# Construindo um portfólio de ativos robusto

## Retomando o projeto:

Vamos estudar um pouco de finanças com Python, pode me ajudar?

Perfeito, alguns pontos que preciso alinhar contigo. Desejo que nosso estudo gere alguns *scripts* Python e que façamos uso das boas práticas de programação e documentação, desejo que nosso sistema seja modular e robusto e escalável, e apresente facilidade de expansão. Outro detalhe, é desejo que ao final todos os módulos possam ser reunidos em um sistema, usando *React*, *NodeJs* e PostgreSQL. Então devemos construir nossos *scripts* Python, de forma que possamos testá-los individualmente, mas que ao final possamos reunir todos os *scripts* em um *pipeline*, devemos estar aptos para gerar os *endpoints* necessários para integração com o **front**, para a parte gráfica devemos usar bibliotecas com **Plotly** e **Dash**, o uso do **Matplotlib** deve ser desencorajado, o que acha dessa proposta? É factível? Lembrando, que nosso foco neste momento são os *scripts* Python. Tenho vários *scripts* prontos e já testados que posso lhe fornecer para que entenda a lógica por trás e possamos dar continuidade ao projeto, tenho um ambiente montado para teste, vou anexar o arquivo **requirements.txt**, para que o conheça e avalie. O objetivo desse projeto é criar um sistema capaz de buscar ativos na B3, compor um portfólio ótimo, avaliar seu desempenho no curto prazo (6 meses e no longo prazo 5 anos).

Os scripts foram pensados como ferramentas de suporte e construção passo-a-passo dessa solução, queremos que nossos scripts sejam robustos e escaláveis, embora tenhamos definido um conjunto de ativos para construção e testes, é desejo que no futuro o sistema possa trabalhar com qualquer conjunto de ativos. Desejo que analise cada um deles (*scripts* - para que possa se familiarizar com o projeto), tenho também alguns arquivos que estou usando para desenvolver o projeto (veja a estrutura de arquivos e os arquivos .csv que temos até o momento), gere a partir disso uma estrutura para darmos sequência ao projeto e conseguirmos nosso objetivo, que deve passar por simulação de Monte Carlo (já finalizamos para curto e longo prazo) e se for possível com ferramentas de ML (é neste ponto que estamos e até aqui não tivemos sucesso) desejamos um algoritmo de ML capaz de prever o retorno do portfólio para os próximo 6 meses e também os próximos 5 anos. Os *scripts* devem que vou lhe fornecer devem servir para seu conhecido do projeto e das práticas de desenvolvimento que estamos usando (para os próximos scripts você deve manter a mesma estrutura) assim, terá um exemplo de documentação para que os novos *scripts* sigam o mesmo roteiro e mantenha o padrão. Posso anexar os scripts e os arquivos primeiro? O que acha dessa proposta. Após sua análise detalhada, preciso que gere uma proposta para os próximos passos, mas antes de qualquer proposta, avalie se precisa de mais alguma informação, tire todas as suas dúvidas comigo, feito isso, faça uma proposta de próximos passos, pode ser? Note que estamos exatamente no ponto para o desenvolvimento do script de ML.

**BackPython/DADOS/**

Texto

Descrição gerada automaticamente

## Texto Descrição gerada automaticamente

## Nosso roteiro passará por:

1. **Análise de Dados Financeiros**:
   1. Importação e manipulação de dados financeiros com pandas.
   2. Extração de dados de APIs financeiras, como Yahoo Finance ou Alpha Vantage.
2. **Visualização de Dados**:
   1. Criação de gráficos de preços e indicadores financeiros.
   2. Análise de tendências e volatilidade com Plotly e Dash.
3. **Análise de Portfólio**:
   1. Riscos e retornos.
   2. Otimização de portfólios com a Fronteira Eficiente.
4. **Modelagem Financeira**:
   1. Regressão linear para previsão de preços.
   2. Modelagem com Simulação de Monte Carlo.
   3. Na Simulação de Monte Carlo, seria interessante dividir os dados em treinamento teste e validação (usar os 2 últimos meses devem ser usados para validação), desejamos ver se o nosso portfólio otimizado está mais próximo do cenário de 5%, 50% ou 95%, queremos inclusive saber o quanto estamos próximos destes cenários.
5. **Estratégias de Negociação**:
   1. Criação de backtests de estratégias.
   2. Análise de médias móveis ou bandas de Bollinger.
   3. Queremos também entender qual o momento certo de compra de um ticket, analisar e interpretar gráficos de candlestick e gerar insights.
6. **Instrumentos Financeiros**:
   1. Precificação de opções com o modelo Black-Scholes.
   2. Cálculo de taxas de juros e valor presente.

## O que pretendemos com esse estudo:

1. **Estrutura Modular**:
   * Dividir o código em funções e classes reutilizáveis.
   * Organizar em pacotes e módulos para fácil manutenção e expansão.
2. **Boas Práticas**:
   * Seguir PEP-8 (o guia de estilo para Python).
   * Utilizar tipagem estática com **type hints** para melhorar a legibilidade e reduzir erros.
   * Documentar o código com docstrings.
3. **Testabilidade**:
   * Escrever testes unitários com pytest ou outra biblioteca.
   * Garantir que cada módulo possa ser testado de forma isolada.
4. **Preparação para Integração**:
   * Projetar funções e classes para gerar saídas claras (e.g., JSON), facilitando a integração via APIs.
   * Usar frameworks como **FastAPI** ou **Flask** para criar endpoints, quando necessário.
5. **Escalabilidade e Eficiência**:
   * Escolher estruturas de dados e algoritmos adequados para grandes volumes de dados financeiros.
   * Garantir que os módulos possam lidar com integração futura com bancos de dados como PostgreSQL.

### Módulo – BP\_mod0\_anal\_xlsx.py

1. Integração direta:
   1. O script agora utiliza OUTPUT\_DIR e RUN\_DASHBOARD diretamente do BP\_mod1\_config.py.
2. Automação e flexibilidade:
   1. Automatiza a verificação do caminho de entrada e saída.
3. Permite ativar o dashboard com base na configuração RUN\_DASHBOARD.
4. Próximos passos:
   1. Verifique se o arquivo BP\_mod1\_config.py está no mesmo diretório que o script principal, ou ajuste o PYTHONPATH para que seja encontrado corretamente.

### Módulo – BP\_mod0\_csv\_xlsx.py

O script **BP\_mod0\_csv\_xlsx.py** tem um objetivo claro: converter arquivos CSV para XLSX com o tratamento específico da coluna Date. Apesar de funcional, há espaço para melhorias em termos de qualidade, legibilidade e integração com o sistema maior.

**Melhorias propostas:**

1. **Centralização de configurações**:
   * Usar parâmetros configuráveis do **BP\_mod1\_config.py**, como caminhos de entrada e saída.
2. **Modularidade**:
   * Organizar o script em funções reutilizáveis para carregamento, processamento e salvamento.
3. **Tratamento de erros e logs**:
   * Substituir print por logs estruturados com níveis como INFO e ERROR.
4. **Flexibilidade**:
   * Permitir caminhos de entrada e saída como argumentos dinâmicos.

### Módulo BP\_mod\_0\_limp\_arquivos.py

O script **BP\_mod0\_limp\_arquivos.py** tem como objetivo realizar a limpeza de diretórios e seus conteúdos, o que é útil para manutenção ou reinicialização de projetos. Apesar de funcional, ele pode ser aprimorado para maior robustez, flexibilidade e integração com outros módulos.

**Melhorias propostas:**

1. **Modularidade**:
   * Encapsular a lógica de limpeza em uma função reutilizável.
   * Adicionar a possibilidade de passar a lista de diretórios dinamicamente.
2. **Logs estruturados**:
   * Substituir print por logging para mensagens consistentes e configuráveis.
3. **Integração**:
   * Incorporar as configurações de diretórios a partir do **BP\_mod1\_config.py**.
4. **Prevenção de erros críticos**:
   * Adicionar um sistema de confirmação para evitar remoções acidentais.
   * Gerenciar exceções para falhas durante a remoção.

### BP\_mod1\_config.py

**Alterações planejadas:**

1. **Flexibilidade no START\_DATE**:
   * Criar um mecanismo para permitir que START\_DATE seja ajustado dinamicamente com base no ticket com menor período histórico.
   * Adicionar suporte para registrar o ativo responsável por essa data inicial.
2. **Separação de configurações fixas e dinâmicas**:
   * Manter a configuração estática atual enquanto permite ajustes dinâmicos realizados por scripts específicos.
3. **Documentação e modularidade**:
   * Adicionar explicações claras para o comportamento dinâmico de START\_DATE e demais configurações.
4. **Portabilidade para novos tickets**:
   * Assegurar que o sistema seja expansível sem limitações aos ativos atualmente listados.

**Alterações principais:**

1. **Flexibilidade em START\_DATE**:
   * Introduzido DEFAULT\_START\_DATE com 10 anos atrás.
   * Função atualizar\_start\_date permite ajustar dinamicamente com base nos dados históricos reais.
2. **Registro de responsável por START\_DATE**:
   * O ativo é armazenado em START\_DATE\_ASSET.
3. **Diretórios dinâmicos**:
   * Criação do diretório OUTPUT\_DIR se não existir.

### BP\_mod3\_prepara\_dados.py

**Estrutura do Script BP\_mod3\_prepara\_dados.py**

**Objetivo**

* Limpar, organizar e preparar os dados históricos para otimização de portfólio.
* Calcular retornos dos ativos e benchmark.
* Gerar um arquivo consolidado para uso nos próximos scripts.

**Funcionalidades**

1. **Leitura e Validação**:
   * Carregar dados históricos de ativos e benchmarks.
   * Verificar e tratar valores ausentes.
2. **Cálculo de Retornos**:
   * Gerar retornos logarítmicos para ativos e benchmark.
3. **Normalização de Período**:
   * Alinhar os ativos e benchmarks ao menor período comum.
4. **Geração de Arquivo Consolidado**:
   * Criar um arquivo com os retornos e as datas consolidados.

Com base na sua observação, é uma boa decisão salvar o arquivo final como .csv, já que ele será consumido por outros scripts e o .csv é mais prático para automação.

Sobre os dados iguais a 0 ou negativos no arquivo gerado, vamos avaliar as possíveis causas:

**Causas dos Dados Negativos ou Zerados**

1. **Retornos Logarítmicos Negativos**:
   * Retornos negativos ocorrem naturalmente em períodos em que o preço do ativo caiu. Isso é esperado, mas pode ser problemático se não for tratado corretamente em cálculos posteriores.
2. **Zeros em Retornos**:
   * Valores de retorno igual a zero geralmente ocorrem por:
     + Preços consecutivos iguais em dias úteis.
     + Dados ausentes (tratados como 0 durante algum processo).
3. **O Benchmark (ˆBVSP)**:
   * Como mencionado, ele é o benchmark e não deve participar do processo de composição do portfólio. Ele pode estar no arquivo para referência, mas será excluído no momento de formar o portfólio.

**Refatoração Proposta**

1. **Filtrar o Benchmark**:
   * O ˆBVSP será mantido no arquivo apenas para comparações, mas será excluído ao preparar os dados para otimização do portfólio.
2. **Remoção de Dados Inválidos**:
   * Remover linhas com valores NaN ou resultados incoerentes (e.g., valores zero em retornos).
3. **Salvar como .csv**:
   * O arquivo final será salvo como .csv no mesmo diretório.
4. **Adicionar Logs Descritivos**:
   * Informar ao usuário quais ajustes foram feitos, como linhas removidas ou benchmarks filtrados.

**Possíveis Causas para Valores Ausentes**

1. **Primeira Linha dos Retornos**:
   * O cálculo de retornos logarítmicos (pct\_change) sempre gera um NaN na primeira linha, pois não há um valor anterior para calcular o retorno.
2. **Dados Inválidos no Arquivo Original**:
   * Valores ausentes ou incorretos no arquivo original podem gerar NaN nos cálculos.
3. **Processo de Conversão de Datas**:
   * Se a coluna "Date" não foi corretamente parseada, algumas linhas podem ter sido interpretadas como NaN.

**Pontos a Melhorar**

1. **Benchmark (ˆBVSP)**:
   * O script remove o benchmark, mas não utiliza dados reais do ˆBVSP para comparação. Adicionar a série de retornos do benchmark e calcular sua performance seria ideal.
2. **Configuração de Parâmetros**:
   * Muitos valores estão hardcoded, como tickers, restrições setoriais e caminhos de arquivos. Idealmente, esses parâmetros poderiam ser configurados externamente.
3. **Validação do Retorno Esperado**:
   * Adicionar uma verificação clara se o portfólio supera o benchmark em pelo menos 8%.
4. **Modularização**:
   * Algumas funções poderiam ser organizadas para facilitar a reutilização e expansão.
5. **Log Detalhado**:
   * Melhorar o registro de logs para monitorar o processo.

**Critérios para Exclusão de Ativos**

1. **Baixa Rentabilidade**:
   * Ativos com retornos médios negativos ou abaixo de um limite aceitável.
2. **Alta Volatilidade**:
   * Ativos com desvio padrão dos retornos acima de um limite predefinido.
3. **Dados Insuficientes**:
   * Ativos com muitos valores ausentes ou inconsistências.
4. **Setores Não Desejados** (opcional):
   * Ativos pertencentes a setores que não atendem aos objetivos estratégicos do portfólio.

### Scripts do BP\_mod2.py (foram criados para ajustes nos tickets, seleção previa dos tickets candidatos ao portfólio ótimo)

**Características de Robustez**

1. **Critérios de Retorno Flexíveis**:
   * O limite de retorno mínimo pode ser ajustado facilmente (padrão: 15%).
   * Isso permite que o script seja adaptado para diferentes estratégias de investimento.
2. **Entrada Dinâmica**:
   * Ele pode processar qualquer arquivo de dados históricos que tenha:
     + **Datas como índice** (Date).
     + **Tickets como colunas** com os preços diários ou ajustes correspondentes.
   * O script detecta automaticamente os tickets que não atendem aos critérios e os exclui.
3. **Saídas Modulares**:
   * Gera dois arquivos:
     + **filtered\_data.csv**: Retornos anualizados dos ativos aprovados.
     + **historical\_data\_filtered.csv**: Histórico de preços/dados filtrado.
   * Esses arquivos são independentes e podem ser usados por outros scripts sem necessidade de ajustes.
4. **Gerenciamento de Erros**:
   * Mensagens de erro informativas e logs detalhados ajudam a identificar problemas rapidamente (ex.: arquivos ausentes, índices incorretos).
5. **Adaptabilidade a Portfólios Diversos**:
   * O script é capaz de lidar com:
     + Qualquer quantidade de ativos.
     + Dados históricos com diferentes períodos e frequências, desde que estejam uniformes dentro do arquivo.

**Limitações Potenciais**

1. **Homogeneidade dos Dados**:
   * Todos os ativos no arquivo de dados históricos devem ter a mesma frequência temporal (ex.: dados diários).
   * Se houver discrepâncias, será necessário tratá-las antes de rodar o script.
2. **Dependência do Índice Date**:
   * O índice do arquivo deve ser Date. Se o arquivo não for estruturado corretamente, isso pode causar erros.
3. **Benchmark Não Incluído Automaticamente**:
   * Atualmente, o benchmark (^BVSP) é removido, mas pode ser necessário adicioná-lo em scripts futuros para comparações.
4. **Sem Suporte para Dados Faltantes Extensos**:
   * O script presume que os dados faltantes já foram tratados. Caso contrário, a filtragem de ativos pode ser afetada.

**Recomendações para Uso com Outros Portfólios**

1. **Verifique o Arquivo de Entrada**:
   * Certifique-se de que o arquivo histórico tenha a estrutura adequada (colunas de tickets e índice Date).
2. **Ajuste o Critério de Retorno**:
   * Alterar o limite mínimo de retorno no código:

python

Copiar código

limite\_minimo = <novo\_valor\_em\_porcentagem>

1. **Integração com Benchmarks ou Setores**:
   * Se necessário, adicione benchmarks ou informações setoriais antes de filtrar os ativos.
2. **Execução em Batch**:
   * O script pode ser facilmente adaptado para processar múltiplos portfólios alterando o arquivo de entrada.

## Scripts faltantes

**1. BP\_mod3\_curto\_prazo.py​(BP\_mod3\_curto\_prazo)**

* **Objetivo**: Simular e analisar tendências de curto prazo para o portfólio com base em EGARCH, Monte Carlo e retornos logarítmicos.
* **Complexidade**: Moderada. Integra diferentes métodos e fornece insights práticos para o portfólio.
* **Dependências**: Dados históricos limpos e pesos do portfólio.
* **Observação**: Poderia ser ajustado para maior flexibilidade na escolha dos modelos.

**2. BP\_mod3\_dashboard.py​(BP\_mod3\_dashboard)**

* **Objetivo**: Criar um dashboard interativo para análise de resultados usando Dash e Plotly.
* **Complexidade**: Alta, devido à criação de visualizações detalhadas e integração de dados.
* **Dependências**: Dados gerados por simulações, percentis e benchmarks.
* **Observação**: É útil refatorar para que funcione como um módulo integrado aos outros scripts.

**3. BP\_mod3\_LSTM\_EGARCH.py​(BP\_mod3\_LSTM\_EGARCH)**

* **Objetivo**: Combina previsões LSTM com EGARCH para modelar retornos futuros e volatilidade.
* **Complexidade**: Alta, devido à integração de modelos avançados.
* **Dependências**: Dados históricos e pesos do portfólio.
* **Observação**: Poderia ser ajustado para modularidade, especialmente ao lidar com dados de entrada e saída.

**4. BP\_mod3\_main\_pipeline.py​(BP\_mod3\_main\_pipline)**

* **Objetivo**: Orquestrar os diferentes módulos, gerenciando simulações, otimização de portfólio e visualizações.
* **Complexidade**: Alta, pois exige a integração perfeita entre todos os módulos.
* **Dependências**: Simulações de Monte Carlo, otimização de portfólio e dashboard.
* **Observação**: Pode ser o passo final, consolidando todo o pipeline.

**5. BP\_mod3\_portfolio\_otimo\_mk.py**

* **Objetivo**: Otimizar o portfólio com restrições setoriais e outros critérios.
* **Complexidade**: Moderada.
* **Dependências**: Dados históricos dos ativos e setores.
* **Observação**: Serve como base para alimentar scripts dependentes do portfólio otimizado.

**6. BP\_mod3\_validacao.py​(BP\_mod3\_validacao)**

* **Objetivo**: Validar o portfólio gerado, simulando EGARCH e retornos anuais.
* **Complexidade**: Moderada.
* **Dependências**: Dados históricos e pesos do portfólio.
* **Observação**: Pode ser trabalhado após a geração do portfólio otimizado.

**Sugestão de Ordem de Trabalho**

1. **BP\_mod3\_portfolio\_otimo\_mk.py**:
   * Garantir que o portfólio otimizado esteja funcional para alimentar os demais scripts.
2. **BP\_mod3\_curto\_prazo.py**:
   * Simular retornos e validar sua funcionalidade com base no portfólio otimizado.
3. **BP\_mod3\_LSTM\_EGARCH.py**:
   * Refinar previsões de longo prazo e volatilidade, integrando ao pipeline.
4. **BP\_mod3\_validacao.py**:
   * Validar as métricas geradas, garantindo consistência com o portfólio.
5. **BP\_mod3\_dashboard.py**:
   * Criar o dashboard para visualização interativa dos resultados.
6. **BP\_mod3\_main\_pipeline.py**:
   * Consolidar e integrar todos os módulos em um fluxo orquestrado.

### Próximos Passos para retornar no dia 30/11/2024

**1. Análise e Validação do Portfólio Otimizado**

* **Tarefa**: Revisar o arquivo portfolio\_otimizado.csv, garantindo que:
  + Os pesos setoriais estejam alinhados com as restrições.
  + O retorno esperado anual continue consistente com os cálculos e alinhado ao benchmark.
* **Ação**: Gerar gráficos ou tabelas para validar visualmente a composição do portfólio.

**2. Simulações de Monte Carlo**

* **Tarefa**: Avaliar o comportamento do portfólio otimizado em diferentes cenários econômicos.
* **Ação**: Usar o script BP\_mod3\_validacao.py para realizar simulações e identificar possíveis vulnerabilidades no portfólio.

**3. Dashboard Final**

* **Tarefa**: Integrar os resultados em um dashboard interativo com métricas principais:
  + Retornos históricos e projetados.
  + Composição setorial e por ativo.
* **Ação**: Refinar o script BP\_mod3\_dashboard.py para exibir os dados do portfólio otimizado.

**4. Backtesting do Portfólio**

* **Tarefa**: Implementar e testar o portfólio contra dados históricos.
* **Ação**: Usar o script BP\_mod3\_main\_pipline.py para integrar os módulos e realizar backtesting.

**5. Aprimoramento de Modelos**

* **Tarefa**: Revisar e potencialmente aprimorar os modelos de otimização.
  + Considerar LSTM ou EGARCH no script BP\_mod3\_LSTM\_EGARCH.py para análises avançadas de volatilidade e previsibilidade.

**Conclusão**

Esses passos nos levarão a um sistema completo, que não só otimiza, mas também valida e apresenta o portfólio de forma clara e profissional. Amanhã podemos começar pelo **passo 1** para revisar e validar o portfólio otimizado.

Tenha uma ótima noite de descanso, e amanhã retomamos com energia total! 🚀✨

**domingo, 1 de dezembro de 2024**

## Retomando e dando continuidade ao projeto

**Próximos Passos**

Com base no que já foi feito e no roteiro acima, sugiro avançar na seguinte ordem:

1. **Simulação de Monte Carlo**:
   * Usar os dados de retornos históricos para gerar cenários futuros (curto e longo prazo).
   * Implementar a divisão dos dados e validar os resultados.
   * Gerar gráficos para comparar os cenários (histograma de retornos esperados, gráficos de densidade).
2. **Análise de Candlesticks**:
   * Identificar padrões nos dados históricos de preços.
   * Desenvolver uma função que sinalize possíveis momentos de compra.
3. **Planejar e Prototipar ML**:
   * Após a simulação de Monte Carlo e candlesticks, iniciar com modelos simples de previsão.

**Análise da Proposta**

1. **Divisão dos Dados**:
   * **Treinamento**: Dados históricos do início até o final de setembro de 2024.
   * **Validação**: Dois meses finais (outubro e novembro de 2024).
   * Essa divisão é ideal, pois permite testar a robustez da Simulação de Monte Carlo (MC) com dados reais que já conhecemos.
2. **Validação do Simulador**:
   * Projetaremos os retornos para os dois meses de validação (outubro e novembro) com base nos dados de treinamento.
   * Compararemos as projeções com os dados reais, verificando a proximidade com os percentis 5%, 50% e 95%.
   * Essa validação é essencial para ajustar o modelo antes de avançar para projeções de curto e longo prazo.
3. **Inclusão do Portfólio e Benchmark**:
   * **Portfólio Otimizado**:
     + Plotar o comportamento do portfólio otimizado dentro dos cenários projetados pela MC.
     + Identificar qual cenário é mais representativo (e.g., 50%) para o portfólio.
   * **Índice BOVESPA**:
     + Adicionar o benchmark (^BVSP) para comparar seu comportamento com os cenários de MC.
     + Isso oferece insights sobre como o mercado geral se alinha com os cenários projetados.
4. **Gráficos e Visualizações**:
   * Um gráfico que inclui:
     + Cenários projetados (com destaque para os percentis).
     + Linha representando o portfólio otimizado.
     + Linha representando o índice BOVESPA.
   * Isso ajudará a visualizar o alinhamento e as discrepâncias entre os cenários, o portfólio e o benchmark.
5. **Próximos Passos (após validação)**:
   * Uma vez validado, podemos gerar cenários futuros de curto (6 meses) e longo prazo (5 anos).
   * Ajustar o modelo com base na performance observada durante a validação.

**Plano de Implementação**

**1. Preparação dos Dados**

* Carregar dados históricos e dividir em treino (até setembro de 2024) e validação (outubro e novembro de 2024).

**2. Simulação de Monte Carlo**

* Usar dados de treino para calcular média e desvio padrão dos retornos.
* Gerar cenários de MC para os dois meses de validação.
* Calcular os percentis 5%, 50% e 95% para cada dia dos dois meses.

**3. Comparação com Dados Reais**

* Comparar os preços simulados com os preços reais de outubro e novembro.
* Identificar o cenário mais próximo da realidade.

**4. Visualização**

* Gráfico combinando:
  + Cenários simulados (áreas de confiança para 5%, 50% e 95%).
  + Linha do portfólio otimizado.
  + Linha do índice BOVESPA.

**5. Ajustes**

* Avaliar a precisão do modelo e ajustar, se necessário, antes de avançar para as projeções futuras.

**Benefícios do Método**

1. **Validação Rigorosa**:
   * Testa o simulador em um ambiente controlado antes de usá-lo para projeções longas.
2. **Insights Aprofundados**:
   * Permite entender como o portfólio otimizado se comporta em relação ao mercado e aos cenários de MC.
3. **Base Sólida para Projeções Futuras**:
   * Ajusta o modelo com base em dados reais antes de avançar.

**Etapa 1: Validação dos Scripts Existentes**

* **Objetivo**: Garantir que os scripts fornecidos funcionam corretamente e produzem os resultados esperados.
* **Ações**:
  1. Executar cada script em sequência com os dados fornecidos.
  2. Validar os arquivos intermediários gerados em relação ao esperado.
  3. Identificar ajustes necessários para a execução fluida.

**Etapa 2: Adição de Simulação de Monte Carlo**

* **Objetivo**: Incluir simulações para analisar risco e projeções de retorno para diferentes portfólios.
* **Ações**:
  1. Criar um módulo específico para a simulação, reutilizando dados já preparados.
  2. Testar cenários com diferentes níveis de volatilidade e horizonte de tempo.
  3. Gerar visualizações (gráficos) para interpretar os resultados.

**Etapa 3: Estruturação para Machine Learning**

* **Objetivo**: Planejar e implementar módulos para previsão de desempenho dos ativos.
* **Ações**:
  1. Definir o objetivo (ex.: previsão de preços ou retornos).
  2. Preparar os dados para o modelo (normalização, features relevantes).
  3. Implementar modelos básicos, como regressões ou redes neurais simples.

**Etapa 4: Documentação e Padronização**

* **Objetivo**: Tornar os scripts mais acessíveis e fáceis de expandir.
* **Ações**:
  1. Refatorar os scripts para seguir padrões consistentes (ex.: docstrings detalhados).
  2. Criar um guia de uso para cada módulo.
  3. Estruturar pastas e arquivos de forma lógica.

**Próximos Passos - Etapa 2: Introdução de Simulações de Monte Carlo**

1. **Objetivo**:
   * Simular cenários de desempenho do portfólio.
   * Analisar volatilidade, possíveis ganhos e perdas.
   * Visualizar distribuições de resultados com gráficos.
2. **Plano para o Script BP\_mod5\_monte\_carlo.py**:
   * **Entrada**: Usaremos os dados já preparados (ex.: prepared\_data.csv ou portfolio\_otimizado.csv).
   * **Simulação**:
     + Gerar várias trajetórias de retornos baseadas na distribuição histórica.
     + Calcular métricas como VaR (Value at Risk) e CVaR (Conditional Value at Risk).
   * **Saída**:
     + Tabelas e gráficos com a distribuição dos retornos simulados.
3. **Documentação**:
   * Manteremos o padrão de documentação e logging para consistência.

**O que será feito:**

1. **Simulação de Monte Carlo**:
   * **Trajetórias**: Gerar várias simulações de retornos para o portfólio ótimo, baseando-se nas distribuições históricas de retornos (como média e volatilidade dos ativos).
   * **Percentis**: Calcular as trajetórias de 5%, 50% (mediana) e 95%, que representarão os cenários pessimista, mediano e otimista.
2. **Análise Comparativa**:
   * **Métricas para o Portfólio Ótimo**:
     + Retorno esperado.
     + Risco (volatilidade).
     + Sharpe Ratio.
   * **Métricas para Monte Carlo**:
     + Distribuição de resultados (gráficos densidade e dispersão).
     + Valor no Risco (VaR) e Valor no Risco Condicional (CVaR).
3. **Visualização**:
   * Comparar diretamente as métricas do portfólio ótimo com as trajetórias do Monte Carlo.
   * Visualizar graficamente os cenários de 5%, 50% e 95% para prever o comportamento do portfólio em diferentes condições.

**Proposta de Horizontes para MC:**

1. **Curto Prazo (6 meses)**:
   * Período ideal para avaliar volatilidade e impactos de curto prazo, como flutuações de mercado.
   * Útil para verificar se o portfólio ótimo é resiliente em condições adversas.
2. **Longo Prazo (5 anos)**:
   * Permite avaliar a estabilidade do portfólio em termos de retornos e comportamento no horizonte de investimentos mais amplos.
   * Mostra a capacidade do portfólio em superar momentos de crise e capturar crescimento.

**Como Simular:**

* As trajetórias de Monte Carlo serão calculadas para cada horizonte de tempo.
* Em cada caso, os retornos diários (ou mensais, dependendo da granularidade escolhida) serão simulados até o período especificado.
* A partir dessas simulações, podemos calcular e visualizar os cenários de 5%, 50% e 95% para os dois horizontes.

1. **Cálculo do Crescimento do Portfólio Ótimo**:
   1. Usaremos os dados do portfólio ótimo (ex.: pesos dos ativos e retornos históricos) para calcular o crescimento acumulado ao longo do tempo.
   2. Isso será feito para o mesmo horizonte de tempo das simulações (6 meses e 5 anos).
2. **Adição ao Gráfico**:
   1. Adicionaremos uma linha ao gráfico representando o crescimento acumulado do portfólio ótimo.
   2. A linha será destacada para facilitar a visualização em relação às trajetórias simuladas.

**Proposta para o Script BP\_mod6\_ML.py**

1. **Objetivo Geral**:
   * Previsão de preços ou retornos dos ativos com base nos dados históricos e em features derivadas.
2. **Estrutura do Script**:
   * **Preparação dos Dados**:
     + Normalização, criação de janelas temporais para aprendizado sequencial e separação em conjuntos de treino/teste.
   * **Modelos de Machine Learning**:
     + Começaremos com um modelo simples (ex.: regressão linear) para estabelecer uma baseline.
     + Expandiremos para modelos mais complexos, como redes neurais simples (MLP) ou baseadas em sequência (LSTM ou GRU), se necessário.
   * **Avaliação**:
     + Métricas de desempenho, como RMSE (Root Mean Squared Error), MAE (Mean Absolute Error) e R².
3. **Flexibilidade e Modularidade**:
   * Estruturar o script para facilitar a adaptação a diferentes ativos e objetivos (ex.: previsão de curto ou longo prazo).

**Solução Proposta:**

1. **Tratar os Arquivos Durante a Execução**:
   * Sempre que possível, os dados serão tratados diretamente na memória, sem salvar arquivos intermediários.
   * Isso reduz a necessidade de múltiplos arquivos e mantém o fluxo mais limpo.
2. **Gerar Arquivos Novos Quando Necessário**:
   * Caso o tamanho dos dados ou etapas complexas justifiquem, criaremos arquivos temporários com nomes claros e sugestivos, como:
     + normalized\_data.csv para dados normalizados.
     + training\_data.csv para dados de treino, caso seja útil para análises futuras.
   * Esses arquivos serão armazenados em uma pasta separada, como DADOS/ML, para organização.
3. **Princípio de Reprodutibilidade**:
   * Todos os passos, como normalização ou divisão em treino/teste, serão reproduzíveis, garantindo que os resultados possam ser replicados sem depender de arquivos intermediários.

## IMPORTANTE para ML previsão de curto prazo,

**1. LSTM para Volatilidade**

* **Vantagens:**
  + O LSTM (Long Short-Term Memory) é uma rede neural recorrente (RNN) ideal para lidar com padrões não lineares e dados temporais altamente voláteis.
  + Pode identificar tendências mais complexas de volatilidade ao longo do tempo, mesmo quando os dados são ruidosos ou apresentam mudanças abruptas.
  + Permite capturar padrões de longo prazo na volatilidade que o EGARCH pode não detectar.
* **Como Aplicar:**
  + Treinar o modelo LSTM com a **volatilidade histórica condicional** como alvo (usando EGARCH para calcular as volatilidades históricas).
  + Usar o LSTM para projetar a **volatilidade futura** nos próximos 6 meses.
  + Utilizar essas previsões de volatilidade para corrigir ou ajustar a escala de retornos projetados pelo EGARCH.

**2. EGARCH para Projeções**

* **Vantagens:**
  + EGARCH é muito eficiente para modelar os retornos e capturar heterocedasticidade (mudanças na variância condicional ao longo do tempo).
  + Sua abordagem matemática robusta ainda é ideal para capturar choques de volatilidade e retornos médios ao longo do tempo.
  + Quando combinado com previsões de volatilidade mais precisas (via LSTM), pode produzir retornos projetados mais confiáveis.
* **Como Aplicar:**
  + Ajustar o modelo EGARCH aos **retornos logarítmicos históricos**, como já implementado.
  + Utilizar as previsões de volatilidade do LSTM para substituir ou ajustar as estimativas de volatilidade geradas pelo EGARCH.

**3. Fluxo de Trabalho Integrado**

1. **Etapa 1**: Ajustar o modelo EGARCH para os retornos logarítmicos e calcular a volatilidade condicional histórica.
2. **Etapa 2**: Treinar o LSTM com as volatilidades históricas (saída do EGARCH) para aprender padrões não lineares.
3. **Etapa 3**: Usar o LSTM para projetar a volatilidade futura.
4. **Etapa 4**: Combinar os retornos projetados pelo EGARCH com as volatilidades projetadas pelo LSTM.
5. **Etapa 5**: Exibir os resultados em um dashboard unificado, destacando as projeções de retornos e volatilidade.

**Benefícios dessa Abordagem**

* **Precisão Aprimorada**:
  + O LSTM pode lidar melhor com dados voláteis e não lineares, complementando a matemática linear do EGARCH.
* **Flexibilidade**:
  + Permite explorar tanto padrões curtos quanto longos nos dados de volatilidade.
* **Melhor Interpretação**:
  + Combinando os dois métodos, conseguimos melhores insights sobre riscos (via volatilidade) e retornos esperados.

**Desafios Potenciais**

1. **Treinamento do LSTM**:
   * O modelo LSTM exige mais dados e poder computacional para ajustar os parâmetros de forma eficaz.
2. **Integração de Modelos**:
   * Combinar as previsões de LSTM com EGARCH exige cuidados, especialmente para garantir consistência temporal.
3. **Validação**:
   * Precisamos garantir que a combinação dos modelos seja rigorosamente validada com dados fora da amostra.

**Próximos Passos**

Se você concordar, posso começar a implementar essa abordagem híbrida. Precisaremos:

1. Configurar o treinamento do LSTM para volatilidade.
2. Ajustar a integração das previsões do LSTM ao EGARCH.
3. Refatorar o dashboard para apresentar as previsões híbridas.

**02/12/2024**

## Retomando o projeto com foco em ML

* **Centralização**: Um único ponto para reunir os resultados de todas as etapas (Monte Carlo, Machine Learning, e métricas de desempenho).
* **Manutenção Simplificada**: Facilitará futuras modificações ou expansões no dashboard.
* **Fluxo Integrado**: Garantirá que todas as informações (previsões, retornos históricos, comparação com benchmark) sejam apresentadas de forma coesa e bem organizada.

**Proposta para o Script Final do Dashboard:**

1. **Estruturação do Script**:
   * **Seção 1: Configurações**:
     + Importar bibliotecas essenciais (Dash, Plotly, Pandas).
     + Definir parâmetros globais (horizonte de previsão, benchmark, etc.).
   * **Seção 2: Integração de Resultados**:
     + Incorporar os dados gerados nos passos anteriores:
       - **Simulações de Monte Carlo**: Exibir cenários e métricas como VaR e CVaR.
       - **Previsões de ML**: Gráficos e tabelas comparando os retornos previstos com o benchmark.
       - **Pesos do Portfólio**: Mostrar a composição do portfólio otimizado.
   * **Seção 3: Gráficos Interativos**:
     + Gráficos de distribuição, previsões e evolução dos retornos ao longo do tempo.
     + Comparações diretas entre o portfólio e o benchmark.
   * **Seção 4: Métricas Detalhadas**:
     + Tabelas detalhadas de desempenho (ex.: RMSE, R²) para as previsões.
     + Análise de como o portfólio otimizado supera o benchmark.
   * **Seção 5: Layout e Callbacks**:
     + Definir o layout do dashboard (títulos, estilos, tabelas).
     + Adicionar callbacks para interatividade (ex.: escolha de horizonte de tempo, seleção de ativos).
2. **Fluxo de Dados**:
   * Garantir que o script carregue os dados do pipeline consolidado (arquivo filtrado e pesos atribuídos).
   * Atualizar dinamicamente com novos resultados (Monte Carlo ou previsões futuras).
3. **Resultados no Dashboard**:
   * **Gráficos**:
     + Simulações de Monte Carlo.
     + Previsões de retornos para 6 meses e 5 anos.
     + Evolução dos retornos do portfólio versus benchmark.
   * **Tabelas**:
     + Pesos dos ativos no portfólio.
     + Métricas de desempenho (ex.: VaR, CVaR, retorno esperado).
     + Comparação direta com o benchmark.

**Próximos Passos:**

1. Criar o script para consolidar os dados com os pesos.
2. Estruturar o novo script do dashboard com todas as funcionalidades integradas.
3. Garantir que o layout seja interativo, com insights claros para tomada de decisão.

**Abordagem Correta:**

1. **Treinamento com Dados Históricos**:
   * Usar retornos históricos (passados) como **target** no treinamento.
   * Incluir features relevantes como:
     + Retornos passados (1M, 6M, etc.).
     + Volatilidade (rolling windows).
     + Correlação entre ativos.
2. **Projeção para o Futuro**:
   * Depois de treinar o modelo, usar as **features mais recentes** para prever os próximos retornos esperados.
   * A previsão pode ser estendida para múltiplos períodos usando projeção iterativa (ex.: prever 1 mês à frente, depois usar esse dado para projetar 2 meses, e assim por diante).
3. **Integração com Portfólio Otimizado**:
   * Incorporar os pesos do portfólio otimizado no modelo, para que as previsões considerem a alocação ideal e suas variações esperadas.

**Fluxo Ajustado para Projeção Futura:**

1. **Treinamento**:
   * Treinar o modelo com **features passadas** e **retornos históricos** como targets.
2. **Predição**:
   * Usar os **dados mais recentes disponíveis** para prever os retornos esperados nos próximos períodos.
   * O modelo aprende a prever **comportamentos esperados** com base no histórico.
3. **Projeção Iterativa (Opcional)**:
   * Para prever múltiplos períodos (6 meses ou 5 anos), usar uma abordagem iterativa:
     + Prever o próximo período.
     + Usar essa previsão para calcular novas features e prever o período seguinte.

**Tarefa Principal: Comportamento do Portfólio Ótimo**

1. **Base de Dados**:
   * **historical\_data\_filtered.csv**: Contém os dados históricos dos ativos (2014-12-01 em diante).
   * **portfolio\_otimizado.csv**: Define os pesos dos ativos no portfólio ótimo.
2. **Objetivo**:
   * Construir uma nova base de dados com uma **coluna adicional portfolio\_otimo** que calcule o valor do portfólio ótimo em cada registro (linha) usando os preços dos ativos e seus respectivos pesos.
3. **Treinamento do Modelo**:
   * Usar essa nova base como entrada para treinar um modelo que aprenda o comportamento do portfólio ótimo.
4. **Projeção Futura**:
   * Usar o modelo treinado para projetar o comportamento do portfólio nos próximos 6 meses e 5 anos.

**Divisão em Dois Scripts**

**Script 1: BP\_mod6\_prepare\_portfolio.py**

1. **Entrada**:
   * historical\_data\_filtered.csv: Dados históricos dos ativos.
   * portfolio\_otimizado.csv: Pesos dos ativos.
2. **Processamento**:
   * Para cada linha da base histórica, calcular: Portfoˊlio Oˊtimo=∑(Prec¸o do Ativo×Peso do Ativo)\text{Portfólio Ótimo} = \sum (\text{Preço do Ativo} \times \text{Peso do Ativo})Portfoˊlio Oˊtimo=∑(Prec¸​o do Ativo×Peso do Ativo)
   * Adicionar o resultado na coluna portfolio\_otimo.
3. **Saída**:
   * Salvar a nova base como portfolio\_comportamento.csv.

**Script 2: BP\_mod7\_train\_and\_project.py**

1. **Entrada**:
   * portfolio\_comportamento.csv: Base consolidada com o comportamento do portfólio.
2. **Treinamento**:
   * Usar o histórico para treinar um modelo que aprenda o comportamento do portfólio ótimo.
3. **Projeção**:
   * Usar as features mais recentes para projetar o comportamento esperado do portfólio nos próximos 6 meses e 5 anos.
4. **Saída**:
   * Resultados das previsões:
     + Salvar as projeções em um arquivo (ex.: projecoes\_portfolio.csv).
     + Gerar gráficos e métricas de desempenho.

**Plano para o Próximo Script (BP\_mod7\_train\_and\_project.py)**

1. **Objetivo Geral**:
   * Treinar um modelo de Machine Learning para aprender o comportamento do portfólio baseado nos dados históricos.
   * Realizar projeções futuras para os próximos **6 meses** e **5 anos**.
2. **Etapas do Script**:

**1. Preparação dos Dados**

* + Carregar a base portfolio\_comportamento.csv.
  + Dividir os dados em **treinamento, teste e validação**:
    - Reservar os últimos **60 dias** para validação (outubro e novembro).
    - Usar os dados restantes para treinamento e teste.

**2. Treinamento do Modelo**

* + Usar algoritmos de regressão para modelar o comportamento do portfólio:
    - Ex.: Random Forest, XGBoost, ou até LSTM (para dados temporais mais avançados).
  + Validar o modelo com o conjunto de teste.

**3. Projeção Futura**

* + Usar os dados mais recentes (últimos registros da base) para prever:
    - Comportamento esperado do portfólio nos próximos **6 meses**.
    - Comportamento esperado nos próximos **5 anos**.
  + Salvar as projeções e gerar gráficos comparativos com o histórico.

**4. Saída**

* + Gerar as seguintes saídas:
    - Arquivo projecoes\_portfolio.csv contendo as projeções.
    - Gráficos comparando as projeções com o histórico.

**Exemplos Práticos de Uso de ML em Previsões Financeiras:**

* **Previsão de Preço de Ações**: Usando dados históricos de preço, volume de negociação, indicadores econômicos, e até mesmo **notícias financeiras** (usando técnicas de NLP para análise de sentimento), é possível treinar modelos de ML para prever a direção futura do preço das ações.
* **Risco de Crédito**: Modelos de ML podem ser usados para prever a probabilidade de inadimplência de um cliente com base em seus históricos financeiros, comportamentos de pagamento, etc.
* **Análise de Sentimento**: ML pode ser usado para analisar o sentimento em torno de empresas ou mercados a partir de dados não estruturados (como notícias ou postagens em redes sociais), o que pode afetar as previsões do mercado financeiro.
* **Previsão de Volatilidade**: Técnicas de ML podem ser usadas para prever a volatilidade do mercado com base em dados históricos de preços, ajudando a entender o risco.

**Aprendizado Supervisionado x Não Supervisionado**

1. **Aprendizado Supervisionado**:
   * No aprendizado supervisionado, o modelo **aprende a partir de dados rotulados**, ou seja, temos **entradas (features)** e **saídas (rótulos ou labels)** que nos ajudam a treinar o modelo.
   * No caso da previsão de preços de ações, os **dados históricos de preços de ações** servem como **entradas** (features), enquanto o **preço futuro das ações** é a **saída** (rótulo ou label). O modelo "aprende" a fazer previsões com base nos padrões históricos, ajustando-se para prever o valor futuro.

**Exemplo**:

* + Dado o **preço das ações de hoje** e dos últimos dias (como variáveis de entrada), o modelo tenta prever o **preço da ação amanhã ou em 6 meses**.
  + **Modelo**: O modelo tenta **minimizar o erro** entre o valor predito (futuro) e o valor real (observado).

**Tipos de modelos usados**:

* + **Regressão Linear**: Simples, para prever um valor contínuo.
  + **Redes Neurais (MLP, LSTM)**: Modelos mais complexos, especialmente úteis para séries temporais como o preço das ações.
  + **Árvores de Decisão**: Para tarefas de regressão ou classificação.

1. **Aprendizado Não Supervisionado**:
   * No aprendizado não supervisionado, o modelo **não tem rótulos** e precisa **encontrar padrões ou agrupamentos** nos dados por conta própria.
   * **Exemplo**: Usar **clustering** para agrupar ações em diferentes categorias, com base em características como volatilidade, tipo de indústria, etc. No entanto, esse tipo de aprendizado não é utilizado para **prever preços**, mas para **encontrar padrões** nos dados sem rótulos.

**Exemplo de Algoritmos**:

* + **K-means clustering**: Agrupamento de ações baseadas em suas semelhanças.
  + **PCA (Análise de Componentes Principais)**: Redução de dimensionalidade para identificar as principais variáveis que explicam a variação nos preços.

**Classificação no Contexto de Previsão de Preços de Ações**

Embora você esteja tratando da **previsão de preços** (o que é um problema de **regressão**), a **classificação** pode ser aplicada em outros cenários, como:

1. **Prever se o preço de uma ação vai subir ou cair**:
   * Em vez de prever o valor exato, você pode **transformar o problema em uma tarefa de classificação**, onde você prevê **se o preço vai aumentar ou diminuir**.
   * Você criaria duas classes: **"subir"** ou **"cair"**.
   * **Exemplo**: Se o preço da ação em t+1 for maior que o preço da ação em t, você classifica como "subir", caso contrário "cair".
2. **Classificação de ações com base em desempenho**:
   * Outra forma de **classificação** seria **classificar as ações em grupos** como: **"boas", "neutras" e "ruins"** com base em certos indicadores financeiros (ex: lucros, volume de transações, etc.), usando um algoritmo como **Random Forest** ou **SVM (Máquinas de Vetores de Suporte)**.

**Resumo**

* **Para previsão de preços de ações**: estamos **usando aprendizado supervisionado** (onde temos dados históricos e o preço futuro como rótulo). O objetivo é prever um valor contínuo (preço), o que caracteriza um **problema de regressão**.
* **Classificação** pode ser aplicada caso você queira prever a **direção do preço** (se vai subir ou cair), mas essa não é a tarefa principal no seu caso, onde o foco é prever o **valor do preço**.

Com base na imagem que você enviou, parece que você possui dados históricos de preços de ações (como VALE3.SA, PETR4.SA, etc.) ao longo de um período de tempo (de 2014 a 2024). Esses dados incluem informações diárias de preços para essas ações, o que é uma excelente base para aplicar aprendizado supervisionado em um modelo preditivo, como regressão.

Para prever preços futuros de ações com aprendizado supervisionado, normalmente:

1. **Variáveis independentes (features)**: Usar as colunas de preços das ações como variáveis de entrada para o modelo. Por exemplo, os preços de ações em dias anteriores podem servir como entrada para prever o preço futuro.
2. **Variável dependente (target)**: A variável alvo que você deseja prever. Nesse caso, o preço da ação para o próximo dia, por exemplo.

Uma abordagem comum seria:

* Criar variáveis adicionais com base nos preços, como a variação percentual diária, médias móveis, volatilidade, etc.
* Utilizar essas variáveis como características para treinar um modelo preditivo, como uma regressão linear, modelos baseados em árvores (ex: XGBoost) ou redes neurais, dependendo da complexidade e das características dos dados.

**Etapas que você pode seguir:**

1. **Pré-processamento dos dados**:
   * Tratar valores ausentes.
   * Criar features derivadas, como variação percentual diária, médias móveis, etc.
2. **Divisão dos dados**:
   * Separar os dados em treino e teste.
   * Treinar o modelo com os dados históricos e testar em dados futuros.
3. **Modelo preditivo**:
   * Usar modelos de aprendizado supervisionado (como regressão) para prever o preço da ação no futuro (por exemplo, no próximo dia ou próximo mês).

Você está absolutamente certo ao observar que a situação dos dados financeiros é diferente de outros conjuntos de dados mais tradicionais, como o das flores de íris. Na base de dados das flores de íris, você possui variáveis independentes (como o comprimento e a largura das pétalas e sépalas) e uma variável alvo (a espécie da flor), permitindo uma tarefa de **classificação**.

No seu caso, ao trabalhar com **dados financeiros**, você tem uma série temporal dos preços das ações ao longo do tempo, com cada linha representando um pregão (com dados como a data e o preço de fechamento dos ativos). A principal diferença aqui é que, sem uma variável alvo explícita como "espécie da flor", você está lidando com um **problema de previsão de séries temporais**.

**Como lidar com a falta de uma variável alvo (target variable)?**

Nos dados financeiros, ao invés de uma variável alvo claramente definida (como no caso das flores), a variável alvo normalmente seria o **valor futuro** de uma variável de interesse (no caso, o preço de fechamento de uma ação em um futuro próximo). Ou seja, o objetivo é **prever o preço de fechamento do ativo para um determinado período no futuro**, com base nos dados passados. Esse tipo de problema é tipicamente abordado como uma **previsão de séries temporais**.

**Como podemos gerar uma variável alvo?**

Em problemas de previsão de séries temporais (como o de previsão de preço de ações), a variável alvo é **gerada a partir dos próprios dados históricos**, ou seja, o que você precisa fazer é definir o "tempo futuro" que deseja prever. Por exemplo:

* **Previsão para o próximo dia**: Se você deseja prever o preço da ação no próximo dia, a variável alvo seria o preço de fechamento da ação no próximo pregão. Para isso, você cria uma nova coluna com o preço do próximo dia.
* **Previsão para 6 meses (curto prazo)**: Você pode criar uma variável alvo para o preço de fechamento da ação 6 meses à frente.
* **Previsão para 5 anos (longo prazo)**: De maneira semelhante, a variável alvo pode ser o preço de fechamento da ação em 5 anos.

**Como isso funciona na prática?**

Vamos ilustrar com um exemplo:

1. **Colunas de entrada (features)**:
   * **Preços históricos**: preço de fechamento dos ativos (como VALE3.SA, PETR4.SA, etc.).
   * **Indicadores técnicos** (se aplicável): como médias móveis, variação percentual, volatilidade, etc.
2. **Variável alvo (target)**:
   * O preço de fechamento em uma data futura (por exemplo, o preço 1 dia após a data em que você está analisando).
   * Ou, se preferir uma previsão de longo prazo, pode-se calcular o preço da ação em uma data futura (6 meses ou 5 anos) e usar isso como alvo.

**Diferença entre aprendizado supervisionado e séries temporais:**

* **Aprendizado supervisionado**: Em problemas de classificação ou regressão, temos variáveis independentes (features) e uma variável dependente (target). No caso dos preços das ações, a variável dependente seria o preço futuro que queremos prever.
* **Séries temporais**: É um tipo específico de aprendizado supervisionado, onde as variáveis de entrada são os dados de séries temporais (ou seja, os preços passados). A principal diferença é que, no aprendizado supervisionado clássico, os dados podem ser independentes, enquanto em séries temporais, os dados têm dependências temporais (ou seja, o preço de hoje depende do preço de ontem e de dias anteriores).

**Conclusão:**

Portanto, para o seu caso específico, é totalmente possível usar **aprendizado supervisionado** para prever o comportamento futuro dos preços das ações. O que você precisa fazer é gerar uma variável alvo a partir dos dados históricos, que será o **preço futuro** de uma ação (no próximo dia, 6 meses à frente, etc.). Depois, é só treinar um modelo de aprendizado supervisionado para prever essa variável alvo com base nas variáveis de entrada (preços históricos e possivelmente outros indicadores).

**Sugestões para Ajustar o Modelo:**

1. **Gerar a variável alvo corretamente**: A variável alvo precisa ser ajustada para refletir o valor do ativo 6 meses à frente, e não apenas fazer previsões com base nos dados atuais. Precisamos garantir que os dados sejam "projetados" corretamente para esse futuro. O processo seria algo como gerar os preços de 6 meses à frente com base nas séries temporais, e não tentar prever diretamente.
2. **Modelo de Séries Temporais**: Em vez de utilizar **Regressão Linear**, pode ser necessário um modelo de **Séries Temporais** mais sofisticado, como **ARIMA** ou **LSTM** (para redes neurais), para lidar com previsões de séries temporais com dados históricos.
3. **Visualização do Futuro**: O gráfico de previsão precisa ser ajustado para mostrar a tendência esperada para os próximos 6 meses, em vez de simplesmente projetar os valores históricos. Podemos criar um gráfico mais alinhado com a projeção real para os próximos meses.

A divisão proposta em três scripts para o seu projeto é excelente, pois separa claramente as responsabilidades e facilita a organização e a manutenção do código. Aqui está como podemos estruturar:

1. **Script de Preparação dos Dados**:
   * Responsável por carregar, limpar e transformar os dados no formato adequado para os modelos de previsão.
   * Usa o caminho de dados configurado no script de configuração (BP\_mod1\_config.py) para salvar os dados preparados.
   * Implementará técnicas de feature engineering, como cálculo de indicadores técnicos.
2. **Script de Previsão de Curto Prazo (6 meses)**:
   * Implementará modelos específicos para prever os próximos 6 meses com base nos dados preparados.
   * Foco em modelos como ARIMA, Prophet ou regressão linear com recursos temporais.
3. **Script de Previsão de Longo Prazo (5 anos)**:
   * Usará técnicas como LSTMs ou Transformers para séries temporais, priorizando capturar tendências de longo prazo.

Com o script de configuração (BP\_mod1\_config.py), os diretórios e parâmetros globais estão centralizados e já existe um padrão bem documentado. Isso facilita a integração entre os scripts.

**Próximos Passos**

1. Configurar o **script de preparação dos dados** para processar os dados enviados.
2. Definir os pipelines iniciais para curto e longo prazo com base nos dados processados.

Na preparação dos dados, o objetivo principal é transformar as séries temporais brutas em um formato que possa ser usado diretamente pelos modelos de machine learning ou séries temporais. Aqui está um resumo das etapas que planejo realizar:

**Etapas de Preparação dos Dados**

1. **Carregar e Validar os Dados**:
   * Garantir que os dados estão no formato esperado, conforme descrito na sua documentação: completos, sem valores ausentes e alinhados aos dias úteis.
2. **Converter a Coluna de Data**:
   * Transformar a coluna Date no tipo datetime para facilitar operações temporais.
3. **Ordenar os Dados**:
   * Ordenar cronologicamente para evitar inconsistências.
4. **Feature Engineering**:
   * Adicionar colunas úteis para os modelos, como:
     + **Retornos logarítmicos diários**: log⁡(PtPt−1)\log(\frac{P\_t}{P\_{t-1}})log(Pt−1​Pt​​), onde PtP\_tPt​ é o preço do ativo no dia ttt.
     + **Médias móveis** (curta, média e longa): para capturar tendências.
     + **Volatilidade** (desvio-padrão dos retornos em uma janela móvel).
     + **Indicadores Técnicos** (ex.: RSI, MACD) para enriquecer os dados.
5. **Normalização ou Padronização**:
   * Transformar os valores das séries para escalas comparáveis, caso necessário para modelos baseados em aprendizado de máquina.
6. **Divisão em Conjuntos de Treino e Validação**:
   * Dividir os dados para validação temporal com uma janela deslizante.
7. **Salvar os Dados Preparados**:
   * Salvar os dados processados em um arquivo (prepared\_data.csv) no diretório configurado no script BP\_mod1\_config.py.

**Objetivo Final**

Transformar os dados históricos para que eles sejam:

* **Limpos** e prontos para análise.
* **Ricos em informações** derivadas de indicadores financeiros e estatísticos.
* **Compatíveis** com os diferentes tipos de modelos que serão usados nos scripts de curto e longo prazo.

Sim, você está absolutamente correto! Ao calcular os retornos logarítmicos, é possível obter valores nulos ou indefinidos em situações específicas, como:

1. **Preços zerados ou negativos**: Embora incomum para ativos financeiros, se ocorrer, o logaritmo será indefinido.
2. **Primeiro registro na série**: Não há preço anterior para calcular o retorno logarítmico.

**Como Lidar com Isso**

1. **Preços Zerados ou Negativos**:
   * **Validação Prévia**: Verificar se existem preços negativos ou zero e tratá-los, por exemplo, marcando-os como erros ou substituindo-os (caso sejam exceções pequenas).
2. **Primeiro Registro**:
   * Definir explicitamente que o retorno logarítmico do primeiro registro será nulo ou zero.
   * Na prática, o primeiro valor será ignorado, pois normalmente os modelos não precisam dessa observação isolada.
3. **Valores Faltantes Após o Cálculo**:
   * Remover ou preencher os valores ausentes com estratégias apropriadas, dependendo do contexto.

**Minha Proposta**

Durante a etapa de preparação:

* **Validarei os dados para identificar inconsistências (valores zero ou negativos)**.
* **Gerarei os retornos logarítmicos cuidadosamente**, com lógica para tratar o primeiro registro e quaisquer inconsistências.
* **Documentarei e logarei quaisquer ajustes feitos**, para que você tenha clareza de como os dados foram transformados.

**Adaptação ao Cenário**

1. **Valores Zerados nos Retornos Logarítmicos**:
   * Quando não houver variação no valor do ativo entre dias consecutivos (preço do dia t é igual ao preço do dia t−1t) o logaritmo será zero, pois:

Diagrama

Descrição gerada automaticamente

1. **Manutenção da Integridade Temporal**:
   * Mesmo para dias sem variação, a informação é relevante. Portanto, ao invés de excluir esses registros, vou mantê-los com um retorno de **zero**.
2. **Tratamento do Primeiro Registro**:
   * Para o primeiro dia da série (onde não há preço anterior), será atribuído um valor nulo ou explicitamente ignorado pelos modelos.
3. **Documentação do Processo**:
   * Todos os passos e escolhas feitas serão registrados, incluindo como os retornos foram tratados.

**05/12/2024 15:10**

**Divisão dos Dados**:

* **Treinamento e Teste**:
  + Usar **80%** da base total (exceto os dois últimos meses de pregões) para treinamento.
  + Usar **20%** da base restante para testes.
* **Validação**:
  + Os **dois últimos meses de pregões** (aproximados) da base servem como período de validação.

**Modelo LSTM**:

* **Treinar o modelo** com o conjunto de treinamento.
* **Avaliar o modelo** com o conjunto de teste.
* **Validar o modelo** comparando previsões no período de validação com os dados reais.

**Previsão Futura**:

* Usar o modelo treinado para projetar **6 meses além do período de validação** (aproximadamente 120 dias úteis).

**Pontos Críticos a Garantir**

1. **Cálculo correto dos dois meses de pregões**:
   * Basear-se no campo Date para identificar o índice de início do período de validação.
   * Verificar se o número de registros no período de validação é suficiente para a lógica do modelo.
2. **Separação clara dos conjuntos**:
   * Treinamento e teste devem excluir o período de validação.
   * Garantir que look\_back seja ajustado ao tamanho de cada conjunto, evitando inconsistências.
3. **Lógica do look\_back**:
   * Deve ser consistente e derivado do tamanho dos dados disponíveis.
4. **Previsão futura coerente**:
   * Usar o final do período de validação como ponto de partida para projetar os próximos 6 meses.