



Instituto Politécnico
de Viana do Castelo

Licenciatura em Engenharia Informática

DATA SCIENCE

~ Data Exploration/Data Statistics~

Usando JUPYTER

Jorge Ribeiro

• jribeiro@estg.ipvc.pt

Conteúdos

- 
- 1. Introdução
 - 1.1. Objetivos
 - 2. Linguagem de Programação Python
 - 3. Jupyter Notebook

4. Análise de dados

- 4.1. Análise de Estatísticas Simples
- 4.2. Análise de Distribuições Estatísticas
- 4.3. Análise da Correlação e Intervalo de Confiança

5. Aprendizagem Supervisionada:

Criação de Modelo classificador/Previsão:

- Árvores de Decisão;
- Regressão Linear
- Redes Neuronais Artificiais

6. Aprendizagem Não supervisionada:

Clustering – K-Means

7. Bibliografia e Referências Web

■ 1.1. Objetivos

Assim, os principais objetivos deste trabalho, são:

1. Análise estatística simples de atributos do conjunto de dados: *Insol.csv*
2. Utilização da linguagem de programação Python em conjugação com a ferramenta Jupyter Notebook;
3. Exploração da biblioteca *Scipy*, para a análise das distribuições estatísticas, bem como a correlação entre atributos;

■ 1.1. Objetivos

Assim, os principais objetivos deste trabalho, são:

1. Análise estatística simples de atributos do conjunto de dados: *Insol.csv*

- número de elementos da amostra
- Mínimo (min) - valor menor da amostra
- Máximo (max) - valor maior da amostra
- Moda - valor mais comum da amostra
- Mediana - valor central da amostra
- Média - valor esperado de cada atributo
- 1º/3º Quartil - 25% dos valores da amostra estão abaixo/acima deste valor
- Desvio padrão
- Analisar a Distribuição Normal
- Analisar Distribuição Binomial
- Poisson - deveras utilizada em simulações de tempos
- T-student - utilizada quando não se sabe qual o valor do desvio padrão da população original
- Histogramas
- HeatMap
- **Scatterplot Matrix**

■ 2. Linguagem de Programação Python

Python é uma linguagem de programação, muito flexível e de alto nível. A sua utilização na análise de dados é uma mais valia, visto que esta linguagem é suportada por um elevado número de bibliotecas e frameworks, cujo o objetivo é fornecer um alargado número de funções e procedimentos matemáticos. Neste trabalho as bibliotecas utilizadas incluem:

-  **pandas** → Utilizada para a leitura e manipulação do conjunto de dados CSV.
-  **NumPy** → Utilizada para a manipulação e geração de matrizes de dados.
-  **SciPy** → Utilizada para a geração de distribuições estatísticas e métodos de determinação de correlação entre atributos.
-  **matplotlib** → Permite a geração de gráficos e a sua personalização.

3. Jupyter Notebook

<https://towardsdatascience.com/beginners-guide-to-jupyter-notebook-8bb85b85085>

DATA SCIENCE

Beginner's Guide to Jupyter Notebook

From the setup to the descriptive statistics



Shinichi Okada Dec 18, 2019 · 9 min read ★



Table of Contents

Introduction

1. [Installing Anaconda](#)
2. [Jupyter Notebook help and shortcuts](#)
3. [Python libraries](#)
4. [Finding data](#)
5. [Importing data](#)
6. [Data attributes](#)
7. [Missing Value](#)
8. [Dealing with NaN values](#)
9. [How to deal categorical data](#)
10. [Descriptive statistics](#)
11. [Basic plotting](#)
12. [Matplotlib](#)
13. [Seaborn](#)
14. [What's next?](#)

3. Jupyter Notebook

Installing Anaconda

We are going to install Anaconda. Anaconda is the standard platform for Python data science. Anaconda will install all necessary packages for our purpose.

From <https://www.anaconda.com/distribution/>, download a 3.7 version and install it.

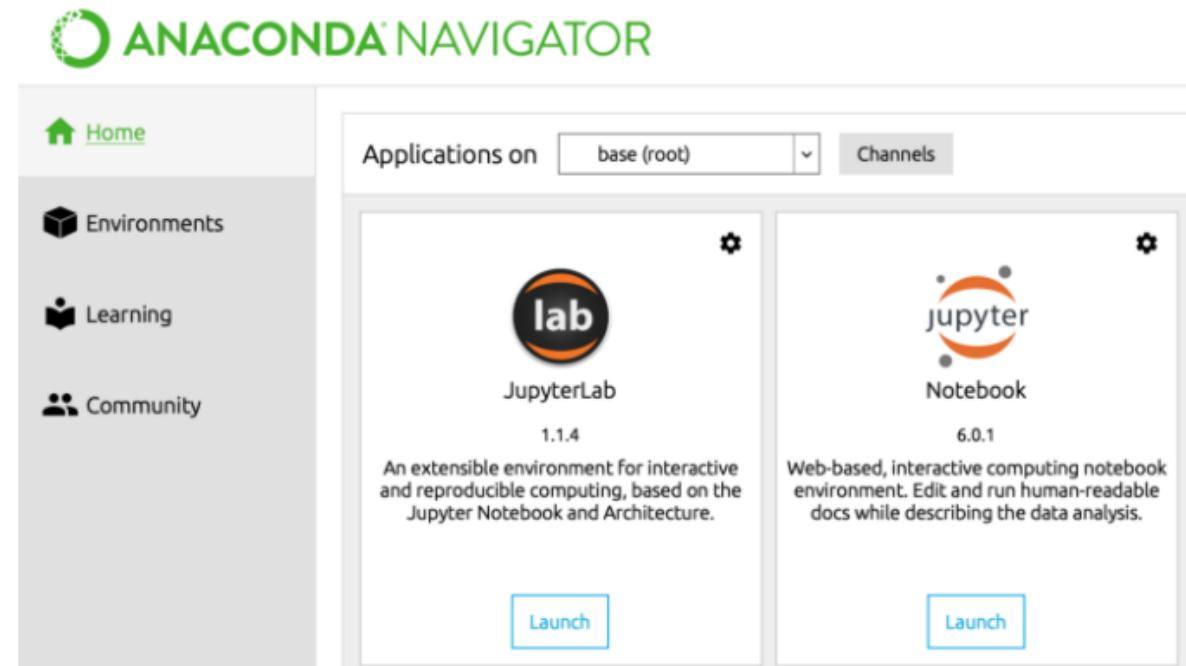
After installing the Anaconda, start Anaconda-Navigator.

Python 3.7 version

Download

64-Bit Graphical Installer (654 MB)

64-Bit Command Line Installer (424 MB)



3. Jupyter Notebook

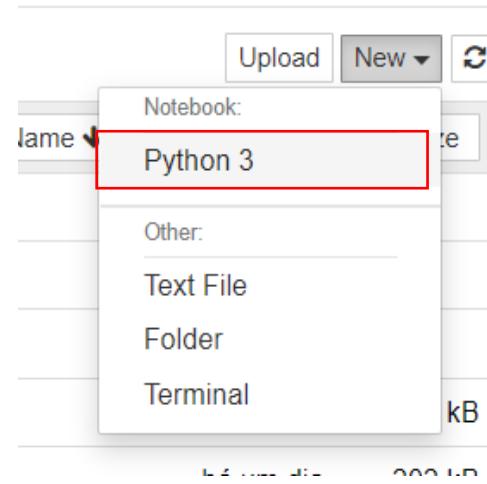
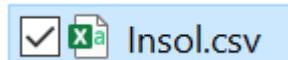


Figura 1 – Criação de um novo documento Python3

The screenshot shows the Jupyter Notebook interface with the title 'jupyter'. It has tabs for 'Files', 'Running', and 'Clusters'. Below the tabs, it says 'Select items to perform actions on them.' A sidebar shows a file tree with '0' items. The main area lists files under 'Jupyter Notebook Projects': 'BD', 'Datasets', 'Exploracao_Statsmodels.ipynb', and 'Exploração Python para Análise de Dados.ipynb'. The last two files are highlighted with a red box.

Figura 2 – Documento Python3 criado

3. Jupyter Notebook



Screenshot of a Microsoft Excel spreadsheet titled "Insol.csv". The spreadsheet contains data from row 1 to 25. The columns are labeled A through S. Column A is "Distrito", B is "NACE1", C is "NACE2", D is "Numero_E", E is "Num_ano", F is "Variacao_1", G is "Variacao_2", H is "Variacao_3", I is "Variacao_4", J is "Variacao_5", K is "Variacao_6", L is "Variacao_7", M is "Fundo_de REPV", N is "Prazo_me", O is "Liquidez_f", P is "Liquidez_r", Q is "Solvabilid", R is "Autonomia", and S is "Envolvimento". The data includes various numerical values and some zeros.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	
1	Distrito	NACE1	NACE2	Numero_E	Num_ano	Variacao_1	Variacao_2	Variacao_3	Variacao_4	Variacao_5	Variacao_6	Fundo_de REPV	Prazo_me	Liquidez_f	Liquidez_r	Solvabilid	Autonomia	Envolvimento	
2	1	14	141	225	9	-0,05512	-12,1834	-0,08162	-0,35388	-0,46474	-0,42497	0,466459	-753912	11,46809	53,17243	0,4873	0,343348	0,420554	0,296049
3	1	13	139	28	10	-0,16378	-7,68548	-0,12059	0,002009	-0,12144	-0,23107	-0,10421	1557403	10,57665	132,829	2,813255	2,393234	1,962532	0,662451
4	5	14	141	80	9	-0,19374	-1,48674	-0,11889	0,055864	-0,12365	0,113843	-0,07489	1675145	8,152964	31,24659	1,973253	0,871358	1,198545	0,545154
5	5	14	141	175	8	-0,15679	-1,36623	-0,12281	-0,10939	-0,12707	0,079843	-0,10614	1529773	4,928096	46,36006	1,904464	2,999926	0,749995	0,
6	1	13	139	140	10	-0,05991	-0,74267	-0,01961	0,002116	-0,04153	-0,09325	0,057831	9857170	7,650846	127,4404	6,690793	5,976631	7,780287	0,886108
7	5	14	141	229	10	0,020864	-0,61464	0,132667	0,494631	0,188162	0,088135	-0,01607	58434	1,608043	116,5806	1,013334	0,360715	0,326416	0,246089
8	1	13	139	65	10	-0,08596	-0,60063	-0,00199	0,01484	0,013302	0,001617	-0,08576	3996879	5,18485	170,8351	3,944611	2,982618	3,594406	0,782344
9	1	13	139	34	10	-0,07863	-0,59062	-0,0795	0,02465	-0,05761	0,074862	-0,29814	420195	15,49001	118,8717	1,157321	0,983513	0,243605	0,195886
10	5	14	141	86	10	-0,16217	-0,52054	-0,00454	0,068854	-0,0172	-0,2356	0,156905	2082650	8,152641	57,4209	2,648355	2,096994	1,892837	0,654319
11	1	14	141	70	9	-0,07562	-0,50851	-0,08813	0,002347	-0,10053	-0,11414	-0,03824	542115	9,160035	235,4504	1,262342	1,08453	0,597614	0,374067
12	1	14	141	31	7	-0,16702	-0,43561	0,004583	0,330842	-0,04857	-0,40228	0,530072	310650	35,05098	80,32141	1,105851	1,0176	0,285748	0,222243
13	1	13	139	59	10	-0,08924	-0,38221	0,054113	0,002292	0,072321	0,051509	-0,00772	401087	3,139482	125,5802	1,10921	0,595283	0,411424	0,291496
14	1	13	139	29	10	-0,22866	-0,36234	-0,00378	0,027406	0,015427	-0,04664	-0,19952	2028368	19,57706	104,7851	4,811318	4,48375	4,18357	0,807083
15	1	13	133	104	10	-0,04767	-0,34273	-0,18407	-0,34645	-0,20245	-0,26117	-0,1582	-345492	21,21079	79,38747	0,845176	0,742395	0,478967	0,323852
16	1	14	143	43	5	-0,05845	-0,33675	-0,01253	0,012287	0,003292	0,43192	-0,24755	4728184	91,67727	57,80023	3,363112	3,264324	2,532935	0,716949
17	1	13	139	17	10	0,009048	-0,2898	0,078333	0,065912	0,098692	0,062071	-0,11671	1005063	2,946807	110,2962	1,709869	0,835486	0,85336	0,460439
18	1	14	143	503	10	-0,23097	-0,27235	-0,12444	-0,38955	0,603331	0,020348	-0,56104	-497682	4,37861	205,4851	0,964122	0,692909	0,404117	0,287809
19	5	13	139	208	10	-0,09663	-0,21654	-0,03393	-0,00739	-0,01878	-0,04747	-0,11239	14111107	3,056496	140,1219	2,53205	1,390462	1,974241	0,66378
20	1	14	141	205	10	-0,00573	-0,16277	0,092729	0,124865	0,107842	-0,29578	-0,11115	4275123	23,44662	199,9215	5,9403	5,645787	5,293599	0,841108
21	3	14	141	87	4	-0,26404	-0,16273	-0,27452	0,505864	-0,34478	-0,11149	-0,06233	620310	33,19089	55,22674	2,560583	2,244351	2,773944	0,735025
22	1	14	141	48	10	0,040989	-0,12098	-0,09213	0,032559	-0,09288	0,267466	-0,08978	1483450	3,749221	76,46889	3,441499	1,986366	3,556124	0,780515
23	1	13	131	32	10	0,012374	-0,09437	-0,1088	0,003121	-0,12099	-0,21174	-0,09254	5485358	0,792045	93,71502	32,44409	8,840153	56,54909	0,982623
24	1	13	132	262	6	-0,13858	-0,05692	-0,02474	0,02456	-0,01751	0,201115	-0,0579	8869976	1,771673	97,68788	5,169147	2,446871	5,249259	0,839981
25	5	13	139	211	10	-0,03193	-0,03823	-0,04825	0,006409	-0,04053	-0,16357	-0,07162	15572068	4,58929	91,84438	6,888249	5,345567	8,089329	0,889981

■ 3. Jupyter Notebook

  Insol.csv

O conjunto de dados tem uma taxa de empresas em insolvência (variável Insolvencia) 19.4%.

Relativamente às variáveis categóricas:

- **Distrito** – 1 – Braga; 2 – Bragança; 3 – Viana do Castelo; 4 – Vila Real e 5 – Porto
- **NACE1** – 13 – Fabricação de textéis e 14 – Indústria do vestuário
- **NACE2** – 131 - Preparação e fiação de fibras textêis, 132 - Tecelagem de textêis, 133 - Acabamento de textêis, 139 - Fabricação de outros textêis, 141 - Confeção de artigos de vestuário excepto artigos de peles com pelo, 142 - Fabricação de artigos de pele com pelo e 143 - Fabricação de artigos de malha
- **Certificacao_legal_contas** – 0 – Não e 1 – Sim
- **Insolvencia** – 0 – Não e 1 – Sim

As restantes variáveis são quantitativas e o nome traduz o significado da variável.

3. Jupyter Notebook

The screenshot shows the Kaggle interface for the 'insolv1' dataset. The left sidebar includes links for Home, Competitions, Datasets, Code, Discussions, Courses, and More. Under 'RECENTLY VIEWED', items like 'insolv1', 'insolvencias', and 'Notebook ArvoresD...' are listed. The main page displays the 'Insol.csv' file (885.62 kB). The file preview shows the first few rows of data:

# Distrito	# NACE1	# NACE2	# Numero_Empregados	# Num_anos_disponivel	# Variacao
0 total values	0 total values	0 total values	0 total values	0 total values	0 total values
1	14	141	225	9	-0,0551%
1	13	139	28	10	-0,1637%
5	14	141	80	9	-0,1937%
5	14	141	175	8	-0,1567%
1	13	139	140	10	-0,0599%
5	14	141	229	10	0,02086%

On the right, the 'Data Explorer' panel shows the file 'Insol.csv' with version 1 (885.62 kB).

3. Jupyter Notebook

The screenshot shows a Jupyter Notebook interface on a web browser. The title bar reads "kaggle.com/code/jribeiro2018/2022-insolv-viewdata/edit". The notebook is titled "2022-Insolv-ViewData". The left sidebar contains various icons for file operations like new, open, save, and run. The main area displays Python code for data analysis:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as st

# Configuring matplotlib to display and save images with a white background
plt.rcParams['figure.facecolor'] = 'white'

# Reading the data from the csv file
raw_data = pd.read_csv('../input/insolv1/Insol.csv', decimal=',', sep=";")

np.set_printoptions(suppress=True)

# Defining some regular bins
# Bin for variation coefficient
bins_var = [-1, -0.8, -0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6, 0.8, 1]

# Bin for values between 0 and 1
bins_from_zero = [0, 0.2, 0.4, 0.6, 0.8, 1]
```

The right sidebar shows the "Input" section with a folder named "insolv1" containing "Insol.csv", and the "Output" section showing a total size of "44.1MB / 19.6GB" with a folder path "/kaggle/working". There are also sections for "Settings", "Schedule a notebook run", and "Code Help" with a search bar for "Find Code Help".

3. Jupyter Notebook

Informação dos Dados Importados

[3]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2738 entries, 0 to 2737
Data columns (total 33 columns):
 #   Column           Non-Null Count  Dtype  
 ---  -- 
 0   Distrito          2738 non-null    int64  
 1   NACE1             2738 non-null    int64  
 2   NACE2             2738 non-null    int64  
 3   Numero_Empregados 2738 non-null    int64  
 4   Num_anos_disponiveis_base 2738 non-null    int64  
 5   Variacao_das_vendas 2738 non-null    object  
 6   Variacao_resultado_operacional 2738 non-null    object  
 7   Variacao_do_ativo 2738 non-null    object  
 8   Variacao_dos_capitais_proprios 2738 non-null    object  
 9   Variacao_do_ativo_circulante 2738 non-null    object  
 10  Variacao_das_existencias 2738 non-null    object  
 11  Variacao_do_imobilizado 2738 non-null    object  
 12  Fundo_de_maneio 2738 non-null    int64  
 13  REPV              2738 non-null    object  
 14  Prazo_medio_de_recebimento 2738 non-null    object  
 15  Liquidez_geral 2738 non-null    object  
 16  Liquidez_reduzida 2738 non-null    object  
 17  Solvabilidade 2738 non-null    object  
 18  Autonomia_financeira 2738 non-null    object  
 19  Endividamento 2738 non-null    object  
 20  Estrutura_financeira 2738 non-null    object  
 21  Passivo_de_curto_prazo_a_dividir_passivo_total 2738 non-null    object  
 22  Custos_dos_encargos_financeiros_dividir_resultado_operacional 2738 non-null    object  
 23  Rendibilidade_operacional_vendas 2738 non-null    object  
 24  Rendibilidade_liquida_das_vendas 2738 non-null    object  
 25  Rendibilidade_do_ativo 2738 non-null    object  
 26  Rendibilidade_capitais_proprios 2738 non-null    object  
 27  Passivo_curto_prazo_dividir_vendas 2738 non-null    object  
 28  Peso_das_amortizacoes_dividir_vendas 2738 non-null    object  
 29  Peso_encargos_financeiros_dividir_vendas 2738 non-null    object  
 30  Produtividade_por_trabalhador 2738 non-null    object  
 31  Certificacao_legal_contas 2738 non-null    int64  
 32  Insolvencia 2738 non-null    int64  
dtypes: int64(8), object(25)
memory usage: 706.0+ KB
```

3. Jupyter Notebook

Conversão das colunas do Tipo ‘Object’ para Float para permitir realizar Cálculos Estatísticos

```
data["Variacao_das_vendas"] = data["Variacao_das_vendas"].str.replace(",","").astype(float)
data["Variacao_resultado_operacional"] = data["Variacao_resultado_operacional"].str.replace(",","").astype(float)
data["Variacao_do_ativo"] = data["Variacao_do_ativo"].str.replace(",","").astype(float)
data["Variacao_dos_capitais_proprios"] = data["Variacao_dos_capitais_proprios"].str.replace(",","").astype(float)
data["Variacao_das_existencias"] = data["Variacao_das_existencias"].str.replace(",","").astype(float)
data["Variacao_do_imobilizado"] = data["Variacao_do_imobilizado"].str.replace(",","").astype(float)
data["REPV"] = data["REPV"].str.replace(",","").astype(float)
data["Prazo_medio_de_recebimento"] = data["Prazo_medio_de_recebimento"].str.replace(",","").astype(float)
data["Liquidez_geral"] = data["Liquidez_geral"].str.replace(",","").astype(float)
data["Liquidez_reduzida"] = data["Liquidez_reduzida"].str.replace(",","").astype(float)
data["Solvabilidade"] = data["Solvabilidade"].str.replace(",","").astype(float)
data["Autonomia_financeira"] = data["Autonomia_financeira"].str.replace(",","").astype(float)
data["Endividamento"] = data["Endividamento"].str.replace(",","").astype(float)
data["Estrutura_financeira"] = data["Estrutura_financeira"].str.replace(",","").astype(float)
data["Passivo_de_curto_prazo_a_dividir_passivo_total"] = data["Passivo_de_curto_prazo_a_dividir_passivo_total"].str.replace(",","").astype(float)
data["Custos_dos_encargos_financeiros_dividir_resultado_operacional"] = data["Custos_dos_encargos_financeiros_dividir_resultado_operacional"].str.replace(",","").astype(float)
data["Rendibilidade_operacional_vendas"] = data["Rendibilidade_operacional_vendas"].str.replace(",","").astype(float)
data["Rendibilidade_liquida_das_vendas"] = data["Rendibilidade_liquida_das_vendas"].str.replace(",","").astype(float)
data["Rendibilidade_do_ativo"] = data["Rendibilidade_do_ativo"].str.replace(",","").astype(float)
data["Rendibilidade_capitais_proprios"] = data["Rendibilidade_capitais_proprios"].str.replace(",","").astype(float)
data["Passivo_curto_prazo_dividir_vendas"] = data["Passivo_curto_prazo_dividir_vendas"].str.replace(",","").astype(float)
data["Peso_das_amortizacoes_dividir_vendas"] = data["Peso_das_amortizacoes_dividir_vendas"].str.replace(",","").astype(float)
data["Peso_encargos_financeiros_dividir_vendas"] = data["Peso_encargos_financeiros_dividir_vendas"].str.replace(",","").astype(float)
data["Produtividade_por_trabalhador"] = data["Produtividade_por_trabalhador"].str.replace(",","").astype(float)
data["Variacao_do_ativo_circulante"] = data["Variacao_do_ativo_circulante"].str.replace(",","").astype(float)
```

3. Jupyter Notebook

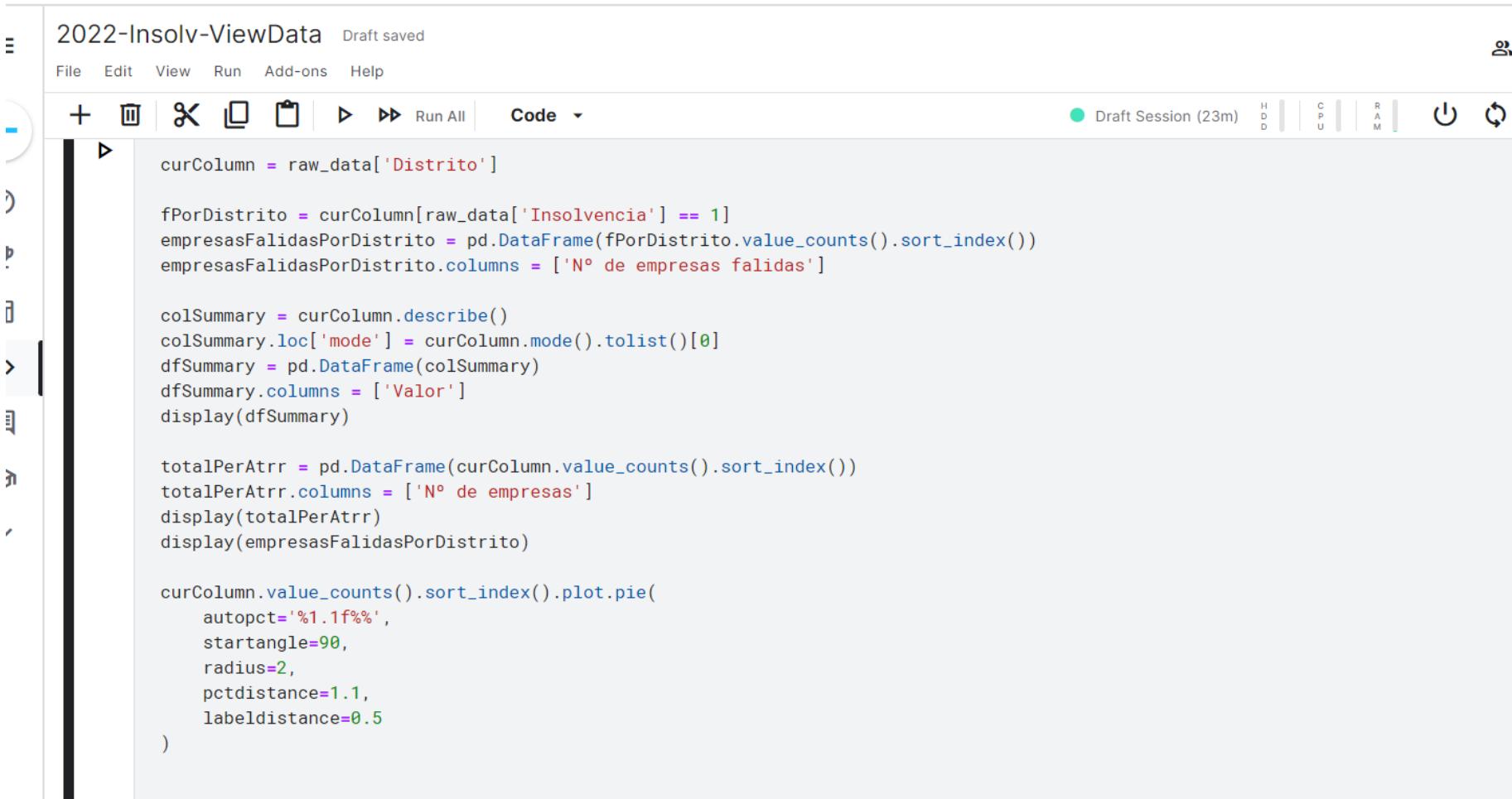
Conversão das colunas do Tipo ‘Object’ para Float para permitir realizar Cálculos Estatísticos

[5]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2738 entries, 0 to 2737
Data columns (total 33 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Distrito          2738 non-null    int64  
 1   NACE1             2738 non-null    int64  
 2   NACE2             2738 non-null    int64  
 3   Numero_Empregados 2738 non-null    int64  
 4   Num_anos_disponiveis_base 2738 non-null    int64  
 5   Variacao_das_vendas 2738 non-null    float64 
 6   Variacao_resultado_operacional 2738 non-null    float64 
 7   Variacao_do_ativo 2738 non-null    float64 
 8   Variacao_dos_capitais_proprios 2738 non-null    float64 
 9   Variacao_do_ativo_circulante 2738 non-null    float64 
 10  Variacao_das_existencias 2738 non-null    float64 
 11  Variacao_do_imobilizado 2738 non-null    float64 
 12  Fundo_de_maneio 2738 non-null    int64  
 13  REPV              2738 non-null    float64 
 14  Prazo_medio_de_recebimento 2738 non-null    float64 
 15  Liquidez_geral 2738 non-null    float64 
 16  Liquidez_reduzida 2738 non-null    float64 
 17  Solvabilidade 2738 non-null    float64 
 18  Autonomia_financeira 2738 non-null    float64 
 19  Endividamento 2738 non-null    float64 
 20  Estrutura_financeira 2738 non-null    float64 
 21  Passivo_de_curto_prazo_a_dividir_passivo_total 2738 non-null    float64 
 22  Custos_dos_encargos_financeiros_dividir_resultado_operacional 2738 non-null    float64 
 23  Rendibilidade_operacional_vendas 2738 non-null    float64 
 24  Rendibilidade_liquida_das_vendas 2738 non-null    float64 
 25  Rendibilidade_do_ativo 2738 non-null    float64 
 26  Rendibilidade_capitais_proprios 2738 non-null    float64 
 27  Passivo_curto_prazo_dividir_vendas 2738 non-null    float64 
 28  Peso_das_amortizacoes_dividir_vendas 2738 non-null    float64 
 29  Peso_encargos_financeiros_dividir_vendas 2738 non-null    float64 
 30  Produtividade_por_trabalhador 2738 non-null    float64 
 31  Certificacao_legal_contas 2738 non-null    int64  
 32  Insolvencia 2738 non-null    int64  
dtypes: float64(25), int64(8)
memory usage: 706.0 KB
```

3. Jupyter Notebook



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** 2022-Insolv-ViewData Draft saved
- Toolbar:** File, Edit, View, Run, Add-ons, Help, Code dropdown, Draft Session (23m), and system status indicators for HDD, CPU, and RAM.
- Code Cell:** Contains the following Python code:

```
curColumn = raw_data['Distrito']

fPorDistrito = curColumn[raw_data['Insolvencia'] == 1]
empresasFalidasPorDistrito = pd.DataFrame(fPorDistrito.value_counts().sort_index())
empresasFalidasPorDistrito.columns = ['Nº de empresas falidas']

colSummary = curColumn.describe()
colSummary.loc['mode'] = curColumn.mode().tolist()[0]
dfSummary = pd.DataFrame(colSummary)
dfSummary.columns = ['Valor']
display(dfSummary)

totalPerAttr = pd.DataFrame(curColumn.value_counts().sort_index())
totalPerAttr.columns = ['Nº de empresas']
display(totalPerAttr)
display(empresasFalidasPorDistrito)

curColumn.value_counts().sort_index().plot.pie(
    autopct='%1.1f%%',
    startangle=90,
    radius=2,
    pctdistance=1.1,
    labeldistance=0.5
)
```

3. Jupyter Notebook

n - número de elementos da amostra;

[16]:

```
# Número de elementos da amostra
data[data.columns[0]].count() # (slowest, but avoids counting NaN values in the first column)
```

[16... 2738

[17]:

```
# Outra forma
print(f'Número de linhas = {len(data)}')
print(f'Número de colunas = {len(data.columns)}')
```

Número de linhas = 2738
Número de colunas = 33

Mínimo (min) - valor menor da amostra;

Máximo (max) - valor maior da amostra;

[18]:

```
# Para encontrar o valor mais baixo vamos percorrer o nosso CSV

row_count = data.shape[0]
column_list = []
for i in range(row_count):
    item = data.at[i, data.columns.values[0]] # Primeira Coluna
    column_list.append(float(item))           #parse float e adicionar à lista

print(max(column_list))      #Print - Distrito Número Máximo
print(min(column_list))      #Print - Distrito Número Minimo
```

5.0
1.0

Moda - valor mais comum da amostra;

Mode

Mode is the number occurring most often in the dataset.

- It is only meaningful if we have many repeated values in our dataset
- If no value is repeated, there is **no mode**
- A dataset can have **one mode, multiple modes or no mode**.

```
9]:  
# Moda da coluna NACE2  
data['NACE2'].mode()
```

```
9... 0    141  
      dtype: int64
```

3. Jupyter Notebook

Média - valor esperado de cada atributo;

Mean

Mean (usually referred to Arithmetic Mean, also called Average) is calculated as sum of all numbers in the dataset and dividing by the total number of values

Arithmetic Mean

$$\text{Arithmetic mean} = \frac{\text{Sum of all numbers}}{\text{No. of values in the set}} \quad \text{or}$$

$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

Arithmetic mean of our data is calculated as,

```
mean = (35 + 15 + 97 + 17 + ...) / 315
```

```
]:
```

```
# Média do Número de Empregados
data['Número_Empregados'].mean()
```

```
... 29.096420745069395
```

Mediana - valor central da amostra (ordenar valores, se n é ímpar a é o valor do meio, se é par, é o valor médio das 2 observações centrais);

Median

Median is the middle value, when the data is sorted in ascending order. Half of the data points are smaller and half of data points are larger than the median.

.]:

```
# Mediana do Fundo_de_maneio
data['Fundo_de_maneio'].median()
```

L... 22087.5

3. Jupyter Notebook

1º/3º Quartil - 25% dos valores da amostra estão abaixo/acima deste valor;

Interquartile Range (IQR)

Interquartile range or **IQR** is the amount spread in middle 50% of the dataset or the distance between *first Quartile* (Q_1) and *third Quartile* (Q_3)

- First Quartile (Q_1) = Median of data points to left of the median in ordered list (25th percentile)
- Second Quartile (Q_2) = Median of data (50th percentile)
- Third Quartile (Q_3) = Median of data points to right of the median in ordered list (75th percentile)
- $IQR = Q_3 - Q_1$

2]:

```
# Primeiro Quartil, Mediana, 3 Quartil
data['Num_anos_disponiveis_base'].quantile([0.25,0.5,0.75])
```

:2...

```
0.25    4.0
0.50    4.0
0.75    7.0
Name: Num_anos_disponiveis_base, dtype: float64
```

3. Jupyter Notebook

Variância e Desvio padrão



```
# Passo 1: Calcular Média( $\mu$ )
numeroEmpregados = data['Numero_Empregados']
media = numeroEmpregados.mean()
print(f'Média = {media}')

# Passo 2: calculate numerator part - sum of ( $x - \text{Média}$ )
numeroEmpregados_numerator = sum([(x - media) ** 2 for x in numeroEmpregados])

# Passo 3: Calcular a variância
numeroEmpregados_variancia = numeroEmpregados_numerator / len(numeroEmpregados)
print(f'Variância = {numeroEmpregados_variancia}')

# Passo 4: Calcular o Desvio Padrão
numeroEmpregados_desvioPadrao = math.sqrt(numeroEmpregados_variancia)
print(f'Desvio Padrão = {numeroEmpregados_desvioPadrao}')
```

Média = 29.096420745069395
Variância = 3208.2434422656183
Desvio Padrão = 56.641358054566616

+ Code

+ Markdown

3. Jupyter Notebook

Variância e Desvio padrão

Variance and Standard deviation

Standard deviation and variance measures the spread of a dataset. If the data is spread out largely, standard deviation (and variance) is greater.

In other terms,

- if more data points are closer to the mean, standard deviation is less
- if the data points are further from the mean, standard deviation is more

Formula for variance for **population** is given as,

$$\text{Variance} = \sigma^2 = \frac{\sum_{i=1}^n (x_i - \mu)^2}{n}$$

where, μ is the **mean** of the dataset

Standard deviation is just the square root of variance

$$\text{Standard deviation} = \sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n}}$$

Note:

For **Sample**, we use $n - 1$ instead of n , \bar{x} - mean of sample

$$\text{Standard deviation} = S_{sample} = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$$

Let's take `Insol.csv` dataset.

Desvio Médio Absoluto

Desvio Médio Absoluto (MAD)

Mean absolute deviation is the average distance between mean and each data point.

$$\text{Mean absolute deviation (MAD)} = \frac{\sum |x_i - \bar{x}|}{n}$$

Let's calculate mean absolute deviation for Fundo_de_Maneio

```
▶ fundo_de_maneio = data['Fundo_de_maneio'].mad()  
print(f'Desvio Médio Absoluto = {fundo_de_maneio}')
```

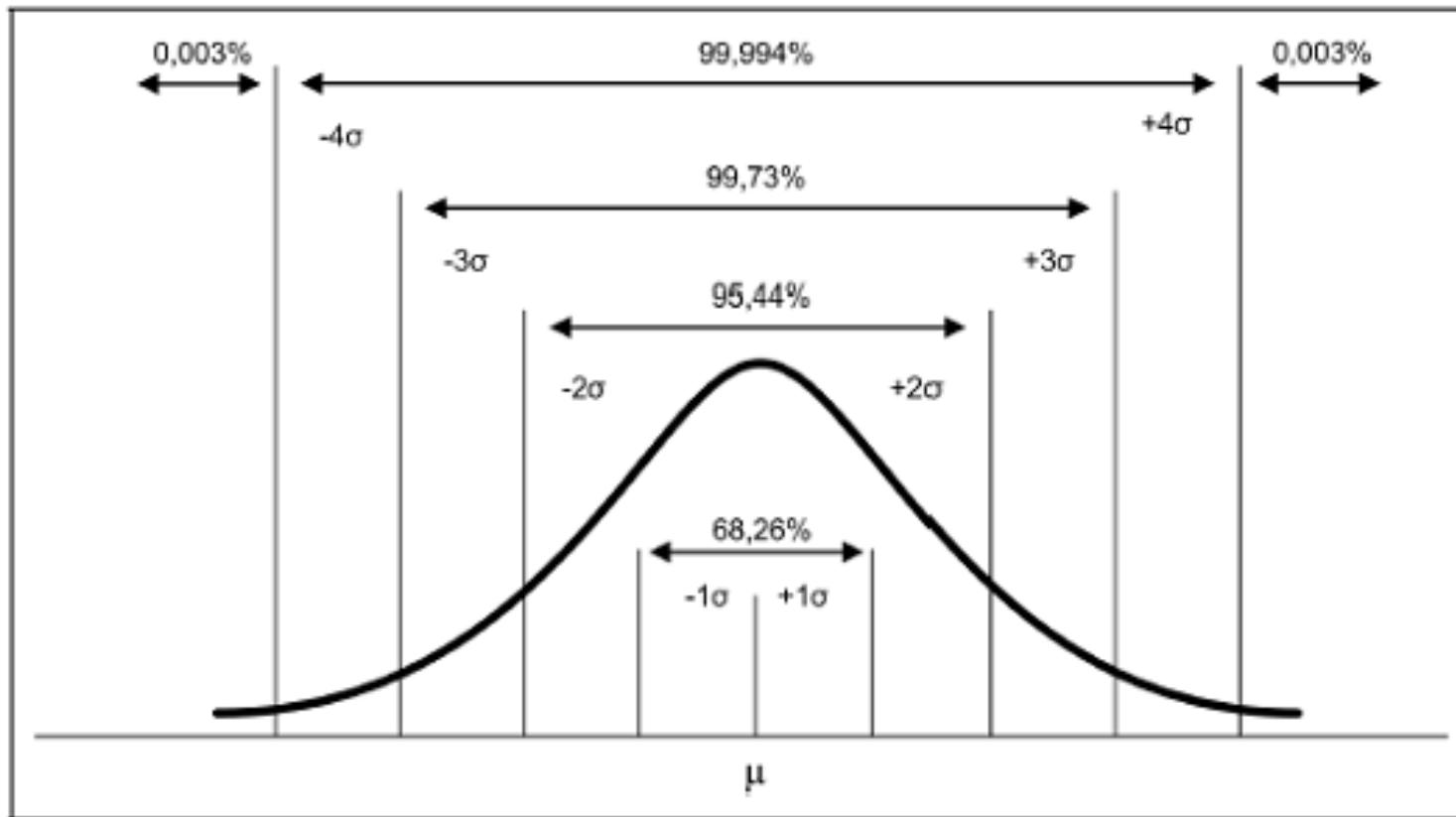
Desvio Médio Absoluto = 538645.1840978444

Analisar a Distribuição Normal (ou gaussiana);

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\}$$

A funcionalidade para distribuição normal é implementada por argumentos que combinam as letras acima com o termo norm. Por default as funções assumem a distribuição normal padrão

Analisar a Distribuição Normal (ou gaussiana);



3. Jupyter Notebook

Distribuição Normal

```
▶ fundo_de_maneio = data['Fundo_de_maneio']

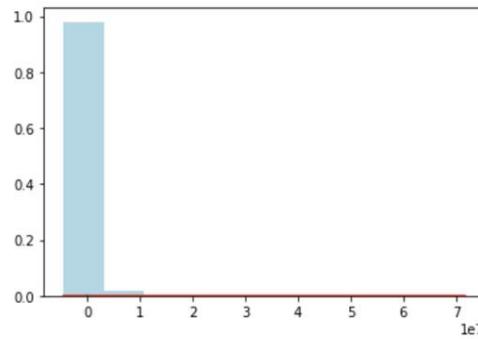
# Get mean (mu) and std (sigma)
fundo_de_maneio_media, fundo_de_maneio_std = fundo_de_maneio.mean(), fundo_de_maneio.std()

# Plot histogram (normalized) - LIGHT-BLUE
fundo_de_maneio.hist(color='lightblue', weights = np.zeros_like(fundo_de_maneio) + 1.0 / fundo_de_maneio.count())

# Plot line graph - RED
fundo_de_maneio.value_counts(sort=False, normalize=True).plot.line(color='red')

# Normal distribution for random points between 1 to 10 with mean, std.
random_data = np.arange(1, 10, 0.001)
pyplot.plot(random_data, stats.norm.pdf(random_data, fundo_de_maneio_media, fundo_de_maneio_std), color='green')
```

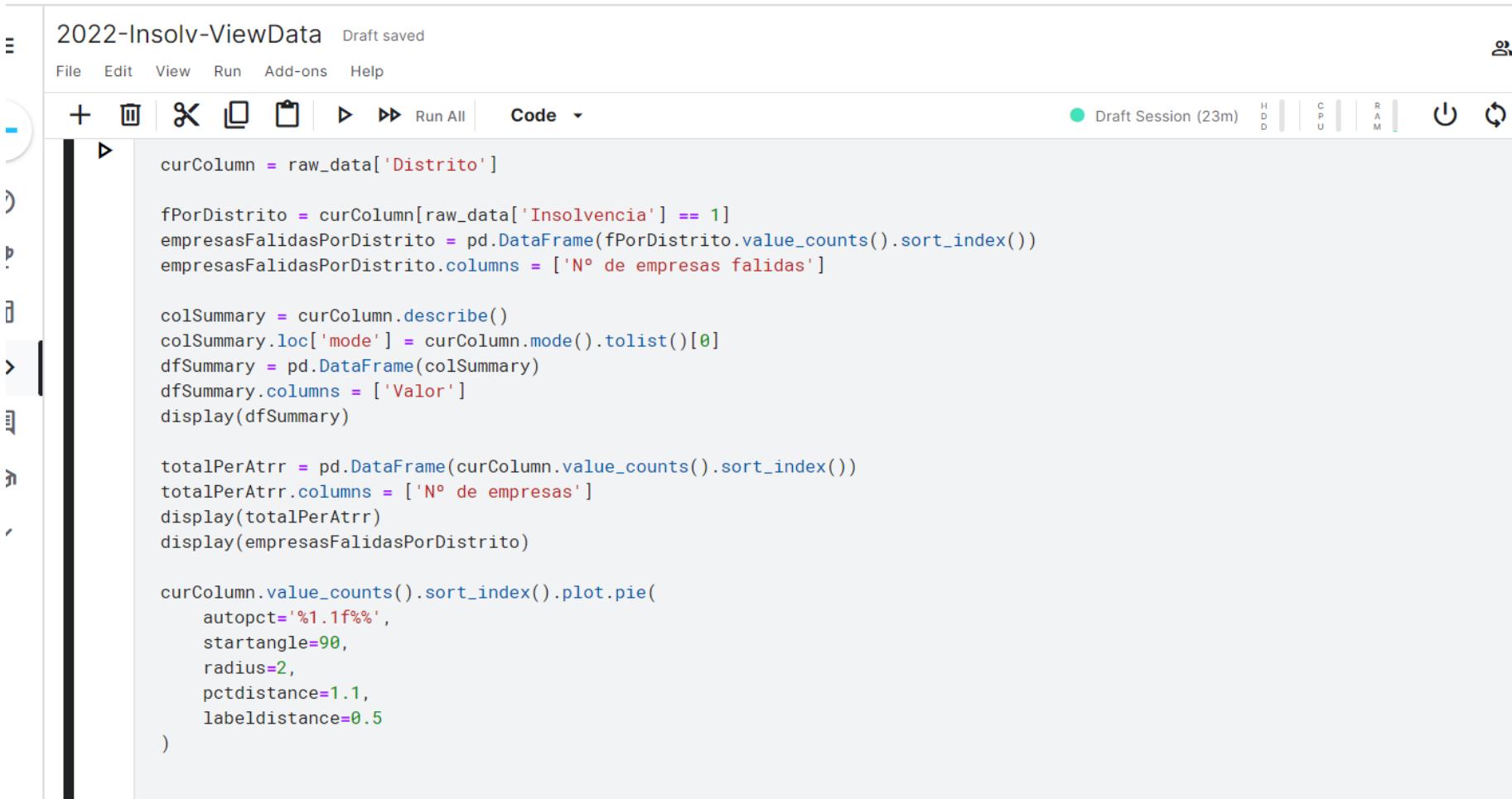
14... [<matplotlib.lines.Line2D at 0x7f0c37210650>]



+ Code

+ Markdown

3. Jupyter Notebook



The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** 2022-Insolv-ViewData Draft saved
- Toolbar:** File, Edit, View, Run, Add-ons, Help, Code dropdown, Draft Session (23m), and system status indicators for HDD, CPU, and RAM.
- Code Cell:** Contains Python code for data analysis, specifically focusing on the 'Distrito' column of a dataset named 'raw_data'. The code includes:
 - Filtering by Insolvencia == 1.
 - Counting companies per district.
 - Describing the column.
 - Displaying the summary.
 - Counting total companies per district.
 - Displaying the total count.
 - Creating a pie chart for the 'Distrito' values.

3. Jupyter Notebook

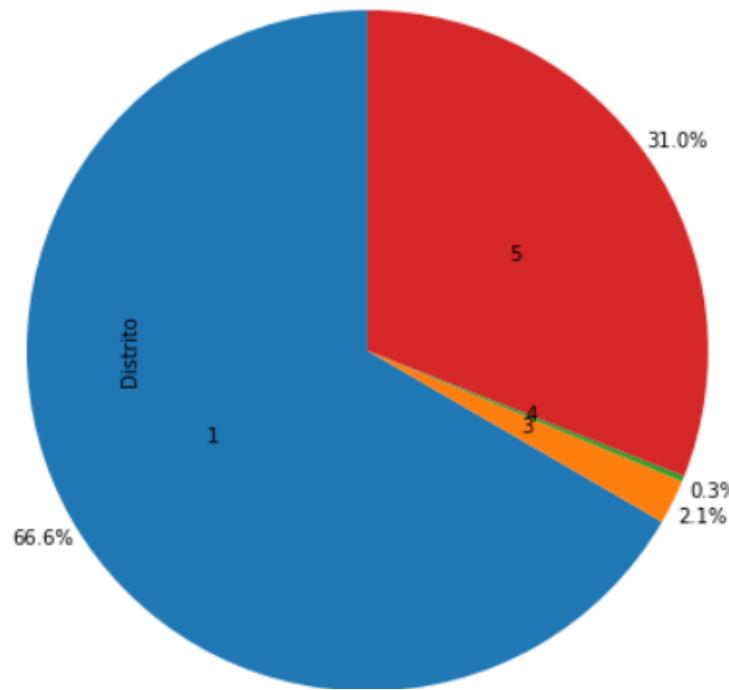
	Valor
count	2738.000000
mean	2.290358
std	1.845332
min	1.000000
25%	1.000000
50%	1.000000
75%	5.000000
max	5.000000
mode	1.000000

	Nº de empresas
1	1824
3	58
4	7
5	849

Nº de empresas falidas

1	349
3	12
4	3
5	167

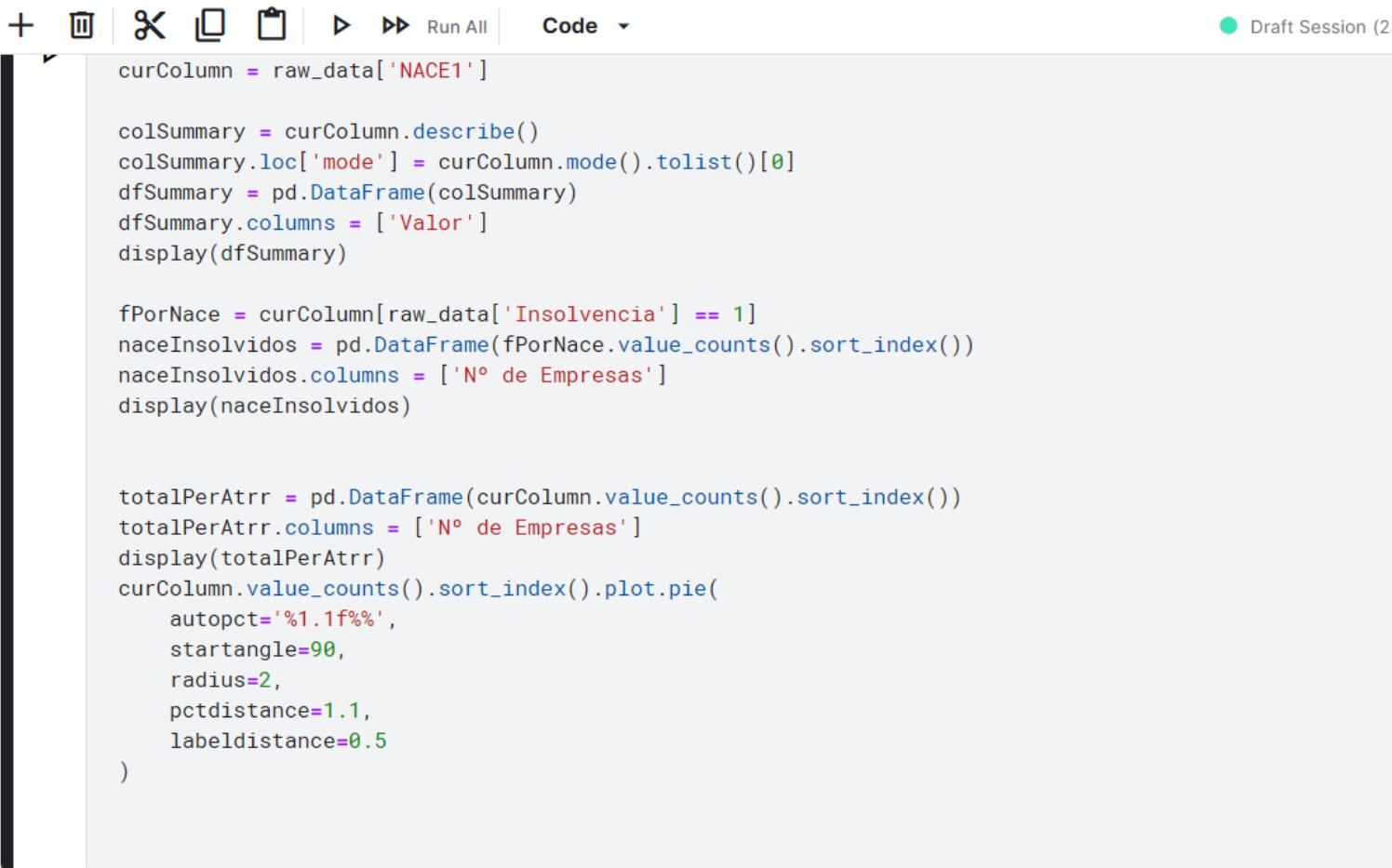
[9]: <AxesSubplot:ylabel='Distrito'>



3. Jupyter Notebook

2022-Insolv-ViewData Draft saved

File Edit View Run Add-ons Help



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

```
curColumn = raw_data['NACE1']

colSummary = curColumn.describe()
colSummary.loc['mode'] = curColumn.mode().tolist()[0]
dfSummary = pd.DataFrame(colSummary)
dfSummary.columns = ['Valor']
display(dfSummary)

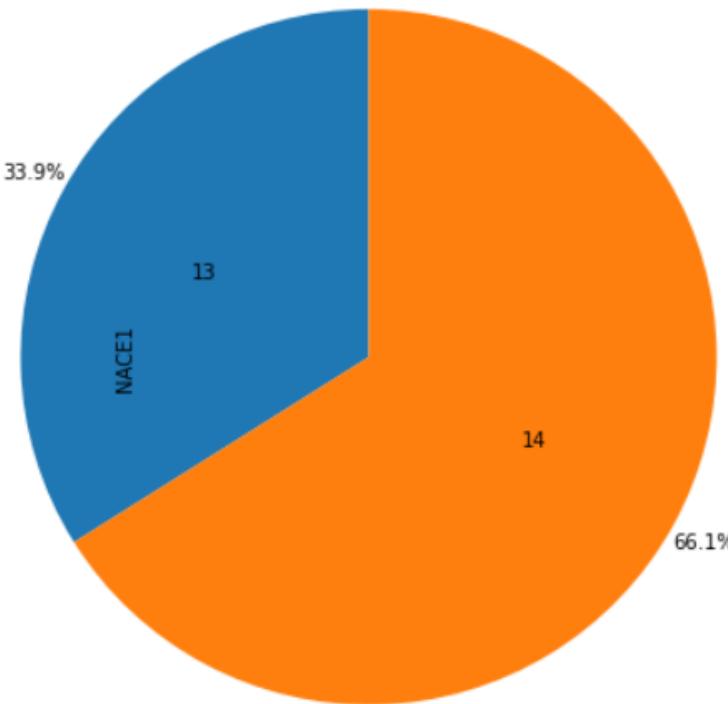
fPorNace = curColumn[raw_data['Insolvencia'] == 1]
naceInsolvidos = pd.DataFrame(fPorNace.value_counts().sort_index())
naceInsolvidos.columns = ['Nº de Empresas']
display(naceInsolvidos)

totalPerAtrr = pd.DataFrame(curColumn.value_counts().sort_index())
totalPerAtrr.columns = ['Nº de Empresas']
display(totalPerAtrr)
curColumn.value_counts().sort_index().plot.pie(
    autopct='%1.1f%%',
    startangle=90,
    radius=2,
    pctdistance=1.1,
    labeldistance=0.5
)
```

3. Jupyter Notebook

	Valor
count	2738.000000
mean	13.660701
std	0.473558
min	13.000000
25%	13.000000
50%	14.000000
75%	14.000000
max	14.000000
mode	14.000000

[10]: <AxesSubplot:ylabel='NACE1'>



	Nº de Empresas
13	143
14	388

	Nº de Empresas
13	929
14	1809

■ 3. Jupyter Notebook

```
> curColumn = raw_data['NACE2']
var_nace2 = curColumn.unique()
var_nace2.sort()

colSummary = curColumn.describe()
colSummary.loc['mode'] = curColumn.mode().tolist()[0]
dfSummary = pd.DataFrame(colSummary)
dfSummary.columns = ['Valor']
display(dfSummary)

totalPerAtrr = pd.DataFrame(curColumn.value_counts().sort_index())
totalPerAtrr.columns = ['Nº de empresas']
display(totalPerAtrr)

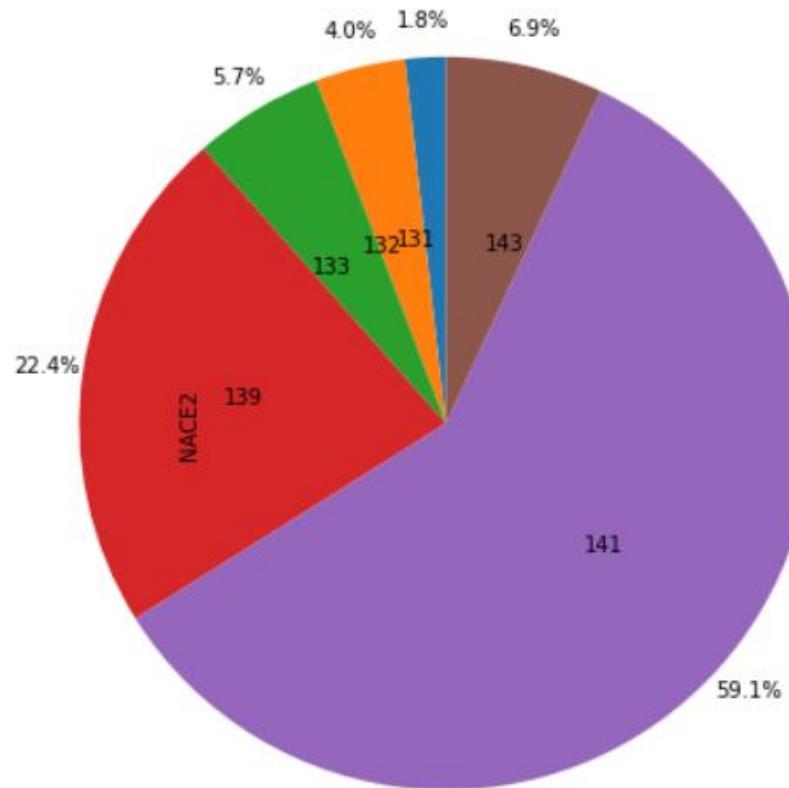
curColumn.value_counts().sort_index().plot.pie(
    autopct='%1.1f%%',
    startangle=90,
    radius=2,
    pctdistance=1.1,
    labeldistance=0.5
)
```

3. Jupyter Notebook

	Valor
count	2738.000000
mean	139.694302
std	2.856009
min	131.000000
25%	139.000000
50%	141.000000
75%	141.000000
max	143.000000
mode	141.000000

	Nº de empresas
131	49
132	109
133	157
139	614
141	1619
143	190

[11]: <AxesSubplot:ylabel='NACE2'>



■ 3. Jupyter Notebook



```
curColumn = raw_data['Numero_Empregados']
var_empregados = curColumn.unique()
var_empregados.sort()

colSummary = curColumn.describe()
colSummary.loc['mode'] = curColumn.mode().tolist()[0]
dfSummary = pd.DataFrame(colSummary)
dfSummary.columns = ['Valor']

bins = [1, 100, 200, 300, 400, 500, 600, 1200, 1400]
curColumn = pd.cut(curColumn, bins)

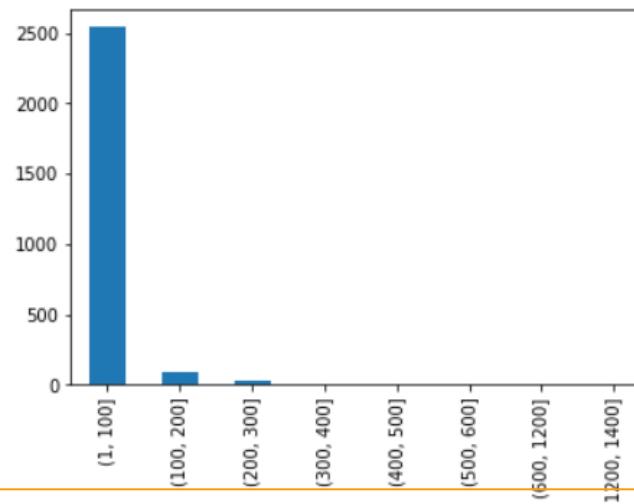
dfIntervals = pd.DataFrame(curColumn.value_counts().sort_index())
dfIntervals.columns = ['Número de Empresas dentro do intervalo']

display(dfSummary)
display(dfIntervals)
curColumn.value_counts().sort_index().plot.bar()
```

3. Jupyter Notebook

	Valor	Número de Empresas dentro do intervalo
count	2738.000000	(1, 100] 2540
mean	29.096421	(100, 200] 93
std	56.651704	(200, 300] 24
min	1.000000	(300, 400] 7
25%	7.000000	(400, 500] 3
50%	13.000000	(500, 600] 5
75%	30.000000	(600, 1200] 2
max	1302.000000	(1200, 1400] 1
mode	7.000000	

[12]: <AxesSubplot:>



■ 3. Jupyter Notebook



```
curColumn = raw_data['Num_anos_disponiveis_base']
var_num_anos_disponiveis_base = curColumn.unique()

colSummary = curColumn.describe()
colSummary.loc['mode'] = curColumn.mode().tolist()[0]
dfSummary = pd.DataFrame(colSummary)
dfSummary.columns = ['Valor']

dfIntervals = pd.DataFrame(curColumn.value_counts().sort_index())
dfIntervals.columns = ['Número de registos (anos disponíveis base)']

display(dfSummary)
display(dfIntervals)

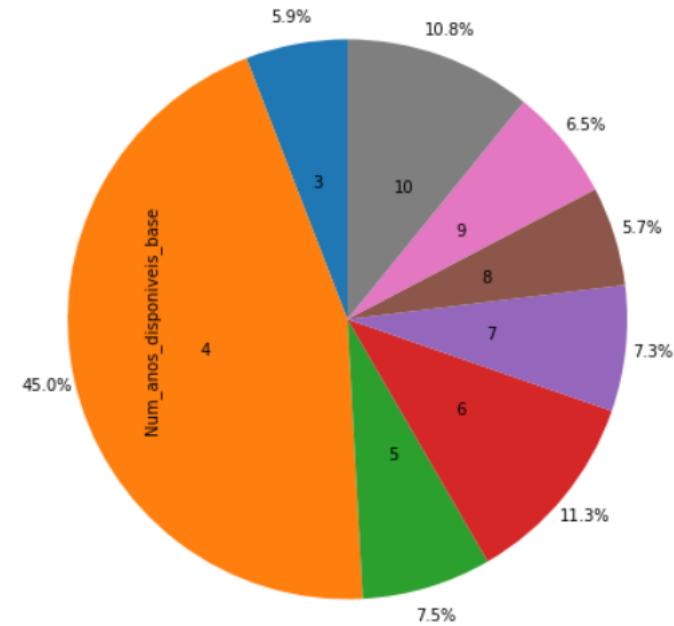
curColumn.value_counts().sort_index().plot.pie(
    autopct='%1.1f%%',
    startangle=90,
    radius=2,
    pctdistance=1.1,
    labeldistance=0.5
)
```

3. Jupyter Notebook

	Valor
count	2738.000000
mean	5.663988
std	2.216363
min	3.000000
25%	4.000000
50%	4.000000
75%	7.000000
max	10.000000
mode	4.000000

	Número de registos (anos disponíveis base)
3	161
4	1232
5	205
6	310
7	199
8	157
9	177
10	297

[13]: <AxesSubplot:ylabel='Num_anos_disponiveis_base'>



■ 3. Jupyter Notebook

```
curColumn = raw_data['Variacao_das_vendas']
var_variacao_vendas = curColumn.unique()
var_variacao_vendas.sort()

colSummary = curColumn.describe()
colSummary.loc['mode'] = curColumn.mode().tolist()[0]
colSummary.loc['confidence-interval'] = st.norm.interval(alpha=0.95, loc=np.mean(curColumn), scale=st.sem(curColumn))
dfSummary = pd.DataFrame(colSummary)
dfSummary.columns = ['Valor']

curColumn = pd.cut(curColumn, bins_var)
dfIntervals = pd.DataFrame(curColumn.value_counts().sort_index())
dfIntervals.columns = ['Nº empresas com variação de vendas dentro do intervalo']

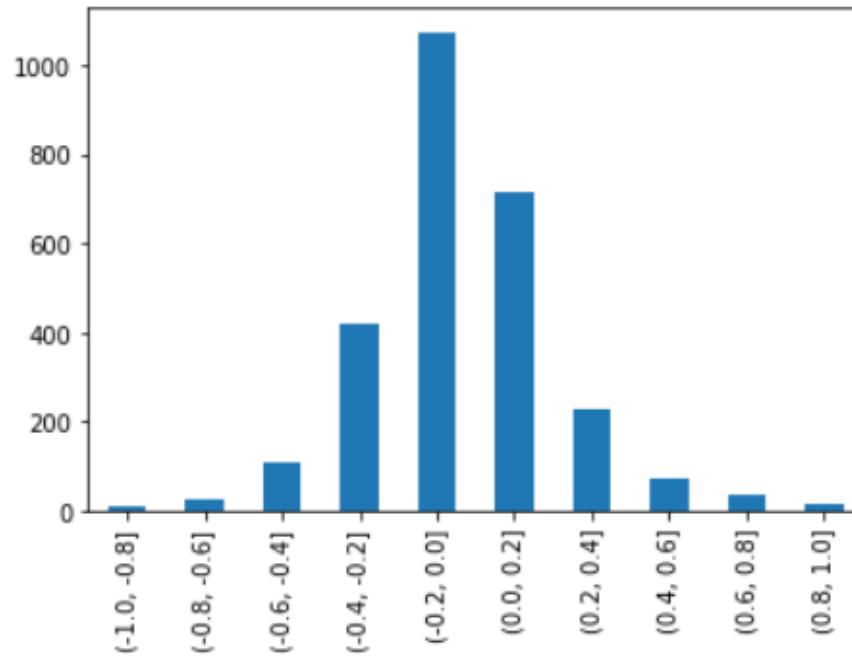
display(dfSummary)
display(dfIntervals)
curColumn.value_counts().sort_index().plot.bar()
```

3. Jupyter Notebook

	Valor
count	2738.0
mean	0.102269
std	5.804741
min	-0.993504
25%	-0.169661
50%	-0.047984
75%	0.082523
max	303.061661
mode	0.0
confidence-interval	(-0.1151581873260547, 0.31969652684029864)

Nº empresas com variação de vendas dentro do intervalo	
(-1.0, -0.8]	8
(-0.8, -0.6]	28
(-0.6, -0.4]	106
(-0.4, -0.2]	420
(-0.2, 0.0]	1074
(0.0, 0.2]	716
(0.2, 0.4]	228
(0.4, 0.6]	71
(0.6, 0.8]	36
(0.8, 1.0]	15

[14]: <AxesSubplot:>



■ 3. Jupyter Notebook

```
curColumn = raw_data['Variacao_resultado_operacional']
var_variacao_resultado_op = curColumn.unique()
var_variacao_resultado_op.sort()

colSummary = curColumn.describe()
colSummary.loc['mode'] = curColumn.mode().tolist()[0]
colSummary.loc['confidence-interval'] = st.norm.interval(alpha=0.95, loc=np.mean(curColumn), scale=st.sem(curColumn))
dfSummary = pd.DataFrame(colSummary)
dfSummary.columns = ['Valor']

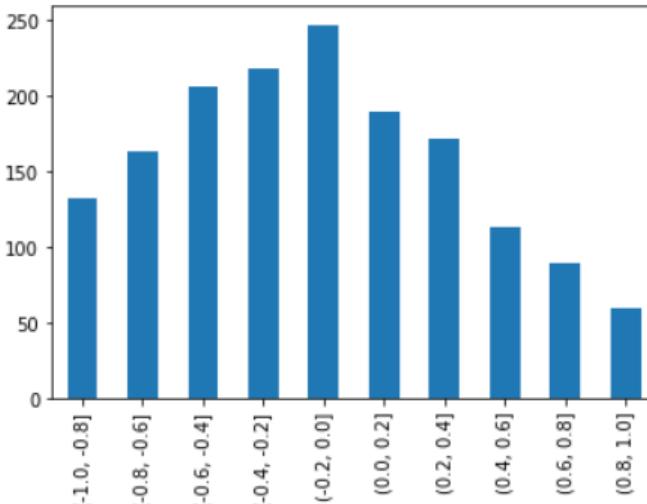
curColumn = pd.cut(curColumn, bins_var)
dfIntervals = pd.DataFrame(curColumn.value_counts().sort_index())
dfIntervals.columns = ['Nº empresas com variação de resultado operacional dentro do intervalo']
display(dfSummary)
display(dfIntervals)
curColumn.value_counts().sort_index().plot.bar()
```

3. Jupyter Notebook

	Valor
count	2738.0
mean	-1.370878
std	22.436321
min	-550.546808
25%	-0.996974
50%	-0.235467
75%	0.475887
max	450.483627
mode	0.0
confidence-interval	(-2.2112722432371172, -0.5304840022544829)

Nº empresas com variação de resultado operacional dentro do intervalo	
(-1.0, -0.8]	132
(-0.8, -0.6]	163
(-0.6, -0.4]	206
(-0.4, -0.2]	218
(-0.2, 0.0]	247
(0.0, 0.2]	189
(0.2, 0.4]	172
(0.4, 0.6]	113
(0.6, 0.8]	89
(0.8, 1.0]	60

[16]: <AxesSubplot:>



■ 3. Jupyter Notebook



```
curColumn = raw_data['Variacao_do_ativo']
var_variacao_ativo = curColumn.unique()
var_variacao_ativo.sort()

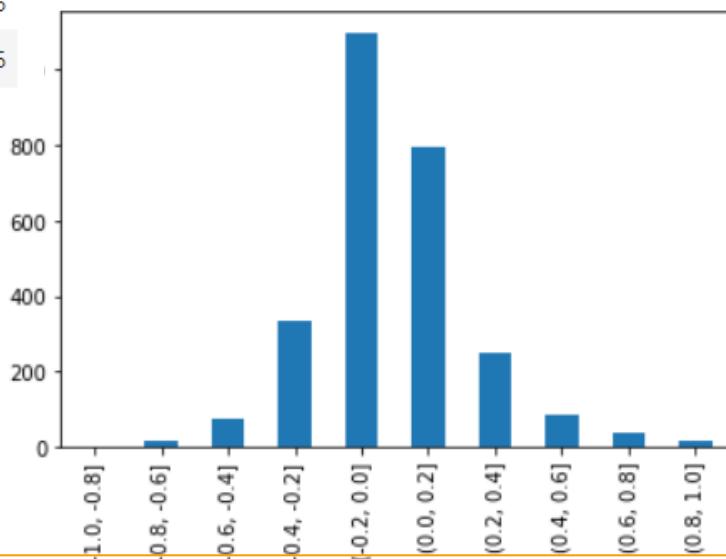
colSummary = curColumn.describe()
colSummary.loc['mode'] = curColumn.mode().tolist()[0]
dfSummary = pd.DataFrame(colSummary)
dfSummary.columns = ['Valor']

curColumn = pd.cut(curColumn, bins_var)
dfIntervals = pd.DataFrame(curColumn.value_counts().sort_index())
dfIntervals.columns = ['Nº empresas com variação de ativo dentro do intervalo']
display(dfSummary)
display(dfIntervals)
curColumn.value_counts().sort_index().plot.bar()
```

3. Jupyter Notebook

Nº empresas com variação de ativo dentro do intervalo			
count	2738.0	(-1.0, -0.8]	0
		(-0.8, -0.6]	16
mean	0.0	(-0.6, -0.4]	72
std	0.1	(-0.4, -0.2]	332
min	-0.1	(-0.2, 0.0]	1099
25%	-0.1	(0.0, 0.2]	798
50%	-0.0	(0.2, 0.4]	251
75%	0.1	(0.4, 0.6]	87
max	8.1	(0.6, 0.8]	36
mode	0.000000	(0.8, 1.0]	16

```
asSubplot:>
```



■ 3. Jupyter Notebook

```
# Configuring the matplotlib figure size and title
fig, ax = plt.subplots(figsize=(30,30))
ax.set_title('Correlation between variables in dataset')

# Excluding the categorical columns from dataframe
filtered_data = raw_data.drop(["Distrito", "NACE1", "NACE2", "Certificacao_legal_contas", "Insolvencia"], axis=1)
heatmap = sns.heatmap(filtered_data.corr(), vmin=-1, vmax=1, annot=True, ax=ax, cmap="icefire")

# Saving the image to a file
plt.savefig('correlation_matrix.png', bbox_inches="tight")
```

3. Jupyter Notebook

Numero_Empregados	1	0.35	0.047	0.016	0.012	0.0035	0.02	0.016	-0.021	0.63	-0.039	0.00043	-0.0078	-0.018	0.022	0.074	-0.074	-0.0068	-0.087	0.016	0.027	0.029	0.043	-0.011	0.0057	-0.0071	0.065	0.0078
Num_anos_disponiveis_base	0.35	1	0.014	0.017	-0.075	0.0035	-0.072	-0.015	-0.0096	0.17	-0.088	0.016	0.037	0.0048	0.12	0.14	-0.14	-0.016	-0.11	0.045	0.057	0.056	0.074	0.0067	0.018	0.0091	0.083	0.073
Variacao_das_vendas	0.047	0.014	1	0.088	0.2	0.0057	0.17	0.014	0.0029	0.0094	-0.0023	0.0058	-0.0051	-0.0031	-0.0048	0.0051	-0.0051	0.00024	0.015	0.0032	0.0027	0.0045	0.026	0.0033	-0.0054	-0.0094	-0.02	0.015
Variacao_resultado_operacional	0.016	0.017	0.088	1	0.084	0.026	0.076	0.0082	0.0051	0.015	0.0048	0.0011	-0.046	-0.028	0.0086	0.042	-0.042	-0.003	0.034	0.18	0.033	0.031	0.15	0.015	-0.0045	-0.011	0.029	0.084
Variacao_do_ativo	0.012	-0.075	0.2	0.084	1	0.058	0.87	0.066	0.1	0.0056	-0.0017	-0.0074	-0.018	-0.011	-0.038	0.064	-0.064	0.0024	0.025	0.037	0.051	0.059	0.25	0.035	-0.0043	-0.025	-0.041	0.14
Variacao_dos_capitais_proprios	0.0035	0.0035	0.0057	0.026	0.058	1	0.05	0.0032	0.0069	0.0085	-0.0059	0.00096	0.017	0.016	0.027	0.07	-0.07	0.0055	0.0037	0.0069	0.025	0.026	0.12	0.8	0.0036	0.00044	-0.019	0.063
Variacao_do_ativo_circulante	0.02	-0.072	0.17	0.076	0.87	0.05	1	0.068	0.016	0.0014	-0.005	-0.0089	-0.019	-0.011	-0.034	0.03	-0.03	0.0037	0.03	0.026	0.039	0.047	0.21	0.031	-0.00023	-0.01	-0.036	0.12
Variacao_das_existencias	0.016	-0.015	0.014	0.0082	0.066	0.0032	0.068	1	-0.004	-0.0075	-0.012	-0.0026	-0.011	-0.0098	-0.0098	0.005	-0.005	0.0026	0.028	0.0029	0.0037	0.0045	0.016	0.00048	-0.00018	-0.0031	-0.017	-0.0022
Variacao_do_imobilizado	-0.021	-0.0096	0.0029	0.0051	0.1	0.0069	0.016	-0.004	1	-0.0058	-0.0062	-0.002	0.0079	0.008	0.02	0.036	-0.036	-0.001	0.014	0.0025	0.0078	0.0076	0.034	0.00097	-0.0037	-0.0065	-0.012	0.038
Fundo_de_maneio	0.63	0.17	0.0094	0.015	0.0056	0.0085	0.0014	-0.0075	-0.0058	1	-0.022	0.0045	0.078	0.056	0.097	0.076	-0.076	-0.0013	-0.12	0.0092	0.014	0.015	0.046	9.2e-05	-0.00099	-0.0043	0.051	0.054
REPV	-0.039	-0.088	-0.0023	0.0048	-0.0017	-0.0059	-0.005	-0.012	-0.0062	-0.022	1	-0.0072	-0.015	0.0084	-0.0021	-0.013	0.013	-0.0047	0.05	0.011	0.0048	0.0066	-0.0095	-0.014	0.0044	-0.0084	-0.055	-0.0096
Prazo_medio_de_recebimento	0.00043	0.016	-0.0058	0.0011	-0.0074	0.00096	-0.0089	-0.0026	-0.002	0.0045	-0.0072	1	0.022	0.028	0.026	0.013	-0.013	-0.00076	0.0047	-0.0088	0.55	0.4	0.0077	-0.0009	0.76	0.89	0.35	-0.093
Liquidex_geral	-0.0078	0.037	-0.0051	-0.046	-0.018	0.017	-0.019	-0.011	0.0079	0.078	-0.015	0.022	1	0.95	0.4	0.14	-0.14	0.0019	-0.23	-0.058	0.024	0.026	0.064	-0.0018	0.019	0.012	-0.053	-0.011
Liquidex_reduzida	-0.018	0.0048	0.0031	-0.028	-0.011	0.016	-0.011	-0.0098	0.008	0.056	0.0084	0.028	0.95	1	0.3	0.13	-0.13	-0.00017	-0.17	-0.054	0.028	0.029	0.058	-0.002	0.026	0.017	-0.069	-0.0066
Solvabilidade	0.022	0.12	-0.0048	0.0086	-0.038	0.027	-0.034	-0.0098	0.02	0.097	-0.0021	0.026	0.4	0.3	1	0.31	-0.31	-0.011	0.13	0.013	0.061	0.062	0.13	-0.0021	0.036	0.032	-0.097	0.058
Autonomia_financeira	0.074	0.14	0.0053	0.042	0.064	0.07	0.03	0.005	0.036	0.076	-0.013	0.013	0.14	0.13	0.31	1	-1	0.003	0.066	0.049	0.1	0.1	0.6	-0.0087	0.0016	0.00072	-0.11	0.18

■ 3. Jupyter Notebook



```
curColumn = raw_data['Insolvencia']
var_insolvencia = curColumn.unique()
var_insolvencia.sort()

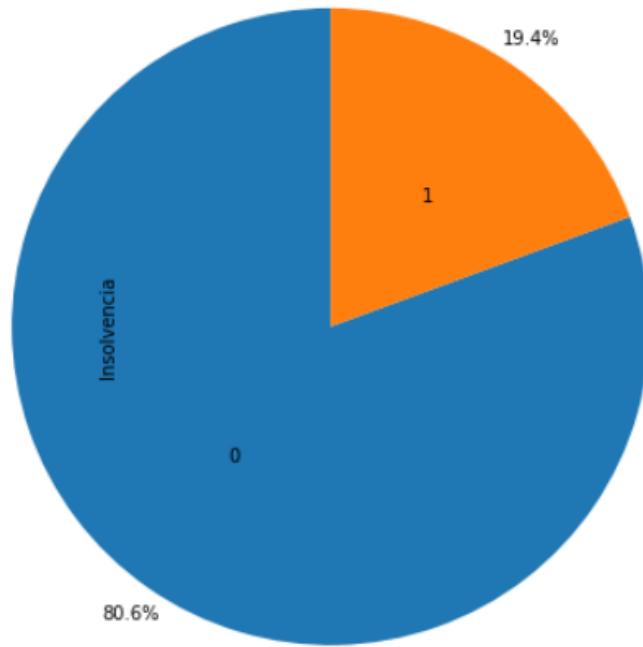
dfIntervals = pd.DataFrame(curColumn.value_counts().sort_index())
dfIntervals.columns = ['Nº empresas em insolvencia (sim/nao)']
display(dfIntervals)
curColumn.value_counts().sort_index().plot.pie(
    autopct='%1.1f%%',
    startangle=90,
    radius=2,
    pctdistance=1.1,
    labeldistance=0.5
)
```

3. Jupyter Notebook

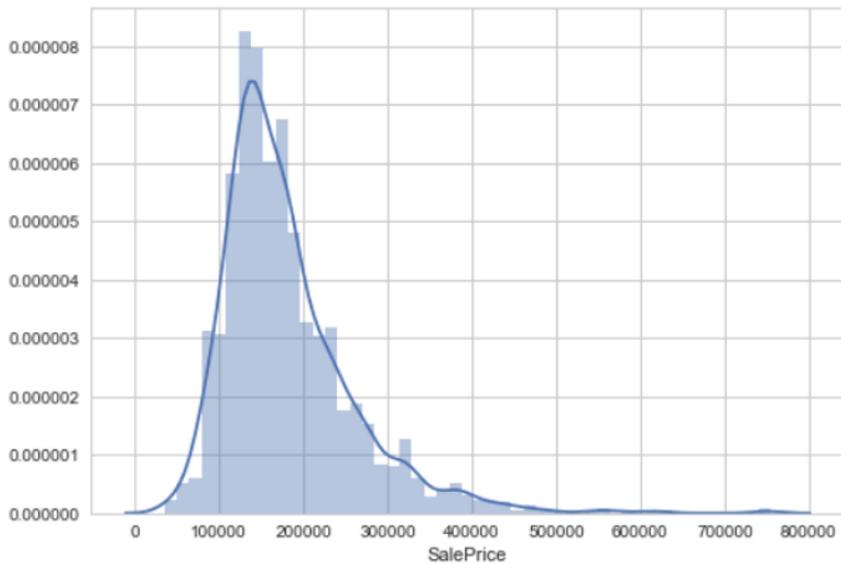
Nº empresas em insolvencia (sim/nao)

0	2207
1	531

[23]: <AxesSubplot:ylabel='Insolvencia'>



```
#SalePrice  
sns.distplot(train['SalePrice'])
```



Correlação de Dados

Coeficiente de Correlação de Pearson

Também chamado de “coeficiente de correlação produto-momento” ou simplesmente de “ ρ de Pearson” mede o grau da correlação (e a direção dessa correlação — se positiva ou negativa) entre duas variáveis. Este coeficiente, normalmente representado por ρ assume apenas valores entre -1 e 1.

A formula desenvolvida por Karl Pearson, há mais de 120 anos, continua a ser a mais utilizada para o cálculo da correlação.

Calcula-se o coeficiente de correlação de Pearson pela seguinte fórmula:

$$\rho = \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\left[\sum_{i=1}^n (x_i - \bar{x})^2 \right] \left[\sum_{i=1}^n (y_i - \bar{y})^2 \right]}}$$

A correlação de Pearson mede a associação linear entre **variáveis contínuas**. É o valor que indica o quanto a relação entre as variáveis pode ser descrita por uma reta.

Correlação de Dados

Interpretando o valor de ρ

- 0.9 a 1 positivo ou negativo indica uma correlação muito forte.
- 0.7 a 0.9 positivo ou negativo indica uma correlação forte.
- 0.5 a 0.7 positivo ou negativo indica uma correlação moderada.
- 0.3 a 0.5 positivo ou negativo indica uma correlação fraca.
- 0 a 0.3 positivo ou negativo indica uma correlação desprezível.

Existem várias possibilidades de interpretação da correlação. Pode-se considerar a inclinação da reta que representa a relação entre as variáveis, pode-se considerar as séries de valores como vetores, e o ρ , em uma interpretação geométrica, representaria o cosseno do ângulo formado entre os vetores, etc. Algumas informações interessantes podem ser encontradas no artigo “[Thirteen ways to look at the correlation coefficient](#)”.

3. Jupyter Notebook

Correlação de Dados

[6]:

```
data.corr()
```

[6]:

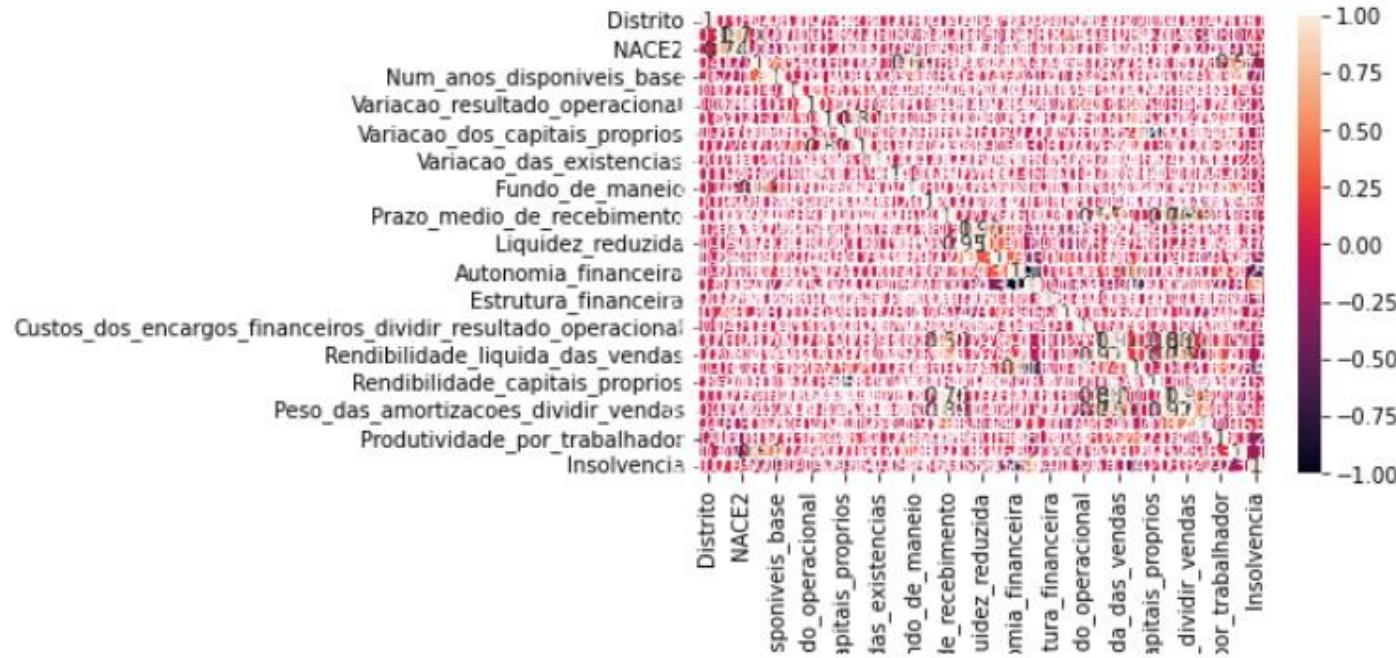
	Distrito	NACE1	NACE2	Numero_Empregados	Num_anos_disponiveis_base	Variacao_das_vendas	Variacao_resultado_operacion
Distrito	1.000000	0.037939	0.037091	0.041098	0.062187	0.019131	0.0107
NACE1	0.037939	1.000000	0.740732	-0.027026	-0.107619	0.036619	0.0016
NACE2	0.037091	0.740732	1.000000	-0.091921	-0.138888	0.057968	0.0156
Numero_Empregados	0.041098	-0.027026	-0.091921	1.000000	0.350247	-0.008386	0.0234
Num_anos_disponiveis_base	0.062187	-0.107619	-0.138888	0.350247	1.000000	-0.084069	0.0346
Variacao_das_vendas	0.019131	0.036619	0.057968	-0.008386	-0.084069	1.000000	0.2903
Variacao_resultado_operacional	0.010782	0.001686	0.015698	0.023409	0.034602	0.290335	1.0000
Variacao_do_ativo	0.047157	0.016935	0.028938	0.015393	-0.078099	0.559880	0.2023
Variacao_dos_capitais_proprios	0.010062	-0.018725	0.003964	-0.004091	0.011201	0.168846	0.2758
Variacao_do_ativo_circulante	0.021154	0.026027	0.018839	0.013656	-0.071728	0.353910	0.2203
Variacao_das_existencias	0.003055	0.016647	0.011263	-0.012314	-0.080497	0.050727	0.0276
Variacao_do_imobilizado	0.032348	-0.027162	0.001465	0.000418	-0.009735	0.084247	0.0185
Fundo_de_maneio	0.014417	-0.092560	-0.059891	0.625267	0.169693	-0.009067	0.0082
REPV	-0.012831	0.040679	0.042099	0.035598	0.019746	0.042391	-0.0242
Prazo_medio_de_recebimento	0.013277	0.101487	0.044559	0.003294	-0.011140	-0.003470	-0.0102
Liquidex_geral	0.028792	-0.045093	-0.023296	0.032797	0.113687	-0.030926	0.0008
Liquidex_reduzida	-0.043541	-0.049664	-0.035347	0.012015	0.057471	-0.031808	-0.0177
Solvabilidade	-0.041281	-0.047932	-0.035403	0.053188	0.126892	-0.016807	0.0257
Autonomia_financeira	-0.032169	-0.066863	-0.066803	0.067769	0.143857	0.022031	0.0967
Endividamento	0.031467	0.036635	0.050356	-0.085084	-0.140283	-0.021254	-0.0975
Estrutura_financeira	0.027322	-0.013030	0.005135	0.039168	0.054555	0.066010	0.1304
Passivo_de_curto_prazo_a_dividir_passivo_total	-0.027886	-0.089991	-0.083434	0.148825	0.189942	0.019554	0.0138
Custos_dos_encargos_financeiros_dividir_resultado_operacional	-0.022257	-0.003285	0.019673	0.001319	0.031020	0.202210	0.4342
Rendibilidade_operacional_vendas	-0.003200	0.015983	0.013421	0.045638	0.069670	0.112831	0.2265
Rendibilidade_liquida_das_vendas	-0.027638	0.031624	0.023319	0.052673	0.055564	0.190195	0.2707
Rendibilidade_do_ativo	-0.010382	-0.046734	-0.033311	0.041195	0.067369	0.151391	0.2852
Rendibilidade_capitais_proprios	-0.022366	0.019480	0.032749	-0.023369	0.001680	0.100414	0.2019

3. Jupyter Notebook



```
import seaborn as sn
m=raw_data.corr()
sn.heatmap(m, annot=True)
```

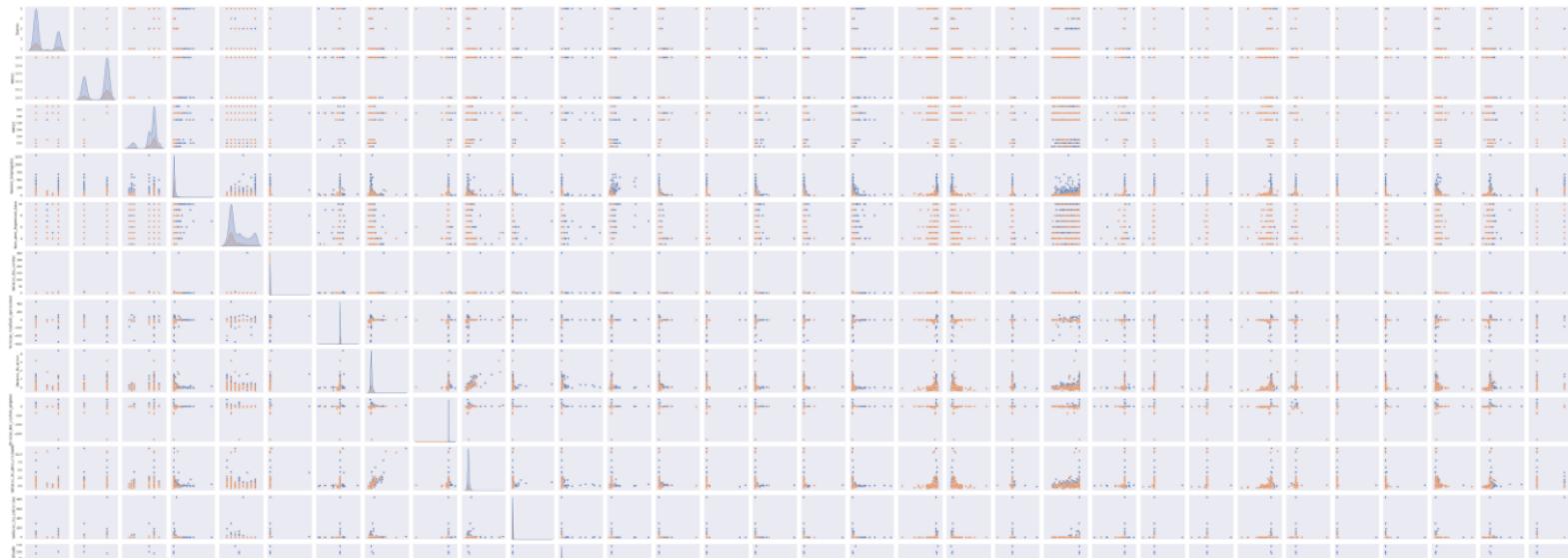
[27]: <AxesSubplot:>



3. Jupyter Notebook

```
sn.set_theme(style="dark")
g=sn.pairplot(raw_data, hue="Insolvencia")
g.savefig("scatterplot_matrix.png")
```

```
sn.set_theme(style="dark")
g=sn.pairplot(raw_data, hue="Insolvencia")
g.savefig("scatterplot_matrix.png")
```



3. Jupyter Notebook

```
sn.set_theme(style="dark")
g=sn.pairplot(raw_data, hue="Insolvencia")
g.savefig("scatterplot_matrix.png")
```

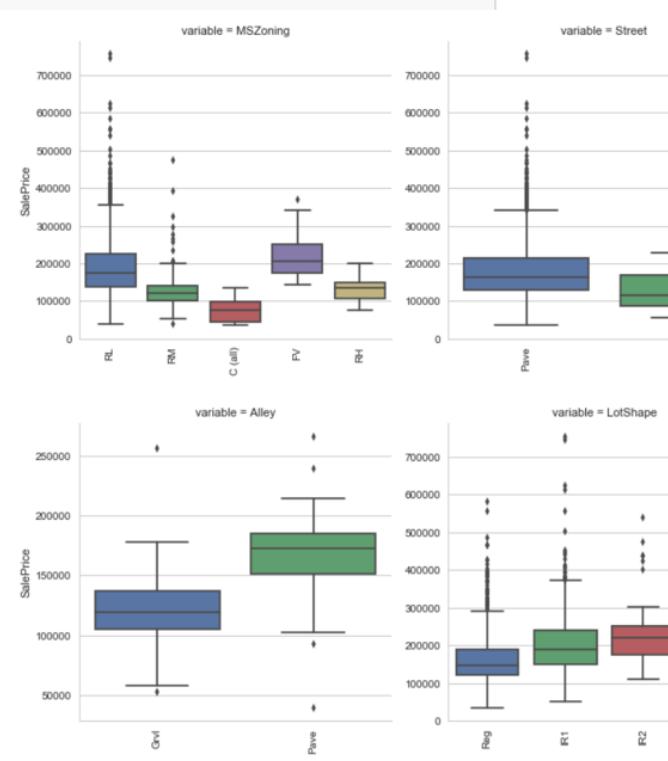
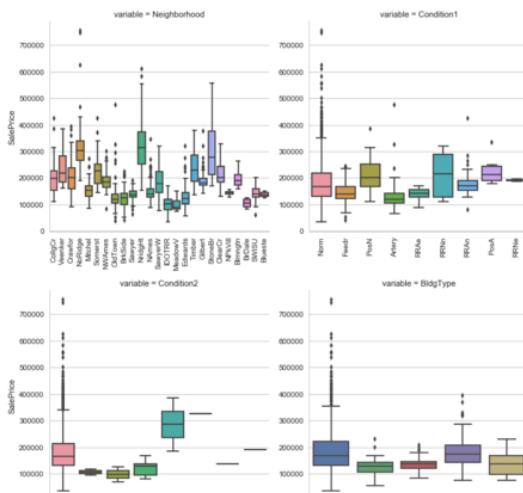


3. Jupyter Notebook

```
def boxplot(x,y,**kwargs):
    sns.boxplot(x=x,y=y)
    x = plt.xticks(rotation=90)

cat = [f for f in train.columns if train.dtypes[f] == 'object']

p = pd.melt(train, id_vars='SalePrice', value_vars=cat)
g = sns.FacetGrid (p, col='variable', col_wrap=2, sharex=False, sharey=False, s
g = g.map(boxplot, 'value','SalePrice')
g
```



■ 1.1. Objetivos

Assim, os principais objetivos deste trabalho, são:

1. Análise estatística simples de atributos do conjunto de dados: *Insol.csv*
2. Utilização da linguagem de programação Python em conjugação com a ferramenta Jupyter Notebook;
3. Exploração da biblioteca Scipy, para a análise das distribuições estatísticas, bem como a correlação entre atributos;

Conteúdos

-  —
- 1. Introdução
 - 1.1. Objetivos
- 2. Linguagem de Programação Python
- 3. Jupyter Notebook

4. Análise de dados

- 4.1. Análise de Estatísticas Simples
- 4.2. Análise de Distribuições Estatísticas
- 4.3. Análise da Correlação e Intervalo de Confiança

5. Aprendizagem Supervisionada:

- Criação de Modelo classificador/Previsão:
 - Árvores de Decisão;
 - Regressão Linear
 - Redes Neuronais Artificiais

6. Aprendizagem Não supervisionada:

Clustering – K-Means

7. Bibliografia e Referências Web

■ 5. Aprendizagem Supervisionada:

Criação de Modelo classificador/Previsão: Árvores de Decisão

1. Árvores de decisão - O que é: Describes decision trees as a type of supervised learning algorithm that uses a tree-like model of decisions and their possible consequences. It includes sections on what it is, types of trees, terminologies, advantages and disadvantages, examples, implementation in Python, R, and a bibliography.

2. Árvores de decisão: Shows a diagram of a decision tree with a root node splitting into two branches, each leading to a decision node, which then splits into terminal nodes.

3. Árvores de decisão: DECISION TREE: Shows a decision tree diagram with a root node splitting into two branches, each leading to a decision node, which then splits into terminal nodes.

4. Árvores de decisão: Describes decision trees as continuous functions that map the space of input variables to discrete values. It includes sections on decision rules, continuous vs categorical variables, and decision regions.

Exemplo 1



1. Vantagens: Lists several advantages: 1. Fácil de entender: Visualizing a decision tree makes the problem easy to understand for people who don't have much analytical training. 2. Utiliza todos os dados: A decision tree uses all the data available for classification. 3. Menor necessidade de limpar dados: Requiring less cleaning of data compared to other methods. 4. Não é necessário ter dados continuos: Can handle both numerical and categorical data. 5. Melhor para tipos de dados: Good for manipulating numerical and categorical data.

2. Desvantagens: Lists disadvantages: 1. Sobrepõe ("Overfitting"): A major difficulty for decision trees. 2. Inadequado para variáveis contínuas: Difficult to work with numerical variables. 3. Menor desempenho: May require more time to train than other methods. 4. Não é paralelo: A decision tree is considered a sequential method.

1. Confusion Matrix: Shows a confusion matrix for a dataset. The columns are 'Previsão de Insolvência' and 'Previsão de Não Insolvência'. The rows are 'Realidade de Insolvência' and 'Realidade de Não Insolvência'. The matrix values are:

	Previsão de Insolvência	Previsão de Não Insolvência
Realidade de Insolvência	1065	46
Realidade de Não Insolvência	19	239

2. Implementação em Python: Shows Python code for creating a decision tree classifier and visualizing it. The code imports 'confusion_matrix' from 'sklearn.metrics', creates a DataFrame with the confusion matrix, and then uses 'tree.DecisionTreeClassifier' and 'tree.export_graphviz' to create a dot file and a PNG visualization.

3. Implementação em Python: Shows a large decision tree visualization generated by the code above.

1. Importar Livrarias necessárias: Shows the R code to import necessary libraries: 'read.csv', 'library(caret)', 'library(rpart)', 'library(rpart.plot)', and 'library(gbm)'.

2. Guardar modelo num ficheiro Output: Shows the R code to save the model to a CSV file: 'model <- rpart(y ~ ., data = df, method = "class")' and 'write.csv(model, "model-output.csv")'.

3. Carregar o modelo de Output e ver o seu Resultado: Shows the R code to load the saved model and make predictions: 'model <- read.csv("model-output.csv", "rb")' and 'ynew <- predict(model, newdata = df)'.

■ 5. Aprendizagem Supervisionada:

Criação de Modelo classificador/Previsão: Regressão Linear

1

2

3

4

Embora estes dois sejam estudados juntos é comum que o professor descreva a regressão como uma forma de prever a variação de uma variável dependente em relação a uma ou mais variáveis independentes. No entanto, a regressão é um tipo de regressão que usa a correlação entre as variáveis para prever a variação da variável dependente.

Uma regressão só é chamada "linear" quando se considera que a relação da resposta às variáveis é tratada como uma função linear ($Ax + b$). Aquelas modelos de regressão que não podem ser traduzidos em uma função linear dos parâmetros são denominados não-lineares.

Uma das formas mais comuns de analisar regressões é através da utilização do método dos Menores Quadrados (MQO) ou, ainda, em inglês, Ordinary Least Squares (OLS). Neste método, visa buscar o melhor valor que os coeficientes possam assumir, de maneira que a diferença entre o valor predito para a função e o real, seja o menor.

$$f(x) = ax + b$$

$$f(x) = ax + b$$

$$a > 0 \quad \text{Crescente}$$

$$a < 0 \quad \text{Decrescente}$$

Variável preditora: É a variável independente que tem o poder de influenciar na variável que nós queremos entender. Na equação acima, ela será o valor de x .

Variável alvo ou dependente:

É a variável que queremos prever. Na equação acima, ela terá como valor o resultado da regressão para cada valor de x .

$$f(x) = ax + b \quad y = ax + b$$

17

18

19

20

21

22

23

24

Exemplo 1



■ 5. Aprendizagem Supervisionada:

Criação de Modelo classificador/Previsão: Redes Neuronais Artificiais

Exemplo 2



■ 5. Aprendizagem Supervisionada:

Criação de Modelo classificador/Previsão: Redes Neuronais Artificiais

Exemplo 2



■ 2. Python - Exploração de Algoritmos de Inteligência Artificial

Com a análise dos dados realizada, passou-se à fase seguinte, correspondente à utilização do conjunto de dados analisado, convertendo todas as ocorrências de valores para pontas, na separação de valores decimais no dataset, para que os seus valores possam ser interpretados corretamente, a este ficheiro atribui-se o nome de `Inst_Finorte_Certo.csv`. Para a exploração de implementações de algoritmos de inteligência artificial, neste caso para a linguagem Python, utilizou-se as implementações da biblioteca scikit-learn, explicando a criação de modelos, assim, as implementações utilizadas foram: árvore de decisão, regressão logística e redes neurais artificiais.

■ 2.1 Árvores de Decisão

Para a utilização da implementação do algoritmo das árvores de decisão, é necessário dividir o conjunto de dados em conjuntos de treino e teste, para o conjunto de treino é importante que seja grande, para que o modelo possa aprender com ele, e para o conjunto de teste, é importante que seja pequeno, para que o modelo não memorize os dados. As colunas consideradas mais importantes (dividindo pelo número de coluna), para a variável X, as colunas selecionadas neste caso foram: Distancia, NACEZ, Número de Empregados, Número de Anos Disponíveis, Autonomia Financeira, Endividamento e Estrutura Financeira. Já para a escolha se a coluna para a qual se quer efetuar a previsão, que neste caso é a coluna de Involvência, última coluna do dataset, selecionada utilizando o selector [:-1].

■ 2.1 Árvores de Decisão

De seguida, efetuou-se a divisão do conjunto de dados, neste caso em 25% para o conjunto de teste e 75% para o conjunto de treino, guardando os registos resultantes da divisão de X para treino em `x_train` e teste em `x_test`, tendo da coluna de modelação, ou seja, em `y`, trein. e `y_test`.

Começa-se a seguir, realizar-se uma transformação dos conjuntos de treino e teste dos valores das colunas em `x_train` e `x_test`, permitindo a padronização do conjunto de dados, um requisito para a maioria das implementações de algoritmos de investigação.

■ 2.1 Árvores de Decisão

Com o conjunto de dados tratado, define-se o modelo classificador utilizando a implementação da biblioteca scikit-learn, para o mesmo é necessário definir o critério de classificação, que neste caso é a entropia, e os valores das colunas consideradas na classificação e `y_train`, com valores das colunas restantes da modelação, para a qual se pretende obter um valor para classificado, neste caso `True` (1) ou `False` (0). Para a obtenção das previsões, ou seja todos os modelos, fornecem-se o conjunto de dados de teste, obtendo as previsões do modelo.

Figura 2 - Importação de conjuntos de dados, bibliotecas e seleção das colunas

Figura 3 - Código para a apresentação dos resultados e valores indicativos do modelo, incluindo a matriz de confusão

Figura 4 - Código para a apresentação dos resultados e valores indicativos do modelo, incluindo a matriz de confusão

Figura 5 - Código para a visualização das curvas de aprendizagem

contendo de todas as colunas menos a coluna de modelação, a qual tem os seus valores armazenados a `x` como a coluna para a qual se pretende obter a classificação final, baseado nos valores das restantes colunas. De seguida, se o conjunto de dados em conjuntos de teste e treino, padronizando os dados para os conjuntos de treino.

Figura 6 - Importação das bibliotecas necessárias, importação do dataset, dividido em conjuntos de teste e treino e padronização dos dados, com o pre-processamento

Figura 7 - Relação entre os valores obtidos de previsão e os valores reais, bem como o aumento do número total de registos de treino

Figura 8 - Redução entre os valores obtidos de previsão e o número de vezes treinado e resultado

Figura 9 - Cálculo para a apresentação da matriz de confusão

Figura 10 - Importação das bibliotecas necessárias, importação do dataset, dividido em conjuntos de teste e treino e padronização dos dados, com o pre-processamento

Figura 11 - Código para a visualização das curvas de aprendizagem

Figura 12 - Relação entre os valores obtidos de previsão e os valores reais, bem como o aumento do número total de registos de treino

Figura 13 - Cálculo para a visualização das curvas de aprendizagem

Figura 14 - Importação das bibliotecas necessárias, importação do dataset, dividido em conjuntos de teste e treino e padronização dos dados, com o pre-processamento

Figura 15 - Código para a visualização das curvas de aprendizagem

Figura 16 - Relação entre os valores obtidos de previsão e os valores reais, bem como a matriz de confusão

Figura 17 - Cálculo para a visualização das curvas de aprendizagem

Figura 18 - Importação das bibliotecas necessárias, importação do dataset, dividido em conjuntos de teste e treino e padronização dos dados, com o pre-processamento

Figura 19 - Código para a visualização das curvas de aprendizagem

Figura 20 - Relação entre os valores obtidos de previsão e os valores reais, bem como a matriz de confusão

Figura 21 - Cálculo para a visualização das curvas de aprendizagem

Figura 22 - Importação das bibliotecas necessárias, importação do dataset, dividido em conjuntos de teste e treino e padronização dos dados, com o pre-processamento

Figura 23 - Código para a visualização das curvas de aprendizagem

Figura 24 - Relação entre os valores obtidos de previsão e os valores reais, bem como a matriz de confusão

Figura 25 - Cálculo para a visualização das curvas de aprendizagem

Figura 26 - Importação das bibliotecas necessárias, importação do dataset, dividido em conjuntos de teste e treino e padronização dos dados, com o pre-processamento

Figura 27 - Código para a visualização das curvas de aprendizagem

Figura 28 - Relação entre os valores obtidos de previsão e os valores reais, bem como a matriz de confusão

Figura 29 - Cálculo para a visualização das curvas de aprendizagem

Figura 30 - Importação das bibliotecas necessárias, importação do dataset, dividido em conjuntos de teste e treino e padronização dos dados, com o pre-processamento

■ 5. Aprendizagem Supervisionada:

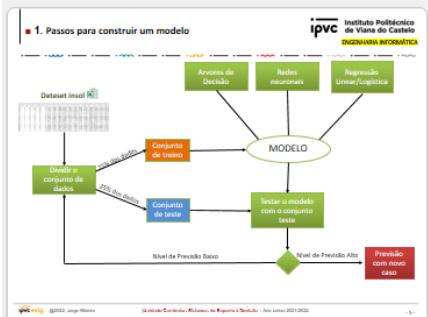
Criação de Modelo classificador/Previsão:

Árvores de Decisão; Regressão Linear e Redes Neuronais Artificiais

Exemplo 3



1



5

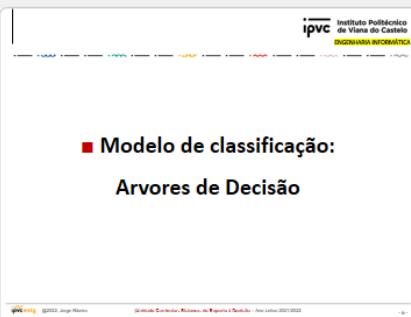
O objetivo deste primeiro ponto do trabalho prático TP2 B é desenvolver e explorar modelos de classificação em Python, os modelos a serem explorados são:

- 1) Árvores de Decisão
- 2) Regressão Linear/Logística
- 3) Redes Neurais

Para treinar e testar os modelos desenvolvidos vai ser usado o dataset [Insol.csv](#), este dispõe de dados sobre um empresário e a informação se esta entrou em Insolvência ou não.

O código foi desenvolvido no Colab, que é um serviço disponível na nuvem da Google que permite o desenvolvimento de código em Python.

2



6

- **iniciae**
- 1. Passos para construir um modelo
- 2. Árvore de decisão
 - 2.1. Import das bibliotecas
 - 2.2. Import do Dataset e tratamento de dados
 - 2.3. Casos de treino e teste
 - 2.4. Previsão do modelo
 - 2.5. Visualização da Árvore de Decisão
 - 2.6. Guardar o modelo
 - 2.7. Previsão com um novo caso
- 3. Regressão Linear/Logística
 - 3.1. Import das bibliotecas
 - 3.2. Import do Dataset e tratamento de dados
 - 3.3. Casos de treino e teste
 - 3.4. Matriz de confusão
 - 3.5. Curva ROC
 - 3.6. Previsão de um novo caso
 - 3.7. Utilização da métrica f1-score

3



7

- Início**
- 4. Redes Neurais**
 - 4.1. Import de biblioteca
 - 4.2. Import do dataset e divisão casos de teste e treino
 - 4.3. Criar modelo e definir as layers
 - 4.4. Compilar e treinar o modelo
 - 4.5. Previsão do modelo e previsão de um novo caso
- 5. Bibliografia**

4



8

■ 5. Aprendizagem Supervisionada:

Criação de Modelo classificador/Previsão:

Árvores de Decisão; Regressão Linear e Redes Neuronais Artificiais

Exemplo 3

■ 3. Regressão Linear/Logística
3.2. Import do dataset e tratamento de dados

```
dataset = pd.read_csv('Insol.csv')  
  
Asim como para as árvores de decisão, foi feito o import do dataset Insol.csv e divididos os dados em features(x) e target(y).
```

■ 3. Regressão Linear/Logística
3.3. Casos de treino e teste

1º Modelo

Os dados foram divididos em 80% para o caso de treino e 20% para o caso de teste utilizando o `train_test_split` do scikit-learn. De seguida foi treinado o modelo e calculada a sua `accuracy` tendo em conta os casos de teste. A precisão do modelo é de 81%, como pretendemos uma precisão maior vamos realizar uma nova divisão de dados e respetivo treino do modelo.

```
# dividir um caso de teste e treino Calcular modelo  
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)  
  
model = LogisticRegression(classifier='liblinear', random_state=42)  
model.fit(x_train, y_train)  
y_pred = model.predict(x_test)  
print("Accuracy logistic regression classifier on test set: {:.2f}".format(model.score(x_test, y_test)))  
  
Accuracy of logistic regression classifier on test set: 0.81
```

■ 3. Regressão Linear/Logística
3.3. Casos de treino e teste

2º Modelo

Para este segundo modelo foram atribuídos 70% dos dados para o caso de treino e 30% para o caso de teste e feita uma distribuição aleatória dos mesmo. Como podemos verificar a precisão do modelo aumentou para 84%.

```
# dividir os casos de teste e treino Calcular modelo  
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state=42)  
  
model = LogisticRegression(classifier='liblinear', random_state=42)  
model.fit(x_train, y_train)  
y_pred = model.predict(x_test)  
print("Accuracy logistic regression classifier on test set: {:.2f}".format(model.score(x_test, y_test)))  
  
Accuracy of logistic regression classifier on test set: 0.84
```

Uma vez que a precisão do modelo é alta vamos guardar o modelo num ficheiro.

```
# salvar o modelo num ficheiro com o pickle  
filename = "model_logreg.pkl"  
pickle.dump(model, open(filename, 'wb'))
```

■ 3. Regressão Linear/Logística
3.4. Matriz da confusão

A matriz da confusão correspondente ao segundo modelo criado é a seguinte. Analisando a matriz, de 658 casos em que a empresa não entra em insolvença o modelo previu que 7 entram em insolvença. De 164 casos em que a empresa entra em insolvença o modelo previu que 127 não entram em insolvença.

	Actual Non Insolvente	Actual Insolvente
Non Insolvente	651	7
Insolvente	164	127

17

■ 3. Regressão Linear/Logística
3.5. Curva ROC

A curva ROC (Receiver Operating Characteristic Curve) é uma representação gráfica para o desempenho do modelo. Para calcular esta curva utilizamos o modelo e o caso de treino.

```
# carregar o modelo  
model = pickle.load(open('filename', 'rb'))  
  
# carregar o novo caso  
new = pd.read_csv('Novo_Testes.xls')  
new = new.drop('Insolvente', axis=1) #remover a tabela Insolvenca  
  
new['proba'] = model.predict_proba(new)[::,1]  
new['proba'] = new['proba'].apply(lambda x: 1 - x)  
  
new[['proba']]
```

Receiver operating characteristic

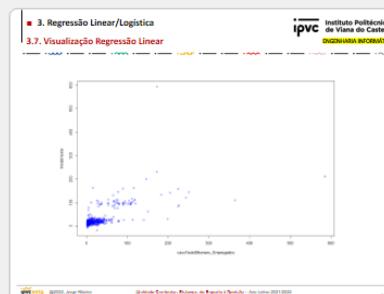
18

■ 3. Regressão Linear/Logística
3.6. Previsão com um novo caso

Comparar por carregar o modelo com melhor precisão que foi guardado no ficheiro `model_logreg.pkl`. De seguida importamos o caso `Novo_Testes.xls` que contém dados de uma nova empresa para previsão e submetemos esses dados ao modelo. A previsão é de que a empresa não entrará em insolvença.

```
# carregar o modelo  
model = pickle.load(open('filename', 'rb'))  
  
# importar novo caso  
new = pd.read_csv('Novo_Testes.xls')  
new = new.drop('Insolvente', axis=1) #remover a tabela Insolvenca  
  
new['proba'] = model.predict_proba(new)[::,1]  
new['proba'] = new['proba'].apply(lambda x: 1 - x)  
  
new[['proba']]
```

19



20

■ Modelo de classificação:
Redes Neuronais

■ 5. Aprendizagem Supervisionada:

Criação de Modelo classificador/Previsão:

Árvores de Decisão; Regressão Linear e Redes Neuronais Artificiais

Exemplo 3

4.1. Import de bibliotecas

```
Para as redes neuronais artificiais fomos importando usando começamos por fazer os seguintes imports. Desde as librarias numpy, pandas e seaborn que são librarías bastante utilizadas no Python. Do scikit-learn importámos os métodos:  
• train_test_split que é usado para fazer a divisão do dataset em caso de treino e caso de teste.  
• StandardScaler usado para criar o modelo
```

```
[1]: #imports  
import keras  
import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
from sklearn.model_selection import train_test_split  
from sklearn.preprocessing import StandardScaler
```

Para criar a rede neuronal foi utilizado o Keras, respetivamente os seguintes imports

```
[2]: #imports  
#Cria a rede Neural Artificial  
from keras.models import Sequential  
from keras.layers import Dense  
from keras import losses
```

4.2. Import do dataset e divisão casos de teste e treino

```
● importar o dataset  
dataset = pd.read_csv('insolencia.csv')  
X = dataset.drop(['insolencia'], axis=1)  
y = dataset['insolencia'  
para dataset insolvencia  
anteriormente a ser calculado
```

```
df = dataset.method_NK200e.head(10)  
df[['R�TIFICO', ... , 'Probabilidade_por_trabalhador', 'Certifica莽ao_legal', 'contas']]  
0 1 14 ... 0.405264 1  
1 1 14 ... 1.059424 1  
2 5 14 ... 1.059424 1  
3 5 14 ... 1.059424 1  
4 1 13 ... 1.786655 1  
  
233 5 14 ... 1.044723 0  
234 5 14 ... 1.059424 0  
235 5 14 ... 1.059424 0  
236 5 14 ... 1.059424 0  
237 1 13 ... 0.244805 0  
  
2788 rows × 14 columns
```

```
[3]: #dividir em conjunto de treino e conjunto de teste  
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

4.3. Criar modelo e definir as layers

```
Os modelos no Keras são definidos como um conjunto de layers, como tal começamos por criar um modelo Sequential.
```

```
model = Sequential()
```

E de seguida definimos as layers que queremos adicionar ao modelo. A primeira layer contém o parâmetro `input_layer` que deve corresponder ao número de features, no nosso caso temos 32 tabelas definidas como features, logo o nosso `input_layer` é 32.

```
[4]:  
#adicionar a Input layer e a primeira hidden layer  
model.add(Dense(32, input_dim=32, activation='relu'))  
model.add(Dense(32, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))
```

Para este exemplo criamos apenas três layers.

4.4. Compilar e treinar o modelo

```
As propriedades definidas para compilar a rede neuronal artificial foram as seguintes:
```

```
[5]: #compilando o nn  
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

Por fim treinamos o modelo com os nossos casos de treino.

```
model.fit(X_train, y_train, epochs=100, validation_data=(X_test, y_test))
```

Epoch	Accuracy
0	0.7682
1	0.7682
2	0.7682
3	0.7682
4	0.7682
5	0.7682
6	0.7682
7	0.7682
8	0.7682
9	0.7682
10	0.7682
11	0.7682
12	0.7682
13	0.7682
14	0.7682
15	0.7682
16	0.7682
17	0.7682
18	0.7682
19	0.7682
20	0.7682
21	0.7682
22	0.7682
23	0.7682
24	0.7682
25	0.7682
26	0.7682
27	0.7682
28	0.7682
29	0.7682
30	0.7682
31	0.7682
32	0.7682
33	0.7682
34	0.7682
35	0.7682
36	0.7682
37	0.7682
38	0.7682
39	0.7682
40	0.7682
41	0.7682
42	0.7682
43	0.7682
44	0.7682
45	0.7682
46	0.7682
47	0.7682
48	0.7682
49	0.7682
50	0.7682
51	0.7682
52	0.7682
53	0.7682
54	0.7682
55	0.7682
56	0.7682
57	0.7682
58	0.7682
59	0.7682
60	0.7682
61	0.7682
62	0.7682
63	0.7682
64	0.7682
65	0.7682
66	0.7682
67	0.7682
68	0.7682
69	0.7682
70	0.7682
71	0.7682
72	0.7682
73	0.7682
74	0.7682
75	0.7682
76	0.7682
77	0.7682
78	0.7682
79	0.7682
80	0.7682
81	0.7682
82	0.7682
83	0.7682
84	0.7682
85	0.7682
86	0.7682
87	0.7682
88	0.7682
89	0.7682
90	0.7682
91	0.7682
92	0.7682
93	0.7682
94	0.7682
95	0.7682
96	0.7682
97	0.7682
98	0.7682
99	0.7682
100	0.7682

25

4. Redes Neuronais

4.5. Precisão do modelo e previsão de um novo caso

```
Uma vez que a accuracy da rede neuronal é aproximadamente 86% vamos utilizar para fazer a previsão de um novo caso.
```

```
[1]: accuracy = model.evaluate(X, y, verbose=0)  
print(accuracy)
```

Foram importados novos dados, `Nova_teste.xls`, relativos a um empresa para ser calculada a probabilidade desta entrar em insolvença. Segundo o modelo de redes neuronais criado a probabilidade da empresa entrar em insolvença é nula.

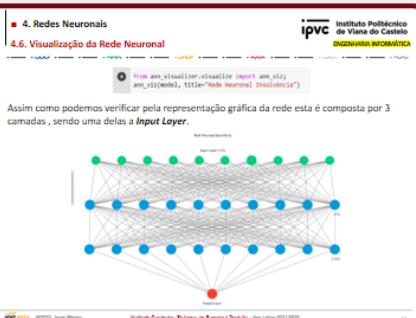
```
[2]: #carregar novo dado  
dataset = pd.read_excel('Nova_teste.xls')  
X_novo = dataset.drop(['insolencia'], axis=1)  
  
y_novo = dataset['insolencia']  
  
#prever  
y_novo[0] == 0  
print("Probabilidade de Nao Insolvencia: ", probabilidade[0][0])  
print("Probabilidade de Insolvencia: ", probabilidade[0][1])
```

26

4. Redes Neuronais

4.6. Visualização da Rede Neuronal

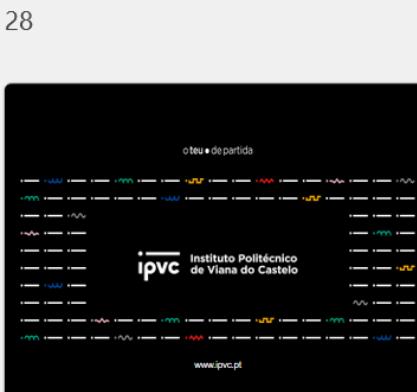
```
Assim como podemos verificar na representação gráfica da rede esta é composta por 3 camadas, sendo uma delas a Input Layer.
```



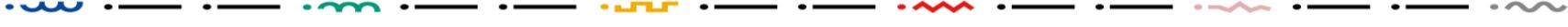
27

5. Bibliografia

- Árvores de decisão:
<https://heartbeat.fritz.ai/decision-tree-classification-in-python-with-scikit-learn-245507adab9>
- Regressão Linear/Logística:
<https://towardsdatascience.com/building-a-logistic-regression-in-python-step-by-step-becc445d6c9d>
- Redes neurais:
<https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/>
- Documentos e weblinks disponibilizados pelo Professor na plataforma Moodle da Unidade Curricular.



Conteúdos

-  —
- 1. Introdução
 - 1.1. Objetivos
- 2. Linguagem de Programação Python
- 3. Jupyter Notebook

4. Análise de dados

- 4.1. Análise de Estatísticas Simples
- 4.2. Análise de Distribuições Estatísticas
- 4.3. Análise da Correlação e Intervalo de Confiança

5. Aprendizagem Supervisionada:

Criação de Modelo classificador/Previsão:

- Árvores de Decisão;
- Regressão Linear
- Redes Neuronais Artificiais

6. Aprendizagem Não supervisionada:

Clustering – K-Means

7. Bibliografia e Referências Web

Bibliografia

- <https://www.kaggle.com/hldergonalves/insolvencia>
- <https://www.kaggle.com/>
- <https://medium.com/fintechexplained/everything-about-python-from-beginner-to-advance-level-227d52ef32d2>
- <https://www.udemy.com/course/python-for-statistical-analysis/>
- <https://www.coursera.org/specializations/statistics-with-python>

■ 6. Bibliografia e Referências Web

[1]. Python Software Foundation. 2021. Python Downloads. Disponível em:
<https://www.python.org/downloads/>;

[2]. Anaconda Inc. 2021. Anaconda Home Page. Disponível em:
<https://www.anaconda.com/>;

[3]. Project Jupyter. 2021. About Us. Disponível em:
<https://jupyter.org/about>;

[4]. NumFOCUS. 2021. pandas. Disponível em:
<https://pandas.pydata.org/>;

■ 6. Bibliografia e Referências Web

[5]. NumPy & The SciPy community. 2021. What is NumPy?. Disponível em:
<https://numpy.org/doc/stable/user/whatisnumpy.html>;

[6]. SciPy developers. 2021. SciPy library. Disponível em:
<https://www.scipy.org/scipylib/index.html>;

[7]. John Hunter, Darren Dale, Eric Firing, Michael Droettboom and the Matplotlib development team. 2021. User's Guide. Disponível em:
<https://matplotlib.org/stable/users/index.html>;

[8]. Michael Waskom. 2020. seaborn: statistical data visualization. Disponível em:
<https://matplotlib.org/stable/users/index.html>;