awari.

Demo Day Awari

Marcos Felipe A. Barroso

awari.

Δ Problematização: Programa para ajudar interessados em investimentos em ações que baseiam-se na análise fundamentalistas de empresas a escolher uma empresa de um amo da economia para aquisição.

Δ Análise Fundamentalista: Metodologia que utiliza dados financeiros, contábeis das empresas que possuem Capital Aberto e negociam partes de sua empresa em bolsa de valores disponíveis para o público em geral.

Δ Valluation: Metodologia para avaliar resultados futuros utilizando premissas e histórico de dados para projeções.

Métrica padrão para o desenvolvimento do Modelo:

CAGR

CAGR é a sigla para *Compound Annual Growth Rate*, ou **taxa de crescimento** anual composta, em português.
Trata-se da taxa de retorno necessária para que um **investimento** parta do seu saldo inicial e chegue até um determinado saldo final.
Ou seja: é a taxa de retorno medida durante o período do investimento, considerando que os lucros foram **constantes** e reaplicados a cada ciclo.

Fórmula: CAGR = (Valor final / Valor inicial) ^ (1/qtd anos) -1

Liquidez	Dívida	Fluxo_de_Caixa
Liquid. corrente	Dívida bruta/PL	FCI/LL
Liquid. imediata	Dívida líq./Ebitda	CAPEX/LL
Capital de giro	EF	CAPEX/FCO
P / Cap. de Giro	ECP	
P / AC Líq.		
Giro dos Ativos		
PME		
PMR		
PMP		
CICLO OPER.		
CICLO DE CX		
11	4	3

Indicadores_Ec_Fin	
Margem bruta	EV/EBIT
Margem Oper.	EV/EBITDA
Margem líquida	Valorização
LPA	Negócios diário
P/L	Volume diário
VPA	Valor de Mercado
P/VP	Valor da Firma
P/EBIT	
P/EBITDA	
P/Ativos	
Ebitda	
Margem Ebitda	
PSR	
ROE	
ROA	
ROIC	
23	

Valluation	
Beta %	1.33
g (Cresc.) %	2.86
IR %	34
N	5
Selic %	6.25
IPCA %	3.25
Ke %	14.24
WACC %	18.41
FCxDesc	R$ 1,277
Valor_Perpetuidade	R$ 2,105
Valor_empresa	R$ 3,382
FCFE	R$ 277
FCFF	R$ 2,400
13	

Fórmula: CAGR = (Valor final / Valor inicial) ^ (1/qtd anos) -1

awari.

Problematização | Coleta de Dados | Preparação de Dados | Análise Exploratória | Modelagem | Comunicação e Visualização

Coleta de Dados em sites públicos (Csv)

Template para padronização das infos financeiras

Geração de base Csv para upload nc Jupyter.

jupyter

Fontes auxiliares:

https://br.financas.yahoo.com/?guccounter=1

https://www.oceans14.com.br/

http://fundamentus.com.br/index.php

https://statusinvest.com.br/

https://www3.bcb.gov.br/expectativas/publico

Empresas selecionadas (Setor de varejo) – Cerca de 6% do valor de mercado da B3

Magalu | viavarejo | RENNER | Guararapes | riachuelo | AREZZO

awari.

Recursos utilizados:

Template para padronização e projeção

M11 | =SEERRO(((K11/B11)^(1/10))-1;0)

	A	B	C	D	E	F	G
1	0	1	2	3	4	5	6
2	**Ano**	**2011**	**2012**	**2013**	**2014**	**2015**	**2016**
3	N_de_acoes	185,714,286	200,000,000	184,126,984	179,452,055	21,885,522	21,969,697
4	Vlr_acao	9.54	12.15	7.55	7.71	17.65	106.17
5	Receita_liquida	6419	7665	8109	9779	8978	9508
6	Custos	-4275	-5146	-5825	-7086	-6399	-6586
7	Resultado_bruto	2144	2519	2284	2693	2579	2922
8	Despesas_oper	-1930	-2370	-1908	-2201	-2239	-2341
9	Res_operacional	214	149	376	492	340	581
10	Res_Financeiro	-165	-172	-243	-360	-486	-503
11	IR_e_CSSL	-36	17	-17	-1	81	9
12	Lucro_liquido	13	-6	116	131	-65	87
13	Depreciacao	87.00	93.00	101.00	113.00	125.00	134.00
14	Ativo_Total	4877	5664	4713	5290	5588	6187
15	Ativo_Circulante	3567	4069	2921	3395	3360	3919
16	Caixa	248	598	771	863	1115	1418
17	Recebiveis	1927	2104	530	618	435	581
18	Estoque	1264	1068	1251	1472	1353	1596
19	Outros_AC	126	298	368	441	457	323
20	Ativo_Nao_Circ	1310	1594	1791	1894	2228	2267
21	Imobilizado	489	575	540	566	578	560
22	Intangivel	448	440	481	488	506	513
23	Outros_ANC	371	578	770	839	1142	1194
24	Passivo_Total	4877	5664	4713	5290	5588	6187
25	Passivo_Circ	3167	3607	2527	2831	2874	3672

Label enconder | MGLU | VVAR | RENNER | GUAR | AREZZO | Indicadores Valluation

Geração do CSV

Problematização | Coleta de Dados | Preparação de Dados | Análise Exploratória | Modelagem | Comunicação e Visualização

awari.

Bibliotecas e Recursos utilizados:

pandas

NumPy

Analise Empresa - Magazine Luiza 2011 ~ 2025

```
In [1]:  1 import pandas as pd
```

```
In [3]:  1 mglu.head()
```

```
In [22]:  1 mglu_ind.drop('Status_2', axis=1, inplace=True)
```

Alterando a coluna alvo (Status) para 0 e 1.

```
In [19]:  1 mglu_hist['Status_1'] = mglu_hist['Status_1'].replace('CAGR_BOM',0)
          2 mglu_hist['Status_1'] = mglu_hist['Status_1'].replace('CAGR_RUIM',1)
          3 mglu_hist.head()
```

```
In [20]:  1 mglu_proj['Status_2'] = mglu_proj['Status_2'].replace('CAGR_BOM',0)
          2 mglu_proj['Status_2'] = mglu_proj['Status_2'].replace('CAGR_RUIM',1)
          3 mglu_proj.head()
```

```
In [21]:  1 mglu_ind['Status'] = mglu_ind['Status'].replace('CAGR_BOM',0)
          2 mglu_ind['Status'] = mglu_ind['Status'].replace('CAGR_RUIM',1)
          3 mglu_ind.head()
```

Tratamento de valores inconsistentes

```
In [24]:  1 mglu_hist.index
Out[24]: RangeIndex(start=0, stop=40, step=1)

In [25]:  1 mglu_hist.isnull().sum()
```

Análise Descritiva

```
In [26]:  1 mglu_hist.describe()
```

Out[26]:

	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
count	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00	40.00
mean	4643757.52	5001092.38	4604142.64	4487435.54	548300.34	550512.95	4769754.76	4770913.38	40485777.37	163754546.11
std	29363860.89	31622599.53	29112875.52	28373677.39	3460217.20	3473509.03	30157093.93	30162196.71	256032994.92	1035645196.51
min	-4275.00	-5146.00	-5825.00	-7086.00	-6399.00	-6586.00	-8378.00	-11053.00	-14332.00	-21657.00
25%	12.13	74.00	41.64	131.00	184.25	125.54	5.00	185.52	-10.75	300.99
50%	455.50	588.00	544.50	604.50	620.00	644.50	826.50	1008.00	2055.00	2753.00
75%	1424.75	1685.75	1531.00	1730.50	1933.25	1952.75	2095.00	2490.50	5649.00	7373.75
max	185714285.70	200000000.00	184126984.10	179452054.80	21885521.89	21969696.97	190731707.30	190764331.20	1619298246.00	6550000000.00

Dividindo do DF

Bibliotecas e Recursos utilizados:

SweetViz

New in 2.0!
- Jupyter, Google Colab (& other notebook) integration
- Report size scaling
- Vertical layout

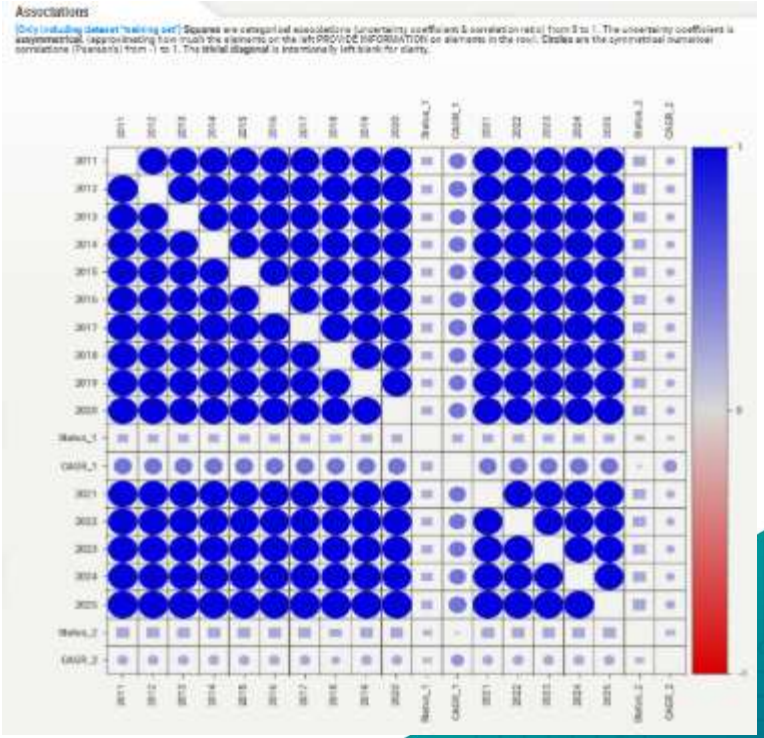
Usando o Sweetviz

```
In [7]: import sweetviz as sv #disclosure: I'm the author
In [8]: my_report = sv.analyze(eglu)
Done! Use 'show' commands to display/save.   [100%] [00:42 -> (00:00 left)]
In [9]: msk = np.random.rand(len(eglu)) < 0.8
        train = eglu[msk]
        test = eglu[~msk]

        train.head() = end
In [10]: my_report = sv.compare([train, 'training set'], [test, 'testing set'])
         my_report.show_html()
Done! Use 'show' commands to display/save.   [100%] [00:02 -> (00:00 left)]
Report SWEETVIZ_REPORT.html was generated! NOTEBOOK/COLAB USERS: the web browser MAY not pop up, regardless, the report is saved in your notebook/colab files.
```

SweetViz 2.0.4

Get updates, docs & report issues here

	training set	testing set
ROWS	30	10
DUPLICATES	0	0
RAM	1.3 kB	0.4 kB
FEATURES	20	20
CATEGORICAL	3	3
NUMERICAL	17	17
TEXT	0	0



Bibliotecas e Recursos utilizados:

PANDAS PROFILING

Usando o Pandas - Profiling

```
In [285]: 1 #pip install pandas-profiling
```

```
In [*]: 1 # Visualização do DF com informações sobre estatistica inclusive.
        2 import pandas_profiling
        3
        4 pandas_profiling.ProfileReport(mglu)
```

Summarize dataset: 73% 24/33 [00:14<00:00, 10.48it/s, Calculate phi_k correlation]

Pandas Profiling Report

Overview | Variables | Interactions | Correlations | Missing values | Sample

Overview | Warnings 20 | Reproduction

Dataset statistics

Number of variables	20
Number of observations	46
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	8.0 KiB
Average record size in memory	163.2 B

Variable types

Categorical	1
Numeric	17

2011
Real number (ℝ)

HIGH CORRELATION

Distinct	38	Mean	4643757.519
Distinct (%)	95.0%	Minimum	-4275
Missing	0	Maximum	185714265.7
Missing (%)	0.0%	Zeros	0
Infinite	0	Zeros (%)	0.0%
Infinite (%)	0.0%	Memory size	448.0 B

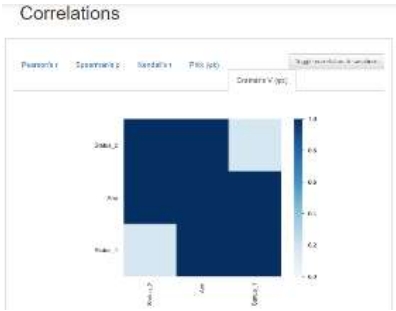
Interactions

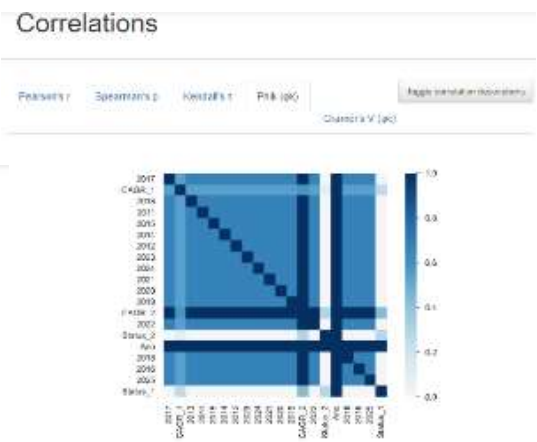
2011	2011
2012	2012
2013	2013
2014	2014
2015	2015
2016	2016
2017	2017
2018	2018
2019	2019
2020	2020
CAGR_1	CAGR_1
2021	2021
2022	2022
2023	2023
2024	2024
2025	2025
CAGR_2	CAGR_2

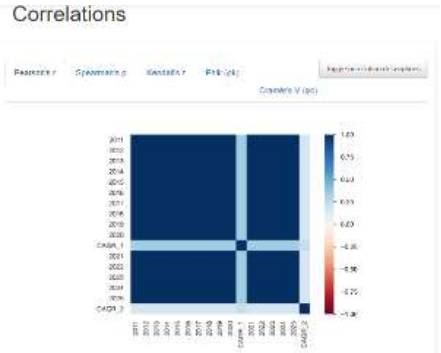
awari.

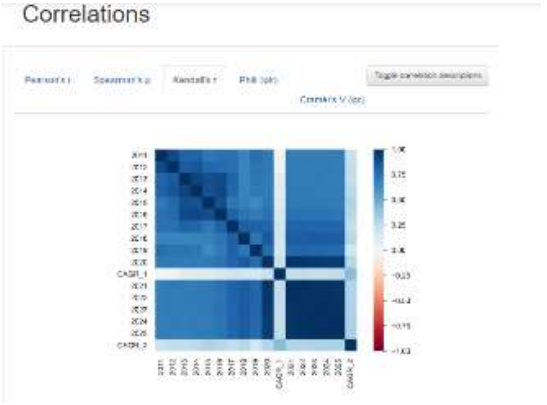
Bibliotecas e Recursos utilizados:

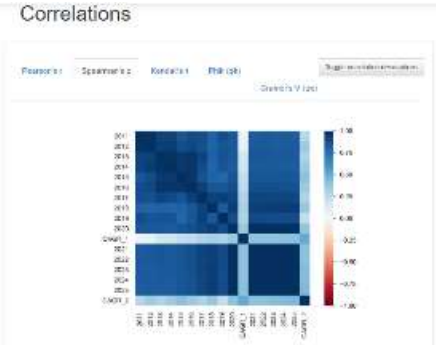
PANDAS PROFILING











Algoritmos de ML utilizados:

- Naïve Bayes;
- Árvores de decisão;
- Random Forest;
- KNN;
- Regressão Logística.

Variável Alvo (Y)

CAGR

- *Divisão de dados entre Treinamento e Teste pelo método Hold-Out (70%/30%);*
- *Trenamento dos algoritmos com e sem escalonamento de valores;*

Bibliotecas e Recursos utilizados:

Problematização | Coleta de Dados | Preparação de Dados | Análise Exploratória | Modelagem | Comunicação e Visualização

awari.

Label Enconder para "hit" do Y

Resultados Sem escalonamento de valores

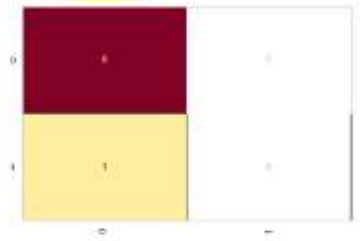
```
In [91]: from sklearn.preprocessing import LabelEncoder
         from sklearn import preprocessing
         from sklearn import utils
         lab_enc = preprocessing.LabelEncoder()
         training_scores_encoded = lab_enc.fit_transform(Y_ren_ind_treinamento)

In [92]: print(training_scores_encoded)
         print(utils.multiclass.type_of_target(Y_ren_ind_treinamento))
         print(utils.multiclass.type_of_target(Y_ren_ind_treinamento.astype('int')))
         print(utils.multiclass.type_of_target(training_scores_encoded))
```

```
In [325]: from sklearn.model_selection import TimeSeriesSplit
          from sklearn.naive_bayes import GaussianNB
          from yellowbrick.classifier import ClassificationReport
          from yellowbrick.datasets import load_occupancy

In [213]: from yellowbrick.classifier import ConfusionMatrix

In [214]: cm = ConfusionMatrix(naive_mglu_proj_data)
          cm.fit(X_mglu_proj_treinamento, Y_mglu_proj_treinamento)
          cm.score(X_mglu_proj_teste, Y_mglu_proj_teste)
Out[214]: 0.6
```

```
In [215]: print(classification_report(Y_mglu_proj_teste, previsoes))
```

	precision	recall	f1-score	support
0	0.62	0.55	0.75	19
1	0.00	0.00	0.00	11
accuracy			0.60	30
macro avg	0.31	0.47	0.38	30
weighted avg	0.39	0.60	0.47	30

```
In [389]: from yellowbrick.classifier import ConfusionMat
          cm = ConfusionMatrix(arvore_mglu_ind)
          cm.fit(X_mglu_ind_treinamento, Y_mglu_ind_trein
          cm.score(X_mglu_ind_teste, Y_mglu_ind_teste)
Out[389]: 0.8571428571428571
```



```
In [390]: print(classification_report(Y_mglu_ind_teste, previsoes))
```

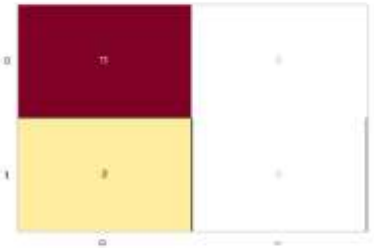
	precision	recall	f1-score	support
0	0.86	1.00	0.92	6
1	0.00	0.00	0.00	1
accuracy			0.86	7
macro avg	0.43	0.50	0.46	7
weighted avg	0.73	0.86	0.79	7

```
In [420]: from yellowbrick.classifier import ConfusionMatrix
          cm = ConfusionMatrix(random_forest_mglu_ind)
          cm.fit(X_mglu_ind_treinamento, Y_mglu_ind_treinamento)
          cm.score(X_mglu_ind_teste, Y_mglu_ind_teste)
Out[420]: 0.8461538461538461
```

```
In [421]: print(classification_
```

	precision
0	0.85
1	0.00
accuracy	
macro avg	0.42
weighted avg	0.72

```
In [450]: from yellowbrick.classifier import ConfusionMatrix
          cm = ConfusionMatrix(knn_mglu_ind)
          cm.fit(X_mglu_ind_treinamento, Y_mglu_ind_treinamento)
          cm.score(X_mglu_ind_teste, Y_mglu_ind_teste)
Out[450]: 0.8461538461538461
```



```
In [451]: print(classification_report(Y_mglu_ind_teste, previsoes))
```

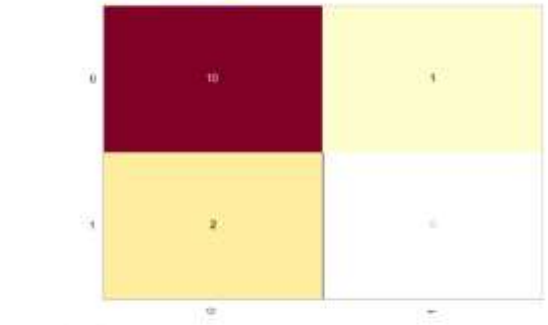
	precision	recall	f1-score	support
0	0.85	1.00	0.92	11
1	0.00	0.00	0.00	2
accuracy			0.85	13
macro avg	0.42	0.50	0.46	13
weighted avg	0.72	0.85	0.78	13

Escalonamento dos valores

Resultados Com escalonamento de valores

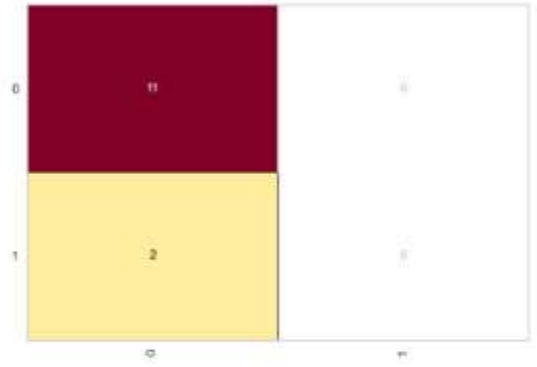
Base Histórica (2011-2020)

```
In [215]: from yellowbrick.classifier import ConfusionMatrix
          cm = ConfusionMatrix(knn_mglu_ind)
          cm.fit(X_mglu_ind_treinamento, Y_mglu_ind_treinamento)
          cm.score(X_mglu_ind_teste, Y_mglu_ind_teste)
Out[215]: 0.7692307692307693
```




```
In [216]: print(classification_report(Y_mglu_ind_teste, previsoes))
              precision    recall  f1-score   support

           0       0.85      0.91      0.87        11
           1       0.00      0.00      0.00         2

    accuracy                           0.77        13
```

```
In [254]: cm = ConfusionMatrix(logistic_mglu_ind)
          cm.fit(X_mglu_ind_treinamento, Y_mglu_ind_treinamento)
          cm.score(X_mglu_ind_teste, Y_mglu_ind_teste)
Out[254]: 0.8461538461538461
```

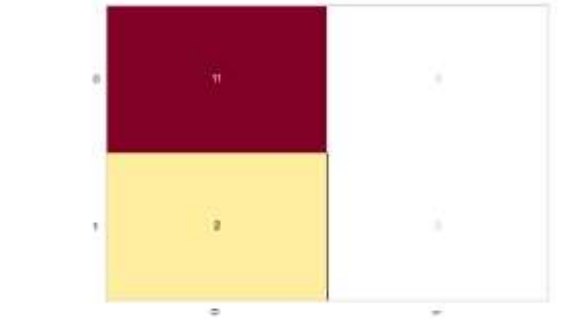



```
In [255]: print(classification_report(Y_mglu_ind_teste, previsoes))
              precision    recall  f1-score   support

           0       0.85      1.00      0.92        11
           1       0.00      0.00      0.00         2

    accuracy                           0.85        13
```

- **Conclusão:**

Diante, dos 5 algoritmos utilizados, que poderiam ser usados para replicar a performance do Cagr diante de uma base histórica das empresas em estudo seriam Random Forest e KNN.

```
In [185]: from yellowbrick.classifier import ConfusionMatrix
          cm = ConfusionMatrix(random_forest_mglu_ind)
          cm.fit(X_mglu_ind_treinamento, Y_mglu_ind_treinamento)
          cm.score(X_mglu_ind_teste, Y_mglu_ind_teste)
Out[185]: 0.8461538461538461
```




```
In [186]: print(classification_report(Y_mglu_ind_teste, previsoes))
              precision    recall  f1-score   support

           0       0.85      1.00      0.92        11
           1       0.00      0.00      0.00         2

    accuracy                           0.85        13
```

awari.

Relatório comparativo

Relatório Final - Valluation / Simulador

In [1]:
```
import pandas as pd
import numpy as np
from pandas_datareader import data as wb
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
pd.set_option('display.float_format', '{:.2f}'.format)
#Comando para evitar notações científicas, nº grandes
```

In [2]:
```
# Importando CSV
dados = pd.read_csv('Ind_TODAS_VALUATION.csv',error_bad_lines=False, delimiter= ';',
                    encoding='ISO-8859-1',thousands=".", decimal=",")
```

In [3]:
```
dados
```

Out[3]:

	Indicador	MGLU	AREZZO	GUAR	RENNER	VVAR
0	FCxDesc	1277.36	-106.41	-2337.82	9502.47	5065.12
1	Valor_Perpetuidade	2104.98	-1126.32	-14783.22	52336.51	28288.56
2	Valor_empresa	3382.34	-1232.73	-17121.04	61838.97	33353.69
3	FCFE	277.47	541.00	2987.00	3970.35	2447.47
4	FCFF	2399.97	160.72	-314.46	1542.63	7087.01

Transpondo a base para gráficos

In [4]:
```
dados2 = dados.T
```

In [5]:
```
dados2.columns = dados2.iloc[0]
```

In [6]:
```
dados2.drop('Indicador',axis=0, inplace=True)
```

In [7]:
```
dados2
```

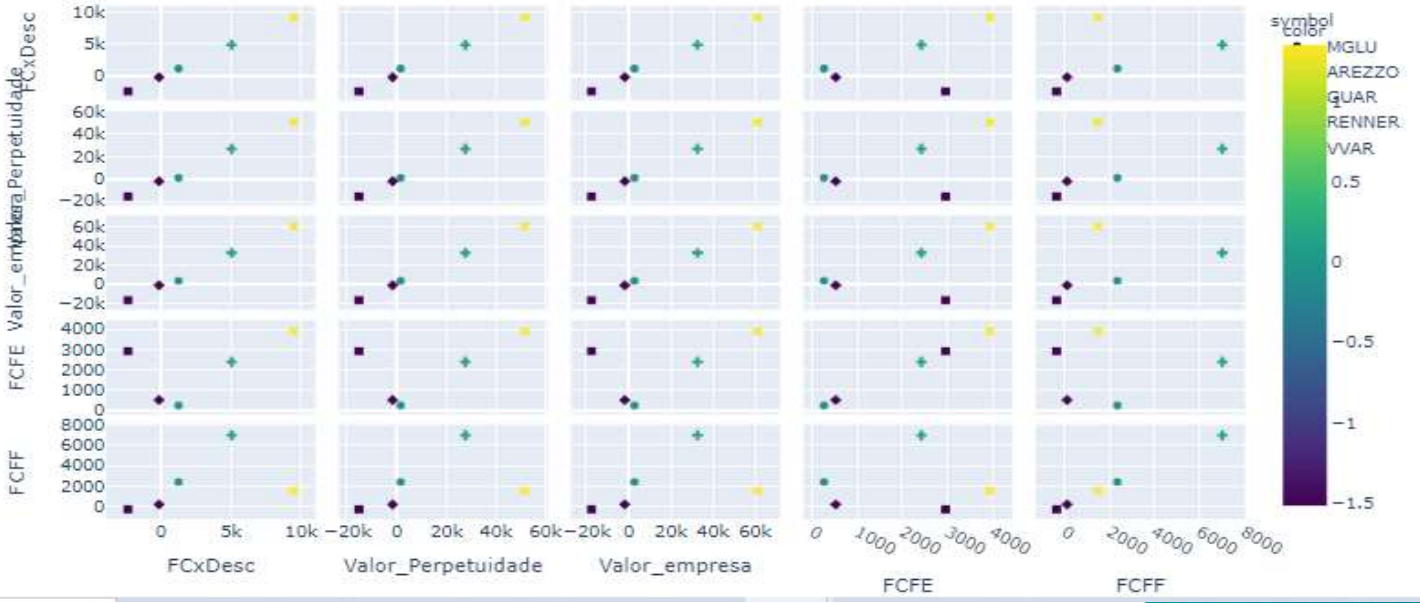
Out[7]:

Indicador	FCxDesc	Valor_Perpetuidade	Valor_empresa	FCFE	FCFF
MGLU	1277.36	2104.98	3382.34	277.47	2399.97
AREZZO	-106.41	-1126.32	-1232.73	541.00	160.72
GUAR	-2337.82	-14783.22	-17121.04	2987.00	-314.46
RENNER	9502.47	52336.51	61838.97	3970.35	1542.63
VVAR	5065.12	28288.56	33353.69	2447.47	7087.01

Relatório comparativo

Gráfico dos indicadores de Valluation das empresas

In [8]:
```
1 grafico = px.scatter_matrix(dados2, dimensions=['FCxDesc', 'Valor_Perpetuidade', 'Valor_empresa', 'FCFE', 'FCFF'], symbol='
2 grafico.show()
```



Problematização ✓ | Coleta de Dados ✓ | Preparação de Dados ✓ | Análise Exploratória ✓ | Modelagem ✓ | Comunicação e Visualização

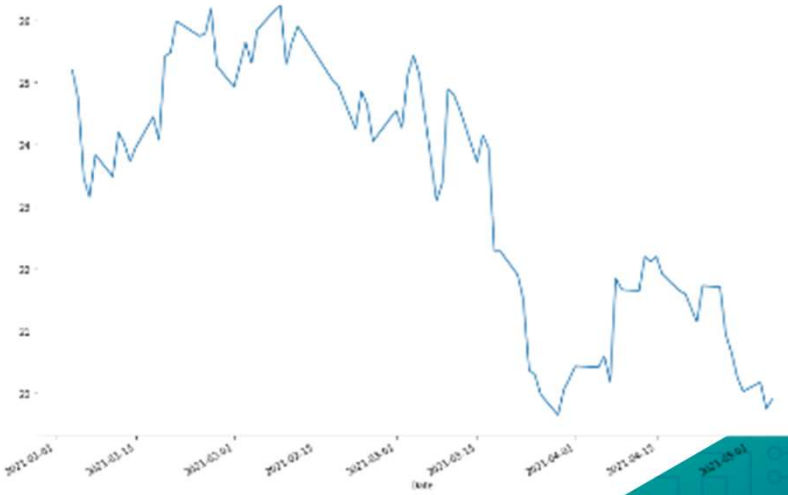
awari.

Coleta do desempenho de Valor da ação

In [9]:

```
data_inicial = "01/01/2021"
data_final = "05/05/2021"

empresas_df = pd.read_excel("EMPRESAS_COT.xlsx")
display(empresas_df)

for empresa in empresas_df['Empresas']:
    print(f"{empresa}:")
    df = web.DataReader(f'{empresa}.SA', data_source='yahoo', start=data_inicial, end=data_final)
    display(df)
    df["Adj Close"].plot(figsize=(15, 10))
    plt.show()
```

Date	High	Low	Open	Close	Volume	Adj Close
2021-01-04	25.58	24.87	25.26	25.20	25706100.00	25.20
2021-01-05	25.18	24.34	25.10	24.76	25431900.00	24.76
2021-01-06	24.66	23.42	24.65	23.46	51799000.00	23.46
2021-01-07	23.85	22.95	23.64	23.16	42146600.00	23.16
2021-01-08	24.30	23.02	23.19	23.84	43988100.00	23.84



Simulador:

1. Informe o valor em R$ desejado para investimento

2. Informe o valor + atual da ação

In [10]:
```
investimento = 1000
acao_mglu = 19.90
acao_renner = 41.45
acao_arz = 79.45
acao_guar = 17.21
acao_vvar = 12.10
```

In [13]:
```
div_mglu = qtd_mglu * 0.28
div_renner = qtd_renner * 0.17
div_arz = qtd_arz * 0.03
div_guar = qtd_guar * 0.20
div_vvar = qtd_vvar * 0.01
```

In [14]:
```
resume = pd.DataFrame([['Qtd', qtd_mglu,qtd_renner,qtd_arz,qtd_guar,qtd_vvar],
                       ['Cotação R$', acao_mglu,acao_renner,acao_arz,acao_guar,acao_vvar],
                       ['Dividendos 21 R$', 0.28,0.17,0.03,0.20,0.01],
                       ['Div. Proj R$', div_mglu, div_renner, div_arz, div_guar, div_vvar]],
                       columns = ['Empresa','Mglu', 'Renner', 'Arezzo', 'Guar','Vvar'])
resume
```

Out[ ]:

	Empresa	Mglu	Renner	Arezzo	Guar	Vvar
0	Qtd	50.25	24.13	12.59	58.11	82.64
1	Cotação R$	19.90	41.45	79.45	17.21	12.10
2	Dividendos 21 R$	0.28	0.17	0.03	0.20	0.01
3	Div. Proj R$	14.07	4.10	0.38	11.62	0.83

Obrigado.

Contatos:

- Linkedin: https://www.linkedin.com/in/marcos-fandrade/
- Github: https://github.com/marcoscomp30
- Celular: (92) 98125-4571