Documentação feita no Desafio Técnico da SmarttBot

v0.1.4

Jean Lucas Almeida Mota

20 de maio de 2020

Contents

Documentação feita para o Desafio Técnico da SmarttBot's

Projeto realizado pelo candidato Jean Lucas Almeida Mota para a vaga de Estágio de Desenvolvimento Estratégias de Investimento (Python), em Belo Horizonte/MG na Empresa SmarttBot.

O Desafio técnico consiste na realização do cálculo de pelo menos dois indicadores técnicos, utilizando o data set: https://www.kaggle.com/mczielinski/bitcoin-historical-data/data#coinbaseUSD_1-min_data_2014-12-01_to_2019-01-09.csv

Este projeto se propõe realizar o cálculo das Médias Móveis Exponenciais e dos Índices de Força Relativa.

Médias Móveis Exponenciais

Média Móvel Exponencial é uma media ponderada do preço dos ultimos N fechamentos de um ativo, dando mais peso para os fechamentos mais próximos.

Sendo que o N é chamado de periodo, e estamos utilizando um periodo de 20. Dependendo do dataset, 20 periodos podem ser 20 minutos, 20 horas, 20 dias ou 20 anos.

No dataset utilizado 20 periodos representam 20 minutos.

No arquivo data/saida.csv esse indicador está representado como 'indicador-1'

Índices de Força Relativa

O indice de força relativa (RSI em inglês) foi desenvolvido por J. Welles Wilder. É um indice com escala de variação fixa, ou seja, varia entre 0 e 100.

É usado para identificar a subvalorização ou sobrevalorização de um ativo. Exemplo: Quanto maior o indice (>70) mais sobrevalorizada o ativo esta, e quanto menor o indice (<30) mais subvalorizada o ativo esta.

No arquivo data/saida.csv esse indicador está representado como 'indicador-1'

Instruções para executar o código

Baixe a versão mais recente do projeto em https://bitbucket.org/jeanhardzz/desafio_smartbot/downloads/?tab=tags

Após baixado, através de linha de comando, dentro do diretório do projeto execute:

python main.py dd/mm/yy H:M dd/mm/yy H:M

Sendo que dd/mm/yy H:M representa uma data e horário. Logo está sendo passado uma data e horário inicio e uma data e horário fim

Se todos os passos foram feitos corretamente, será mostrada a mensagem: O arquivo saida.csv foi criado com sucesso. Seguido de um preview do arquivo.

Modulos

Main

Programa principal

`main.`**`main`**`()`

Onde são coletadas as datas: inicio e fim e chama as classes para o calculo dos indicadores e criação do saida.csv

Classes Package

Classe Dados

Esta classe esta sendo usada para ler um arquivo.csv de candlestisck e gerar uma saida.csv com indicadores.

Essa classe se ultiliza da classe Indicadores para fazero calculos dos indicadores

O arquivo saida.csv contem dois indicadores, sao eles: indicador-1: Representa as Médias Móveis Exponenciais indicador-2: Representa os Índices de Forças Relativas

1

class `classes.dados.`**`Dados`** (caminho_csv, pytest_val=0)

Bases: **`object`**

`corta_excesso` **(**data**)**

Recebe um dataframe e retorna outro dataframe com datas inferiores a data_fim.

Parameters: **data** – Dataframe

Returns: data_cortado

property **`gera_csv_indicadores`**

Gera um arquivo.csv com dois indicadores.
indicador-1: Medias Móveis Exponenciais. indicador-2: Indice de Força Relativa.
Para otimizar, faço um corte nos Dados no data_fim. Assumindo que os indicadores nao usaram dados após data_fim para cálculo.
Após os calculos efetua-se a criação do arquivo.csv e garanto a criação do mesmo. Retornando False caso nao tenho efetuado a criação e True caso tenha sucesso

Returns:

`imprime_datas_inicio_fim` **()**

Usa a função print() para imprimir as datas inicio e fim.

Returns:

`pesquisa_no_dataframe_o_intervalo_datas` **(**data**)**

Recebe um data frame e devolve um novo dataframe no intervalo.
O dataframe retorno esta no intervalo data_inicio - data_fim.

Parameters: **data** – Dataframe.

Returns: data_intervalo

`preenche_data_fim` **(**data_fim**)**

Testa se o valor é valido para ser a data fim.
Primeiro faz um teste de ValueError para garantir que foi escrito uma data no formato exigido.
E depois faz uma conversao de datetime para seconds since the epoch, que é um tipo de dado que conta a quantidade de segundos passados desde January 1, 1970, 00:00:00 (UTC).
E por ultimo testa se a data fim é menor que a data inicio.

Parameters: **data_inicio** – Inicio do intervalo.

Returns:

`preenche_data_inicio` **(**data_inicio**)**

Testa se o valor é valido para ser a data inicial.
Primeiro faz um teste de ValueError para garantir que foi escrito uma data no formato exigido.
E depois faz uma conversao de datetime para seconds since the epoch, que é um tipo de dado que conta a quantidade de segundos passados desde January 1, 1970, 00:00:00 (UTC).

Parameters: **data_inicio** – Inicio do intervalo.

Returns:

`tratando_dados_faltantes` **()**

Retira do DataFrame todos os dados invalidos (NaN).
Pesquisa em todas as colunas se há um valor do tipo Nan. Se houver, remove essa linha do DataFrame.

Returns:

Classe Indicadores

Essa classe possui os metodos para calcular indicadores de candlesticks dado um DataFrame.

Nenhuma das funções aqui deve alterar o DataFrame mas sim retornar o indicador no formato series ou lista, ou ate mesmo um DataFrame cópia do original.

2

class classes.indicadores.**Indicadores**
Bases: **object**

calcula_indices_de_força_relativa (precos, n=14)
Retorna uma lista com os índices de força relativa dado uma Serie de preços
O indice de força relativa (RSI em inglês) foi desenvolvido por J. Welles Wilder. É um indice com escala de

variação fixa, entre 0 e 100. Basicamente, quanto maior o indice (>70) mais sobrevalorizada a ação esta e quanto menor o indice (<30) mais subvalorizada a ação esta.

Parameters:

- **precos** – Serie
- **n** – periodo

returns: lista de rsi

calcula_medias_relativas (dados)
Retorna uma serie com o calculo de todas as medias móveis exponenciais.
Média Móvel Exponencial é basicamente a media ponderada do preço de fechamento dos ultimos N fechamentos, dando mais peso para os fechamentos mais próximos. Aqui o N é chamado de periodo, e estamos utilizando um periodo de 20. Dependendo do dataset, 20 periodos pode ser 20 minutos, 20 horas, 20 dias ou 20 anos.

Parameters: **dados** – DataFrame
Returns: Serie

Tests Package

Testes Dados

Testes Automatizados para a classe Dados.

tests.test_dados.**dados** ()

tests.test_dados.**testa_se_arquivo_csv_foi_criado_com_sucesso** ()
Confere se dados.gera_csv_indicadores gera um arquivo.csv.

Returns:

tests.test_dados.**testa_se_data_fim_recebe_datas_anterioes_a_1970** (dados)
Confere se a classe dados recebe a data fim anterior ao formato utilizado.

Parameters: **dados** –
Returns:

tests.test_dados.**testa_se_data_fim_recebe_datas_compativeis** (dados)
Confere se a classe dados recebe uma data fim no formato correto.

Parameters: **dados** –
Returns:

tests.test_dados.**testa_se_data_fim_recebe_datas_incompativeis** (dados)
Confere se a classe dados recebe uma data fim com o mes e o dia no formato correto.

Parameters: **dados** –
Returns:

tests.test_dados.**testa_se_data_fim_recebe_horas_incompativeis** (dados)
Confere se a classe dados recebe as horas fim no formato correto.

Parameters: **dados** –
Returns:

tests.test_dados.**testa_se_data_inicio_recebe_datas_anterioes_a_1970** (dados)
Confere se a classe dados recebe a data inicio anterior ao formato utilizado.

Parameters: **dados** –

Returns:

tests.test_dados.**testa_se_data_inicio_recebe_datas_compativeis** (dados)
Confere se a classe dados recebe uma data inicio no formato ideal.

Parameters: **dados** –

Returns:

tests.test_dados.**testa_se_data_inicio_recebe_datas_incompativeis** (dados)
Confere se a classe dados recebe uma data inicio com o mes e o dia no formato correto.

Parameters: **dados** –

Returns:

tests.test_dados.**testa_se_data_inicio_recebe_horas_incompativeis** (dados)
Confere se a classe dados recebe as horas inicio no formato correto.

Parameters: **dados** –

Returns:

tests.test_dados.**testa_se_existem_dados_faltantes** (dados)
Confere se há dados nulos do tipo NaN.

Parameters: **dados** –

Returns:

tests.test_dados.**testa_se_retorna_um_data_frame_no_intervalo** (dados)
Confere se a funcao dado.pesquisa_no_dataframe retorna um dataframe no intervalo correto.

Parameters: **dados** –

Returns:

Testes Indicadores

Testes Automatizados para a classe Indicadores.

tests.test_indicadores.**dados** ()

tests.test_indicadores.**indicadores** ()

tests.test_indicadores.**testa_se_o_calculo_dos_indicadores_esta_correto** (dados, indicadores)
Confere se o arquivo.csv gerado após os calculos dos indicadores esta correto.
Essa conferencia é realizada a partir de um arquivo.csv com os calculos feitos previamente.
A função cria uma saida.csv e depois chama esse arquivo para fazer a comparação de dados, isso teve que ser feito dessa forma porque há algum erro de comparação quando se compara um arquivo criado dentro do python e um arquivo externo chamado pelo python, mesmo garantindo que ambos sao identicos.

Parameters:

- **dados** –
- **indicadores** –

Returns: