Nome: Adriano da Silva de Carvalho

Nº USP: 13692400

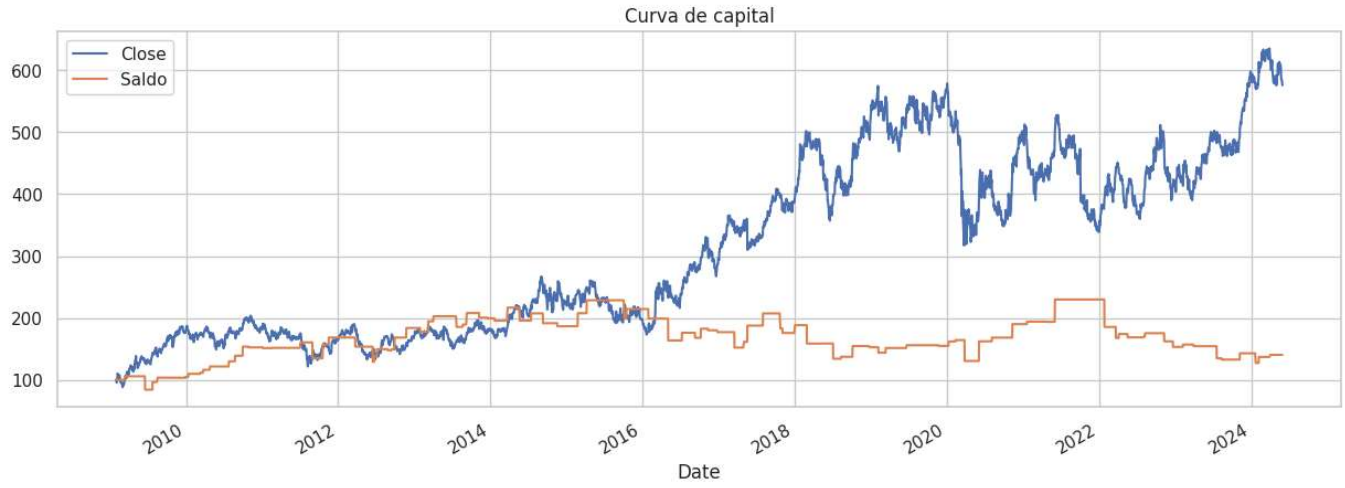
Trading Sistemático

```
%pip install ta
!pip install yfinance --upgrade --no-cache-dir


# Importar bibliotecas necessárias
from ta import add_all_ta_features
from ta.utils import dropna
import yfinance as yf

from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neural_network import MLPClassifier
from sklearn import metrics

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

sns.set(style='whitegrid')

# Baixar dados de ativos
start_date = '2005-01-01'
end_date = '2024-05-30'

itub4 = yf.download('ITUB4.SA', start_date, end_date, '1d')[['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']]
abev3 = yf.download('ABEV3.SA', start_date, end_date, '1d')[['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']]

# Selecionar quais dados serão utilizados com os algoritmos abaixo
dados = itub4.copy()

# Ajustar preços
for index, row in dados.iterrows():
    dados.at[index, 'High'] = row['High'] / row['Close'] * row['Adj Close']
    dados.at[index, 'Low'] = row['Low'] / row['Close'] * row['Adj Close']
    dados.at[index, 'Close'] = row['Adj Close']

# Definição dos pontos de compra e venda nas janelas de "periodo_tam" dias.
dados['Action'] = -1000  # No action

periodo_tam = 11  # Tamanho da janela (número de dias)
total_dados = dados.shape[0]
cont = 0

while cont < total_dados:
    min_val = dados['Close'].iloc[cont:cont + periodo_tam].min()
    max_val = dados['Close'].iloc[cont:cont + periodo_tam].max()
    for i in range(cont, min(cont + periodo_tam, total_dados)):
        if dados['Close'].iloc[i] == min_val:
            dados.loc[dados.index[i], 'Action'] = 0  # Hold
            if i + 1 < total_dados:
                dados.loc[dados.index[i + 1], 'Action'] = 1  # Buy
        elif dados['Close'].iloc[i] == max_val:
            dados.loc[dados.index[i], 'Action'] = 0  # Hold
            if i + 1 < total_dados:
                dados.loc[dados.index[i + 1], 'Action'] = -1  # Sell
        elif dados['Action'].iloc[i] == -1000:
            dados.loc[dados.index[i], 'Action'] = 0  # Hold
    cont += periodo_tam

# Cálculo das bandas de Bollinger
from ta.volatility import BollingerBands
ind_bb = BollingerBands(close=dados['Close'], window=20, window_dev=2)
dados['bb_hb'] = ind_bb.bollinger_hband()
dados['bb_lb'] = ind_bb.bollinger_lband()
dados['bb_mb'] = ind_bb.bollinger_mavg()
dados['bb_hr'] = dados['bb_hb'] / dados['Close']
dados['bb_lr'] = dados['bb_lb'] / dados['Close']
dados['bb_mr'] = dados['bb_mb'] / dados['Close']
```

```
    dados['bb_mr'] = dados['bb_mb'] / dados['Close']

    # Cálculo do indicador RSI
    from ta.momentum import RSIIndicator
    ind_rsi = RSIIndicator(close=dados['Close'], window=2)
    dados['RSI'] = ind_rsi.rsi() / 100.0

    # Exclui as primeiras linhas do data frame, uma vez que nem todos os índices estão disponíveis nessas datas.
    dados = dados.iloc[20:]

    # Criação da matriz de dados de entrada e do vetor de dados de saída dos algoritmos de ML
    din = dados[['bb_hr', 'bb_lr', 'bb_mr', 'RSI']].reset_index().drop(['Date'], axis=1).to_numpy()
    dout = dados['Action'].reset_index().drop(['Date'], axis=1).to_numpy()

    print(din.shape)
    print(dout.shape)

    # Separação dos dados em conjunto de treinamento e conjunto de teste/validação.
    train_n = 1000

    train_in = din[0:train_n]
    train_out = dout[1:train_n + 1]

    test_in = din[train_n:4000]
    test_out = dout[train_n + 1:4001]

    # Algoritmos de aprendizado de máquina. Utilizar apenas um deles em cada teste.
    # Rede Neural MLP com parâmetros solver e activation alterados para melhor desempenho de aprendizado.
    # clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(20,), random_state=1, max_iter=200000, activation='tanh')

    # Árvore de Decisão
    clf = DecisionTreeClassifier(random_state=1, max_depth=7)

    # Random Forest
    # clf = RandomForestClassifier(random_state=1, max_depth=5, n_estimators=5)

    # Treina o modelo com os dados de entrada/saída
    clf.fit(train_in, train_out.ravel())

    # Avaliação dos resultados do conjunto de dados de treinamento
    y_pred_train = clf.predict(train_in)
    print("Accuracy train:", metrics.accuracy_score(train_out, y_pred_train))

    # Avaliação dos resultados do conjunto de dados de teste
    y_pred_test = clf.predict(test_in)
    print("Accuracy test:", metrics.accuracy_score(test_out, y_pred_test))

    # Resultado dos algoritmos para todo o conjunto de dados.
    y_pred_all = clf.predict(din)

    # Inclusão da saída dos algoritmos de ML no DataFrame "dados_ml". É possível comparar essa saída com a saída desejada (ideal).
    dados['ML-Action'] = y_pred_all

    # Cálculo do saldo e de outras estatísticas dos trades
    dados['Saldo'] = 0
    saldo = 100
    comprado = 0
    val_neg = 0
    trades = 0
    trades_pos = 0
    media_pos = 0
    media_neg = 0

    for i in range(0, dados.shape[0]):
        if dados['ML-Action'].iloc[i] == 1 and comprado == 0:
            val_neg = dados['Close'].iloc[i]
            comprado = 1
        elif dados['ML-Action'].iloc[i] == 1 and comprado == -1:
            res_trade = val_neg / dados['Close'].iloc[i] - 1
            if res_trade > 0:
                trades_pos += 1
                media_pos += res_trade
            else:
                media_neg += res_trade
            saldo = saldo * (1 + res_trade)
            trades += 1
            comprado = 0
        elif dados['ML-Action'].iloc[i] == -1 and comprado == 0:
            comprado = -1
            val_neg = dados['Close'].iloc[i]
        elif dados['ML-Action'].iloc[i] == -1 and comprado == 1:
            res_trade = dados['Close'].iloc[i] / val_neg - 1
            if res_trade > 0:
```

```
            trades_pos += 1
            media_pos += res_trade
        else:
            media_neg += res_trade
        saldo = saldo * (1 + res_trade)
        trades += 1
        comprado = 0
    dados.loc[dados.index[i], 'Saldo'] = saldo

if trades_pos > 0:
    print("Trades total:", trades, "Trades positivos:", trades_pos, "(", round(trades_pos / trades * 100, 2), "%)")
    print("Media trades positivos:", round(media_pos / trades_pos * 100, 2), "%, Media trades negativos:", round(media_neg / (trades - tr
    print("Rent. buy and hold:", round((dados['Close'].iloc[-1] / dados['Close'].iloc[0] - 1) * 100, 2), "%")
    print("Rent. ML trade:", round((dados['Saldo'].iloc[-1] / dados['Saldo'].iloc[0] - 1) * 100, 2), "%")
else:
    print("Não foram feitos trades no período.")

# Normalização e apresentação gráfica dos resultados
ref = 1000
dados[['Close', 'Saldo']] = dados[['Close', 'Saldo']] / dados[['Close', 'Saldo']].iloc[ref] * 100
dados[['Close', 'Saldo']].iloc[ref:].plot(figsize=(15, 5), title='Curva de capital')
```

```
[**********100%%**********]  1 of 1 completed
[**********100%%**********]  1 of 1 completed
(4805, 4)
(4805, 1)
Accuracy train: 0.895
Accuracy test: 0.7783333333333333
<ipython-input-2-5ad3977539eb>:117: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  dados['ML-Action'] = y_pred_all
<ipython-input-2-5ad3977539eb>:120: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  dados['Saldo'] = 0
Trades total: 127 Trades positivos: 83 ( 65.35 %)
Media trades positivos: 9.32 %, Media trades negativos: -7.18 %
Rent. buy and hold: 756.86 %
Rent. ML trade: 4647.15 %
<Axes: title={'center': 'Curva de capital'}, xlabel='Date'>
```



Conclusão da utilização de **Trading Sistemático**

Com a adição de 2 novos ativos "IMAB" e "SP500BR", a aplicação do algoritmo de aprendizado MLP (Multi Layer Perceptron), vemos que foi obtido um resultado muito expressivo comparado com os vistos em aula.