02-03	1	749.86	2
12736530	3889709	2021-02-03	2	89.46	1
12736536	3890306	2021-02-03	4	115.79	1
12736585	3892899	2021-02-03	1	6963.06	4
...
13610186	2848227	2021-02-03	1	964.11	2
13610199	2850348	2021-02-03	1	4392.08	3
13610236	2935476	2021-02-03	1	856.99	2
13610281	2938916	2021-02-03	4	376.76	2
13610426	3120285	2021-02-03	1	1713.98	3

13342 rows x 5 columns

Não ocorreram nem valores nulos ou faltantes. Figura 6.

Figura 6 – Verificando valores nulos.



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [32]: ##Verificando valores nulos
##Não temos valores nulos no arquivo
print(df_OperacoesTesouroDireto.isna().sum())
```

```
Codigo do Investidor    0
Tipo Titulo              0
Valor do Titulo          0
Valor                   0
Investidor               0
dtype: int64
```

```
In [ ]:
```

```
In [59]: df_OperacoesTesouroDireto=df_OperacoesTesouroDireto.sort_values(['Codigo do Investidor','Data da Operacao', 'Tipo Titulo'])
```

```
In [34]: df_OperacoesTesouroDireto
```

Out[34]:

	Codigo do Investidor	Tipo Titulo	Valor do Titulo	Valor	Investidor
12736490	3807294	1	0.00	1071.24	3
12736519	3808063	1	0.00	749.86	2
12736530	3889709	2	889.23	89.46	1

As figuras 7, 8 9 e 10 são de gráficos gerados para demonstração de relações entre os valores e os outros dados de Data da Operação, Tipo de Operação e Investidor.

Figura 7 – Gerando o grafico de barras com os Tipos de Operação e os Valor.

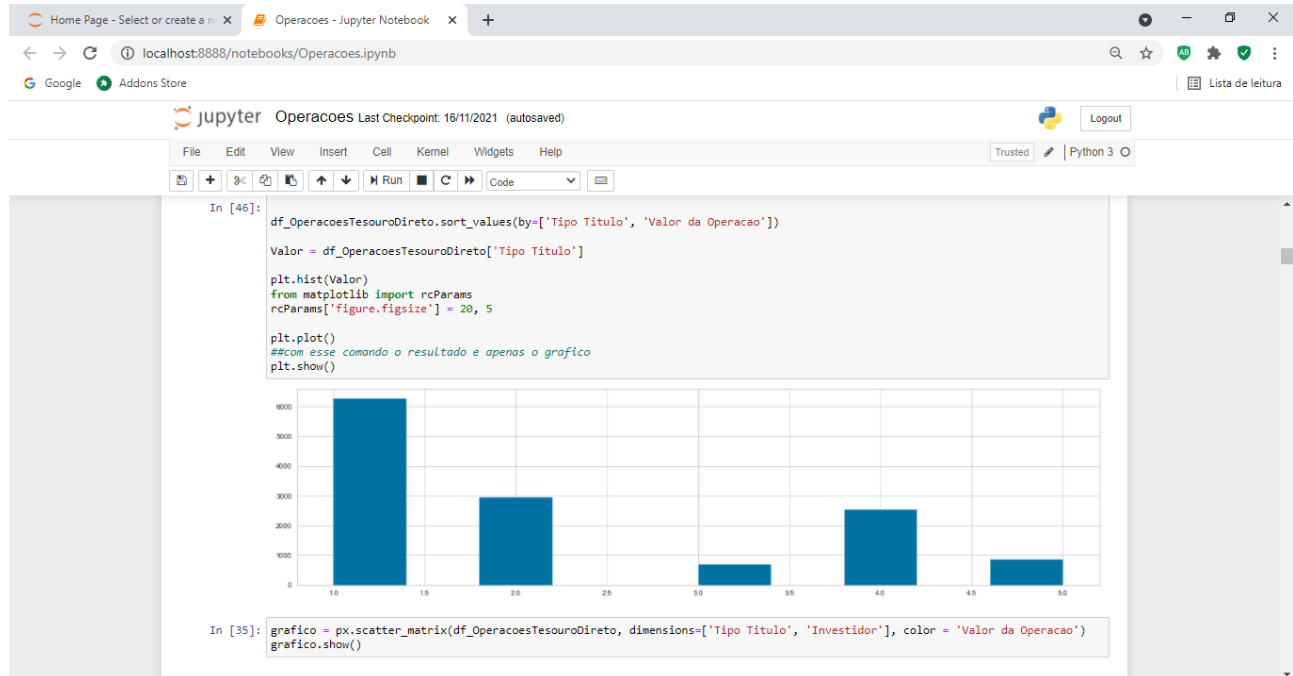


Figura 8 – Grafico de matriz.

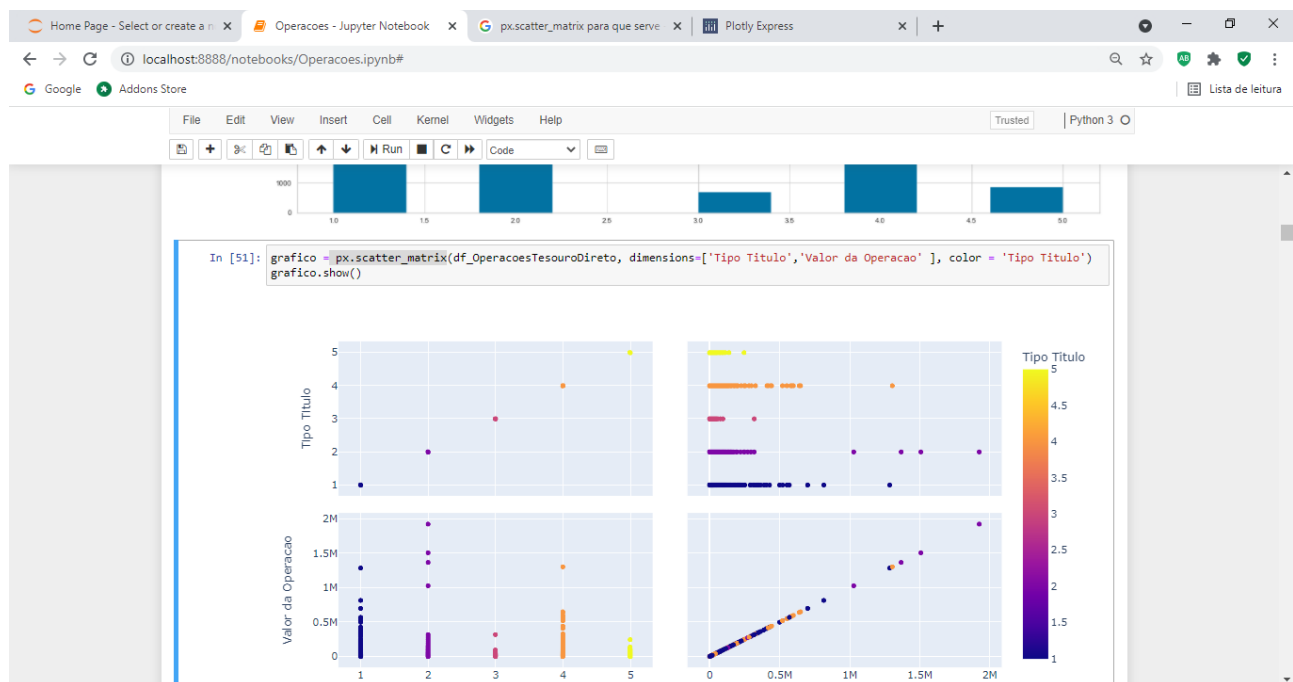


Figura 9 – Histograma de valores e tipo de títulos.

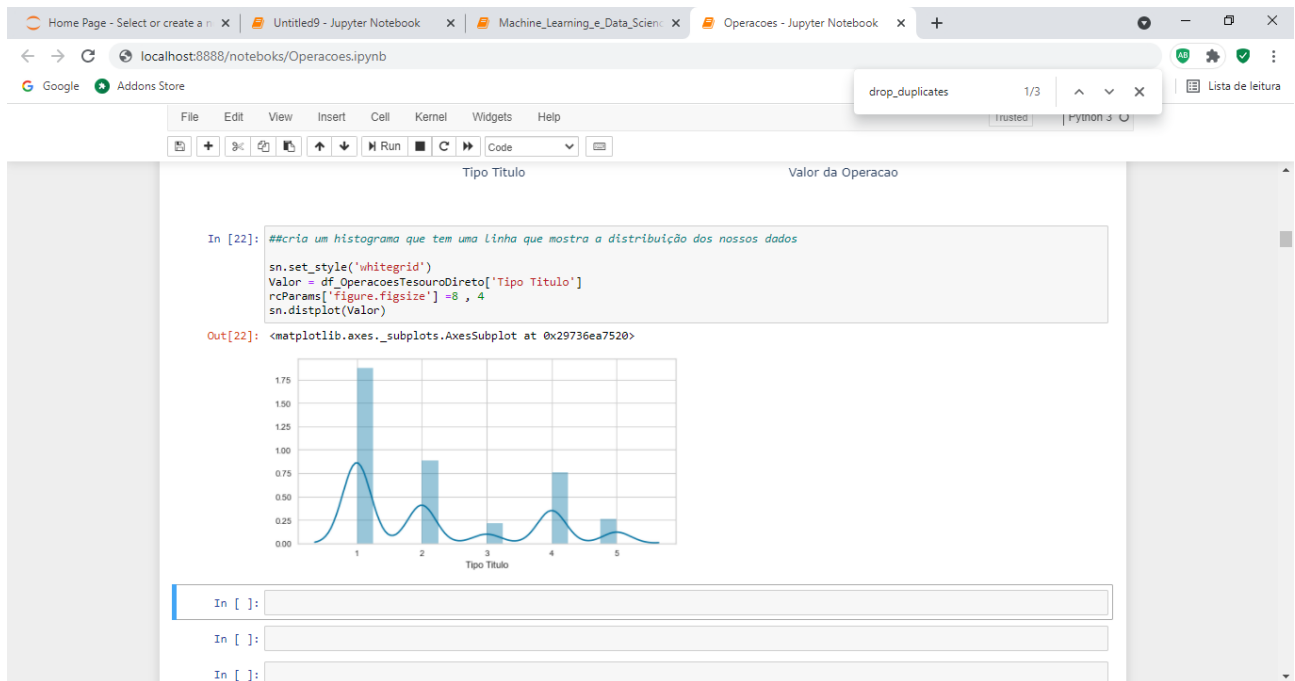
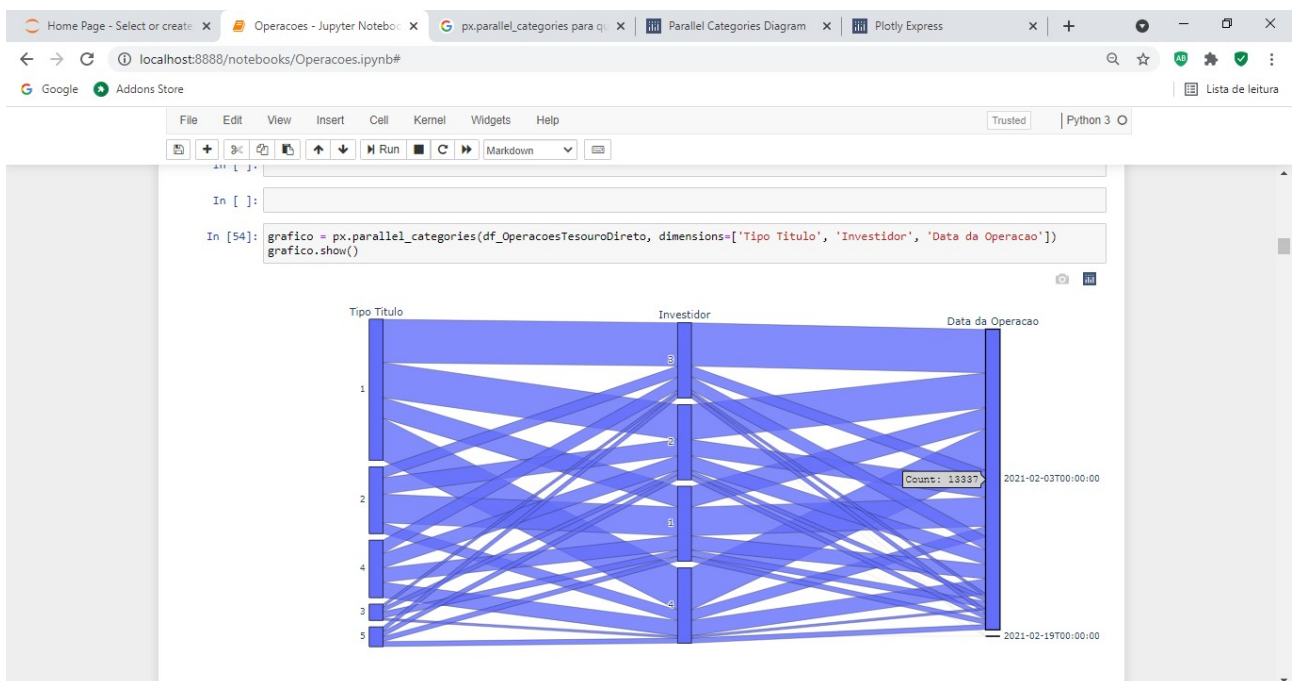


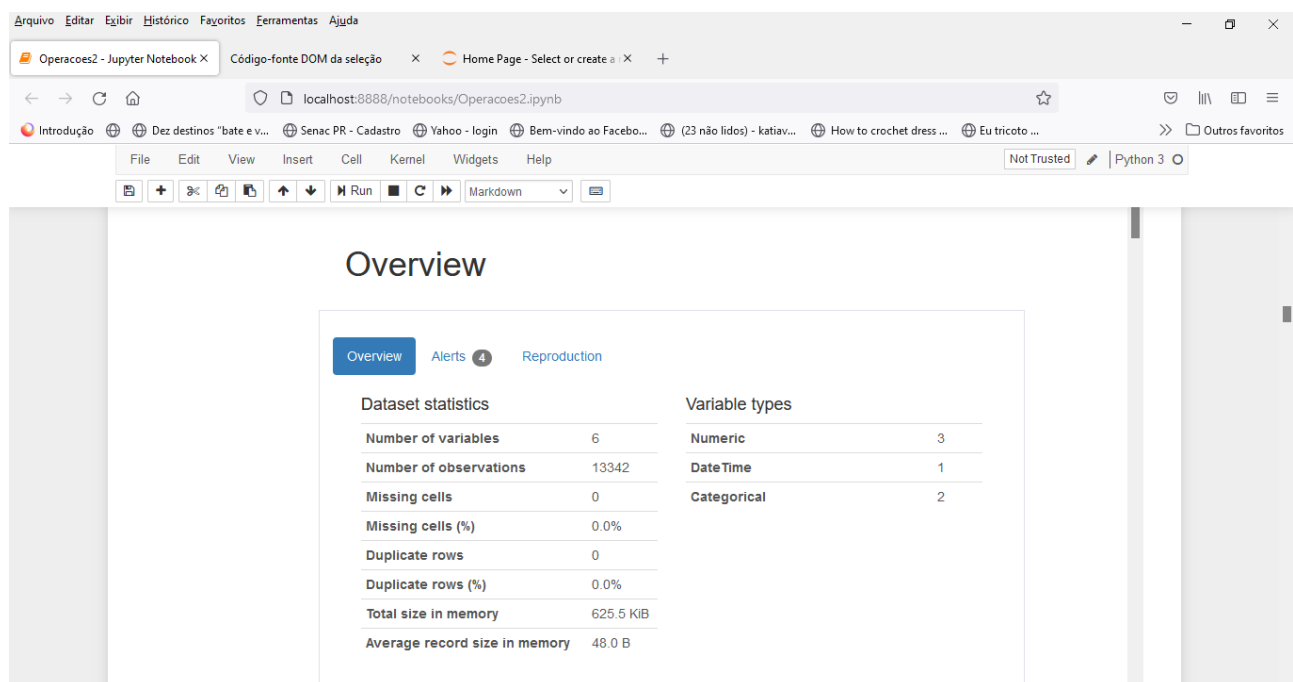
Figura 10 – Gráfico que faz uma relação entre os Tipos de títulos, Investidor e Data da Operação.



4. Análise e Exploração dos Dados

Iniciou-se a fase de análise e exploração dos dados com o método "Profile Report" da biblioteca "pandas_profiling"⁷, que possibilita uma rápida análise exploratória dos dados do *dataframe*: estatística descritiva com medidas de dispersão e medidas de posição, valores extremos, dados faltantes, correlações etc. No *overview* do "Pandas Profiling Report", figura 11, são demonstradas as estatísticas do *dataset* e os tipos de variáveis. Mostra-se que não há dados faltantes, o número total de linhas e colunas do *dataframe* foi processado.

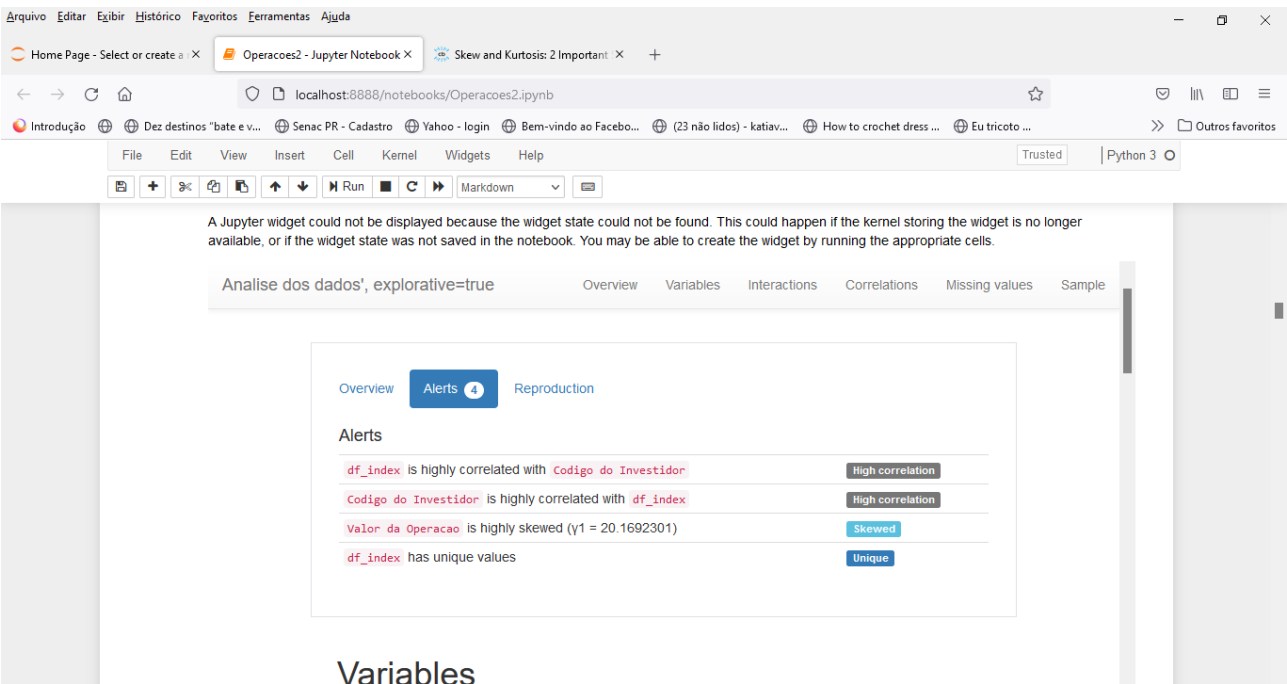
Figura 11 Overview do "Pandas Profiling Report".



O "Pandas Profiling Report" gerou quatro alertas, que foram analisados pontualmente. Figura 12.

7 - <https://github.com/ydataai/pandas-profiling>

Figura 12 – Alertas gerados.



O que mais chama a atenção seria o campo Valor da Operação - **Skewed** o que indica uma distorção de valores. No caso, muito alto. São os outliers. Figura 13 e 14.

Figura 13 – Identificando valores extremos.

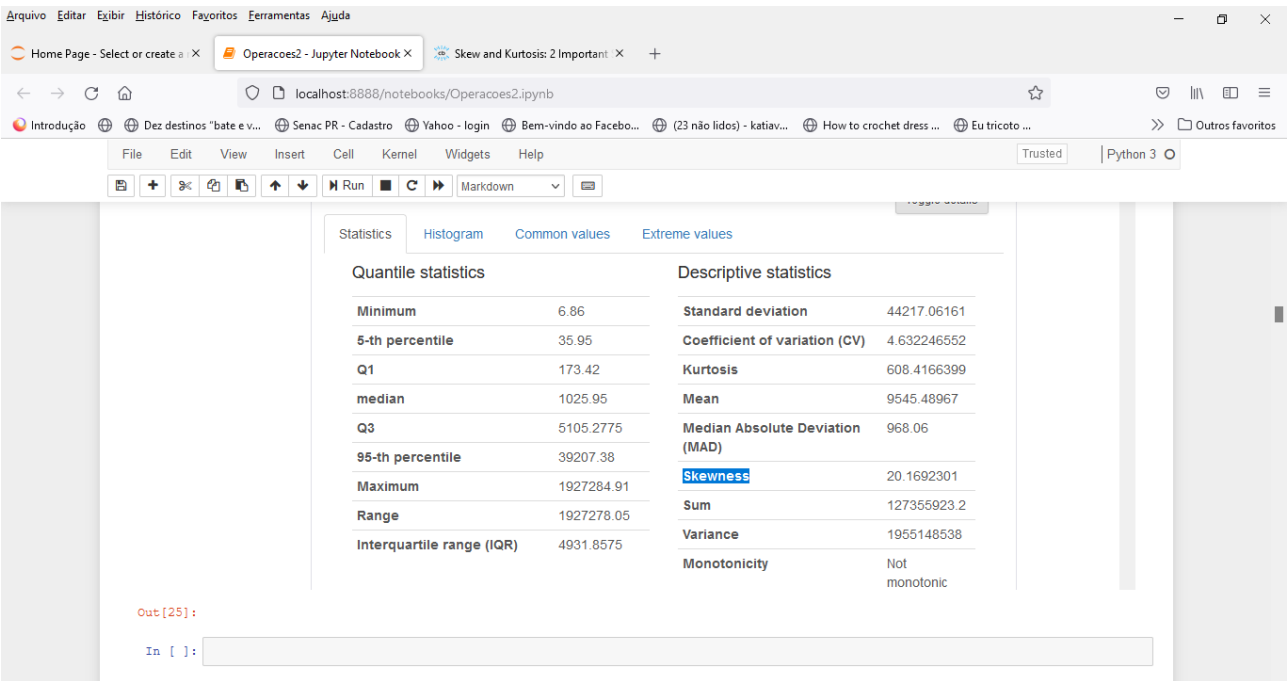
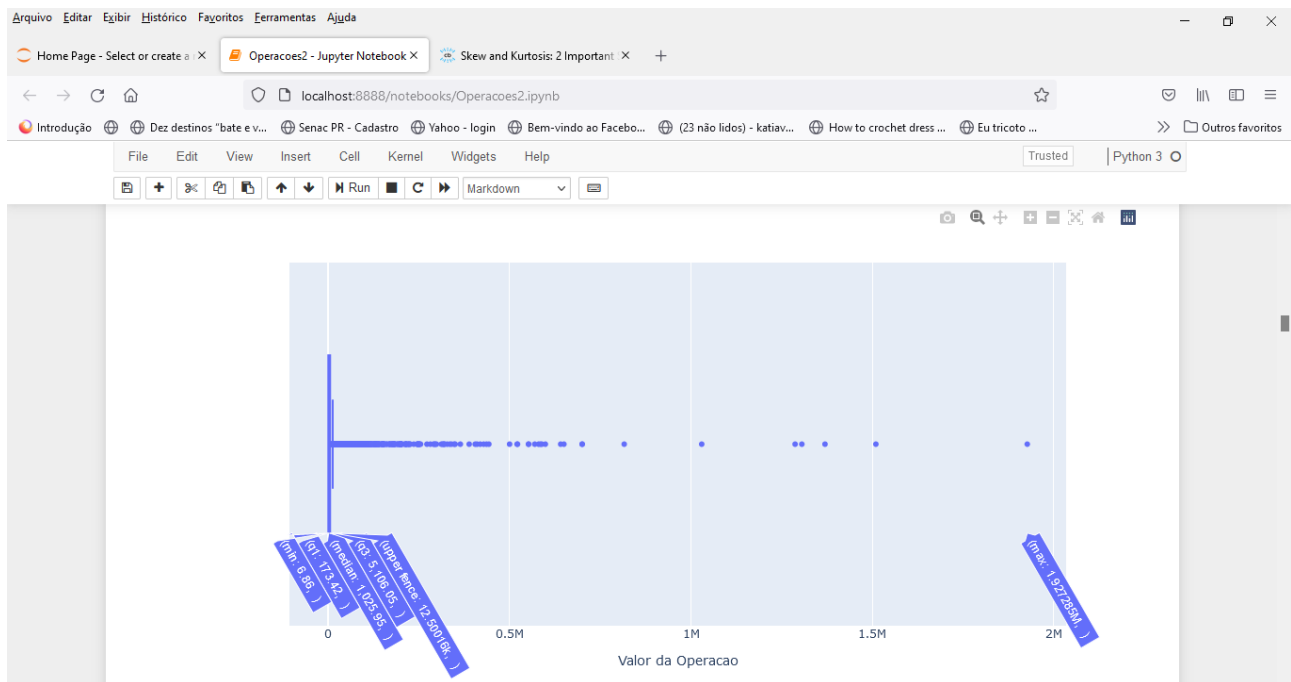


Figura 14 - Boxplot dos Valores das Operações

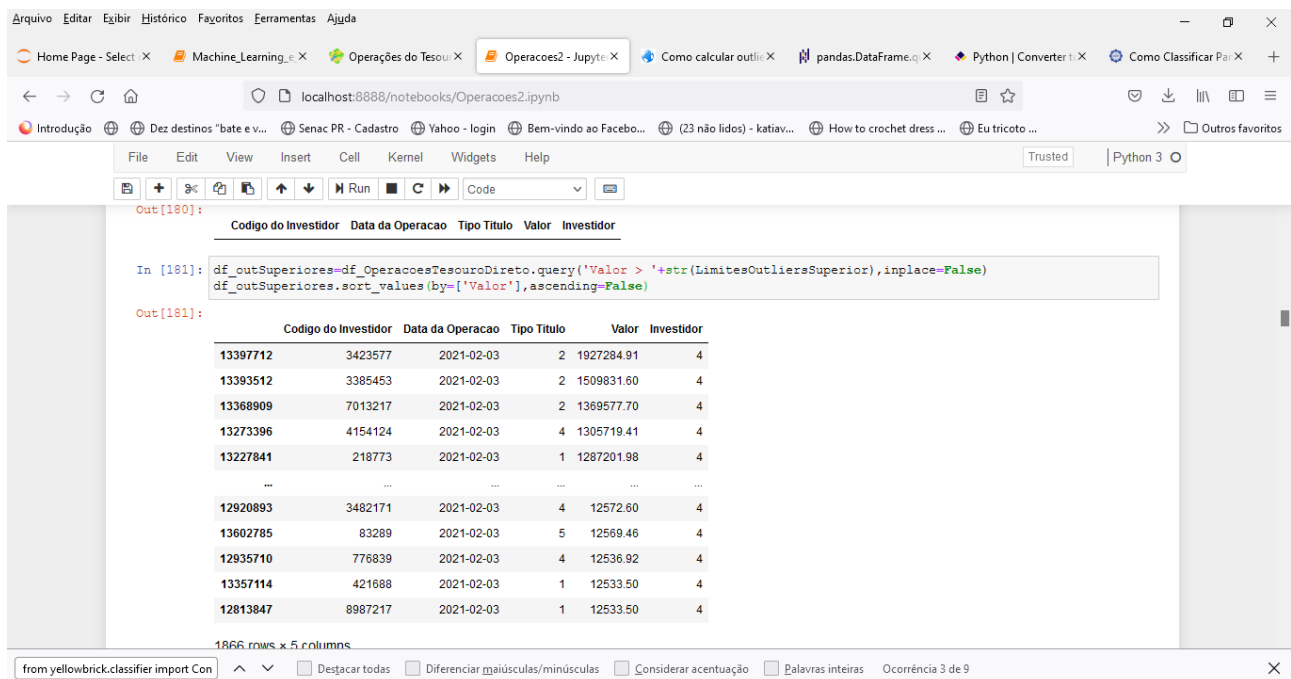


Por fim, analisando-se os valores extremos máximos e mínimos encontrados pelo “Pandas Profiling”, figuras 15 e 16, foi constatado que todos os valores máximos estão contidos na lista de *outliers*. Apesar dessa constatação, decidimos manter o dataset original.

Figura 15 – Identificando valores extremos com Intervalo-Interquartil e criando os limites inferiores e superiores.

```
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
Home Page - Select or create a... Machine_Learning_e... Operacoes2 - Jupyter... Como calcular outliers pandas.DataFrame() Python | Converter t... Como Classificar Pa...
localhost:8888/notebooks/Operacoes2.ipynb
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
Run Code
In [ ]:
Identificando valores extremos com Intervalo-Interquartil
In [141]: Q1=df_OperacoesTesouroDireto['Valor'].quantile(0.25) ## 1o. quartil
Q3=df_OperacoesTesouroDireto['Valor'].quantile(0.75) ## 3o. quartil
IIQ=Q3-Q1##amplitude entre quartis
print('Primeiro quartil',Q1)
print('Terceiro quartil',Q3)
print('IIQ',IIQ)
Primeiro quartil 173.42
Terceiro quartil 5105.2775
IIQ 4931.8575
Calculando limites inferiores e superiores
In [175]: valor = 1.5 * IIQ
LimitesOutliersSuperior = (Q3 + valor)
LimitesOutliersInferior = (Q1 - valor)
In [ ]:
```

Figura 16 - Criando dataset de valores superiores.



```
from yellowbrick.classifier import Con

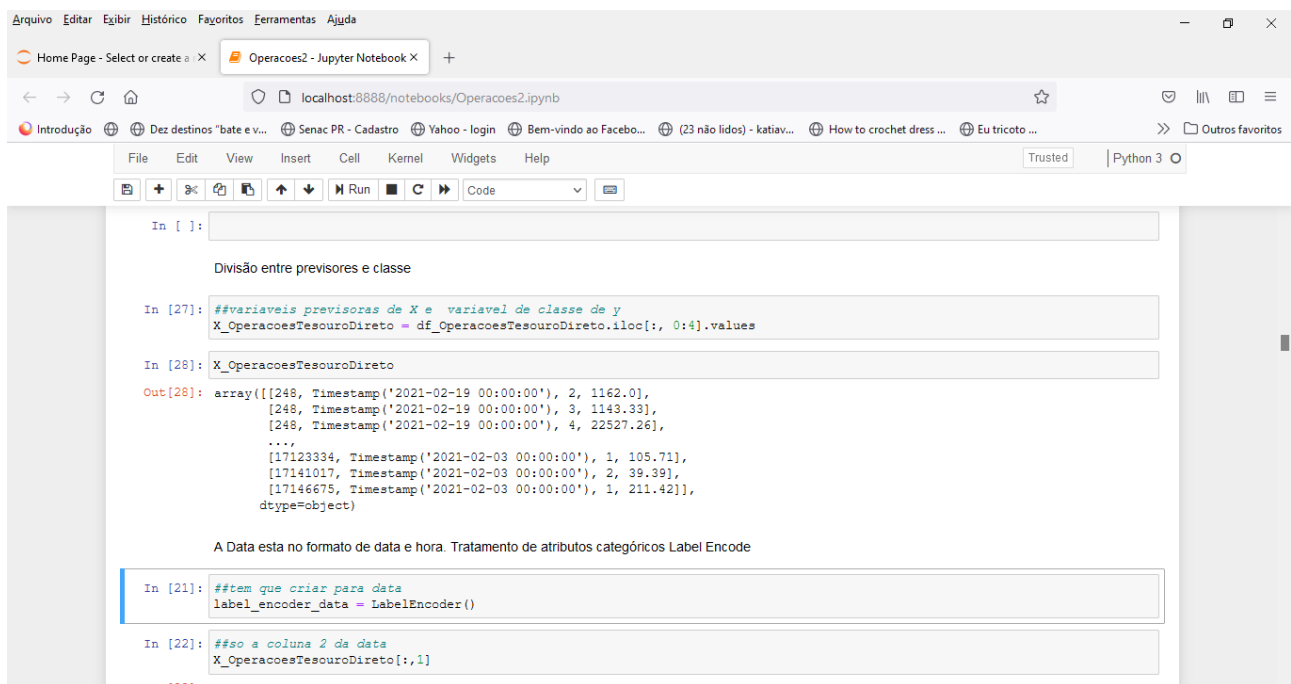
# Destacar todas
# Diferenciar maiúsculas/minúsculas
# Considerar acentuação
# Palavras inteiras
Ocorrência 3 de 9
```

Codigo do Investidor	Data da Operacao	Tipo Titulo	Valor	Investidor	
13397712	3423577	2021-02-03	2	1927284.91	4
13393512	3385453	2021-02-03	2	1509831.60	4
13368909	7013217	2021-02-03	2	1369577.70	4
13273396	4154124	2021-02-03	4	1305719.41	4
13227841	218773	2021-02-03	1	1287201.98	4
...
12920893	3482171	2021-02-03	4	12572.60	4
13602785	83289	2021-02-03	5	12569.46	4
12935710	776839	2021-02-03	4	12536.92	4
13357114	421688	2021-02-03	1	12533.50	4
12813847	8987217	2021-02-03	1	12533.50	4

1866 rows x 5 columns

Com o objetivo de realizar alguns exercícios em um *dataframe* Pandas optou-se por criar um novo *dataset* com o X (previsor) e y (alvo), conforme figura 17.

Figura 17 – Divisão entre previsores.



```
In [ ]:

Divisão entre previsores e classe

In [27]: ##variaveis previsoras de X e variavel de classe de y
X_OperacoesTesouroDireto = df_OperacoesTesouroDireto.iloc[:, 0:4].values

In [28]: X_OperacoesTesouroDireto
Out[28]: array([[248, Timestamp('2021-02-19 00:00:00'), 2, 1162.0],
                [248, Timestamp('2021-02-19 00:00:00'), 3, 1143.33],
                [248, Timestamp('2021-02-19 00:00:00'), 4, 22527.26],
                ...,
                [17123334, Timestamp('2021-02-03 00:00:00'), 1, 105.71],
                [17141017, Timestamp('2021-02-03 00:00:00'), 2, 39.39],
                [17146675, Timestamp('2021-02-03 00:00:00'), 1, 211.42]],
                dtype=object)

A Data esta no formato de data e hora. Tratamento de atributos categóricos Label Encode

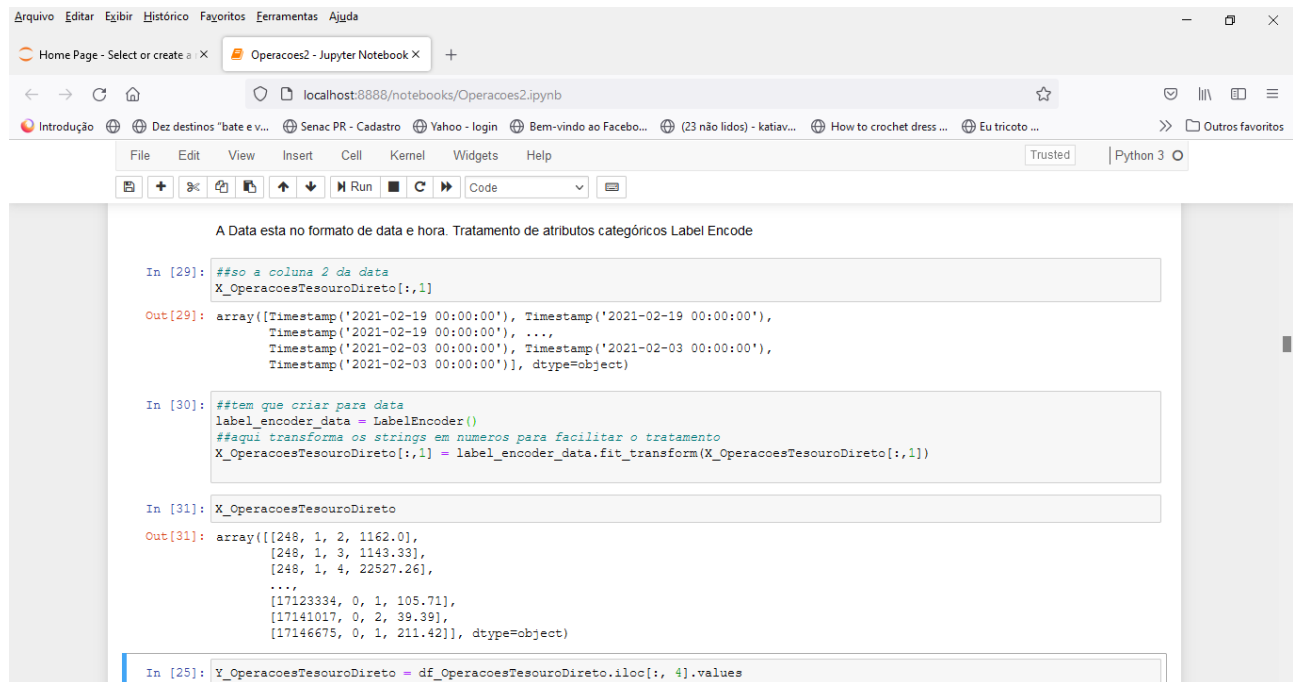
In [21]: ##tem que criar para data
label_encoder_data = LabelEncoder()

In [22]: ##so a coluna 2 da data
X_OperacoesTesouroDireto[:,1]
```

Notou-se a necessidade de tratamento da Data da Operação que está como formato “timestamp” para numérico facilitando a análise dos dados. Figura 18.

Fiz uma codificação com o `LabelEncoder()` e apliquei o `fit_transform` para transformar as datas em código de 0 e 1, pois só existiam duas datas no nosso Dataframe para o mês de fevereiro de 2021.

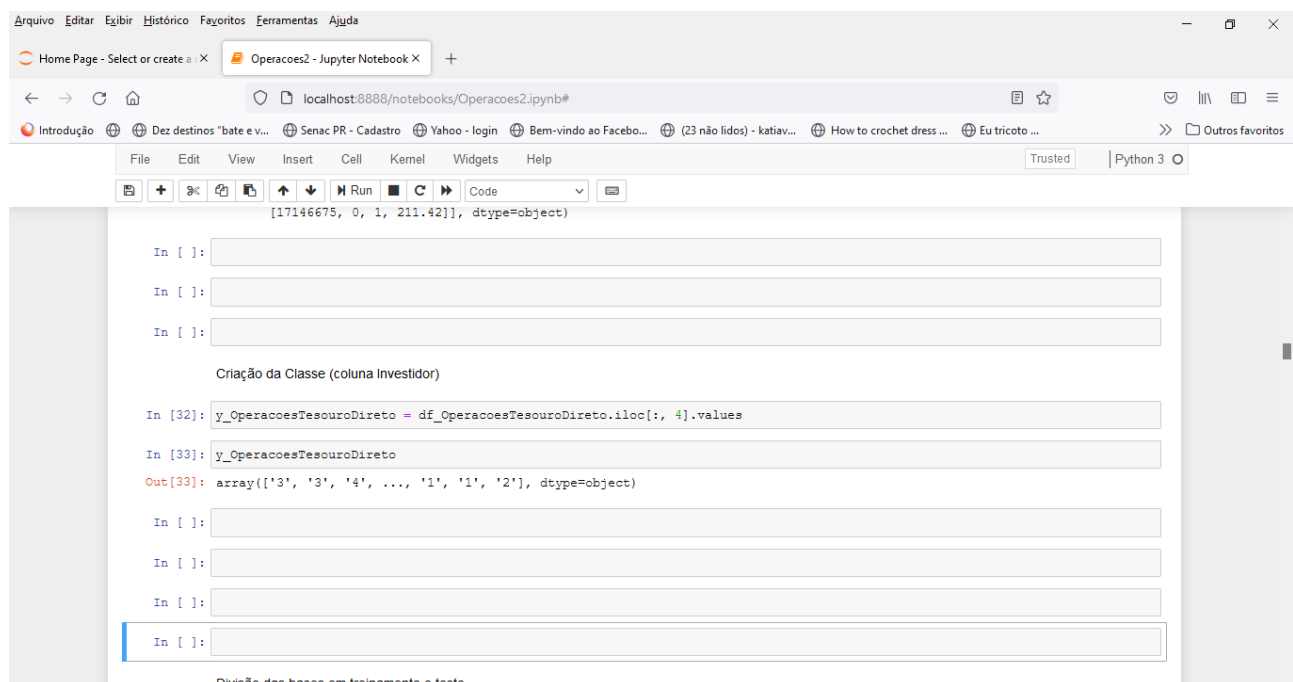
Figura 18 – Tratamento da data da operação que estava como timestamp.



```
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
Operacoes2 - Jupyter Notebook X
localhost:8888/notebooks/Operacoes2.ipynb
Introdução
Dez destinos "bate e v..." Senac PR - Cadastro Yahoo - login Bem-vindo ao Facebo... (23 não lidos) - katiav... How to crochet dress ... Eu tricoto ...
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [29]: ##so a coluna 2 da data
X_OperacoesTesouroDireto[:,1]
Out[29]: array([Timestamp('2021-02-19 00:00:00'), Timestamp('2021-02-19 00:00:00'),
Timestamp('2021-02-19 00:00:00'), ...,
Timestamp('2021-02-03 00:00:00'), Timestamp('2021-02-03 00:00:00'),
Timestamp('2021-02-03 00:00:00')], dtype=object)
In [30]: ##tem que criar para data
label_encoder_data = LabelEncoder()
##aqui transforma os strings em numeros para facilitar o tratamento
X_OperacoesTesouroDireto[:,1] = label_encoder_data.fit_transform(X_OperacoesTesouroDireto[:,1])
In [31]: X_OperacoesTesouroDireto
Out[31]: array([[248, 1, 2, 1162.0],
[248, 1, 3, 1143.33],
[248, 1, 4, 22527.26],
...,
[17123334, 0, 1, 105.71],
[17141017, 0, 2, 39.39],
[17146675, 0, 1, 211.42]], dtype=object)
In [25]: Y_OperacoesTesouroDireto = df_OperacoesTesouroDireto.iloc[:, 4].values
```

A coluna de Investidores foi a definida como o alvo ou classe. Figura 19.

Figura 19 – Criação da coluna alvo ou classe.



```
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
Operacoes2 - Jupyter Notebook X
localhost:8888/notebooks/Operacoes2.ipynb#
Introdução
Dez destinos "bate e v..." Senac PR - Cadastro Yahoo - login Bem-vindo ao Facebo... (23 não lidos) - katiav... How to crochet dress ... Eu tricoto ...
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
[17146675, 0, 1, 211.42]], dtype=object)
In [ ]:
In [ ]:
In [ ]:
Criação da Classe (coluna Investidor)
In [32]: y_OperacoesTesouroDireto = df_OperacoesTesouroDireto.iloc[:, 4].values
In [33]: y_OperacoesTesouroDireto
Out[33]: array(['3', '3', '4', ..., '1', '1', '2'], dtype=object)
In [ ]:
In [ ]:
In [ ]:
In [ ]:
```

Divisão das bases em treinamento e teste

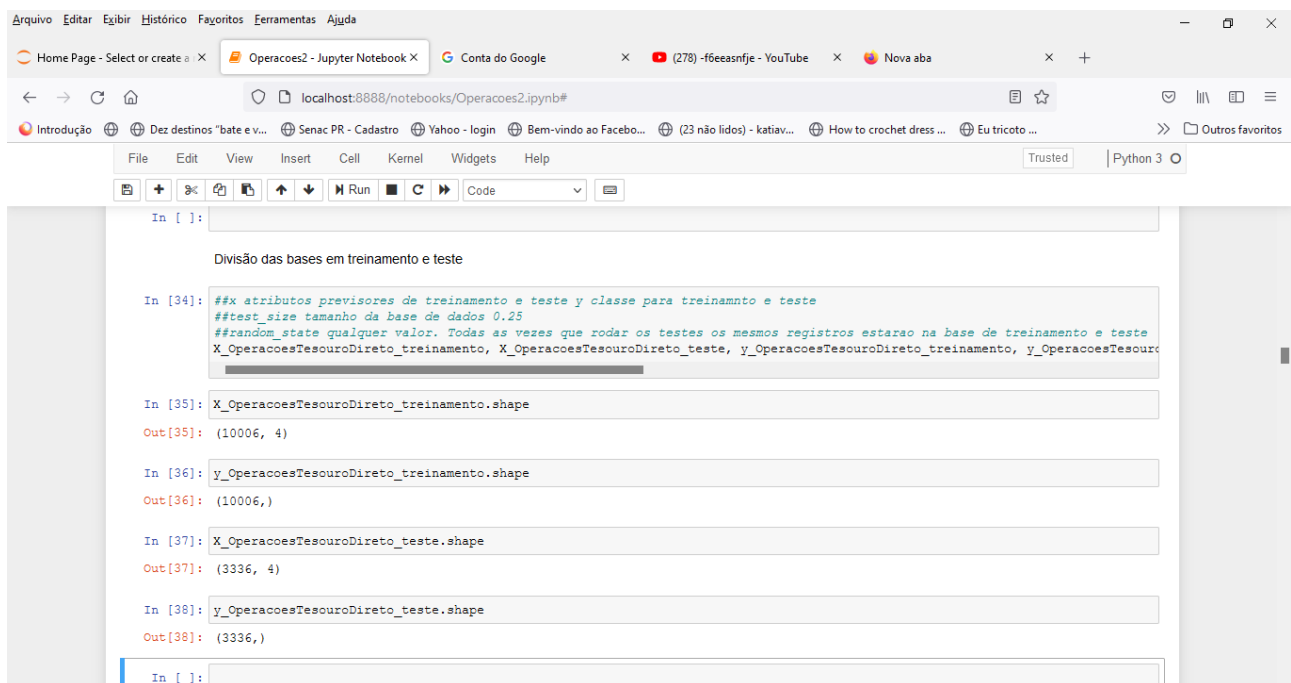
Dados de Treinamento

Conforme podemos imaginar, dados de treino são os dados que serão apresentados ao algoritmo de machine learning para criação do modelo. No caso, estamos representando cerca de 75% da totalidade dos dados.

Dados de Teste

Serão apresentados ao modelo após a sua criação, simulando previsões reais que o modelo realizará, permitindo assim que o desempenho real seja verificado. No caso, estamos representando cerca de 25% da totalidade dos dados.

Figura 20 - Divisão de treinamento e teste.



```
In [ ]:

Divisão das bases em treinamento e teste

In [34]: ##x atributos previsoers de treinamento e teste y classe para treinamnto e teste
##test_size tamanho da base de dados 0.25
##random_state qualquer valor. Todas as vezes que rodar os testes os mesmos registros estarao na base de treinamento e teste
X_OperacoesTesouroDireto_treinamento, X_OperacoesTesouroDireto_teste, y_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_teste = train_test_split(X_OperacoesTesouroDireto, y_OperacoesTesouroDireto, test_size=0.25, random_state=42)

In [35]: X_OperacoesTesouroDireto_treinamento.shape
Out[35]: (10006, 4)

In [36]: y_OperacoesTesouroDireto_treinamento.shape
Out[36]: (10006,)

In [37]: X_OperacoesTesouroDireto_teste.shape
Out[37]: (3336, 4)

In [38]: y_OperacoesTesouroDireto_teste.shape
Out[38]: (3336,)
```

5. Criação de Modelos de *Machine Learning*

Segundo Géron (2019), *machine learning* é a ciência (e a arte) da programação de computadores para que eles possam aprender com os dados⁸. Mas há outras definições mais abrangentes, como o campo de estudo que dá aos computadores a habilidade de aprender sem ser explicitamente programado, de Arthur Samuel (1959). Em relação à possibilidade de serem ou não treinados com supervisão humana, os sistemas de *machine learning* podem ser classificados em: supervisionados, não supervisionados, semi supervisionados e aprendizado por reforço, podendo ser combinados com outras classificações.

A classificação, objetivo do atual projeto, é uma das tarefas típicas do aprendizado supervisionado, em que os dados de treinamento fornecidos ao algoritmo encontram-se rotulados, ou seja, contêm as soluções desejadas.

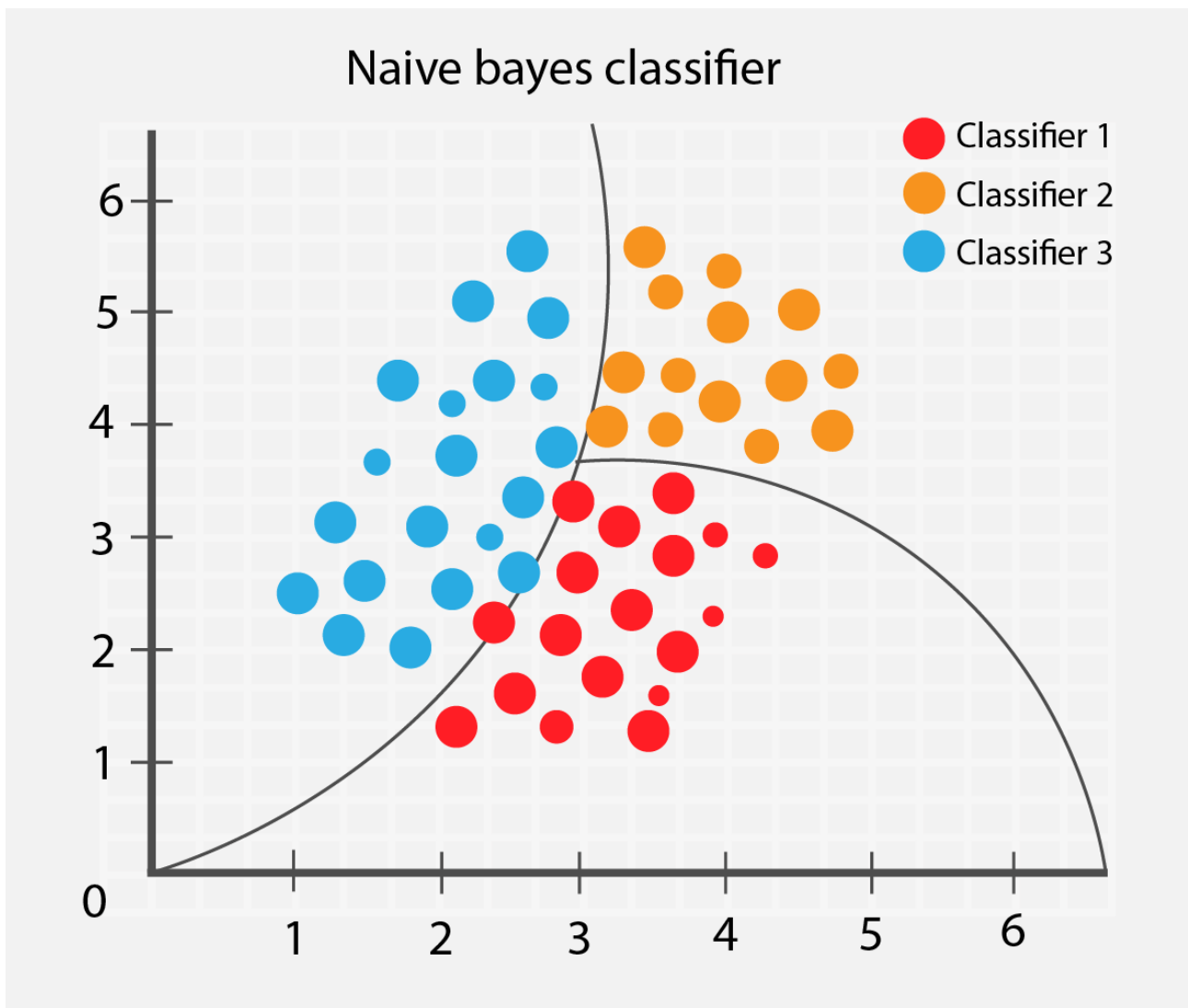
Optou-se por comparar o desempenho de cinco algoritmos de *machine learning* para classificação:

1. Decision Tree Classifier (CART): Árvore de Decisão;
2. Random Forest Classifier (RFC): Floresta Aleatória.
3. k-Nearest Neighbors (k-NN): k-Vizinhos Mais Próximos;
4. Support Vector Machines (SVM): Máquinas de Vetores de de Suporte;e
5. *Neural Networks* (RNAs): Redes Neurais Artificiais

Naive Bayes é um conjunto de algoritmos com base na aplicação do teorema de Bayes com a suposição "ingênua" (*naive*), na verdade uma forte suposição de independência condicional entre os nós filhos no contexto de seu pai. São redes muito simples com apenas um pai (representando o nó não observado) e vários filhos (correspondendo aos nós observados).

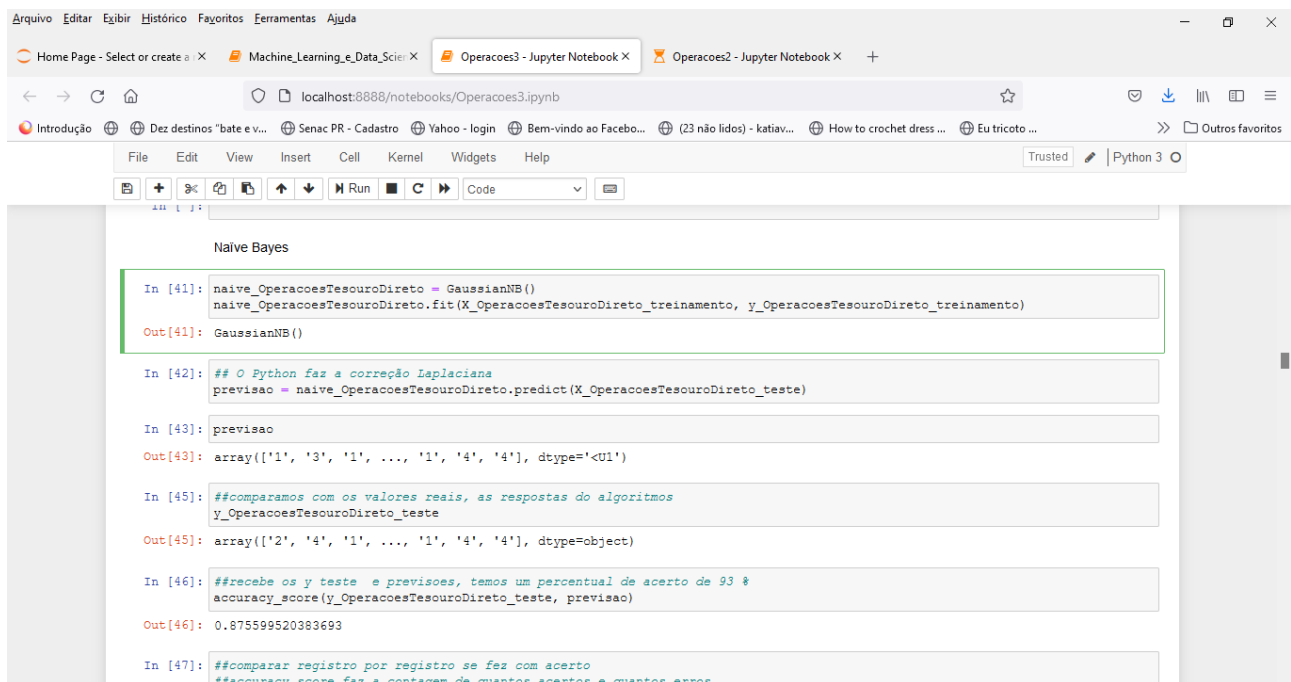
8 - Géron, Aurélien. Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow (p. 4). Edição do Kindle.

Figura 21 – Representação de classificação com Naive Bayes⁹.



9 - Fonte: <https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf>

Figura 22 – Código da classificação com Naive Bayes.



The screenshot shows a Jupyter Notebook titled 'Operacoes3 - Jupyter Notebook' running on a local host. The notebook contains several code cells for Naive Bayes classification. The first cell initializes a Gaussian Naive Bayes model and fits it to training data. The second cell uses the model to predict on test data, applying Laplace correction. The third cell compares the predicted results with the actual test results. The fourth cell calculates the accuracy score, which is approximately 0.875. The fifth cell contains comments about comparing the results register by register.

```
Naive Bayes

In [41]: naive_OperacoesTesouroDireto = GaussianNB()
naive_OperacoesTesouroDireto.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)

Out[41]: GaussianNB()

In [42]: ## O Python faz a correção Laplaciana
previsao = naive_OperacoesTesouroDireto.predict(X_OperacoesTesouroDireto_teste)

In [43]: previsao
Out[43]: array(['1', '3', '1', ..., '1', '4', '4'], dtype='<U1')

In [45]: ##comparamos com os valores reais, as respostas do algoritmos
y_OperacoesTesouroDireto_teste
Out[45]: array(['2', '4', '1', ..., '1', '4', '4'], dtype=object)

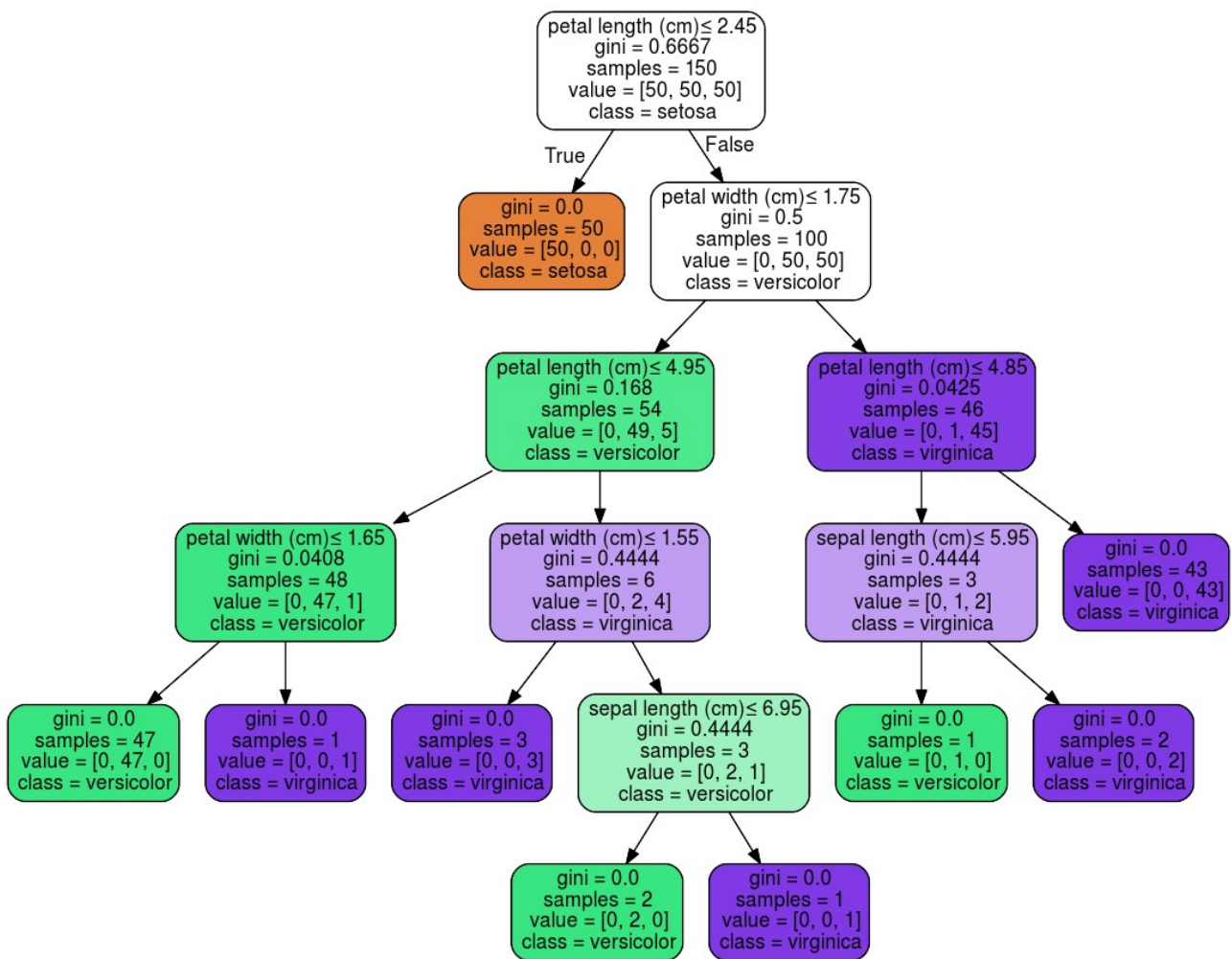
In [46]: ##recebe os y teste e previsoes, temos um percentual de acerto de 93 %
accuracy_score(y_OperacoesTesouroDireto_teste, previsao)
Out[46]: 0.875599520383693

In [47]: ##comparar registro por registro se fez com acerto
##accuracy score faz a contagem de quantos acertos e quantos erros
```

Árvores de Decisão é uma tabela de decisão sob a forma de árvore com nós e folhas sequenciais e interligados que classificam as instâncias, ordenando-as com base nos valores dos recursos. Cada nó em uma árvore de decisão representa uma característica em uma instância a ser classificada, e cada ramo representa um valor que o nó pode assumir. As instâncias são classificadas começando no nó raiz com base em seus valores de recursos.¹⁰ Trata-se de um dos modelos mais práticos e mais utilizados em inferência por indução (representação visual na figura 23).

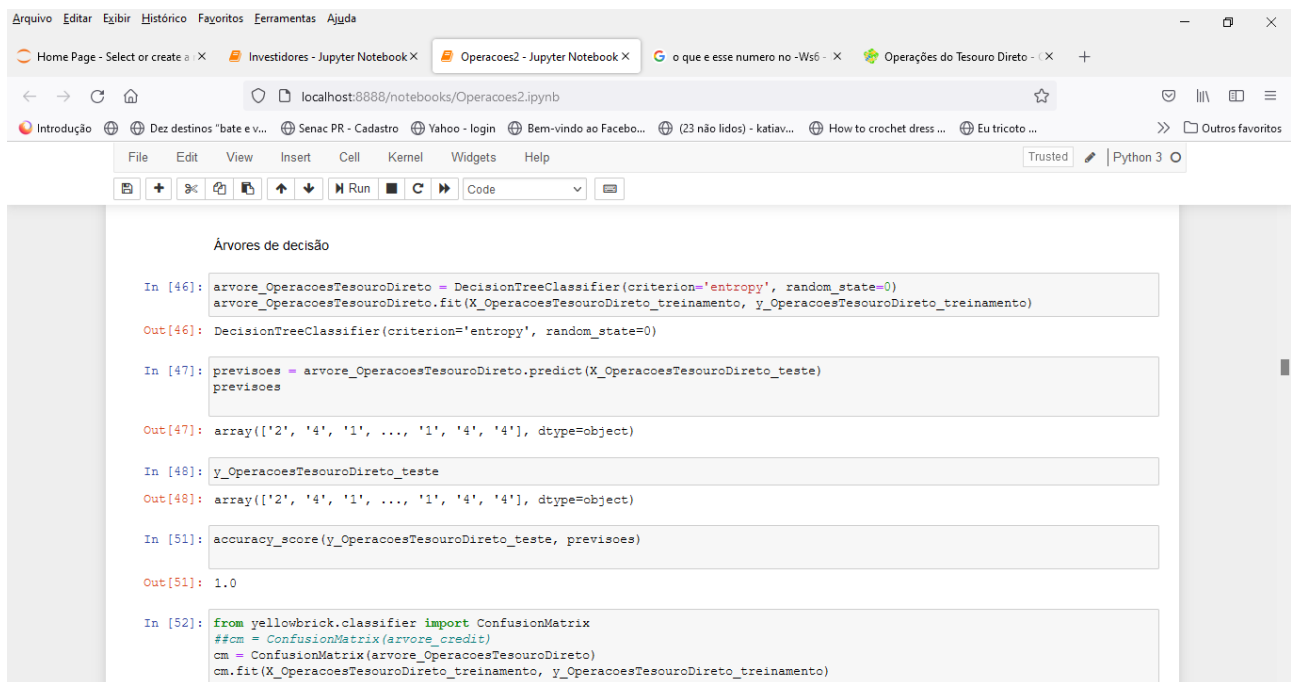
¹⁰ - KOTSIANTIS, Sotiris B.; ZAHARAKIS, Ioannis D.; PINTELAS, Panayiotis E. Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, v. 26, n. 3, p.159-190, 2006.

Figura 23 – Representação visual do classificador Árvores de Decisão¹¹.



11 - Fonte: <https://scikit-learn.org/stable/modules/tree.html>

Figura 24 – Código do classificador Árvores de Decisão.



```
Árvores de decisão

In [46]: arvore_OperacoesTesouroDireto = DecisionTreeClassifier(criterion='entropy', random_state=0)
arvore_OperacoesTesouroDireto.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)

Out[46]: DecisionTreeClassifier(criterion='entropy', random_state=0)

In [47]: previsoes = arvore_OperacoesTesouroDireto.predict(X_OperacoesTesouroDireto_teste)
previsoes

Out[47]: array(['2', '4', '1', ..., '1', '4', '4'], dtype=object)

In [48]: y_OperacoesTesouroDireto_teste

Out[48]: array(['2', '4', '1', ..., '1', '4', '4'], dtype=object)

In [51]: accuracy_score(y_OperacoesTesouroDireto_teste, previsoes)

Out[51]: 1.0

In [52]: from yellowbrick.classifier import ConfusionMatrix
##cm = ConfusionMatrix(arvore_credito)
cm = ConfusionMatrix(arvore_OperacoesTesouroDireto)
cm.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)
cm.score(X_OperacoesTesouroDireto_teste, y_OperacoesTesouroDireto_teste)
```

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

Na prática, significa treinar um conjunto de classificadores de árvores de decisão, cada um em um subconjunto aleatório diferente do conjunto de treinamento, e fazer as previsões, obtendo-as de todas as árvores individuais, e, então, prever a classe que obtém a maioria dos votos¹². No exemplo da figura 25 vemos um *ensemble* de árvores de decisão, em que a Class C teve o maior número de resultados nas árvores individuais, sendo, portanto classificada como resultado final.

12 - Géron, Aurélien. Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow (p. 185). Edição do Kindle

Figura 25 – Representação de Floresta Aleatória¹³.

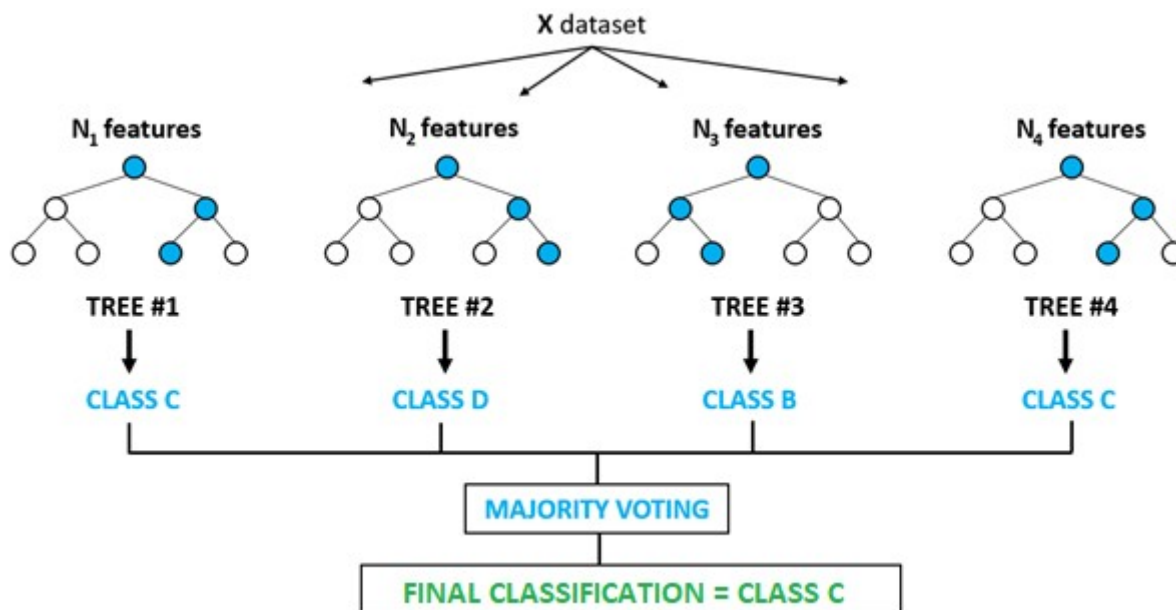


Figura 26 - Código da Representação de Floresta Aleatória.

```
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
Machine_Learning_e_Data_Scie X Operacoes3 - Jupyter Notebook X Operacoes2 - Jupyter Notebook X
localhost:8888/notebooks/Operacoes2.ipynb
Introdução Dez destinos "bate e v... Senac PR - Cadastro Yahoo - login Bem-vindo ao Facebo... (23 não lidos) - katiav... How to crochet dress ... Eu tricotado ...
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
Random Forest
In [56]: ##cria algoritmo
random_forest_OperacoesTesouroDireto = RandomForestClassifier(n_estimators=100, criterion='entropy', random_state = 0)
random_forest_OperacoesTesouroDireto.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)
Out[56]: RandomForestClassifier(criterion='entropy', random_state=0)
In [57]: previsoes = random_forest_OperacoesTesouroDireto.predict(X_OperacoesTesouroDireto_teste)
previsoes
Out[57]: array(['2', '4', '1', ..., '1', '4', '4'], dtype=object)
In [58]: ##dados reais. acertou para todos e menos o ultimo
y_OperacoesTesouroDireto_teste
Out[58]: array(['2', '4', '1', ..., '1', '4', '4'], dtype=object)
In [59]: ##qual o accuracy. Melhorou
from sklearn.metrics import accuracy_score, classification_report
accuracy_score(y_OperacoesTesouroDireto_teste, previsoes)
Out[59]: 1.0
In [60]: ##
```

Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

Home Page - Select or create a... Machine_Learning_e_Data_Scie: X Operacoes3 - Jupyter Notebook X Operacoes2 - Jupyter Notebook X

localhost:8888/notebooks/Operacoes2.ipynb

Introdução Dez destinos "bate e v... Senac PR - Cadastro Yahoo - login Bem-vindo ao Facebo... (23 não lidos) - katiav... How to crochet dress ... Eu trico ... Outros favoritos

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [59]: ##qual o accuracy. Melhorou
from sklearn.metrics import accuracy_score, classification_report
accuracy_score(y_OperacoesTesouroDireto_teste, previsoes)
```

Out[59]: 1.0

```
In [62]: ##
from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(random_forest_OperacoesTesouroDireto)
cm.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)
cm.score(X_OperacoesTesouroDireto_teste, y_OperacoesTesouroDireto_teste)
```

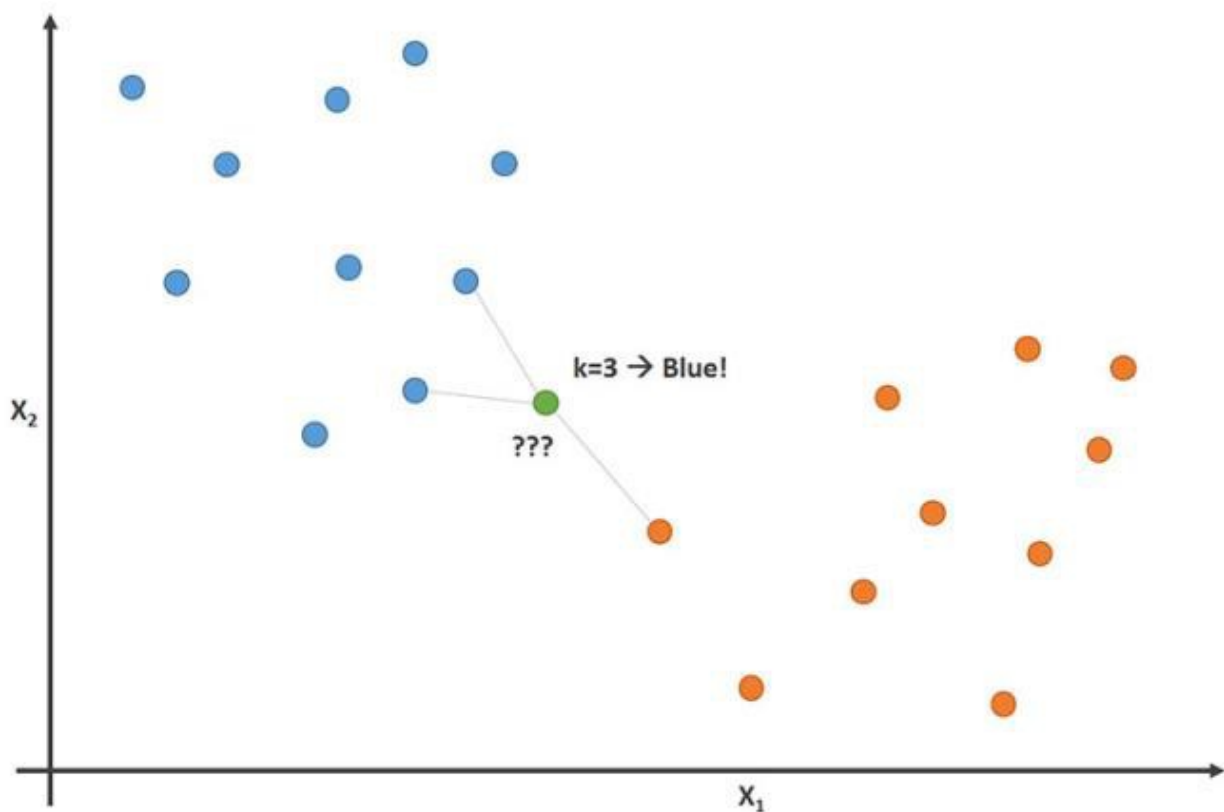
Out[62]: 1.0

```
In [63]: print(classification_report(y_OperacoesTesouroDireto_teste, previsoes))
```

	precision	recall	f1-score	support
1	1.00	1.00	1.00	813
2	1.00	1.00	1.00	869
3	1.00	1.00	1.00	809
4	1.00	1.00	1.00	845
accuracy			1.00	3336
macro avg	1.00	1.00	1.00	3336
weighted avg	1.00	1.00	1.00	3336

K-Vizinhos Mais Próximos é um dos mais simples algoritmos de aprendizagem baseados em instância. Segue o princípio de que as instâncias dentro de um conjunto de dados geralmente existem próximas a outras instâncias que possuem propriedades semelhantes. Se as instâncias são marcadas com um rótulo de classificação, então o valor do rótulo de uma instância não classificada pode ser determinado observando-se a classe de seus vizinhos mais próximos. O k-NN localiza as k instâncias mais próximas da instância da consulta e determina sua classe, identificando o único rótulo de classe mais frequente¹⁴, conforme a figura 27.

Figura 27 – Representação da classificação do algoritmo k-Vizinhos Mais Próximos¹⁵.



14 - KOTSIANTIS, Sotiris B.; ZAHARAKIS, Ioannis D.; PINTELAS, Panayiotis E. Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, v. 26, n. 3, p.159-190, 2006.

15 - Fonte: <https://rapidminer.com/blog/k-nearest-neighbors-laziest-machine-learning-technique/>

Figura 28 – Código da representação da classificação do algoritmo k-Vizinhos Mais Próximos.

macro avg	1.00	1.00	1.00	3336
weighted avg	1.00	1.00	1.00	3336

Aprendizagem baseada em instâncias - knn

```
In [64]: knn_OperacoesTesouroDireto = KNeighborsClassifier(n_neighbors=3, metric='minkowski', p = 2)
knn_OperacoesTesouroDireto.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)

Out[64]: KNeighborsClassifier(n_neighbors=3)

In [65]: previsoes = knn_OperacoesTesouroDireto.predict(X_OperacoesTesouroDireto_teste)
previsoes

Out[65]: array(['2', '3', '2', ..., '3', '4', '4'], dtype=object)

In [66]: y_OperacoesTesouroDireto_teste

Out[66]: array(['2', '4', '1', ..., '1', '4', '4'], dtype=object)

In [68]: from sklearn.metrics import accuracy_score, classification_report
accuracy_score(y_OperacoesTesouroDireto_teste, previsoes)

Out[68]: 0.5422661870503597

In [69]: from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(knn_OperacoesTesouroDireto)
cm.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)
```

As Support Vector Machines, ou Máquinas de Vetores de Suporte em português, estão entre as técnicas mais recentes de aprendizado supervisionado. As últimas pesquisas giram em torno da noção de uma "margem", ou seja, qualquer um dos lados de um hiperplano que separa duas classes de dados, maximizando a margem e criando, assim, a maior distância possível entre o hiperplano de separação¹⁶.

Na prática, significa encontrar o “hiperplano”, uma linha de separação entre os dados de duas classes, buscando maximizar a distância entre os pontos mais próximos em relação a cada uma das classes ou classificações, conforme representação gráfica da figura 29.

16 - KOTSIANTIS, Sotiris B.; ZAHARAKIS, Ioannis D.; PINTELAS, Panayiotis E. Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, v. 26, n. 3, p.159-190, 2006.

Figura 29 – Representação de classificação com Máquinas de Vetores de Suporte¹⁷.

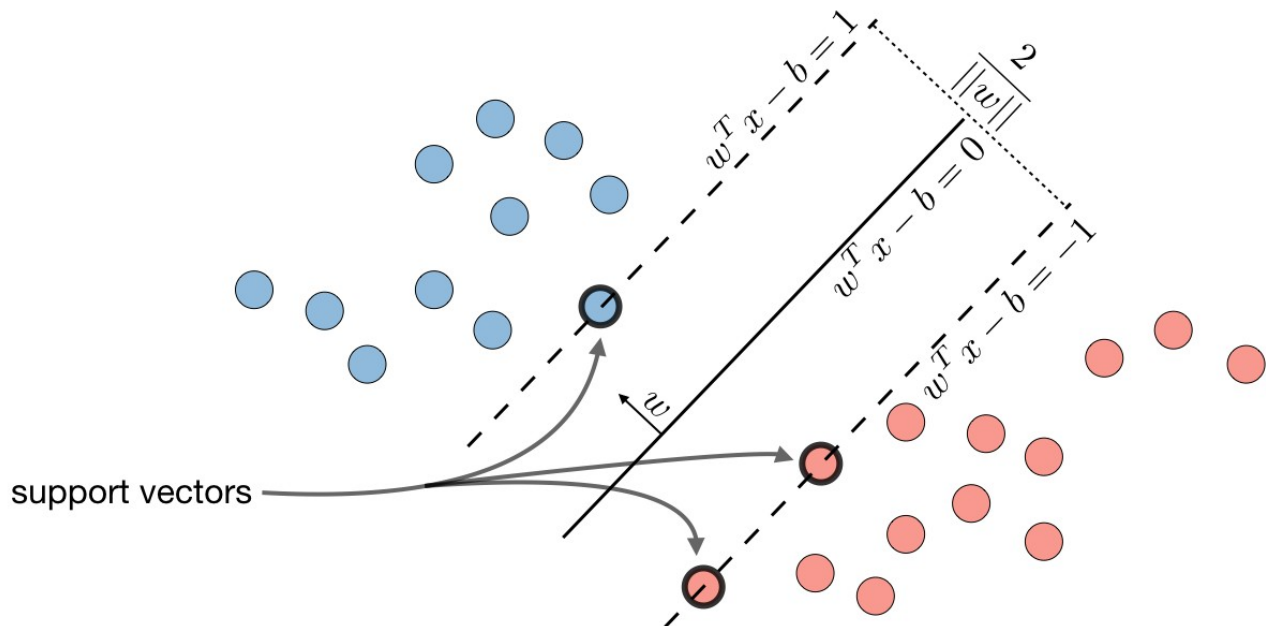


Figura 30 – Código da classificação com Máquinas de Vetores de Suporte.

```
Arquivo  Editar  Exibir  Histórico  Favoritos  Ferramentas  Ajuda

Home Page - Select or create a...  Machine_Learning_e_Data_Scie...  Operacoes3 - Jupyter Notebook X  Operacoes2 - Jupyter Notebook X  +

localhost:8888/notebooks/Operacoes2.ipynb

Introdução  Dez destinos "bate e v...  Senac PR - Cadastro  Yahoo - login  Bem-vindo ao Facebo...  (23 não lidos) - katiav...  How to crochet dress ...  Eu tricot...  >>  Outros favoritos

File  Edit  View  Insert  Cell  Kernel  Widgets  Help  Trusted  Python 3

Out[69]: 0.5422661870503597

SVM

In [70]: ##demora menos e da accuracy baixinho 0.36360911270983215
svm_OperacoesTesouroDireto = SVC(kernel='rbf', random_state=1, C = 2.0) # 2 -> 4
svm_OperacoesTesouroDireto.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)

Out[70]: SVC(C=2.0, random_state=1)

In [71]: previsoes = svm_OperacoesTesouroDireto.predict(X_OperacoesTesouroDireto_teste)
previsoes

Out[71]: array(['2', '4', '4', ..., '2', '4', '3'], dtype=object)

In [72]: y_OperacoesTesouroDireto_teste

Out[72]: array(['2', '4', '1', ..., '1', '4', '4'], dtype=object)

In [73]: from sklearn.metrics import accuracy_score, classification_report
accuracy_score(y_OperacoesTesouroDireto_teste, previsoes)

Out[73]: 0.36360911270983215

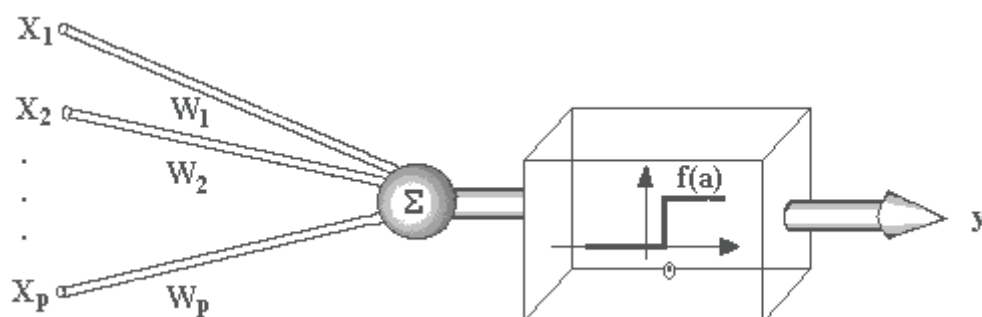
In [74]: from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(svm_OperacoesTesouroDireto)
cm.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)
cm.score(X_OperacoesTesouroDireto_teste, y_OperacoesTesouroDireto_teste)
```

Uma rede neural artificial é composta por várias unidades de processamento, cujo funcionamento é bastante simples. Essas unidades, geralmente são conectadas por canais de comunicação que estão associados a determinado peso. As unidades fazem operações apenas sobre seus dados locais, que são entradas recebidas pelas suas conexões. O comportamento inteligente de uma Rede Neural Artificial vem das interações entre as unidades de processamento da rede¹⁸.

A operação de uma unidade de processamento, proposta por McCullock e Pitts em 1943, pode ser resumida da seguinte maneira:

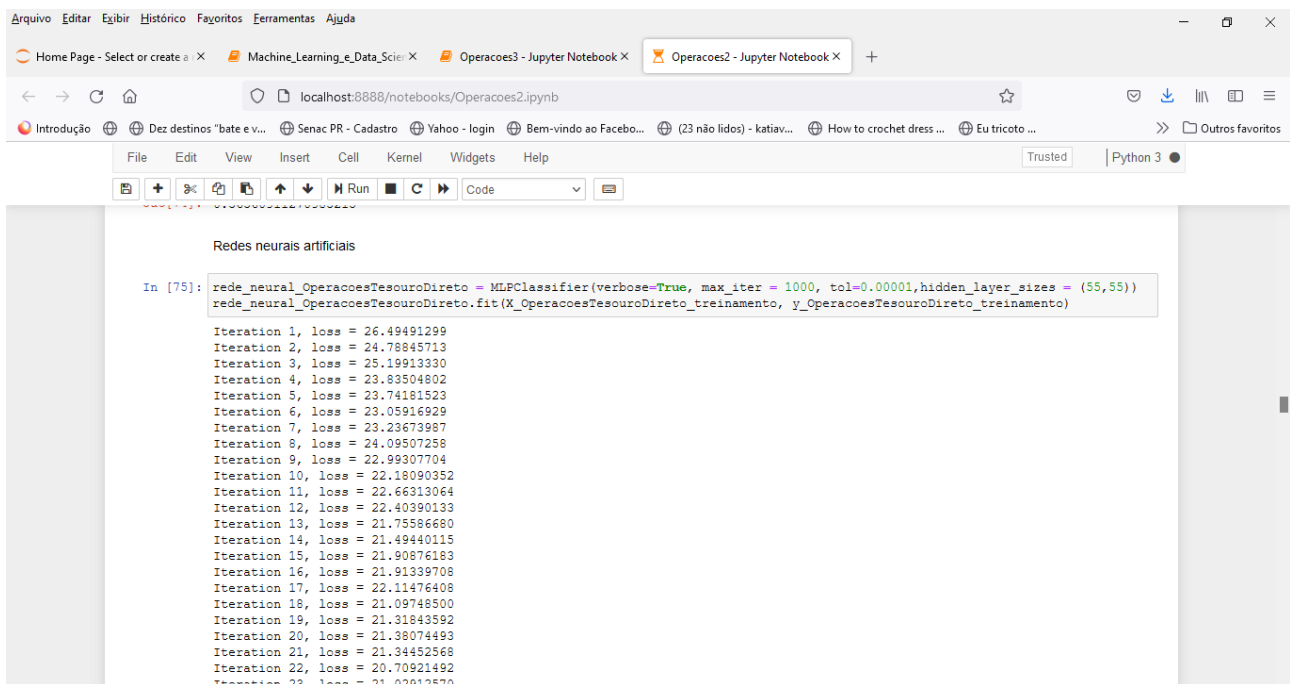
- Sinais são apresentados à entrada;
- Cada sinal é multiplicado por um número, ou peso, que indica a sua influência na saída da unidade;
- É feita a soma ponderada dos sinais que produz um nível de atividade;
- Se este nível de atividade exceder um certo limite (threshold) a unidade produz uma determinada resposta de saída.

Figura 31 – Representação da classificação da Rede Neural.



Esquema de unidade McCullock - Pitts.

Figura 32 – Código da classificação da Rede Neural.

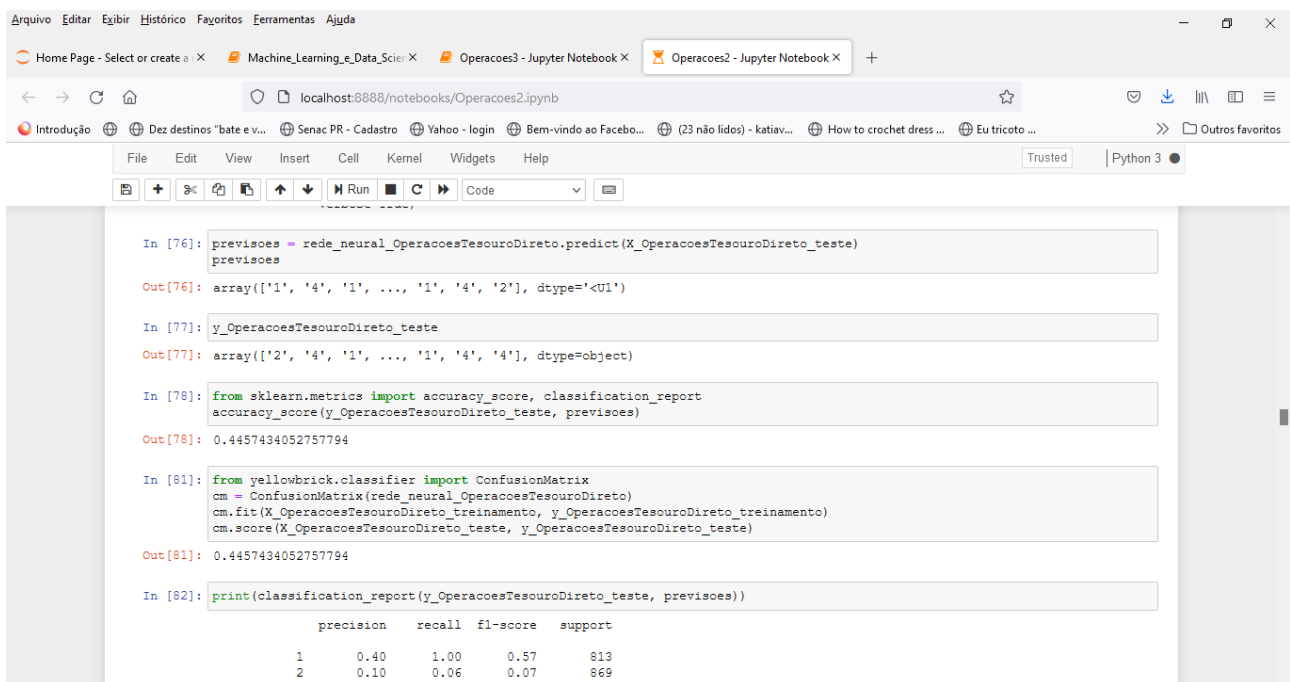


The screenshot shows a Jupyter Notebook interface with the title bar 'Operacoes2 - Jupyter Notebook'. The browser address bar indicates the local host path: 'localhost:8888/notebooks/Operacoes2.ipynb'. The notebook content is titled 'Redes neurais artificiais'. A code cell (In [75]:) contains the following Python code:

```
rede_neural_OperacoesTesouroDireto = MLPClassifier(verbose=True, max_iter = 1000, tol=0.00001, hidden_layer_sizes = (55,55))
rede_neural_OperacoesTesouroDireto.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)
```

The output of the code cell shows the training progress over 23 iterations:

```
Iteration 1, loss = 26.49491299
Iteration 2, loss = 24.78845713
Iteration 3, loss = 25.19913330
Iteration 4, loss = 23.83504802
Iteration 5, loss = 23.74181523
Iteration 6, loss = 23.05916929
Iteration 7, loss = 23.23673987
Iteration 8, loss = 24.09507258
Iteration 9, loss = 22.99307704
Iteration 10, loss = 22.18090352
Iteration 11, loss = 22.66313064
Iteration 12, loss = 22.40390133
Iteration 13, loss = 21.75586680
Iteration 14, loss = 21.49440115
Iteration 15, loss = 21.90876183
Iteration 16, loss = 21.91339708
Iteration 17, loss = 22.11476408
Iteration 18, loss = 21.09748500
Iteration 19, loss = 21.31843592
Iteration 20, loss = 21.38074493
Iteration 21, loss = 21.34452568
Iteration 22, loss = 20.70921492
Iteration 23, loss = 21.02912570
```



The screenshot shows the same Jupyter Notebook interface, now displaying the evaluation of the trained model. The code cell (In [76]:) contains the following Python code:

```
previsoes = rede_neural_OperacoesTesouroDireto.predict(X_OperacoesTesouroDireto_teste)
previsoes
```

The output (Out[76]:) is an array of predicted class labels:

```
array(['1', '4', '1', ..., '1', '4', '2'], dtype='<U1')
```

The next code cell (In [77]:) contains the following Python code:

```
y_OperacoesTesouroDireto_teste
```

The output (Out[77]:) is an array of actual class labels:

```
array(['2', '4', '1', ..., '1', '4', '4'], dtype=object)
```

The next code cell (In [78]:) contains the following Python code:

```
from sklearn.metrics import accuracy_score, classification_report
accuracy_score(y_OperacoesTesouroDireto_teste, previsoes)
```

The output (Out[78]:) is the accuracy score:

```
0.4457434052757794
```

The next code cell (In [81]:) contains the following Python code:

```
from yellowbrick.classifier import ConfusionMatrix
cm = ConfusionMatrix(rede_neural_OperacoesTesouroDireto)
cm.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)
cm.score(X_OperacoesTesouroDireto_teste, y_OperacoesTesouroDireto_teste)
```

The output (Out[81]:) is the accuracy score:

```
0.4457434052757794
```

The final code cell (In [82]:) contains the following Python code:

```
print(classification_report(y_OperacoesTesouroDireto_teste, previsoes))
```

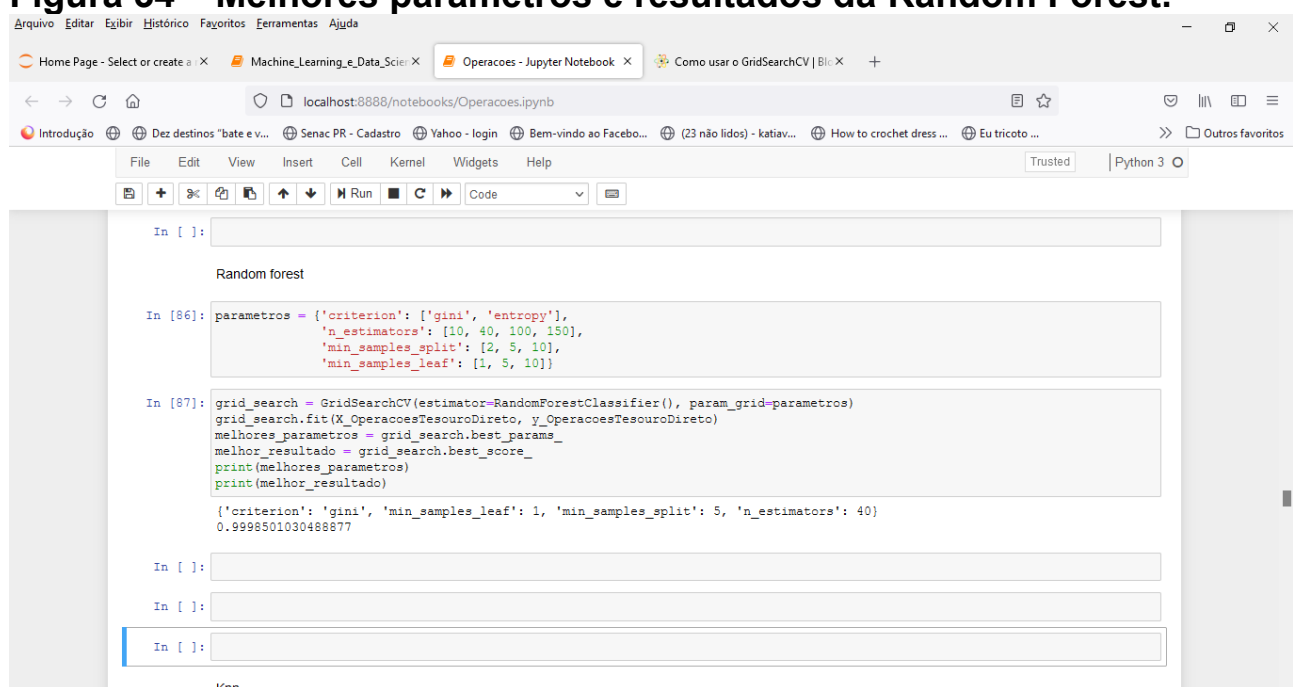
The output shows the classification report:

	precision	recall	f1-score	support
1	0.40	1.00	0.57	813
2	0.10	0.06	0.07	869
4	0.25	0.00	0.00	869

Figura 33 – Melhores parâmetros e resultados da Árvore de Decisão.

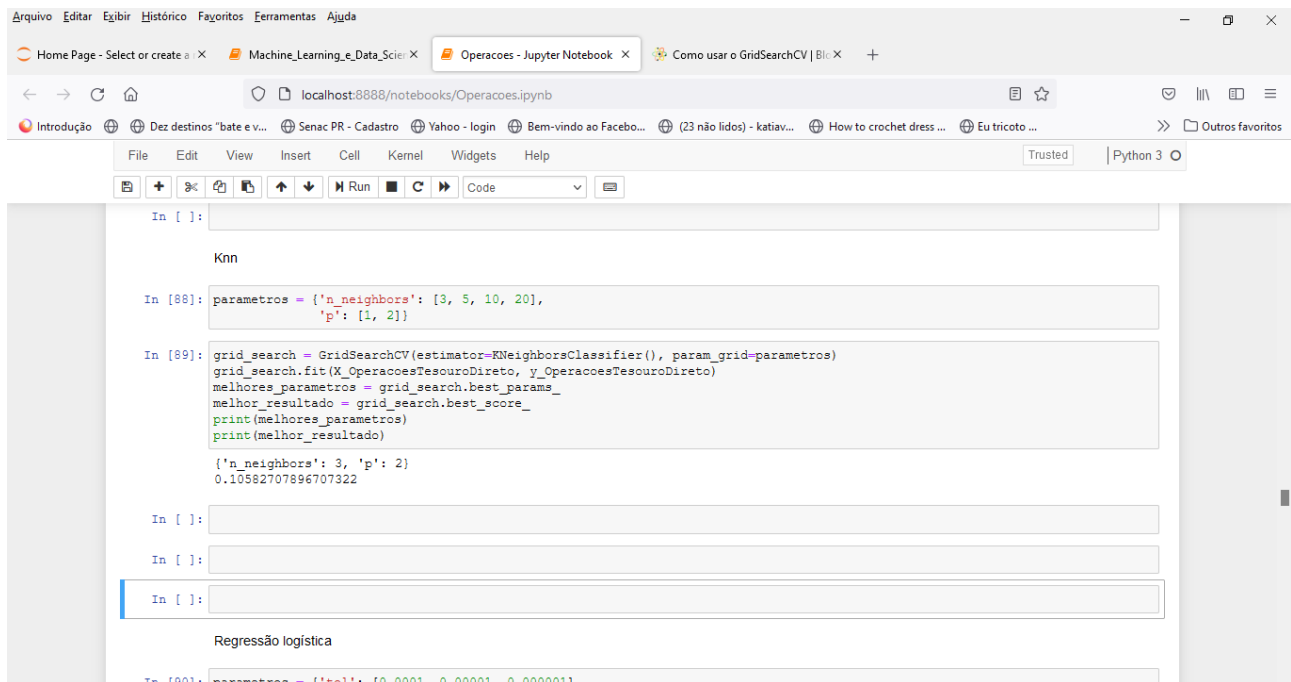


Figura 34 - Melhores parâmetros e resultados da Random Forest.



O terceiro algoritmo a ser avaliado foi Knn, figura 35.

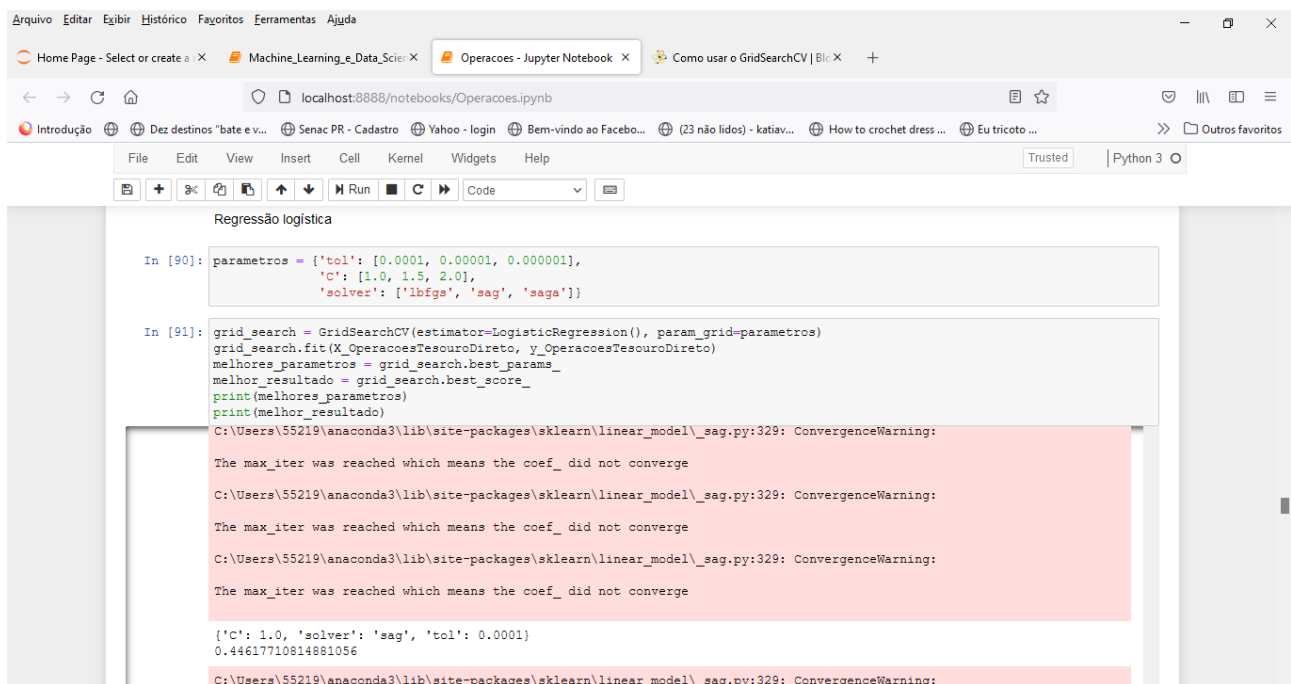
Figura 35 - Melhores parâmetros e resultados do Knn.



```
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
Machine_Learning_e_Data_Scie Operacoes - Jupyter Notebook Como usar o GridSearchCV | Bl...
localhost:8888/notebooks/Operacoes.ipynb
Introdução (23 não lidos) - katiav... How to crochet dress ... Eu tricotico ...
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [ ]:
Knn
In [88]: parametros = {'n_neighbors': [3, 5, 10, 20],
                      'p': [1, 2]}
In [89]: grid_search = GridSearchCV(estimator=KNeighborsClassifier(), param_grid=parametros)
grid_search.fit(X_OperacoesTesouroDireto, y_OperacoesTesouroDireto)
melhores_parametros = grid_search.best_params_
melhor_resultado = grid_search.best_score_
print(melhores_parametros)
print(melhor_resultado)
{'n_neighbors': 3, 'p': 2}
0.10582707896707322
In [ ]:
In [ ]:
In [ ]:
Regressão logística
In [90]: parametros = {'tol': [0.0001, 0.00001, 0.000001],
                      'C': [1.0, 1.5, 2.0],
                      'solver': ['lbfgs', 'sag', 'saga']}
```

O quarto algoritmo a ser avaliado foi Regressão Logística, figura 36.

Figura 36 - Melhores parâmetros e resultados da Regressão Logística.

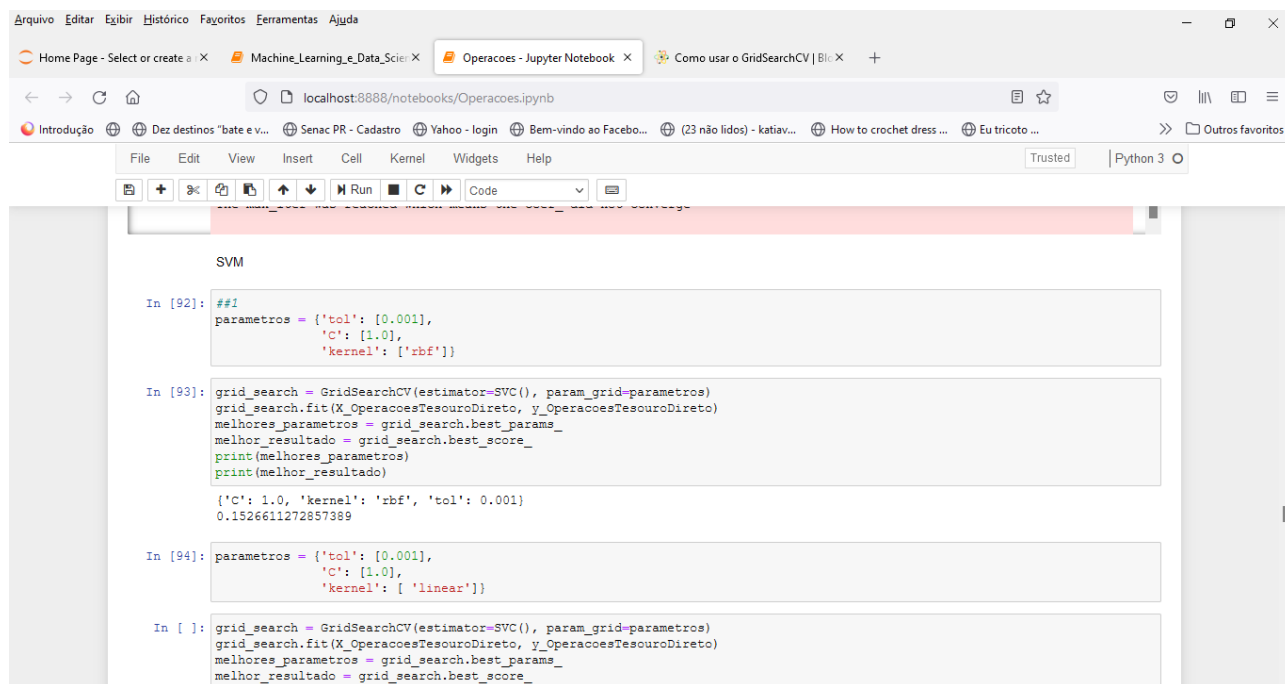


```
Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda
Machine_Learning_e_Data_Scie Operacoes - Jupyter Notebook Como usar o GridSearchCV | Bl...
localhost:8888/notebooks/Operacoes.ipynb
Introdução (23 não lidos) - katiav... How to crochet dress ... Eu tricotico ...
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
Regressão logística
In [90]: parametros = {'tol': [0.0001, 0.00001, 0.000001],
                      'C': [1.0, 1.5, 2.0],
                      'solver': ['lbfgs', 'sag', 'saga']}
In [91]: grid_search = GridSearchCV(estimator=LogisticRegression(), param_grid=parametros)
grid_search.fit(X_OperacoesTesouroDireto, y_OperacoesTesouroDireto)
melhores_parametros = grid_search.best_params_
melhor_resultado = grid_search.best_score_
print(melhores_parametros)
print(melhor_resultado)
C:\Users\55219\anaconda3\lib\site-packages\sklearn\linear_model\_sag.py:329: ConvergenceWarning:
The max_iter was reached which means the coef_ did not converge
C:\Users\55219\anaconda3\lib\site-packages\sklearn\linear_model\_sag.py:329: ConvergenceWarning:
The max_iter was reached which means the coef_ did not converge
C:\Users\55219\anaconda3\lib\site-packages\sklearn\linear_model\_sag.py:329: ConvergenceWarning:
The max_iter was reached which means the coef_ did not converge
{'C': 1.0, 'solver': 'sag', 'tol': 0.0001}
0.44617710814881056
C:\Users\55219\anaconda3\lib\site-packages\sklearn\linear_model\_sag.py:329: ConvergenceWarning:
```

O quinto algoritmo a ser avaliado foi SVM, figura 37.

Como foram muitos parâmetros e a execução foi muito demorada, dividimos em várias execuções e, no final, pegamos o melhor resultado para ser comparado. O código ficou muito grande e para não nos estendermos retiramos o melhor resultado que foi: {'C': 2.0, 'kernel': 'sigmoid', 'tol': 0.001}

Figura 37 - Melhores parâmetros e resultados do SVM.



The screenshot shows a Jupyter Notebook window with the title 'Operacoes - Jupyter Notebook'. The browser address bar indicates the notebook is running on 'localhost:8888/notebooks/Operacoes.ipynb'. The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running cells, and code execution. The code is written in Python 3. The notebook contains several code cells. The first cell, labeled 'In [92]:', defines a dictionary of parameters for GridSearchCV:

```
##1
parametros = {'tol': [0.001],
              'C': [1.0],
              'kernel': ['rbf']}
```

 The second cell, labeled 'In [93]:', performs a GridSearchCV search using SVC as the estimator. It prints the best parameters and the best score:

```
grid_search = GridSearchCV(estimator=SVC(), param_grid=parametros)
grid_search.fit(X_OperacoesTesouroDireto, y_OperacoesTesouroDireto)
melhores_parametros = grid_search.best_params_
melhor_resultado = grid_search.best_score_
print(melhores_parametros)
print(melhor_resultado)

{'C': 1.0, 'kernel': 'rbf', 'tol': 0.001}
0.1526611272857389
```

 The third cell, labeled 'In [94]:', defines a new set of parameters for GridSearchCV:

```
parametros = {'tol': [0.001],
              'C': [1.0],
              'kernel': ['linear']}
```

 The fourth cell, labeled 'In []:', performs another GridSearchCV search using the 'linear' kernel:

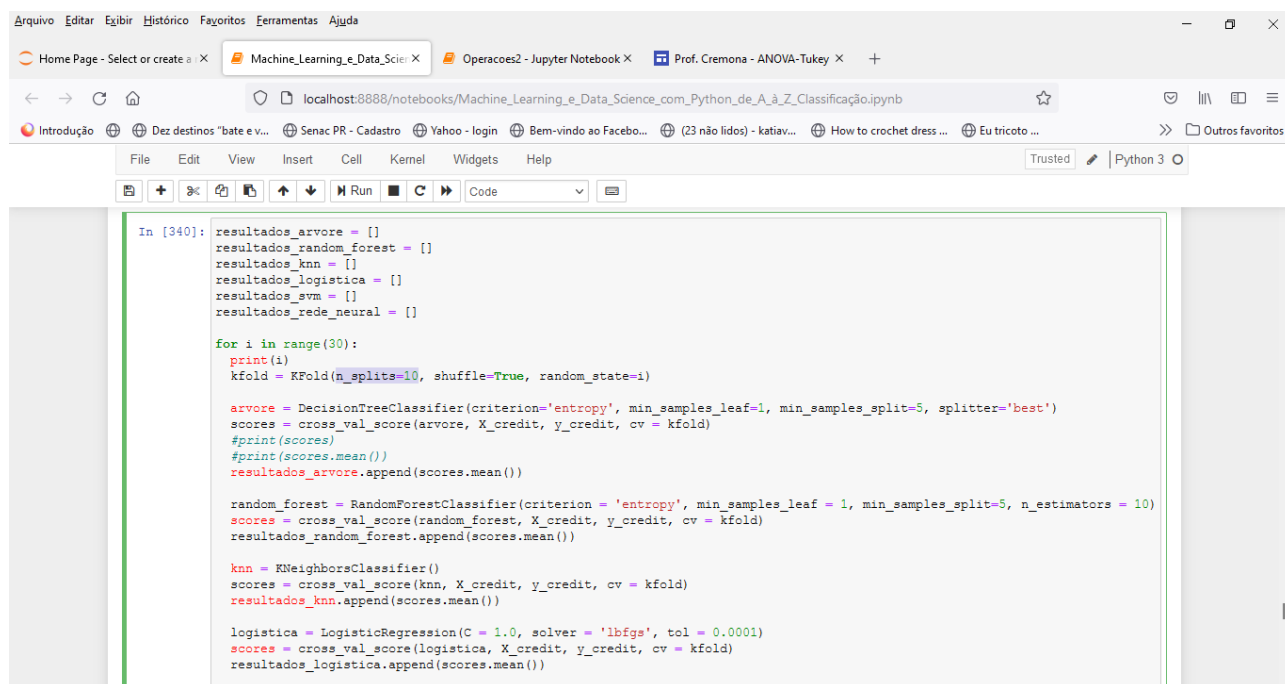
```
grid_search = GridSearchCV(estimator=SVC(), param_grid=parametros)
grid_search.fit(X_OperacoesTesouroDireto, y_OperacoesTesouroDireto)
melhores_parametros = grid_search.best_params_
melhor_resultado = grid_search.best_score_
```

Os parâmetros dos melhores resultados de cada algoritmo de acordo com a melhor acurácia, foram passados para o código de Validação Cruzada.

Foi utilizado o método Kfold e foram feitos `n_splits=10`, isso significa que a cada rodada determinada pelo FOR será feito 10 testes. Como fizemos um total de 30 rodadas, será 10 vezes 30 no total de 300 testes.

Nesse estudo, estamos considerando somente a acurácia. Poderia ser criada para a avaliação do precision e do recall. Figura 38.

Figura 38 – Código da Validação Cruzada.



```
In [340]: resultados_arvore = []
resultados_random_forest = []
resultados_knn = []
resultados_logistica = []
resultados_svm = []
resultados_rede_neural = []

for i in range(30):
    print(i)
    kfold = KFold(n_splits=10, shuffle=True, random_state=1)

    arvore = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=1, min_samples_split=5, splitter='best')
    scores = cross_val_score(arvore, X_credito, y_credito, cv = kfold)
    #print(scores)
    #print(scores.mean())
    resultados_arvore.append(scores.mean())

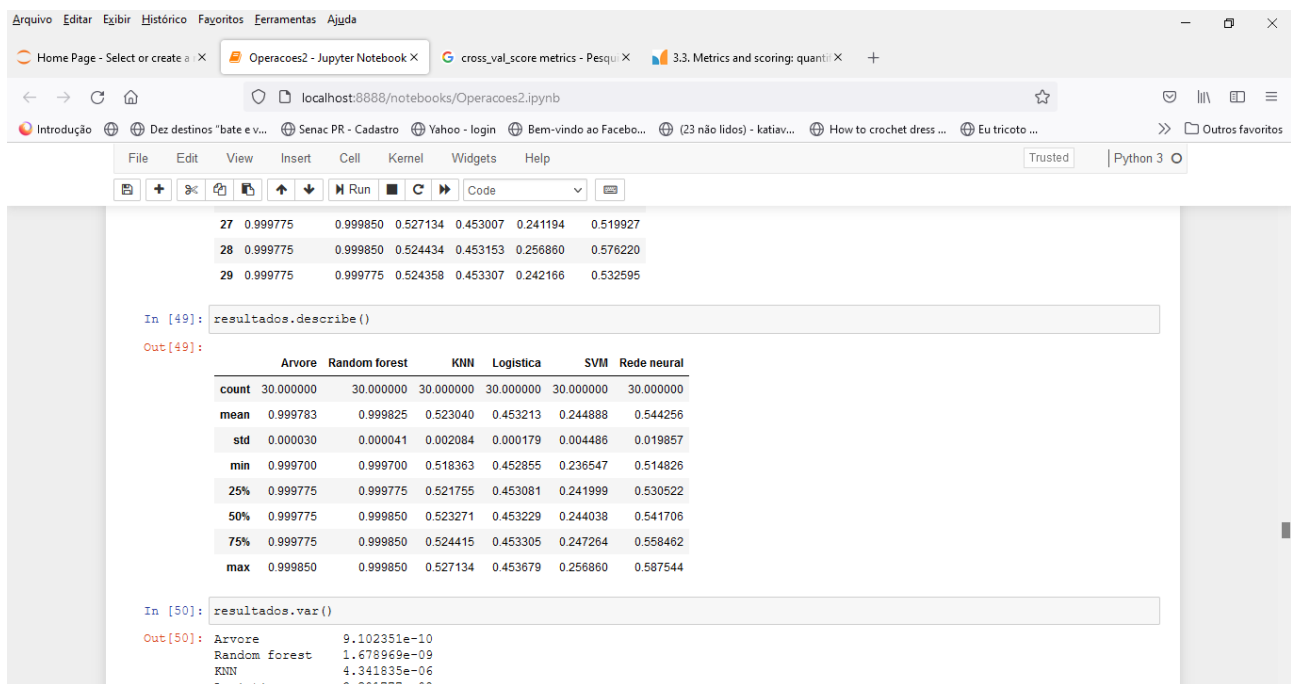
    random_forest = RandomForestClassifier(criterion = 'entropy', min_samples_leaf = 1, min_samples_split=5, n_estimators = 10)
    scores = cross_val_score(random_forest, X_credito, y_credito, cv = kfold)
    resultados_random_forest.append(scores.mean())

    knn = KNeighborsClassifier()
    scores = cross_val_score(knn, X_credito, y_credito, cv = kfold)
    resultados_knn.append(scores.mean())

    logistica = LogisticRegression(C = 1.0, solver = 'lbfgs', tol = 0.0001)
    scores = cross_val_score(logistica, X_credito, y_credito, cv = kfold)
    resultados_logistica.append(scores.mean())
```

Os melhores desempenhos foram os da Random Forest com 0.999825 e em segundo a Árvore de Decisão com 0.999783 de acurácia, dentre os modelos avaliados, as maiores médias e medianas, bem como os maiores valores mínimos e máximos, sendo Random Forest o de melhor resultado, confirmando a tendência atual de ser um dos mais poderosos algoritmos de aprendizado de máquina disponíveis atualmente.

Figura 39 – Resultado do desempenho dos classificadores.



Selecionado um dos algoritmos com melhor desempenho, qual seja, o Random Forest Classifier (Floresta Aleatória), passou-se ao treinamento (figura 40), o resultado obteve um escore de 1,00, ou seja, quase toda a base de testes foi classificada corretamente, conforme os “target_names”, em “Investidor” = 1, “Investidor” = 2, “Investidor” = 3 e “Investidor” = 4.

Figura 40 – Random Forest – Treinamento.

```

Random Forest - Melhor resultado de algoritmo - Treinamento

In [30]: X_OperacoesTesouroDireto_treinamento, X_OperacoesTesouroDireto_teste, y_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_teste = train_test_split(X_OperacoesTesouroDireto, y_OperacoesTesouroDireto, test_size=0.2, random_state=0)
classifier=RandomForestClassifier(n_estimators=1000, random_state=0)
classifier.fit(X_OperacoesTesouroDireto_treinamento, y_OperacoesTesouroDireto_treinamento)
classifier.score(X_OperacoesTesouroDireto_teste, y_OperacoesTesouroDireto_teste)

Out[30]: 1.0

In [48]: predicacao= classifier.predict(X_OperacoesTesouroDireto_teste)
print(predicacao)

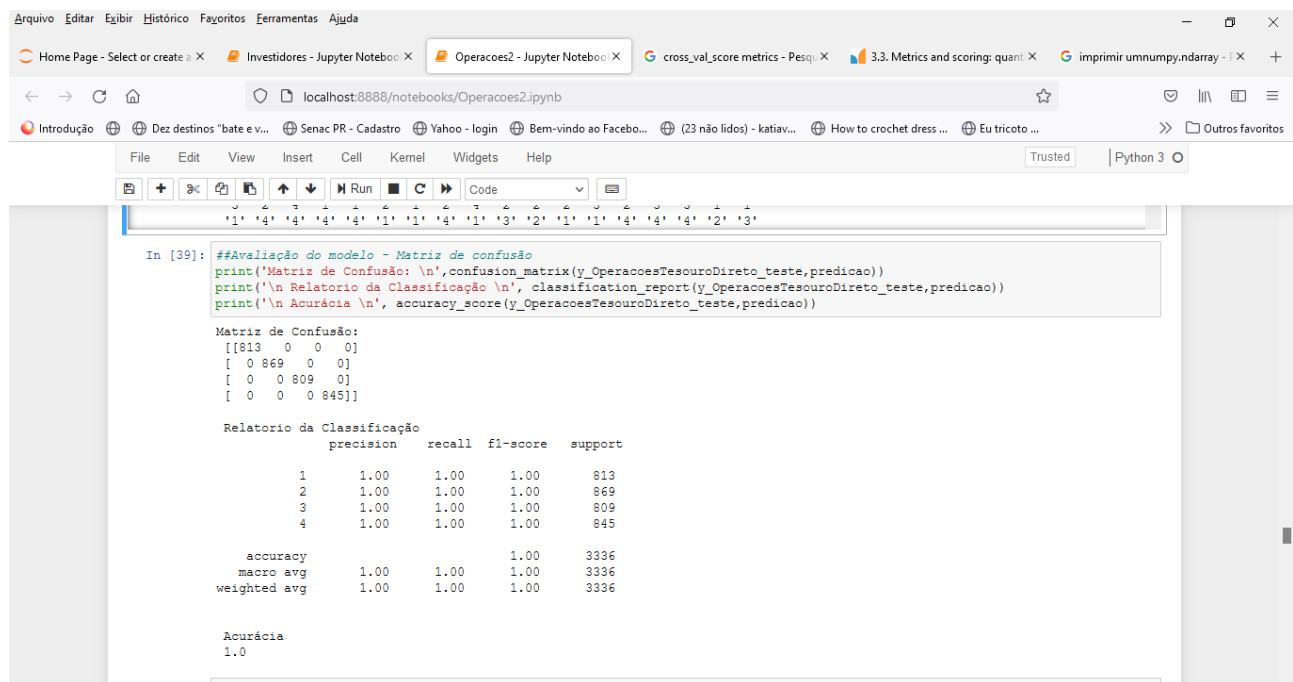
['2' '4' '1' '4' '2' '2' '3' '4' '4' '4' '2' '3' '1' '3' '4' '1' '4' '2'
 '4' '2' '2' '3' '4' '2' '4' '2' '1' '3' '1' '2' '4' '1' '2' '1' '1' '2'
 '3' '3' '4' '2' '1' '4' '4' '1' '3' '2' '2' '2' '1' '3' '2' '3' '4' '2'
 '2' '2' '2' '4' '3' '2' '4' '3' '3' '2' '3' '2' '4' '4' '1' '2' '2' '1'
 '4' '4' '1' '1' '1' '1' '2' '4' '4' '4' '3' '2' '4' '4' '2' '4' '2' '4'
 '4' '4' '1' '3' '2' '2' '2' '2' '2' '4' '3' '3' '4' '2' '2' '1' '4' '3'
 '4' '2' '4' '1' '2' '4' '3' '4' '2' '4' '4' '4' '3' '3' '1' '1' '3' '4'
 '4' '1' '3' '4' '4' '4' '1' '1' '3' '2' '1' '1' '2' '2' '4' '3' '3' '1'
 '3' '4' '4' '1' '3' '2' '3' '2' '1' '2' '4' '3' '2' '1' '2' '3' '2' '2'
 '2' '1' '3' '4' '4' '2' '3' '1' '1' '2' '4' '2' '2' '4' '2' '2' '3' '2'
 '4' '3' '4' '3' '1' '4' '4' '1' '1' '3' '3' '4' '3' '3' '3' '2' '2' '4'
 '3' '2' '4' '3' '4' '3' '4' '1' '1' '2' '4' '2' '1' '4' '4' '1' '3' '1'
 '1' '4' '4' '3' '1' '1' '4' '3' '1' '2' '3' '1' '2' '1' '4' '4' '4' '1'
 '4' '3' '1' '4' '1' '2' '2' '4' '3' '4' '1' '1' '3' '1' '3' '4' '4' '2'
 '4' '1' '4' '1' '4' '4' '1' '1' '3' '4' '4' '2' '4' '1' '4' '2' '1' '3'
 '2' '2' '1' '3' '4' '3' '2' '1' '3' '4' '3' '4' '3' '1' '1' '4' '1' '4'
  
```

Construída a matriz de confusão com o código da figura 41, que mostra as quantidades de classificações corretas e incorretas como verdadeiros positivos, falsos positivos, falsos verdadeiros e falsos negativos, sendo

1. verdadeiro positivo (*true positive* - TP): ocorre quando a classificação que se busca é prevista corretamente (no presente projeto é a quantidade de faixa de investidor previsto corretamente); 2. falso positivo (*false positive* - FP): ocorre quando a classificação que se busca é prevista incorretamente (no presente projeto faixa de investidor previsto incorretamente); 3. falso verdadeiro (*true negative* - TN): ocorre quando a classificação que não se está buscando é prevista corretamente e falso negativo (*false negative* - FN): ocorre quando a classificação que não se está buscando foi prevista incorretamente.

Nos resultados da classificação do modelo, nenhuma das ocorrências de classificação do Investidor foi classificada incorretamente, com “accuracy” de 1,00, “precision”, “recall” e “f1” da inconstitucionalidade igual a 1,00, “precision” e “f1” da constitucionalidade igual a 1 e “recall” igual a 1,00.

Figura 41 – Matriz de confusão do Random Forest.



O mesmo desempenho com o algoritmo Árvore de Decisão (figura 42).

Figura 42 – Árvore de Decisão mesmo desempenho do Random Forest – Treinamento.

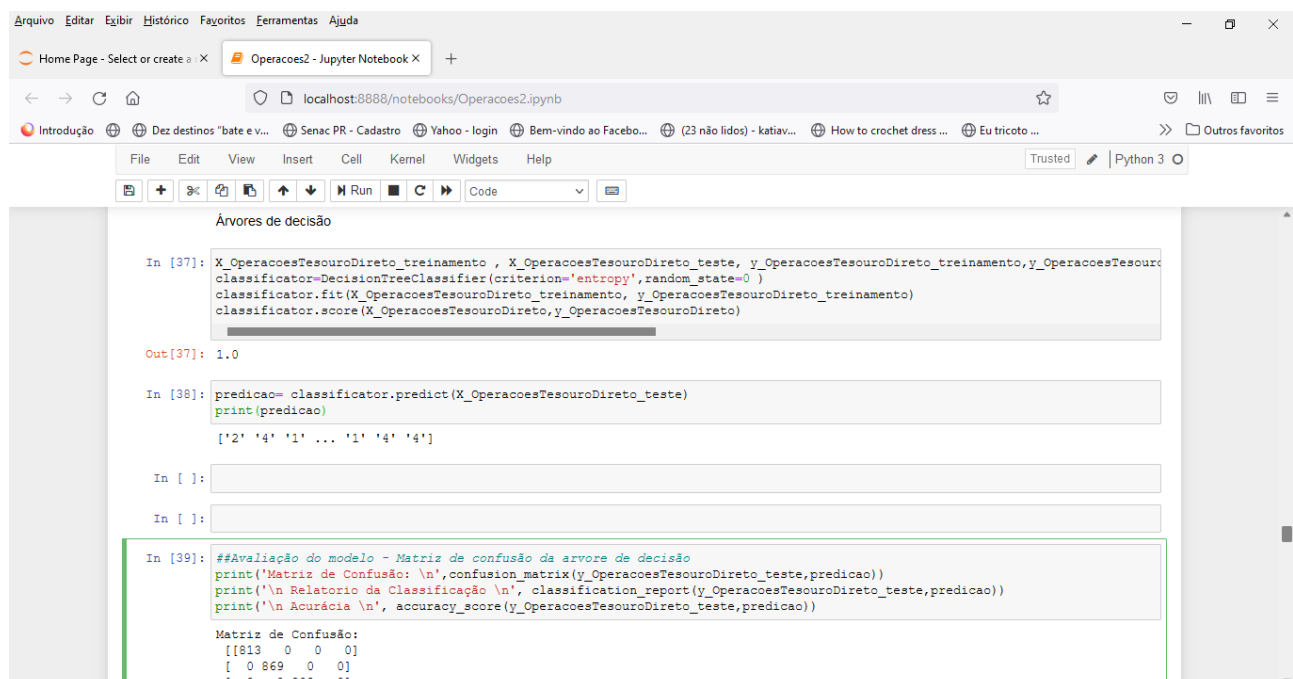
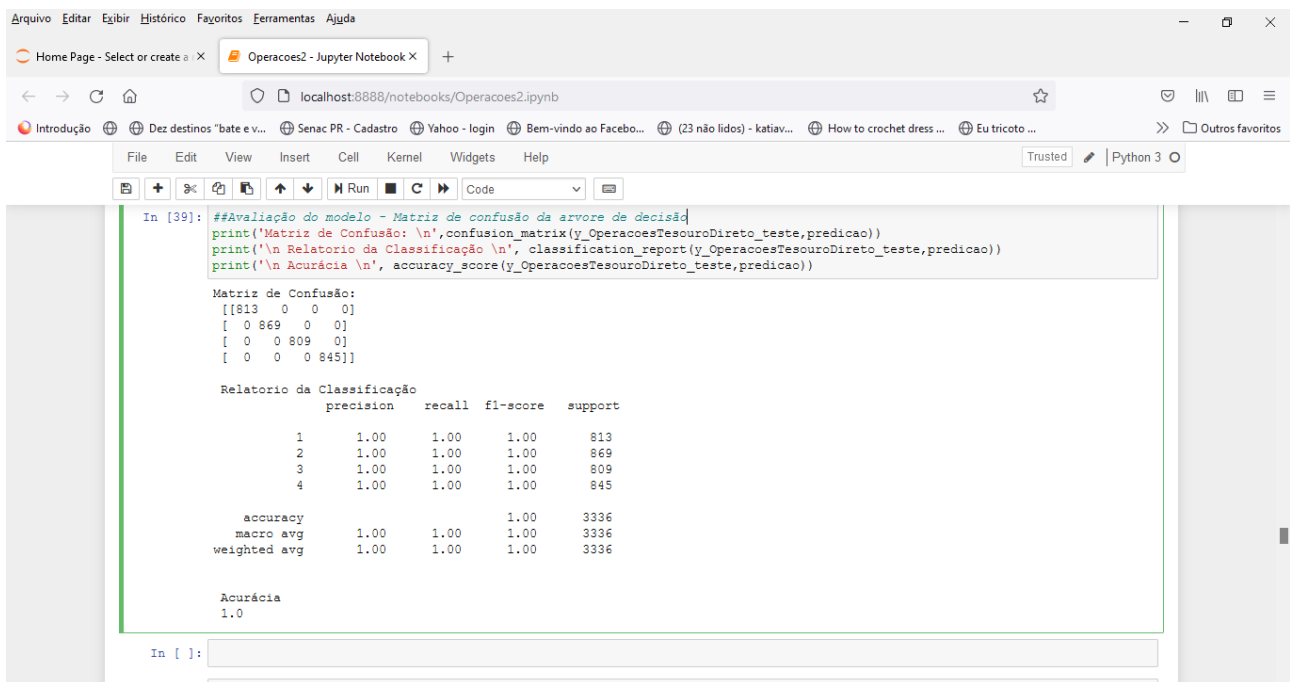


Figura 43 – Matriz de confusão da Árvore de Decisão.








6. Apresentação dos Resultados





Para a apresentação dos resultados obtidos foi utilizado o modelo de Canvas proposto por Dourard¹⁹. O modelo foi preenchido com os resultados observados pela execução dos algoritmos do assunto proposto.

Figura 44 – Modelo de Canvas para projetos de *machine learning* proposto por Dourard.

THE MACHINE LEARNING CANVAS Designed for: Puc Minas Designed by: Katia V

Pinheiro Date: Março de 2022 Iteration: .

PREDICTION TASK 	DECISIONS 	VALUE PROPOSITION 	DATA COLLECTION 	DATA SOURCES 
<p>O resultado do modelo é a predição do resultado das operações do Tesouro Nacional realizada no mês de fevereiro de 2021, possibilitando determinar o estudo do comportamento das operações realizadas. E O nível de valores de investimento nos títulos existentes.</p>	<p>A tarefa de <i>machine learning</i> (ML) é a classificação das operações, tendo como X (previsor) os tipos de Investidor como rótulo ou variável y (alvo). O <i>output</i> é a classificação dos Investidores em Baixo, médio baixo, médio alto e alto Investidor.</p>	<p>O modelo produzirá a classificação , classificação dos Investidores em Baixo, médio baixo, médio alto e alto Investidor. Predizendo a classificação.</p> <p>Valores gerados: - celeridade: o modelo tem uma velocidade de análise muito maior que a análise manual das ações. Podem ser analisadas e classificadas milhares operações em alguns minutos. - eficiência: o percentual de erro na classificação do modelo é muito inferior à análise humana. - redução de custos: alocação da mão de obra técnica e especializada em outras atividades, mais complexas. - otimização da recuperação de créditos tributários: a análise extremamente ágil e em massa das operações com o modelo possibilitará um estudo muito maior para as áreas interessadas.</p>	<p>As fontes de dados serão os dados de uma planilha CSV com os Tipo de Título do Tesouro Direto, Data da Operação, Quantidade, Datas de Vencimentos e Valores do Título e os Valores das Operações e a data das Operações. No caso só foram contempladas as operações de Venda do Tesouro para os Investidores extraídos da web.</p>	<p>O arquivo foi extraído do site do governo - dados.gov.br - de dados do Tesouro Nacional , via <i>web scraping</i> com código Python, utilizando as libs “BeautifulSoup”, “wget” e “request”. Os reduzindo a dimensionalidade. Foi decidido não tratar outliers,</p> <p>Os dados coletados foram muito concisos e não tivemos nenhum dados que tiveram de ser processados com técnicas de NLP, nem colinearidades, inconsistências e dados faltantes.</p>

<p>IMPACT</p> <p></p> <p>SIMULATION</p> <p>As técnicas de validação cruzada poderão ser utilizadas para avaliação do modelo, bem como as métricas de desvio-padrão, coeficiente de variação, valor mínimo e valor máximo dos parâmetros da “accuracy”.</p>	<p>MAKING</p> <p></p> <p>PREDICTIONS</p> <p>As predições serão realizadas após o processamento dos dados e de forma muito rápida.</p>		<p>BUILDING MODELS</p> <p></p> <p>Serão testados sete algoritmos de ML:</p> <p>1. Logistic Regression (RL): Regressão Logística, 2. K-Nearest Neighbors (K-NN): K-Vizinhos mais próximos, 3. Decision Tree Classifier (CART): Árvore de Decisão, 4 - Support Vector Machines (SVM): Máquinas de Vetores de Suporte e 5. Random Forest Classifier (RFC): Floresta Aleatória. O modelo será construído com o algoritmo que tiver o melhor desempenho na classificação.</p>	<p></p> <p>FEATURES</p> <p>Os dados brutos de entrada disponíveis serão: Código do Investidor, Canal da Operação, Data da Operação, Tipo da Operação, Título, Vencimento e valor do Título e Tipo da Operação extraídos do <i>site</i> do dados.gov.br.</p>
	<p>MONITORING</p> <p>A avaliação e monitoramento do modelo poderá ser realizada através da classificação de novas operações financeiras, medindo a acurácia das classes, bem como pela medição do tempo gasto para classificação de novos <i>datasets</i> pelo modelo. Por limitações de tempo, essa etapa não será realizada no presente projeto.</p>			

7. Links

<https://github.com/KatiaVP/PUCMINAS.git>

Link do vídeo:

<https://www.youtube.com/watch?v=n0NFEYRkYx0>

8. Referências

1 - Escovedo, Tatiana (2020-02-27T22:58:59). Introdução a Data Science . Casa do Código. Edição do Kindle.

2 - <https://its.unl.edu/bestpractices/remember-5-ws>

3 - <https://www.python.org/doc/essays/blurb/>

4 - <https://learnpython.com/blog/jupyter-notebook-python-ide-installation-tips/>

5 - <https://dados.gov.br/dataset/operacoes-do-tesouro-direto>

6 - Escovedo, Tatiana (2020-02-27T22:58:59). Introdução a Data Science . Casa do Código. Edição do Kindle.

7 - <https://github.com/ydataai/pandas-profiling>

8 - Géron, Aurélien. Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow (p. 4). Edição do Kindle.

9 - Fonte: <https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf>

10 - KOTSIANTIS, Sotiris B.; ZAHARAKIS, Ioannis D.; PINTELAS, Panayiotis E. Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, v. 26, n. 3, p.159-190, 2006.

11 - Fonte: <https://scikit-learn.org/stable/modules/tree.html>

12 - Géron, Aurélien. Mãos à Obra: Aprendizado de Máquina com Scikit-Learn & TensorFlow (p. 185). Edição do Kindle

13 - adaptado de <https://www.kaggle.com/getting-started/176257>

14 -- KOTSIANTIS, Sotiris B.; ZAHARAKIS, Ioannis D.; PINTELAS, Panayiotis E. Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, v. 26, n. 3, p.159-190, 2006.

15 - Adaptado da fonte <https://rapidminer.com/blog/k-nearest-neighbors-laziest-machine-learning-technique/>

16 - KOTSIANTIS, Sotiris B.; ZAHARAKIS, Ioannis D.; PINTELAS, Panayiotis E. Machine learning: a review of classification and combining techniques. Artificial Intelligence Review, v. 26, n. 3, p.159-190, 2006.

17 - <https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-supervised-learning>

18 - <https://sites.icmc.usp.br/andre/research/neural/>

19 – Adaptado da fonte https://sciencing-com.translate.goog/what-is-the-tukey-hsd-test-12751748.html?_x_tr_sl=en&_x_tr_tl=pt&_x_tr_hl=pt-BR&_x_tr_pto=sc

19 - <https://www.ownml.co/machine-learning-canvas>