Resultado do script:

Os 10 melhores ativos (ID) para investir (Rendimento considerado de 21/06/2021 a 21/06/2022): 812, 609, 805, 810, 803, 714, 1145, 520, 1042, 918.

Esses valores e seus rendimentos estão mostrados na tabela abaixo:

ID	Rendimentos [%]
812	29.760744
609	22.544026
805	22.462552
810	19.903089
803	18.847747
714	17.470657
1145	17.050732
520	16.126571
1042	14.863535
918	14.445243

Foram desconsiderados os ativos com menos que 15 dados coletados, devido ao alto risco, ou seja, pouca precisão do programa.

Script:

```
# Gabriel Prieto Paris
# 21/06/2022

from locale import normalize
from os import sep
import pandas as pd
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt
from pathlib import Path
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score
from datetime import datetime
from sklearn.tree import ExtraTreeRegressor

# Esse programa tem como objetivo encontrar os 10 melhores ativos para
investir.
# Realiza-se uma regressão linear nos dados coletados dos features em função
do tempo, encontrando a reta que mais se aproxima da curva feita por esses
dados.
# depois realiza-se uma regressão linear dos preços coletados em função das
features, encontrando a reta mais próxima da curva dos preços. Assim essa
reta é extrapolada para os dias futuros aos da medição.
# Por fim, calcula-se os rendimentos em porcentagem do dia 21/06/2021 até o
dia 21/06/2022, rankeando-os.
```

```
# LER ARQUIVO CSV
arquivo = pd.read_csv(r'C:\Users\gabri_7fr\Desktop\Exercicio
Sarpen\exercise_data.csv', sep=',')
# print(arquivo.head())

# VETOR COM O NOME DO ATIVO EM ORDEM CRESCENTE
id_ativo = sorted(set(arquivo["id"]))
# print(id_ativo)

# # DESCOMENTAR PARA SALVAR NOVOS ARQUIVOS
# # SEPARANDO OS DADOS DE CADA ATIVO EM ARQUIVOS CSV SEPARADAMENTE
# for i in id_ativo:
#     ativo = arquivo[arquivo["id"]==i]
#     # print(ativo)
#     ativo.to_csv(r'C:\Users\gabri_7fr\Desktop\Exercicio Sarpen\Ativos\out-
{}.csv'.format(i), index=False)

kk = -1
rend = np.zeros((len(id_ativo),1))
for k in id_ativo:
    kk = kk+1
    # LENDO O ARQUIVO DE CADA ATIVO
    tabela = pd.read_csv(r'C:\Users\gabri_7fr\Desktop\Exercicio
Sarpen\Ativos\out-{}.csv'.format(k), sep=',').fillna(0)
    # Retirando dados que tenham poucos pontos para análise, pois tem pouca
precisão
    if len(tabela)<15:
        continue
    print(k)
    dias = tabela["date"]
    # print(dias)

    # VETOR COM OS DIAS CORRIDOS
    t = np.zeros((len(dias),1))
    d1 = datetime.strptime(dias.loc[0], '%Y-%m-%d')
    for i in range(len(dias)):
        d2 = datetime.strptime(dias.loc[i], '%Y-%m-%d')
        t[i,0] =  abs((d2 - d1).days)
    # print(t)

    # QUANTIDADE DE DIAS ATÉ A DATA 21/06/2022
    hoje = datetime.strptime('2022-06-21', '%Y-%m-%d')
    quantidade_dias = abs((hoje - d1).days)+1
    # print(quantidade_dias)
    futuro = np.arange(start=0,stop=quantidade_dias) # Vetor de tempo desde
de o ínicio da coleta de dados

    # NORMALIZAÇÃO DOS VALORES
    del tabela["id"]
```

```
    del tabela["date"]
    tabela_norm = (tabela-tabela.mean())/tabela.std()
    # print(tabela_norm)

    # SEPARANDO ARGUMENTOS (X) DO QUE QUEREMOS CALCULAR (Y)
    x = tabela[['feature 1','feature 2', 'feature 3', 'feature 4', 'feature
5']]
    y = tabela['price']
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,
random_state=0) # Separando 20% dos pontos para teste
    # print(x_train)
    # print(x_test)

    # REALIZANDO A REGRESSÃO
    regressor = LinearRegression().fit(x_train,y_train) # treino com valores
normalizados por default, já que tem unidades diferentes/desconhecidas
    # print(regressor.coef_) # Coeficientes da equação price = c1*feature1 +
c2*feature2 + c3*feature3 + c4*feature4 + c5*feature5 + constante
    # print(regressor.intercept_)
    predicted = regressor.predict(x_test) # Valores de teste
    df = pd.DataFrame(y_test, predicted)
    dfr = pd.DataFrame({'Price':y_test, 'Previsto':predicted})
    # print(dfr) # Tabela de teste

    # NOVO DATAFRAME COM DIAS CORRIDOS SUBISTITUINDO A DATA DO ANO
    tabela.insert(0, 'Dias corridos', t)
    # print(tabela)

    # REGRESSÃO EM ÁRVORE DO FEATURE 1
    x1 = tabela['Dias corridos']
    y1 = tabela['feature 1']
    x_train_1, x_test_1, y_train_1, y_test_1 = train_test_split(t, y1,
test_size=0.2, random_state=0) # Separando 20% dos pontos para teste
    regressor_1 = LinearRegression().fit(x_train_1.reshape(-1,1),y_train_1)
    predicted_1 = regressor_1.predict(x_test_1) # Valores de teste
    # df1 = pd.DataFrame(y_test_1, predicted_1)
    # dfr1 = pd.DataFrame({'feature 1':y_test_1, 'Previsto':predicted_1})
    # print(dfr1) # Tabela de teste

    feature1 = regressor_1.predict(futuro.reshape(-1, 1))

    # REGRESSÃO EM ÁRVORE DO FEATURE 2
    x2 = tabela['Dias corridos']
    y2 = tabela['feature 2']
    x_train_2, x_test_2, y_train_2, y_test_2 = train_test_split(t, y2,
test_size=0.2, random_state=0) # Separando 20% dos pontos para teste
    regressor_2 = LinearRegression().fit(x_train_2.reshape(-1,1),y_train_2)
    predicted_2 = regressor_2.predict(x_test_2) # Valores de teste
    # df2 = pd.DataFrame(y_test_2, predicted_2)
```

```
    # dfr2 = pd.DataFrame({'feature 2':y_test_2, 'Previsto':predicted_2})
    # print(dfr2) # Tabela de teste

    feature2 = regressor_2.predict(futuro.reshape(-1, 1))

    # REGRESSÃO EM ÁRVORE DO FEATURE 3
    x3 = tabela['Dias corridos']
    y3 = tabela['feature 3']
    x_train_3, x_test_3, y_train_3, y_test_3 = train_test_split(t, y3,
test_size=0.2, random_state=0) # Separando 20% dos pontos para teste
    regressor_3 = LinearRegression().fit(x_train_3.reshape(-1,1),y_train_3)
    predicted_3 = regressor_3.predict(x_test_3) # Valores de teste
    # df3 = pd.DataFrame(y_test_3, predicted_3)
    # dfr3 = pd.DataFrame({'feature 3':y_test_3, 'Previsto':predicted_3})
    # print(dfr3) # Tabela de teste

    feature3 = regressor_3.predict(futuro.reshape(-1, 1))

    # REGRESSÃO EM ÁRVORE DO FEATURE 4
    x4 = tabela['Dias corridos']
    y4 = tabela['feature 4']
    x_train_4, x_test_4, y_train_4, y_test_4 = train_test_split(t, y4,
test_size=0.2, random_state=0) # Separando 20% dos pontos para teste
    regressor_4 = LinearRegression().fit(x_train_4.reshape(-1,1),y_train_4)
    predicted_4 = regressor_4.predict(x_test_4) # Valores de teste
    # df4 = pd.DataFrame(y_test_4, predicted_4)
    # dfr4 = pd.DataFrame({'feature 4':y_test_4, 'Previsto':predicted_4})
    # print(dfr4) # Tabela de teste

    feature4 = regressor_4.predict(futuro.reshape(-1, 1))

    # REGRESSÃO EM ÁRVORE DO FEATURE 5
    x5 = tabela['Dias corridos']
    y5 = tabela['feature 5']
    x_train_5, x_test_5, y_train_5, y_test_5 = train_test_split(t, y5,
test_size=0.2, random_state=0) # Separando 20% dos pontos para teste
    regressor_5 = LinearRegression().fit(x_train_5.reshape(-1,1),y_train_5)
    predicted_5 = regressor_5.predict(x_test_5) # Valores de teste
    # df5 = pd.DataFrame(y_test_5, predicted_5)
    # dfr5 = pd.DataFrame({'feature 5':y_test_5, 'Previsto':predicted_5})
    # print(dfr5) # Tabela de teste

    feature5 = regressor_5.predict(futuro.reshape(-1, 1))

    # DATA FRAME COM TODOS OS RESULTADOS DO MODELO
    tabela_completa = pd.DataFrame({'feature 1':feature1,'feature
2':feature2,'feature 3':feature3,'feature 4':feature4,'feature 5':feature5})
    future_price = regressor.predict(tabela_completa)
    date_list = pd.date_range(start=dias.loc[0], periods=quantidade_dias)
```

```
    tabela_completa.insert(0, 'date', date_list)
    tabela_completa.insert(6, 'Price', future_price)
    # print(tabela_completa)
    # # DESCOMENTAR PARA SALVAR NOVOS ARQUIVOS
    # tabela_completa.to_csv(r'C:\Users\gabri_7fr\Desktop\Exercicio
Sarpen\Ativos\Tabelas Completas\tabela_completa-{}.csv'.format(k),
index=False)

    # CÁLCULO DOS RENDIMENTOS EM PORCENTAGEM PARA CADA ATIVO
    compra = tabela_completa[tabela_completa['date']=='2021-06-21']
    venda = tabela_completa[tabela_completa['date']=='2022-06-21']
    rend[kk,0] = ((venda['Price'].to_numpy()-
compra['Price'].to_numpy())/abs(compra['Price'].to_numpy()))*100 # Rendimento
em porcentagem, para poder comparar todos os ativos... valores absolutos no
denominador para evitar mudança de sinal

# print(rend)
# print(rend.shape)

rendimentos = pd.DataFrame({'ID':id_ativo})
rendimentos.insert(1, 'Rendimentos [%]', rend)
r = rendimentos.sort_values(by='Rendimentos [%]', ascending=False)
# r.to_csv(r'C:\Users\gabri_7fr\Desktop\Exercicio
Sarpen\Ativos\Rendimentos\rendimentos.csv', index=False)

# print(r)
melhores = r.head(10)
print('Rendimento em porcentagem dos 10 melhores ativos para investir
(Rendimento considerado de 21/06/2021 a 21/06/2022)')
print(melhores)

id_melhores = melhores["ID"]

# fig1,axs = plt.subplots(2,5)
# for m in range(2):
#     for n in range(5):
#         df_real = pd.read_csv(r'C:\Users\gabri_7fr\Desktop\Exercicio
Sarpen\Ativos\out-{}.csv'.format(812), sep=',')
#         df_regre = pd.read_csv(r'C:\Users\gabri_7fr\Desktop\Exercicio
Sarpen\Ativos\Tabelas Completas\tabela_completa-{}.csv'.format(812), sep=',')
#         df_real.cumsum()
#         df_regre.cumsum()
#         axs[m, n],plt.scatter(df_real['date'],df_real['price'],color='red')
#         axs[m, n],plt.plot(df_regre['date'],df_regre['Price'])

# plt.show()


# fig = plt.subplot()
```

```
# df_real = pd.read_csv(r'C:\Users\gabri_7fr\Desktop\Exercicio
Sarpen\Ativos\out-{}.csv'.format(812), sep=',')
# df_regre = pd.read_csv(r'C:\Users\gabri_7fr\Desktop\Exercicio
Sarpen\Ativos\Tabelas Completas\tabela_completa-{}.csv'.format(812), sep=',')
# df_real.cumsum()
# df_regre.cumsum()
# plt.scatter(df_real['date'],df_real['price'],color='red')
# plt.scatter(df_regre['date'],df_regre['Price'])
# plt.show()

# for ax in axis.flat:
# axs.set(xlabel="Date",
#     ylabel="Price",
#     title="Preço do ativo ao longo do tempo",
#     xlim=[df_real["date"].loc[0], df_real["date"].iloc[-1]])
```