**Tech Challenge 4 – Documentação Técnica**

**1. Introdução**

Este projeto apresenta uma aplicação que integra dois modelos principais, ambos voltados para a compreensão e previsão dos retornos do mercado de ações brasileiro (B3). Com base em conceitos de **Factor Investing**, o sistema foi desenvolvido com o propósito de:

* Implementar modelos de **seleção de carteiras de ações**, e
* Construir variáveis preditivas (**features**) para redes neurais, com o intuito de prever movimentos futuros do mercado.

O modelo teórico base adotado é o **Fama-French Five-Factor Model**, uma evolução do modelo de três fatores originalmente proposto por Fama e French (1993). Essa abordagem é amplamente utilizada na literatura de investimentos baseados em fatores, com o objetivo de explicar e capturar os principais componentes sistemáticos que influenciam os retornos dos ativos.

Neste projeto, combinamos indicadores **fundamentalistas** derivados do modelo Fama-French com **indicadores técnicos**, para alimentar algoritmos de **Machine Learning** e **Deep Learning**. A aplicação foi estruturada sobre uma infraestrutura robusta baseada na **AWS (Amazon Web Services)**, garantindo escalabilidade, eficiência computacional e automação na coleta e processamento dos dados.

**2. Visão Geral da Arquitetura**

O projeto está organizado em três subsistemas integrados, que juntos compõem o pipeline completo de ingestão, tratamento, análise e previsão de dados de mercado:

* **Sistema de Coleta e Armazenamento de Dados (Web Scraping);**
* **Sistema de Processamento e Cálculo de Indicadores Financeiros;**
* **Sistema de Modelagem Preditiva (ML e DL).**

**3. Sessão 1 – Códigos do Laboratório de Modelos de Machine Learning**

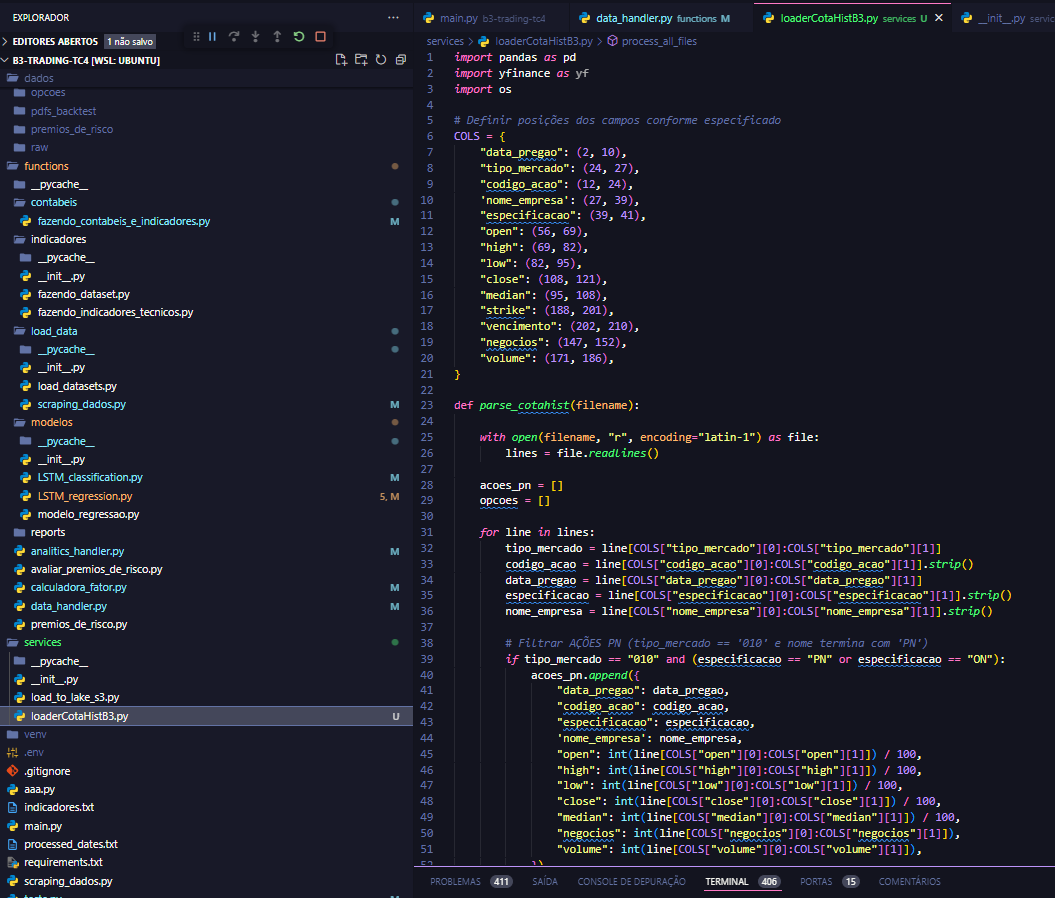
**3.1 Scraping de Dados (loaderCotaHistB3.py)**

O primeiro código executado no processo de ingestão é o loaderCotaHistB3.py. Ele é responsável por carregar os arquivos históricos de cotações disponibilizados pela B3, com nomenclatura padrão COTAHIST\_ANO.TXT.

Esses arquivos, disponibilizados manualmente pela B3, contêm informações fundamentais como:

* Preços de fechamento diário,
* Código das ações,
* Nome da empresa,
* Dados de opções negociadas.

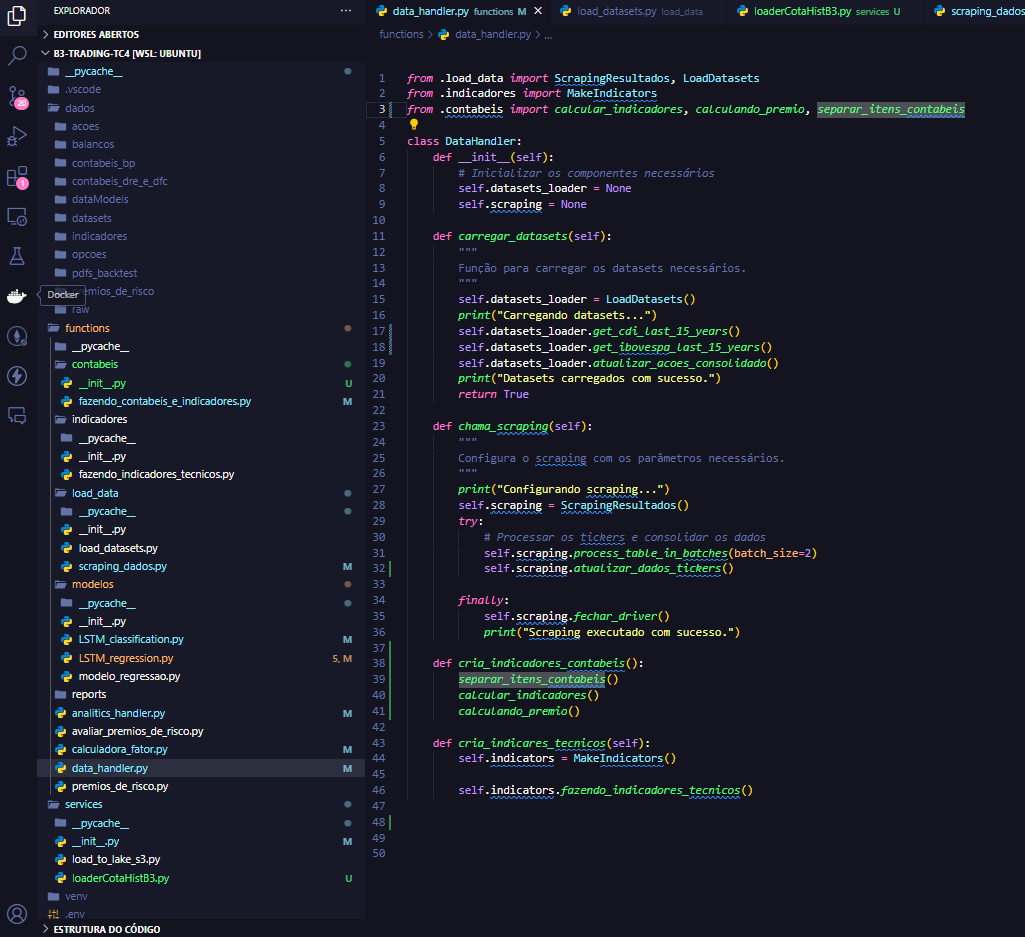
**Importante:** esses arquivos são fornecidos apenas manualmente pela B3, mas são fundamentais para garantir uma análise histórica precisa. O uso do **preço de fechamento ajustado** é inadequado para este tipo de análise, pois compromete a exatidão dos cálculos de indicadores fundamentalistas relacionados ao preço real de negociação dos ativos no passado.  
Portanto, o loaderCotaHistB3.py é **imprescindível para garantir a integridade histórica dos preços**, permitindo análises profundas e consistentes ao longo dos anos.



**3.2 DataHandler – Organização, Consolidação e Scraping**

Após a ingestão inicial dos dados, o módulo main.py instancia o DataHandler.py, que assume um papel central na consolidação e preparação dos dados. Suas principais funções são:

* **Função carregar\_datasets()**:  
  Realiza a leitura de dados externos, incluindo:
  + Históricos da taxa CDI (via API do Banco Central),
  + Histórico do índice Ibovespa (via Yahoo Finance).
* **Função atualizar\_acoes\_consolidado()**:  
  Consolida os dados de preços coletados via COTAHIST com os dados complementares do Yahoo Finance.  
  Esta função também realiza um mapeamento detalhado de ações que **mudaram de ticker ao longo do tempo**, como no caso da antiga **VIIA3 (Via Varejo)**, atualmente **BHIA3 (Casas Bahia)** — garantindo que o histórico de preços e fundamentos contábeis seja preservado corretamente mesmo após reestruturações societárias.

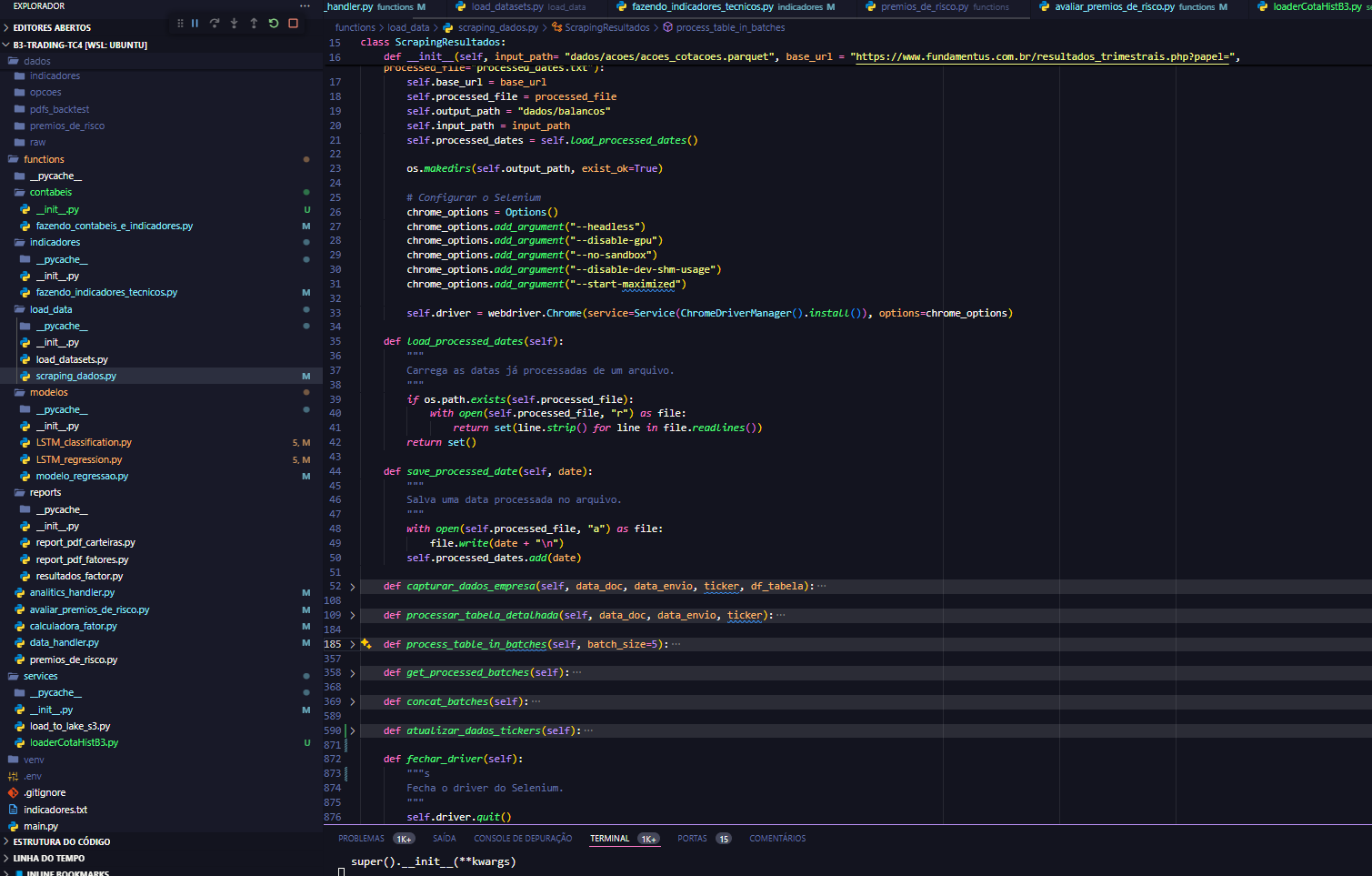


**3.3 Chamada e Execução dos Scrapers Contábeis (chama\_scraping)**

A função chama\_scraping() é um dos núcleos do projeto. Ela é responsável por executar o processo automatizado de scraping dos dados contábeis diretamente do **site oficial da CVM (Comissão de Valores Mobiliários)**. Seu funcionamento envolve:

* Uso da função process\_table\_in\_batches(), que faz a leitura de todas as DREs (Demonstrações do Resultado) trimestrais para cada ação da B3 desde 2010;
* O processo é **extremamente longo**, podendo levar de **1 a 2 semanas** para capturar todo o histórico, devido à latência das requisições por página e trimestre;
* Além da extração, são realizados diversos **ajustes estruturais**, como:
  + Correção de contas acumuladas por trimestre;
  + Ajuste de dados consolidados no quarto trimestre;
  + Uniformização das contas contábeis conforme o padrão da CVM.

Após os tratamentos, os dados extraídos são salvos em um arquivo .parquet nomeado **balancos\_consolidados.parquet**, dentro do diretório dados/balancos/.

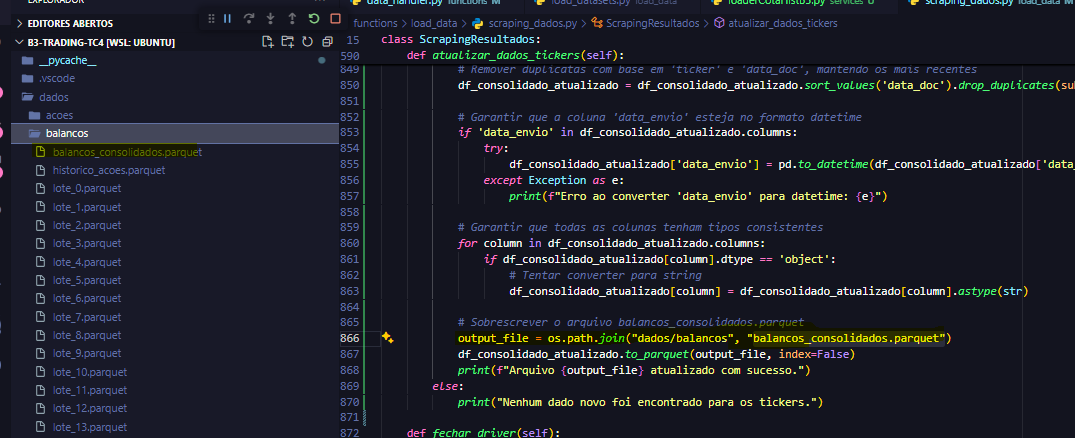


**3.4 Atualização Incremental dos Dados (atualizar\_dados\_tickers())**

O projeto também conta com a função atualizar\_dados\_tickers(), que tem como objetivo manter os dados atualizados de forma **inteligente e incremental**.

Esse módulo verifica, para cada ticker, se houve **envio de novos balanços** (DREs trimestrais) na CVM. Caso haja dados novos, o sistema realiza o scraping apenas do período mais recente, evitando a reprocessamento desnecessário de todo o histórico.

O novo dado é então **concatenado ao dataframe principal**, mantendo a base de dados constantemente atualizada.

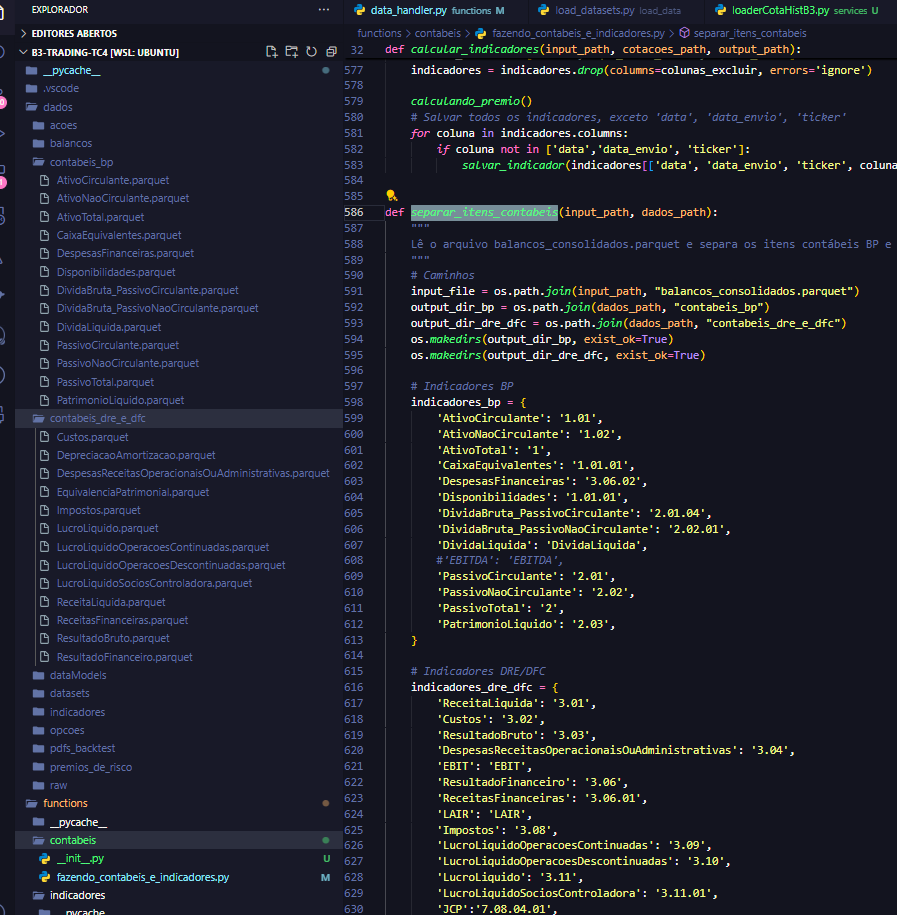


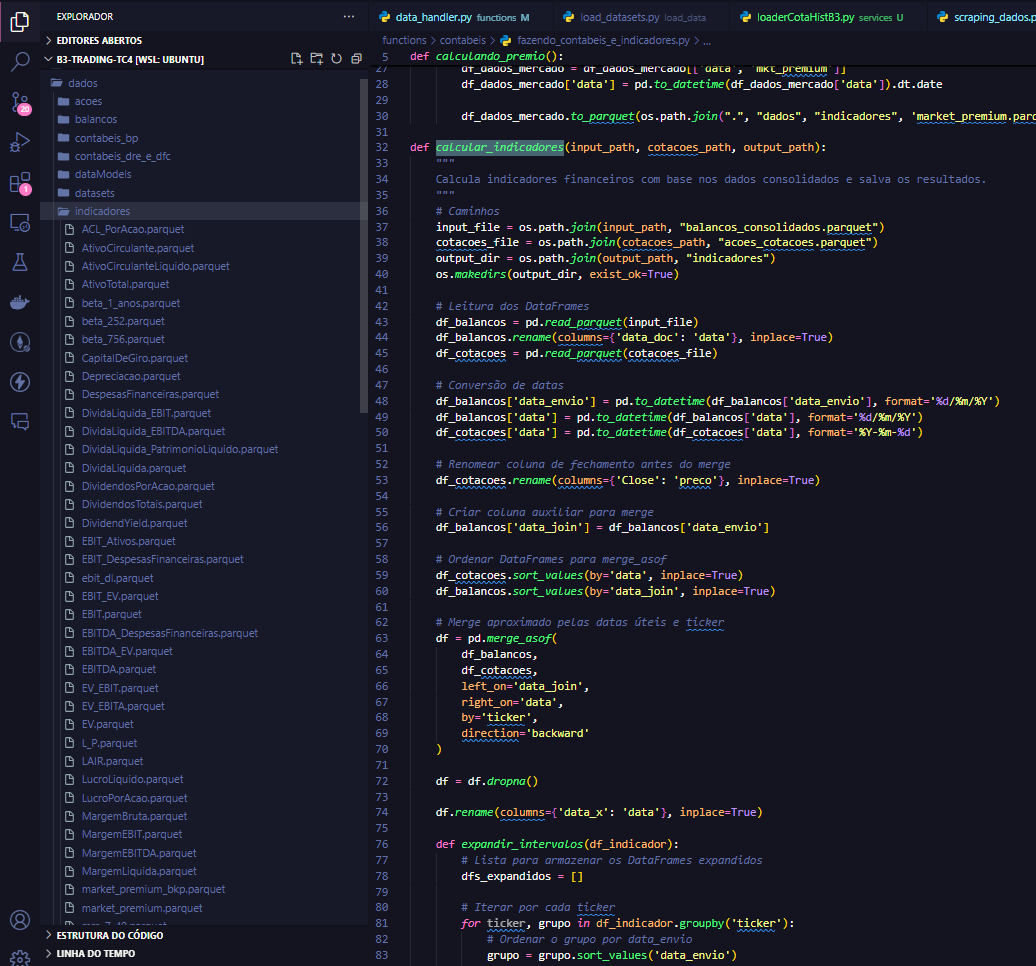
**3.5 Geração de Indicadores Contábeis (cria\_indicadores\_contabeis())**

Com os dados financeiros consolidados, o projeto executa o script cria\_indicadores\_contabeis() para transformar os dados brutos em **indicadores contábeis estruturados**. Essa função realiza:

1. A **explosão dos dataframes consolidados** por conta contábil, gerando arquivos separados conforme a estrutura de indicadores desejada;
2. A geração de mais de **65 indicadores fundamentais**, incluindo:
   * **EBIT**, **EBITDA**, **ROIC**, **ROE**, **Margem Bruta**, entre outros;
3. O uso de **operações vetorizadas** para garantir alta performance no processamento com Python.

Essa etapa transforma os dados contábeis em **features prontas para uso nos modelos de regressão e redes neurais**, sendo uma das bases do projeto.





**4. Sessão 2 – Exploração de Dados**

**4.1 Cálculo do Prêmio de Risco – Função calculando\_premio()**

Após a consolidação dos dados contábeis e geração dos indicadores fundamentalistas, uma das etapas mais relevantes do projeto é a execução da função calculando\_premio(). Essa função é **fundamental para a estrutura estatística do modelo de regressão**, pois permite estimar os **prêmios de risco** associados ao mercado acionário brasileiro.

**Objetivo da função:**

A função calculando\_premio() é responsável por mensurar, mês a mês, o **excesso de retorno do mercado de ações** sobre a **taxa livre de risco (CDI)**. Esse valor é conhecido como prêmio de risco do mercado.

Ou seja, ela calcula:

Premio de Risco=Retorno do IBOV−Retorno do CDI

Esse diferencial representa o risco assumido ao investir em ações em comparação a investimentos considerados mais seguros, como títulos públicos indexados ao CDI. O resultado dessa função será usado posteriormente na **regressão linear** que tentará explicar os retornos ajustados com base nos **betas de cada fator de risco**, os quais também são utilizados nos modelos LSTM e na construção da carteira de investimentos baseada em fatores.

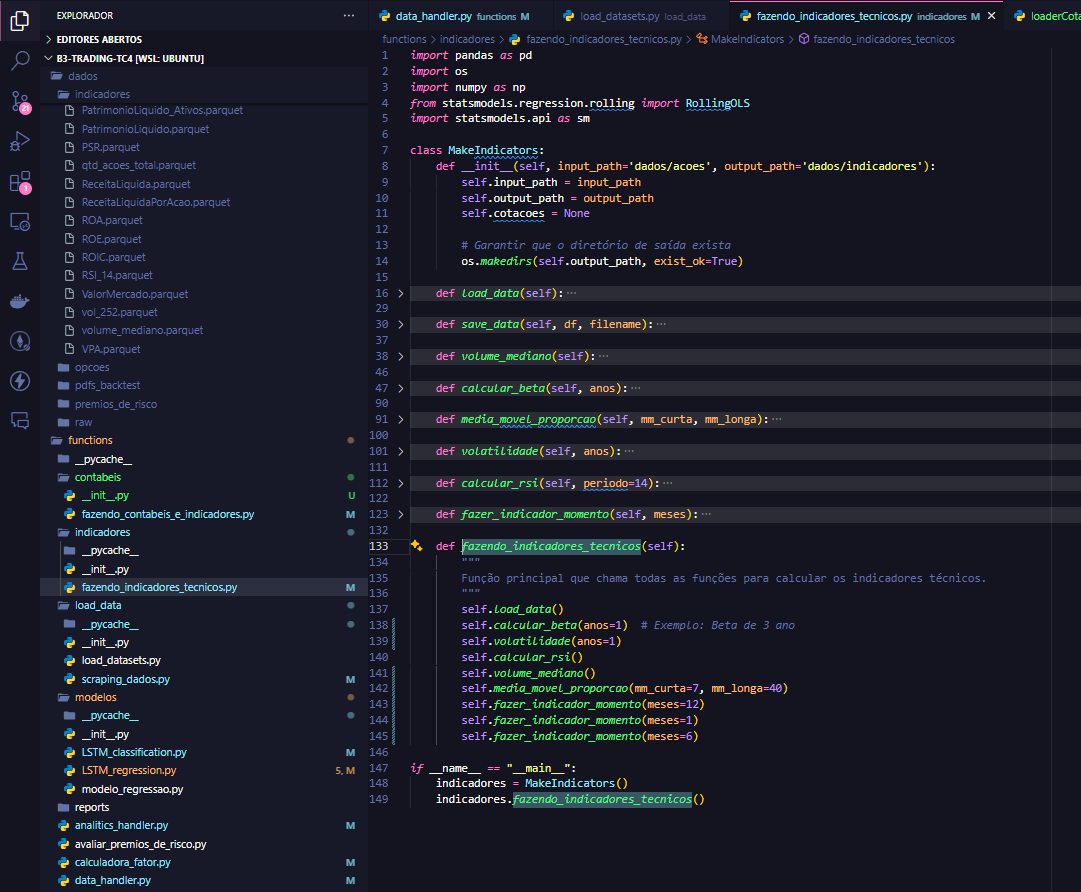


**4.2 Geração de Indicadores Técnicos**

Com os dados contábeis prontos e o prêmio de risco do mercado calculado, o próximo passo é enriquecer o conjunto de dados com **indicadores técnicos**, que são amplamente utilizados em estratégias quantitativas de investimento. Os indicadores técnicos extraídos incluem:

* **Beta:** medida de sensibilidade de uma ação em relação ao Ibovespa;
* **Volatilidade:** desvio padrão dos retornos, indicador de risco do ativo;
* **RSI (Índice de Força Relativa):** identifica zonas de sobrecompra ou sobrevenda;
* **Volume mediano:** média das negociações diárias;
* **Médias móveis e proporções:** cruzamentos de curto e longo prazo;
* **Indicadores de momentum:** crescimento ou queda dos preços nos últimos períodos (1, 6 e 12 meses).

Esses indicadores são essenciais para complementar os fatores fundamentalistas, criando um **conjunto robusto e diversificado de features** que serão exploradas posteriormente nos modelos preditivos.



**4.3 Entendimento Conceitual: O Prêmio de Risco**

O **prêmio de risco** representa o **retorno adicional que um investidor exige** (ou historicamente recebe) por escolher aplicar seu capital em ativos arriscados (como ações) em vez de investimentos considerados livres de risco (como o CDI).

Premio de Risco = Retorno Esperado do Ativo− Retorno Livre de Risco

Como os investidores, em geral, são **aversos ao risco**, é natural que eles exijam uma compensação financeira para assumir incertezas. Essa compensação é justamente o **prêmio de risco**, e sua avaliação é fundamental no contexto do **Factor Investing**.

**Fatores analisados no projeto:**

Durante o projeto, avaliamos o prêmio de risco associado a diferentes fatores clássicos da literatura de investimentos:

* **Alavancagem:** (Dívida Líquida / EBITDA)
* **Valor:** (EBIT / Ativos Totais)
* **Qualidade:** (ex: ROIC elevado ou EBIT / Enterprise Value)
* **Volatilidade:** (Beta 252 dias)
* **Momentum:** (retorno recente de 1, 6 e 12 meses)

Cada fator representa uma característica específica das empresas, e o investidor é **recompensado por assumir exposição a esses riscos estruturais**.

**4.4 Análise por Quartis: Avaliação Empírica dos Fatores**

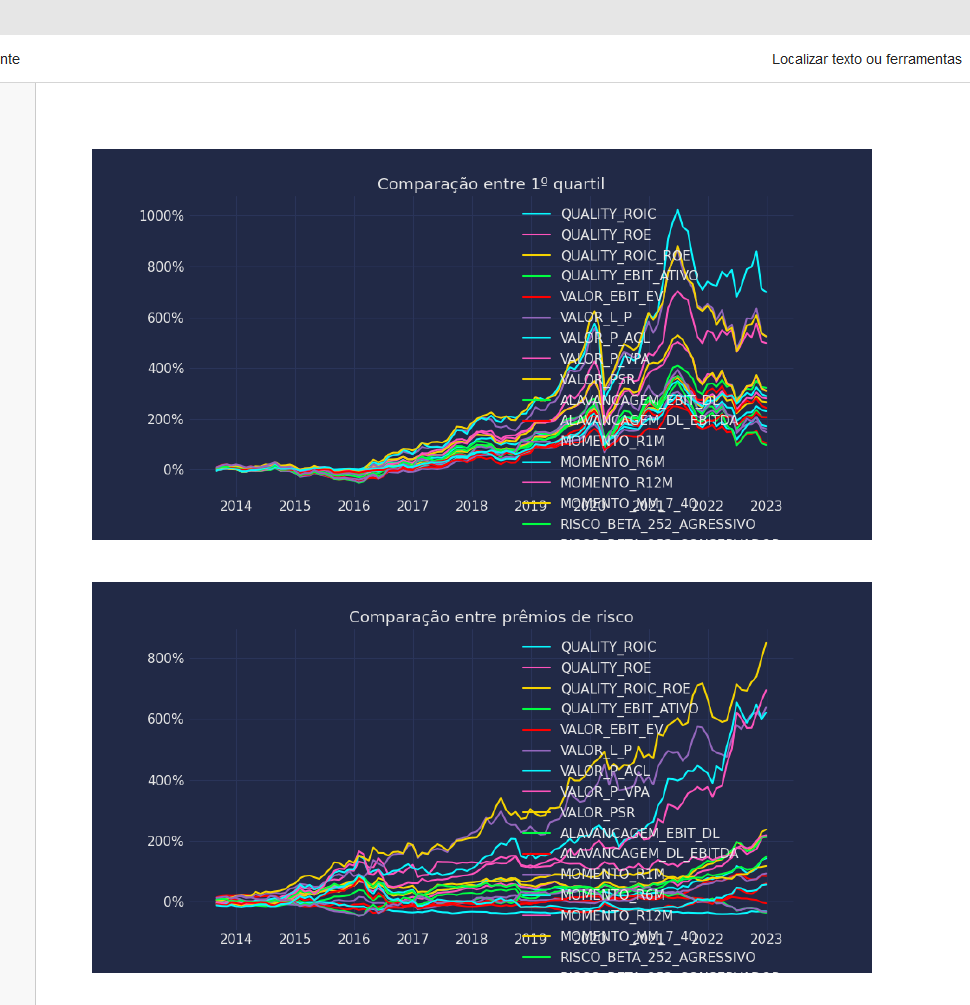
Com base nos fatores acima, segmentamos as empresas em **quartis**, de acordo com o valor de cada indicador. O **1º quartil** representa as empresas com os melhores valores (ex: maior ROIC, maior momentum, menor alavancagem, etc.), enquanto o **4º quartil** representa as empresas com os piores desempenhos em cada fator.

**Metodologia:**

A análise foi feita ao longo de um período de **10 anos**, observando o retorno mensal acumulado de cada quartil.

**Anexo:**  
Um relatório PDF com gráficos gerados a partir do código contém a avaliação de **19 indicadores diferentes**, comparando o desempenho entre os quartis.





**Exemplos de análise retirados do relatório:**

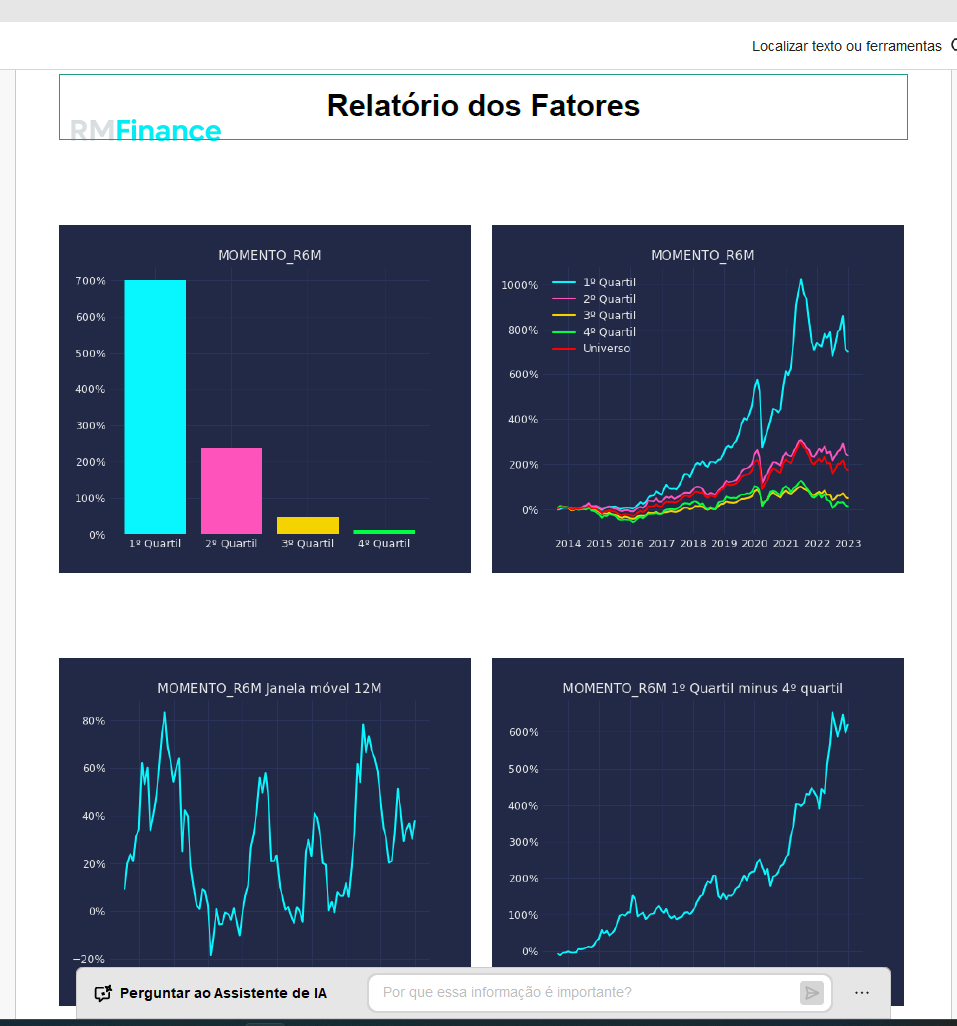
* **Indicador ROIC (fator qualidade):**
  + **Primeira imagem:** o 1º quartil supera o 4º quartil de forma consistente, demonstrando um **bom poder explicativo**.
  + **Segunda imagem:** o retorno da linha azul (1º quartil) está muito próximo ao retorno médio do universo de ações, o que indica **baixo diferencial**. Universo representa todas as ações desses testes. Isso significa que esse indicador não é bom para performar acima do mercado, resumidamente, se comprarmos ações aleatórias do universo teremos um retorno muito aproximado.
  + **Quarta imagem:** o gráfico indica que o % de retorno que o 1 quartil tem sobre o 4 quartil ao longo do tempo testado, o retorno do 1º quartil chega a ser inferior ao 4º em certos períodos, o que **reduz a utilidade do indicador** e demonstra **irracionalidade** **explicativa** para esse indicador.



**Destaque: Fator Momentum**

O indicador de **momentum de 6 meses** foi o que apresentou o melhor desempenho dentre os 19 analisados. As empresas do 1º quartil (que mais subiram nos últimos meses) apresentou retornos **consistentemente superiores**. Isso evidencia que, no mercado brasileiro, **comprar ações que estão subindo tem funcionado muito bem** como estratégia quantitativa

**Observação:**  
Esse comportamento é peculiar do Brasil e **não é observado com a mesma intensidade em mercados como os Estados Unidos**, onde o efeito momentum é geralmente mais contido ou instável.

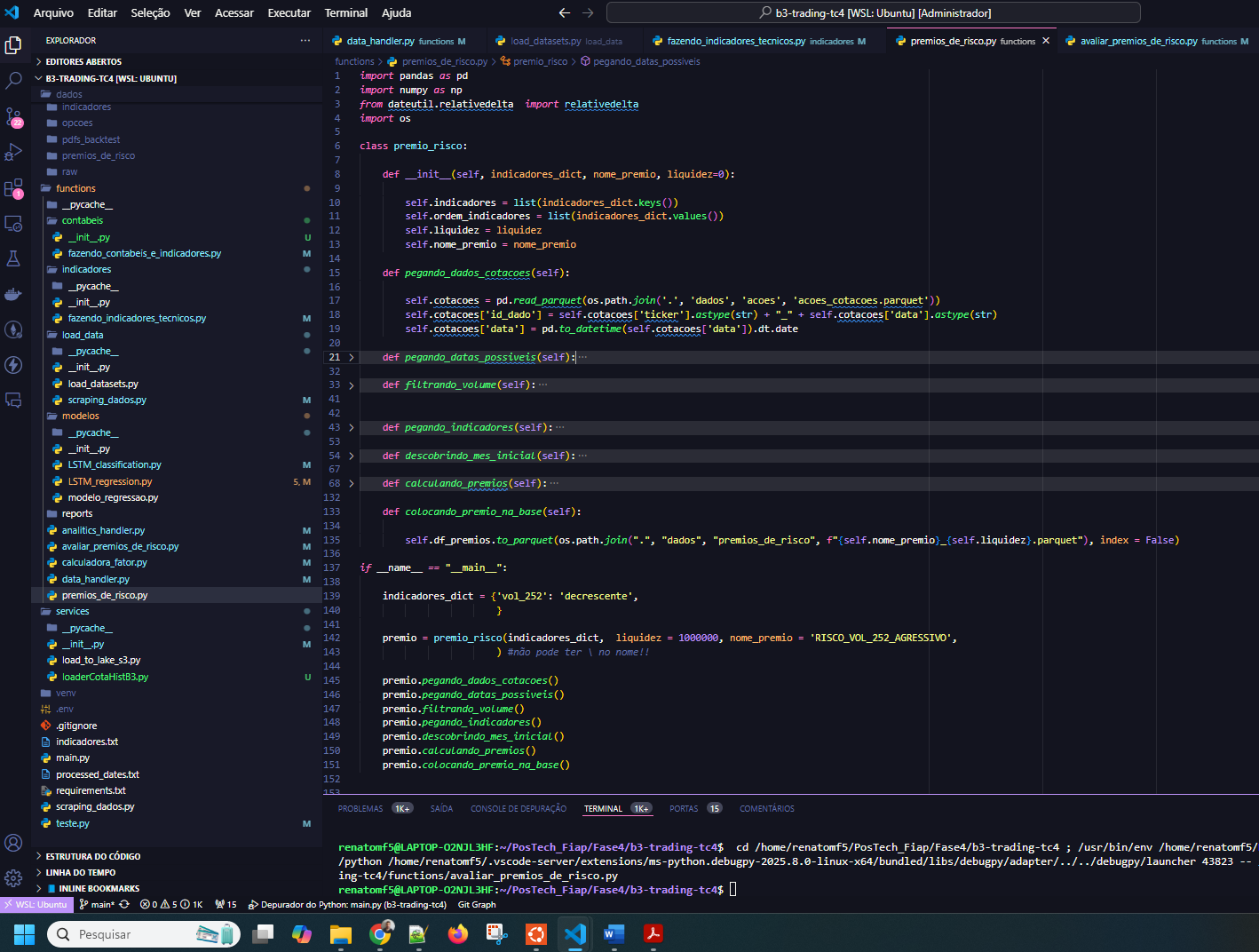


**4.5 Implementação da Análise Exploratória – Códigos**

A lógica completa da análise de prêmios de risco foi implementada no script premios\_de\_risco.py. O funcionamento segue a seguinte dinâmica:

1. A cada mês:
   * As ações são classificadas em **quartis**, com base no valor do indicador em questão.
   * Calcula-se o **retorno médio de cada quartil** naquele mês.
2. Esses dados são acumulados ao longo do tempo e **salvos mês a mês**.
3. Os resultados são armazenados no diretório:

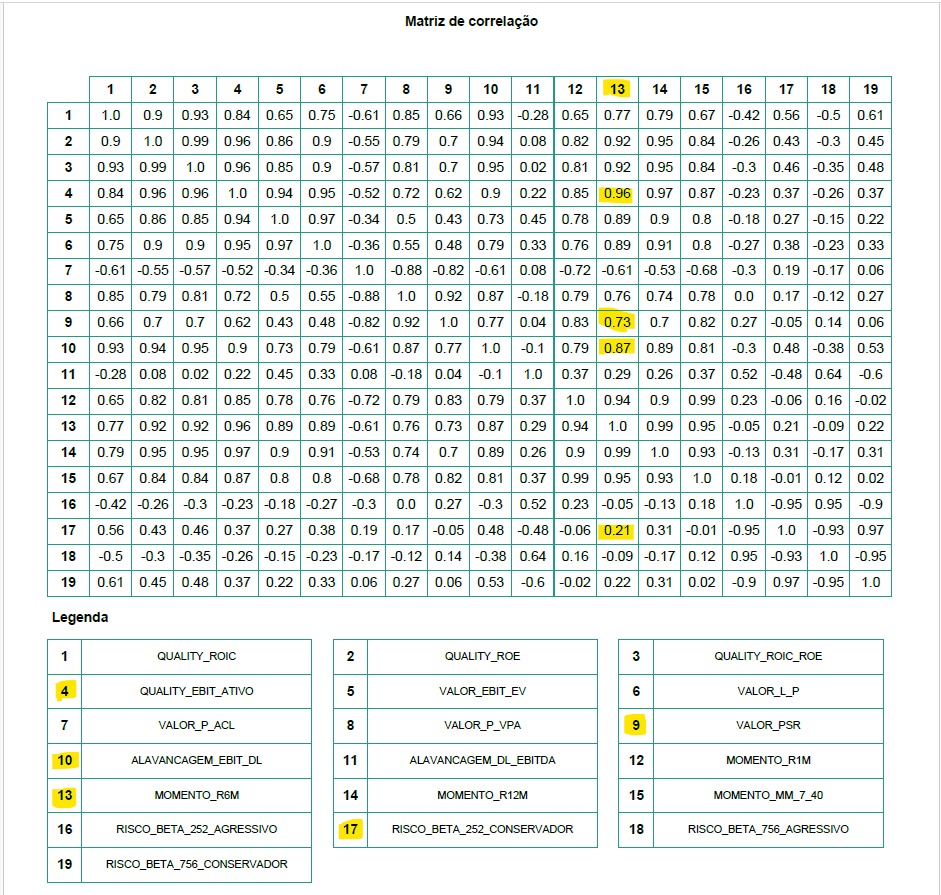
Posteriormente, é executado o script avaliar\_premios\_de\_risco.py, responsável por gerar as **visualizações em PDF**. Esse processo utiliza o código report\_pdf\_fatores.py, localizado na pasta reports, que monta automaticamente os gráficos comparativos e resumos estatísticos de cada indicador.



**4.6 Escolha dos Indicadores com Base na Análise Exploratória**

Após a análise detalhada dos indicadores fundamentalistas e técnicos com base nos prêmios de risco, torna-se necessário definir **quais indicadores serão utilizados nos modelos de previsão**, tanto na construção da carteira baseada em fatores (*Factor Investing*), quanto na **definição das features para os modelos LSTM**.

Uma das ferramentas empregadas para auxiliar nessa seleção foi a **matriz de correlação entre os retornos mensais dos indicadores**. Essa matriz foi gerada e analisada no PDF anexo à etapa anterior. Seu objetivo foi medir a dependência entre os fatores, ajudando a evitar a inclusão de variáveis altamente correlacionadas entre si no mesmo modelo.

  
  
**Conclusão extraída da matriz:**  
Para que a regressão funcione de maneira explicativa e com o mínimo de multicolinearidade, é fundamental **selecionar fatores que apresentem bons prêmios de risco, mas que sejam entre si pouco correlacionados**. Isso garante uma maior independência entre os betas dos fatores e, consequentemente, maior interpretabilidade na explicação dos retornos ajustados.

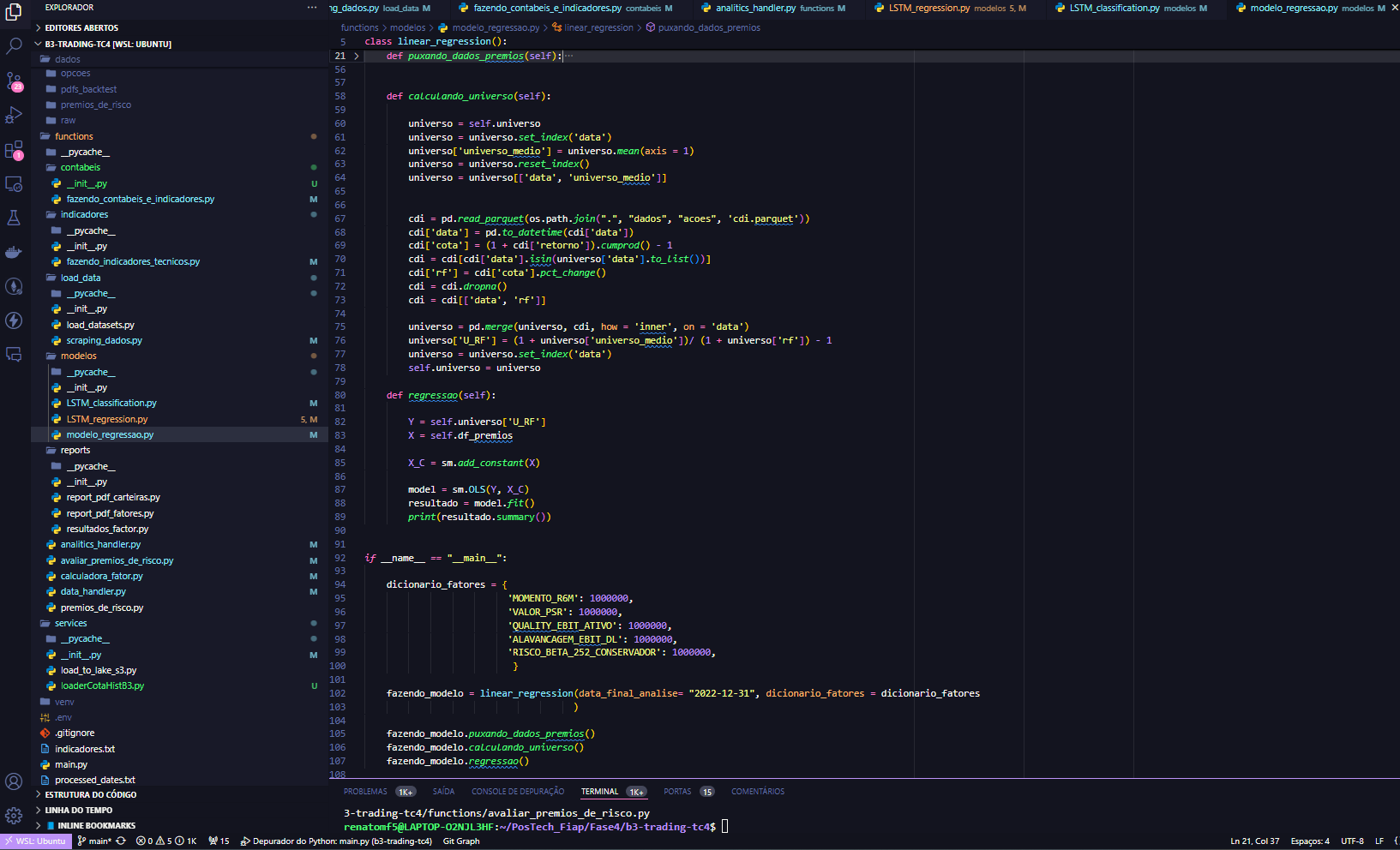
**Critério adotado na seleção:**

1. **Iniciar pelo fator com o melhor desempenho em termos de prêmio de risco** — no nosso caso, foi o **momentum de 6 meses**, conforme comprovado na análise anterior.
2. **Selecionar manualmente os demais indicadores representativos dos fatores (valor, qualidade, alavancagem, risco)**, prezando por baixa correlação cruzada.

**4.7 Aplicação da Regressão Linear (modelo\_regressao.py)**

Com os indicadores selecionados, aplicamos um modelo de regressão linear com o objetivo de **avaliar a relação entre os fatores de risco e o retorno ajustado dos ativos em relação à taxa livre de risco (CDI)**.

O código responsável por essa análise está no script modelo\_regressao.py, localizado na pasta modelos.



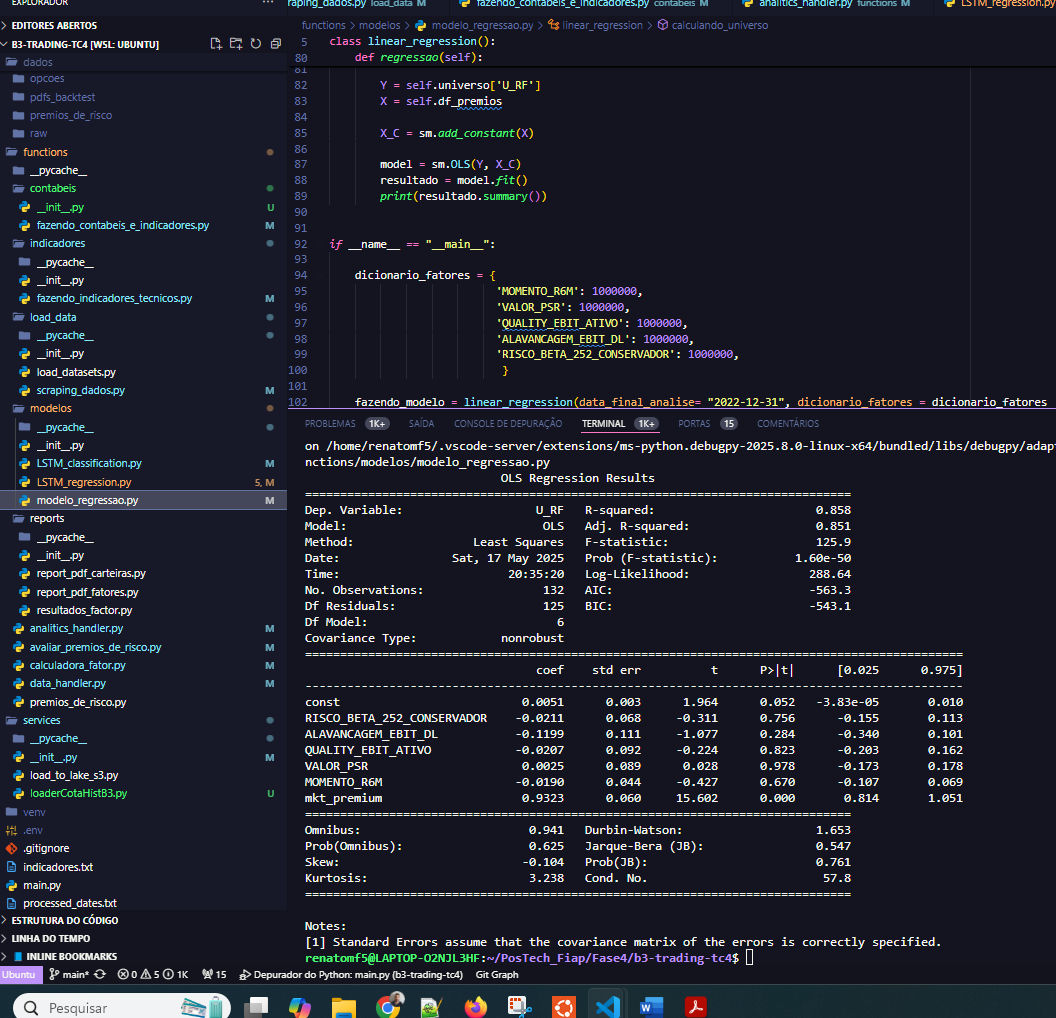
**Objetivos da Regressão Linear:**

* **Quantificar o impacto de cada fator de risco sobre o retorno ajustado**:
  + Cada coeficiente representa o efeito de um fator.
  + Exemplo: um coeficiente positivo para MOMENTO\_R6M indica que ativos com maior momentum tendem a gerar maior retorno ajustado.
* **Avaliar a significância estatística de cada fator**:
  + Utilizamos os **valores-p** para verificar se os coeficientes são estatisticamente relevantes.
* **Avaliar o poder explicativo do modelo**:
  + O **R² (coeficiente de determinação)** é utilizado como métrica principal para indicar qual proporção da variação do retorno ajustado é explicada pelos fatores.

**Melhor avaliação obtida:**

* **R² = 85,8%**, o que indica que **a maior parte da variabilidade do retorno ajustado foi explicada pelos fatores incluídos no modelo**.
* Esse valor representa um **excelente ajuste** e reforça a qualidade da escolha dos indicadores base.

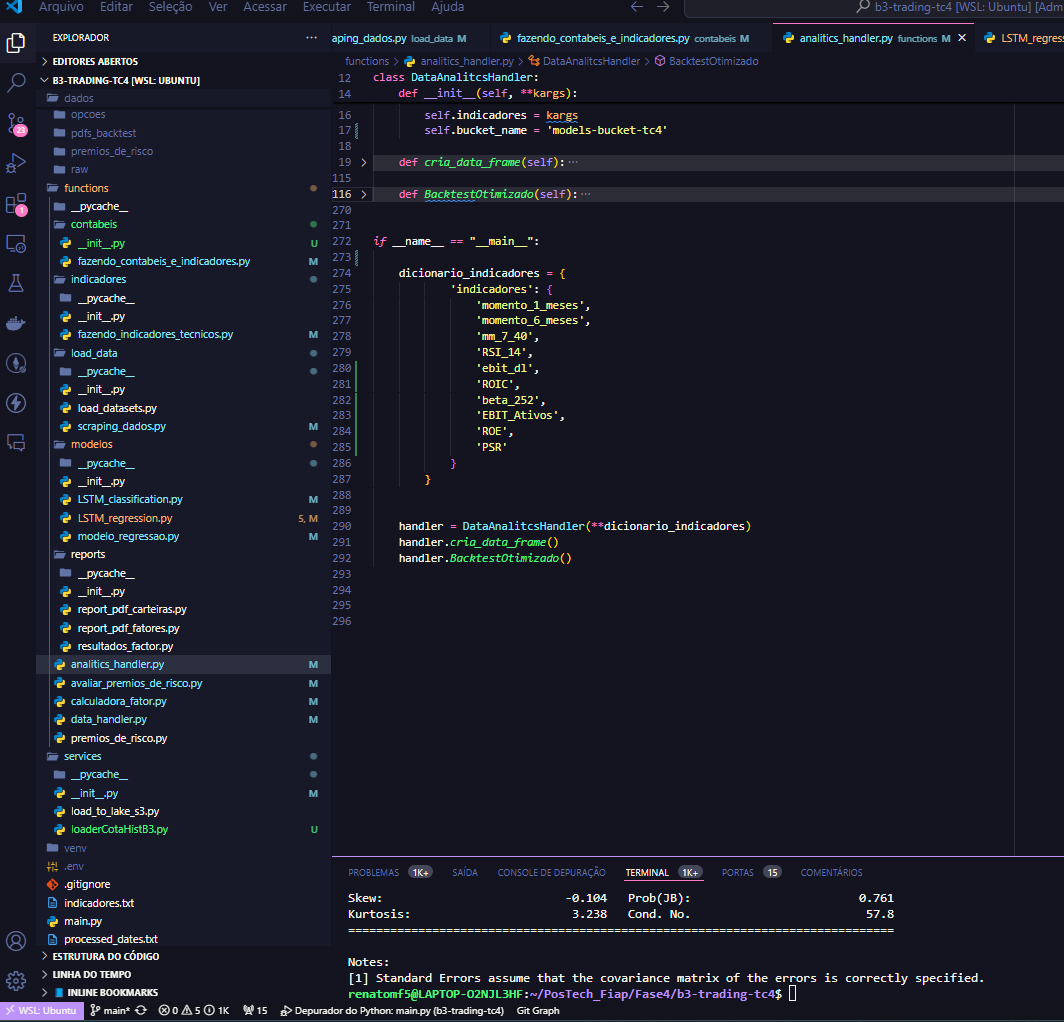
✅ **Ponto positivo:**  
Um R² elevado como esse indica que os fatores escolhidos são altamente relevantes para explicar os retornos ajustados, servindo de base para a modelagem preditiva posterior.



**5. Sessão 3 – Modelagem com Redes Neurais e Carteiras Baseadas em Fatores**

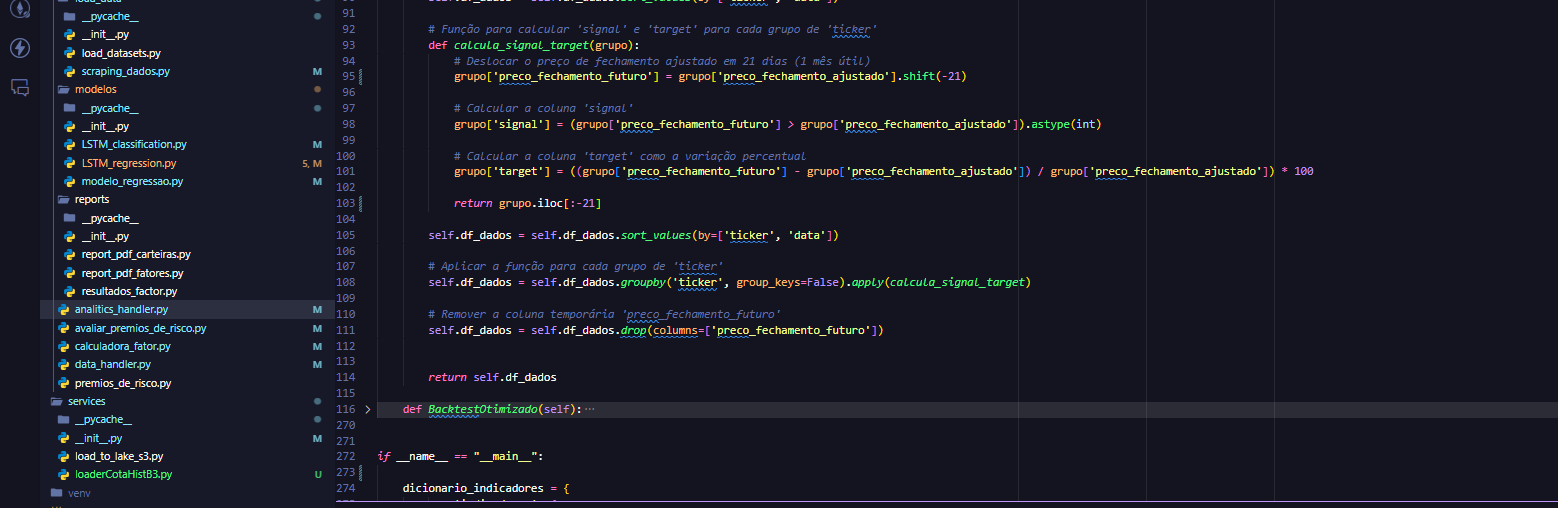
**5.1 Construção das Redes Neurais LSTM**

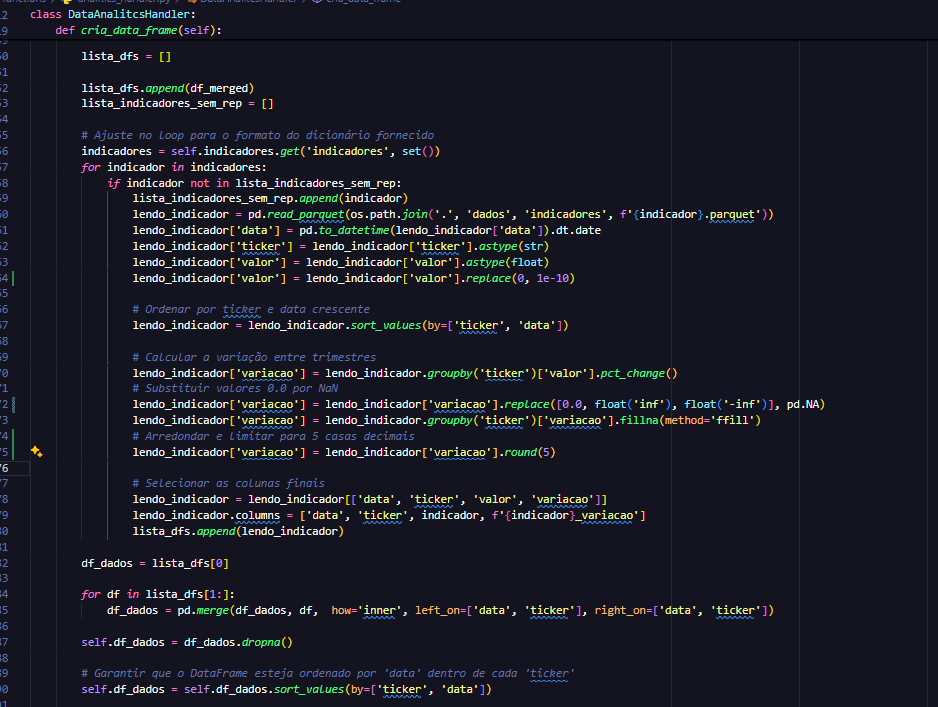
A construção dos modelos de redes neurais foi realizada no módulo analicts\_handler.py, localizado na pasta functions. Este módulo concentra a lógica de criação dos datasets de treino/teste, definição de features e execução dos modelos LSTM.



**Etapas do pipeline de modelagem:**

1. **Definição do dicionário de indicadores (features):**
   * Inclui os fatores selecionados na análise anterior.
   * Foram adicionados também indicadores complementares como:
     + mm\_7\_40 (média móvel curta vs. longa),
     + RSI\_14 (índice de força relativa de 14 dias),
     + momento\_1\_meses (momentum de curto prazo).
2. **Função cria\_data\_frame():**
   * Responsável por consolidar todos os indicadores em um único dataframe.
   * Também adiciona a **feature variação**, que representa a variação temporal de cada indicador — recurso importante para enriquecer o treinamento supervisionado.
3. **Função calcula\_signal\_target():**
   * Calcula o **retorno futuro de 21 dias úteis** (cerca de 1 mês corrido).
   * Com base nesse retorno futuro:
     + **Signal = 1** se o retorno é positivo (indica compra),
     + **Signal = 0** se o retorno é negativo (indica venda).
   * Também é gerada a **coluna target**, com o valor percentual exato do retorno projetado, usada no modelo de regressão LSTM.





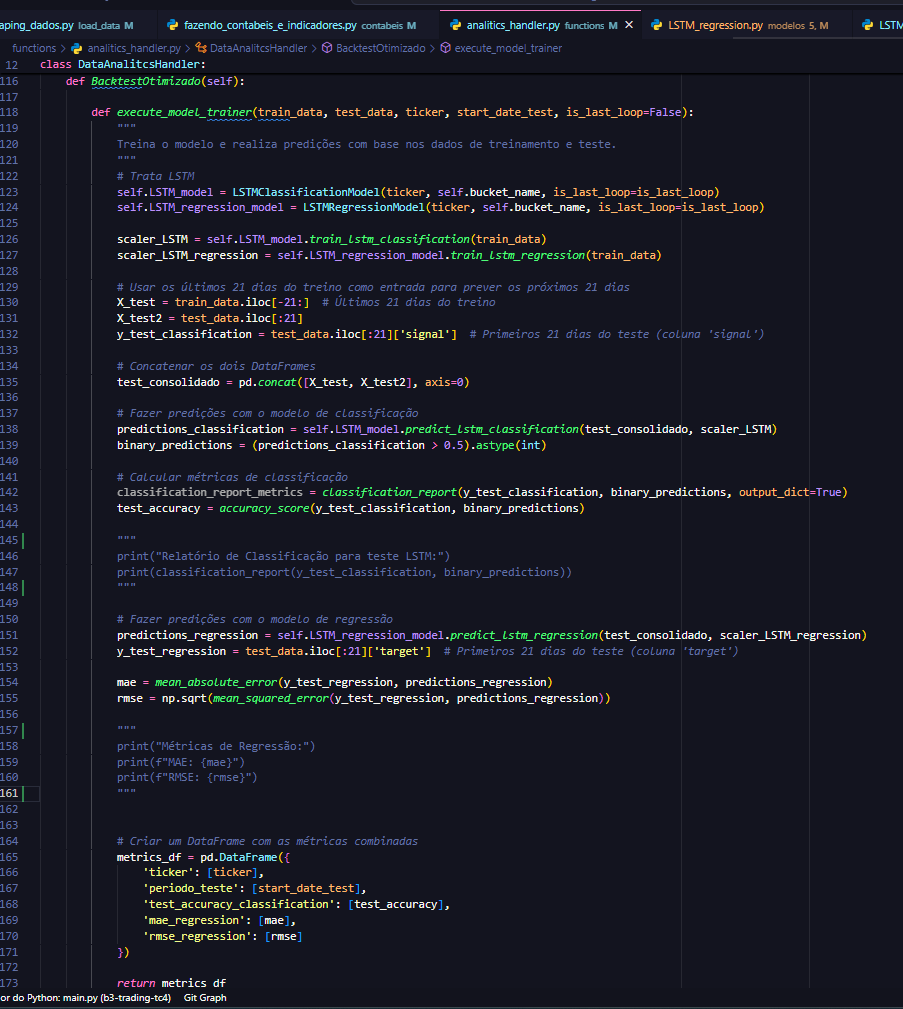
**5.2 Backtests e Treinamento Otimizado**

A função BacktestOtimizado() implementa a lógica de simulação realista dos modelos, dividindo os dados em janelas temporais para treino e teste.

**Características do processo:**

* Para cada **ticker tratado pelo sistema de scraping**, são realizados **3 ciclos de treino e teste (backtests)**.
* Cada conjunto de dados é dividido da seguinte forma:
  + **Treinamento:** até no máximo 4 anos de dados.
  + **Teste:** janela de 30 dias corridos (~21 úteis).
* Para cada janela, é executada a função execute\_model\_trainer, que:
  + Instancia os modelos LSTM (classificação ou regressão),
  + Treina os modelos com os dados de entrada,
  + Realiza as predições,
  + Avalia as métricas de desempenho (ex: acurácia, RMSE, MAE).

📊 **Importante:** essa etapa é essencial para garantir que os modelos generalizam bem em dados futuros, simulando como eles se comportariam em ambiente de produção real.



**6. Sessão 4 – Arquitetura das Redes Neurais Utilizadas**

**6.1 Arquitetura do Modelo LSTM – Classificação**

Nesta seção, detalhamos a arquitetura do modelo LSTM utilizado para **classificação binária**, cujo objetivo é prever se uma ação apresentará retorno positivo ou negativo no futuro próximo (21 dias úteis).



**Arquitetura da Rede:**

* **Tipo de Rede:**
  + Rede Neural LSTM **bidirecional** (*Bidirectional LSTM*), que processa as sequências de dados em **duas direções** (passado e futuro).
  + Isso permite ao modelo capturar contextos temporais mais ricos — fundamental para séries temporais financeiras.
* **Dropout:**
  + Técnica de regularização utilizada para reduzir o **overfitting**.
  + Taxa de **10%** dos neurônios desativados aleatoriamente durante o treinamento.
* **Camada Densa (Fully Connected):**
  + Unidade de saída: **1** (probabilidade de classe).
  + Função de ativação: **Sigmoid**, que retorna valores entre 0 e 1.
  + Regularização L2: penaliza pesos muito altos, melhorando a generalização.

**Compilação e Treinamento:**

* **Otimizador:**
  + **Adam**, um otimizador adaptativo amplamente adotado por sua eficiência e velocidade de convergência.
* **Função de perda:**
  + **Binary Crossentropy**: mede a diferença entre a saída da rede (probabilidade) e os rótulos reais (0 ou 1).
* **Early Stopping:**
  + Interrompe o treinamento se a perda de validação não melhorar após **10 épocas consecutivas**.
  + A opção restore\_best\_weights=True garante que os melhores pesos (com menor perda) sejam restaurados.

**Configurações de Treinamento:**

* **Dados de entrada (X)**: sequências de **21 passos de tempo** por ação.
* **Rótulos (y)**: sinais binários de compra ou venda.
* **Validação:** 10% dos dados separados para validação.
* **Épocas:** até **80 épocas**.
* **Batch size:** 32.
* **Callback:** Early Stopping.

**Pré-processamento:**

* **Normalização dos dados:**
  + Os dados são padronizados para média 0 e desvio padrão 1, otimizando o desempenho da rede.
* **Criação de Sequências (time\_step = 21):**
  + As séries temporais são convertidas em janelas com 21 registros consecutivos.

**Armazenamento na Nuvem (AWS S3):**

* Após o treinamento, o modelo .keras é salvo e automaticamente **enviado para o bucket S3**, possibilitando consultas e predições em tempo real pela API.

**6.2 Arquitetura do Modelo LSTM – Regressão**

O modelo de regressão tem uma arquitetura semelhante ao de classificação, mas com foco na **previsão de valores contínuos**, como a variação percentual futura de uma ação em 21 dias úteis.



**Diferenças Principais:**

* **Objetivo:**
  + Ao invés de classificar, o modelo **prevê um valor real contínuo**, como o retorno percentual futuro.
* **Camada Densa de saída:**
  + Unidade: **1** (valor contínuo).
  + Ativação: **Linear** (sem compressão do valor de saída).
  + Regularização L2: permanece ativa para reduzir overfitting.
* **Função de Perda:**
  + **Mean Squared Error (MSE)**: avalia a média dos quadrados das diferenças entre valores reais e previstos.

**Importante:** A maior diferença entre os modelos reside na função de ativação da camada final e na função de perda.

**Componentes em Comum com o Modelo de Classificação:**

* **Bidirectional LSTM**
* **Dropout de 10%**
* **Early Stopping com restore\_best\_weights**
* **Batch size = 32**
* **Normalização dos dados**
* **Criação de sequências com time\_step = 21**
* **Armazenamento em AWS S3**

**7. Sessão 5 – API de Predição: FastAPI**

**7.1 Estrutura e Funcionalidades da FastAPI**

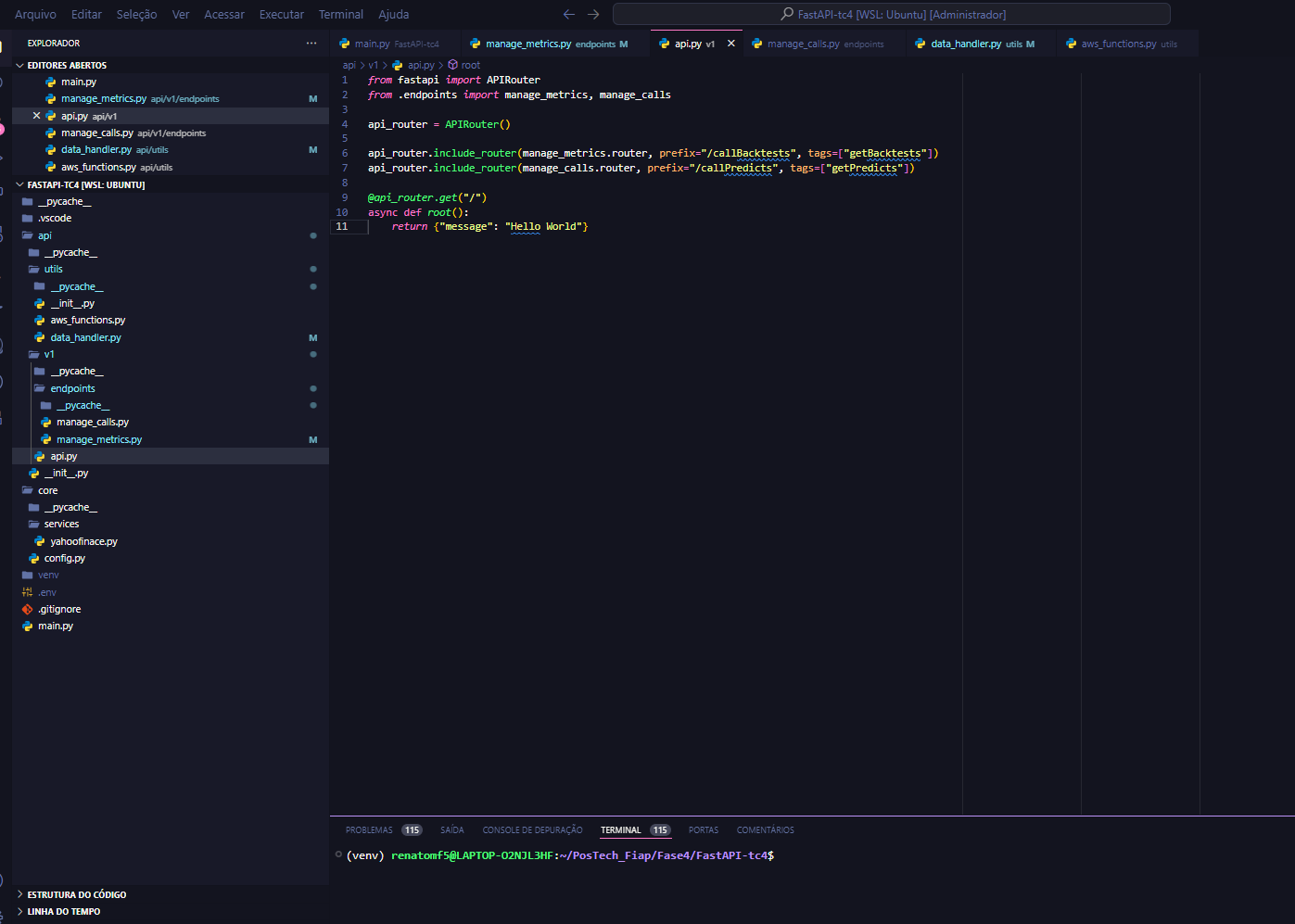
Para operacionalizar os modelos de forma prática, foi desenvolvida uma **API com FastAPI**. Essa API é responsável por:

1. **Carregar os modelos LSTM e os scalers específicos** de cada ativo, diretamente do ambiente AWS S3.
2. **Executar as predições em tempo real**, retornando o preço-alvo e a recomendação de compra ou venda.
3. **Exibir métricas de desempenho** como RMSE, MAE e acurácia para cada ativo/ticker.
4. **Listar os resultados dos backtests realizados** para os modelos LSTM e carteiras baseadas em fatores.
5. **Apresentar a performance histórica das carteiras recomendadas**, com comparação a benchmarks como o Ibovespa.

**7.2 Estrutura dos Códigos da API**

O principal arquivo da API é o api.py, responsável por:

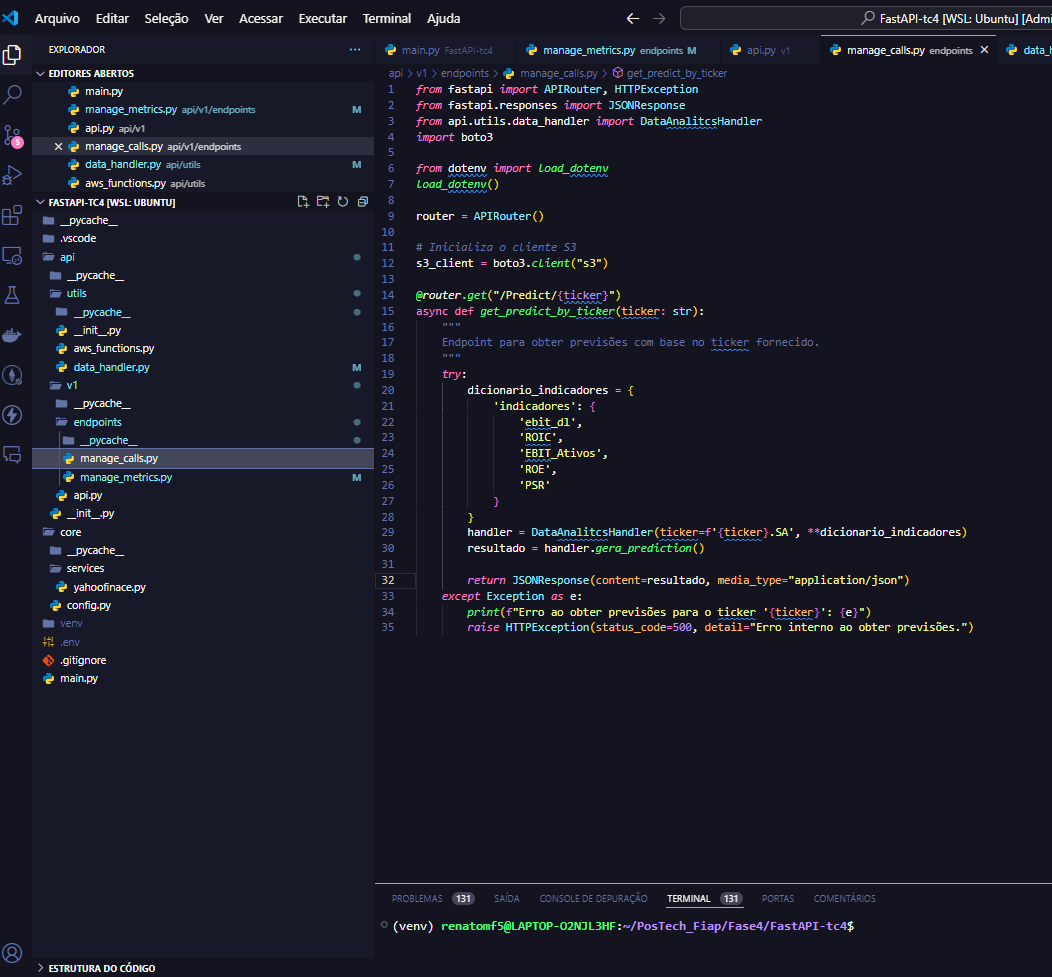
* Criar as rotas de acesso;
* Processar as requisições da interface web;
* Gerenciar a lógica de predição.

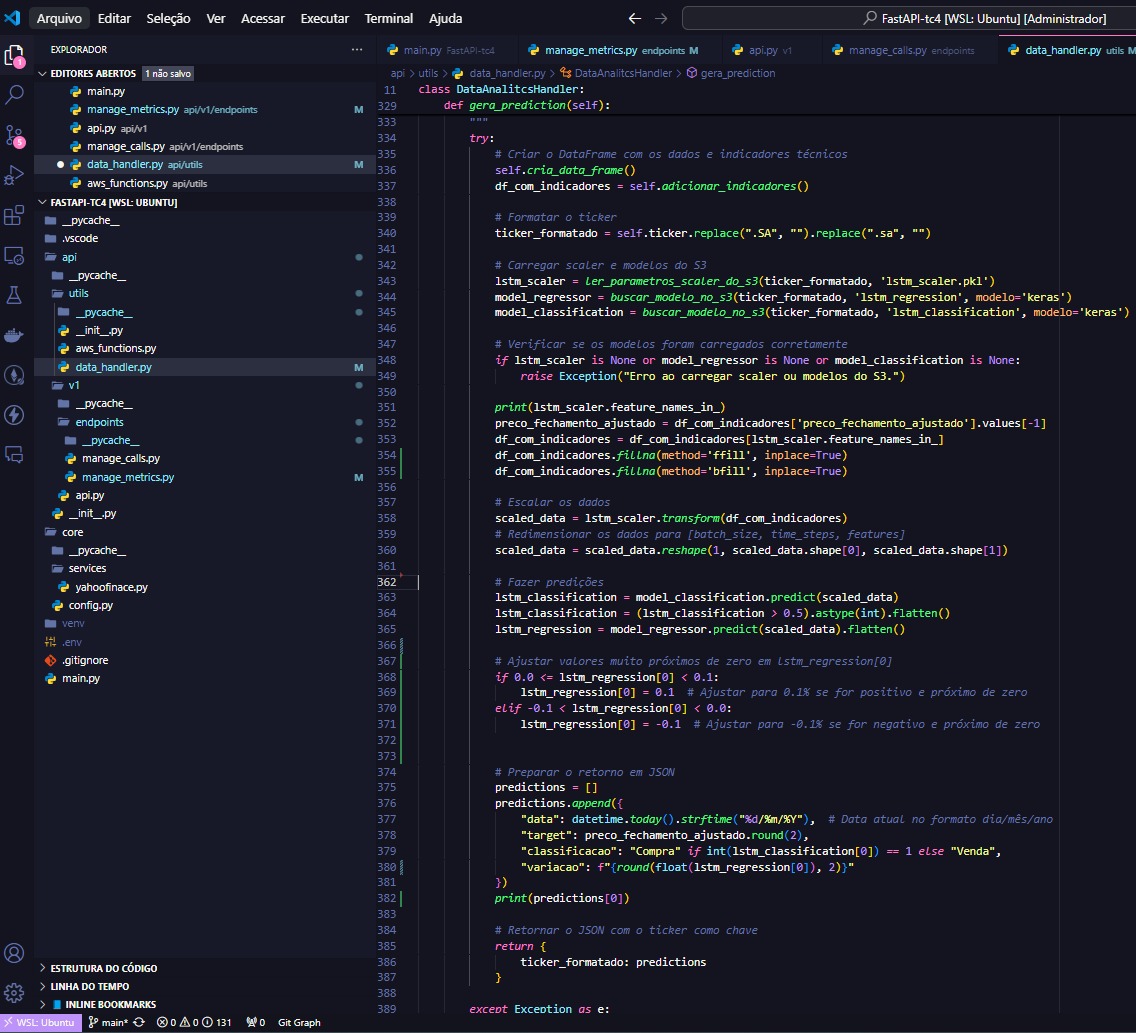
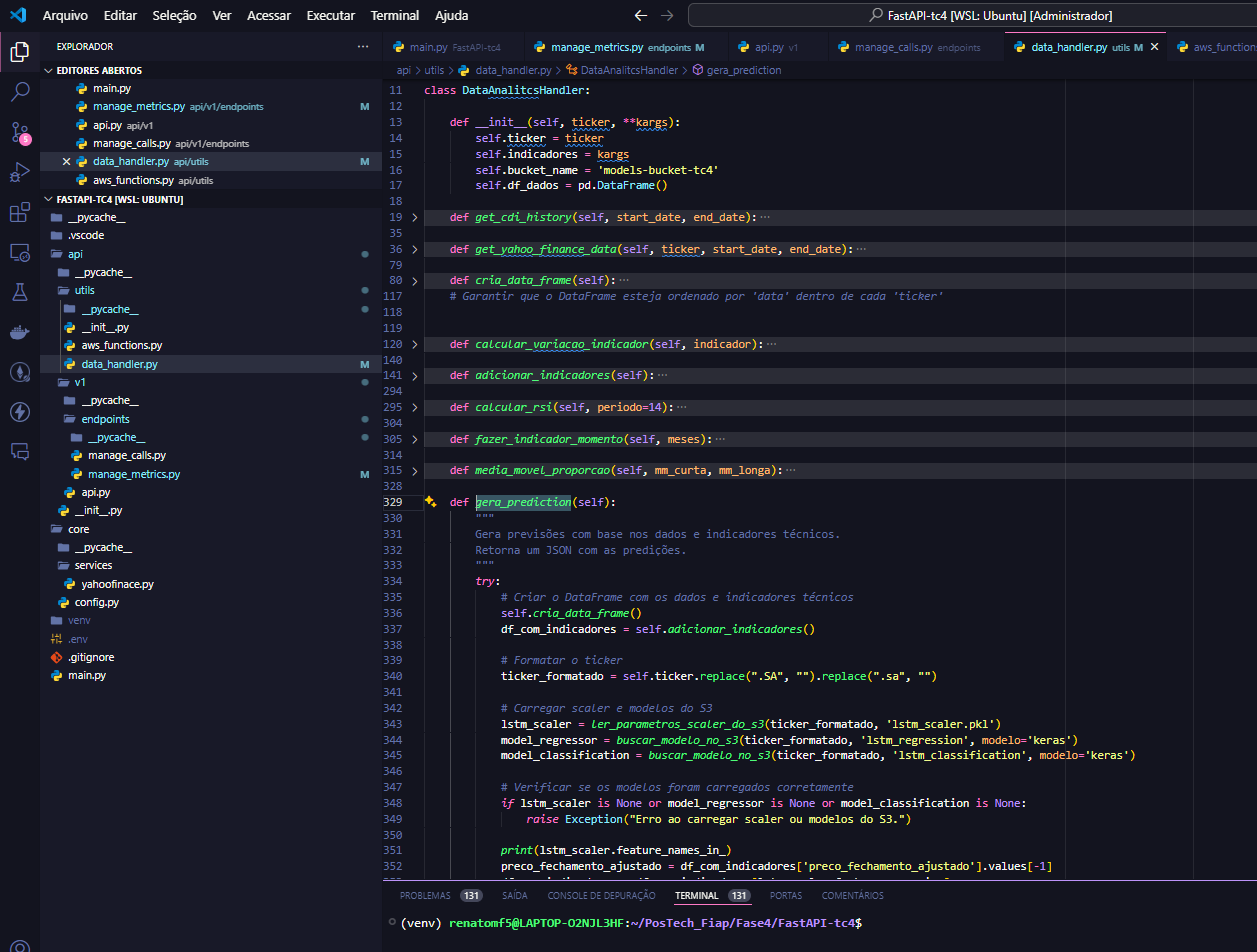


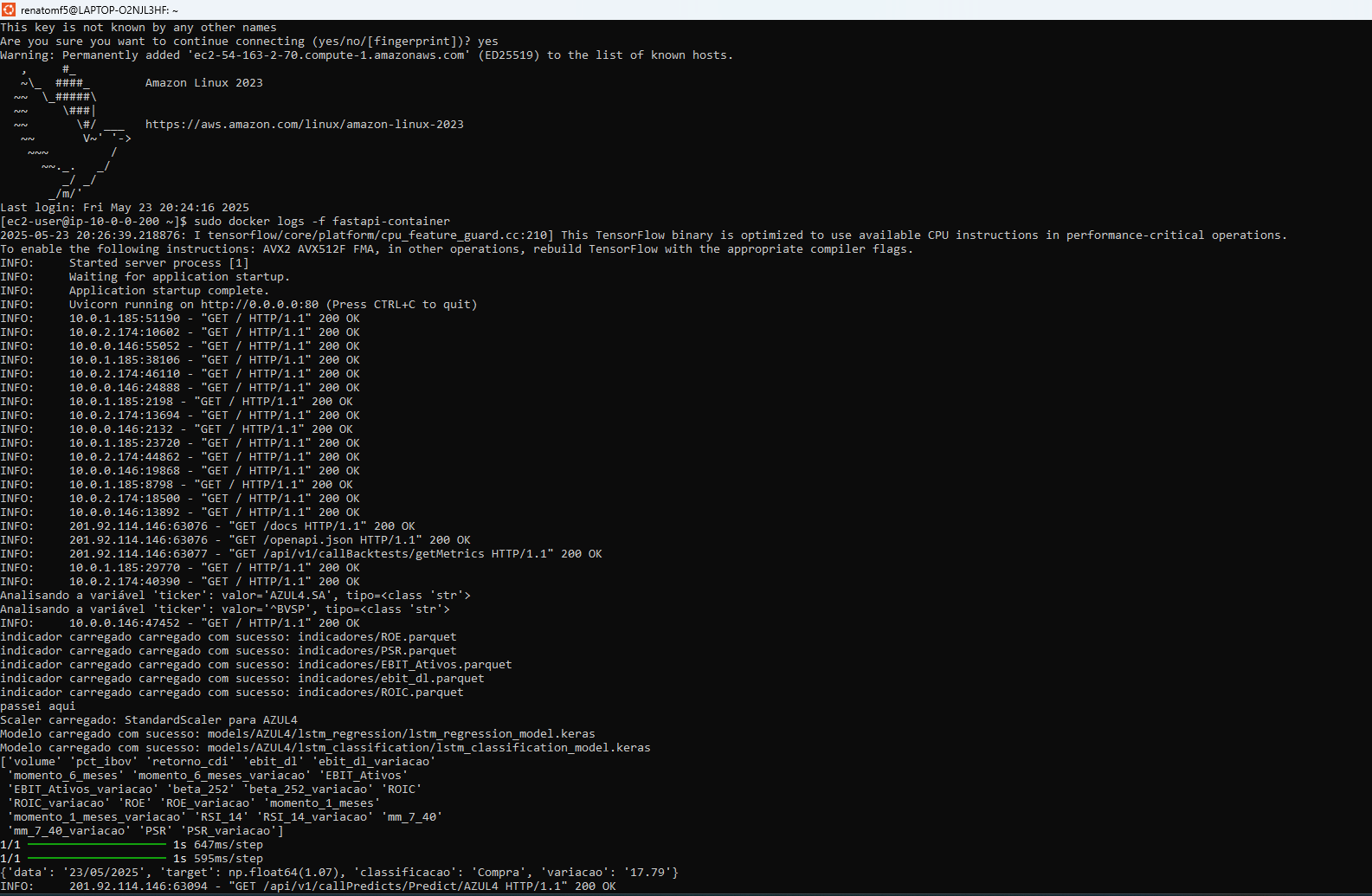
**Processo de Predição (rota principal):**

1. O **ticker** é recebido como parâmetro da plataforma web.
2. Os **indicadores técnicos e fundamentalistas** para o ticker são carregados do **data lake**.
3. É executada a função gera\_prediction(), da classe DataAnalyticsHandler, que:
   * Aplica as **mesmas transformações de dados** usadas no treinamento.
   * Carrega o modelo .keras e os **scalers** corretos do S3.
   * Utiliza os **últimos 21 registros** para montar a sequência de entrada.
   * Gera a predição e retorna os resultados em um JSON, que é exibido na interface web.

**Nota Técnica:**  
Grande parte das funções utilizadas na API reaproveitam a lógica do laboratório de modelagem apresentado anteriormente, garantindo **consistência entre treino e inferência**.

****



Evidencias de funcionamento da API em execução em servidor DOCKER em uma EC2 T3.SMALL  
Evidencias retiradas por conexão SSH da maquina local para a Instancia:  


**8. Sessão 6 – Estrutura dos Buckets S3**

A AWS S3 foi utilizada como **repositório central de modelos, dados e resultados**.

**Principais Buckets:**

* **datalake-tc4**
  + Armazena os indicadores fundamentalistas e técnicos extraídos e atualizados.
* **Bucket de Modelos por Ticker:**
  + Cada ação possui uma pasta com os modelos LSTM treinados e os respectivos scalers.
* **Backtests:**
  + Resultados dos testes de simulação (backtests) são armazenados com os históricos de retorno por ticker.

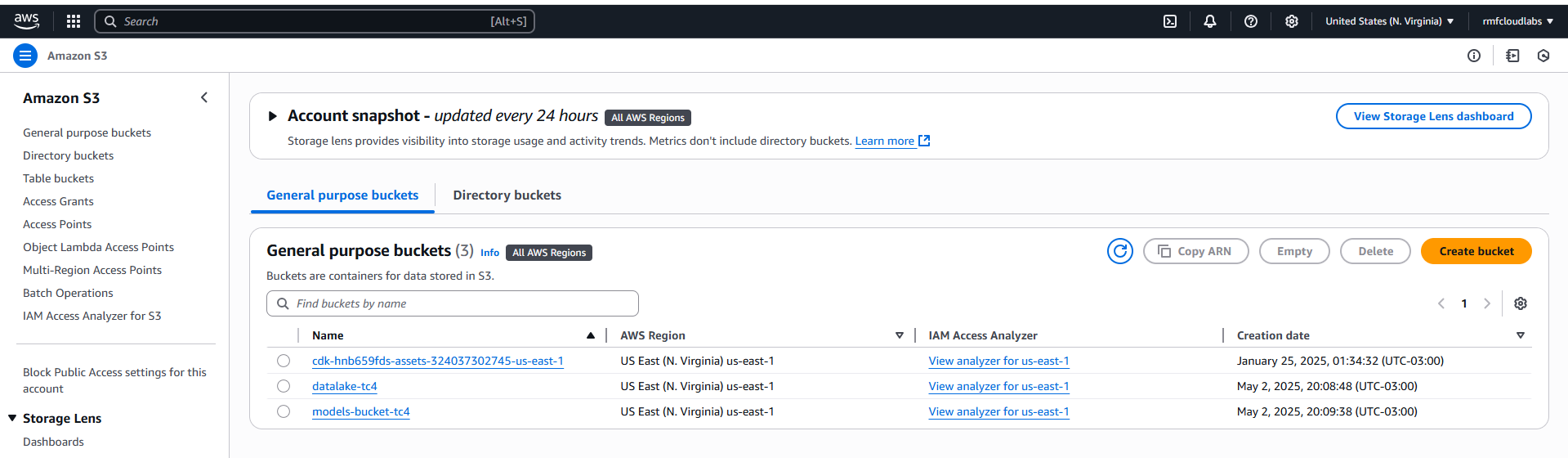
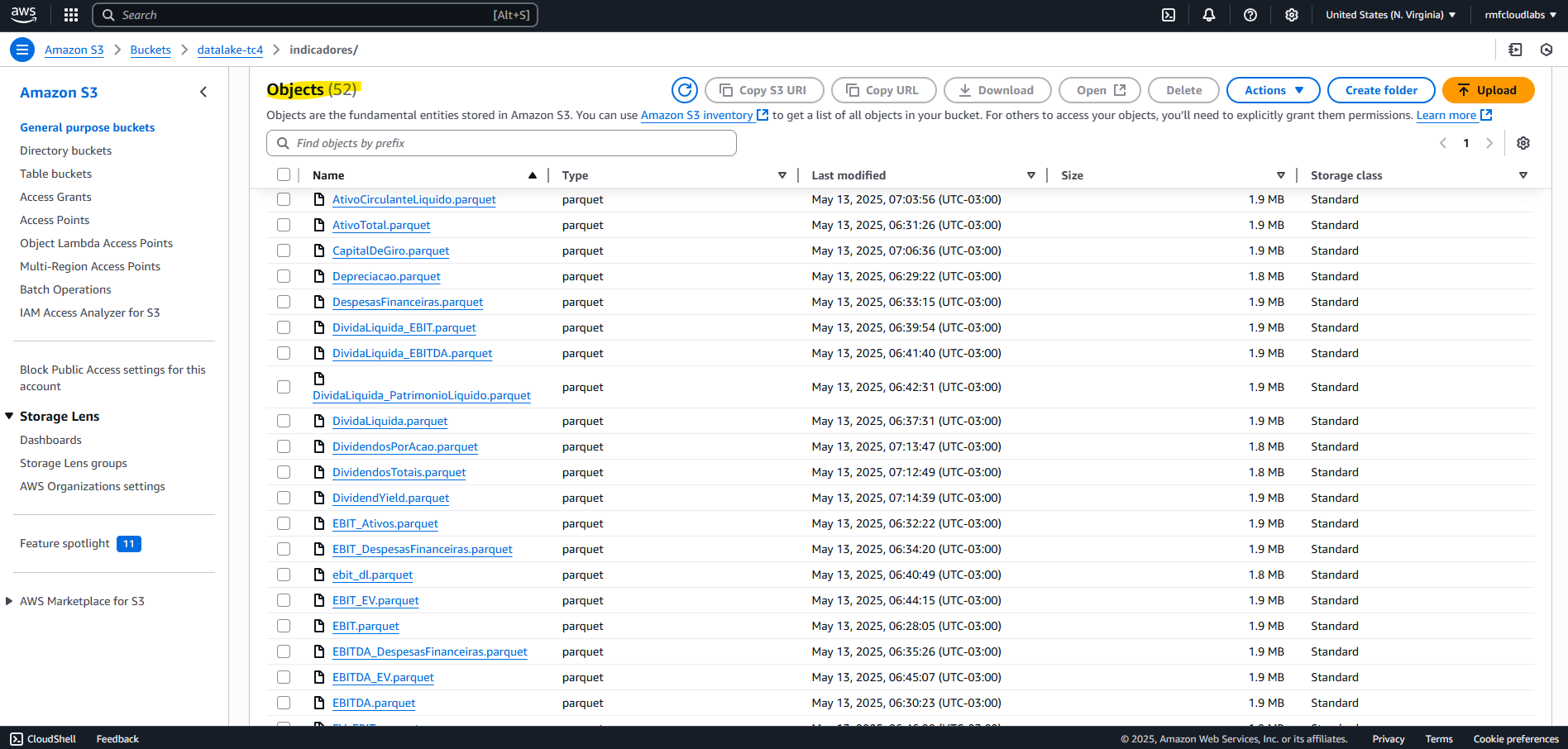
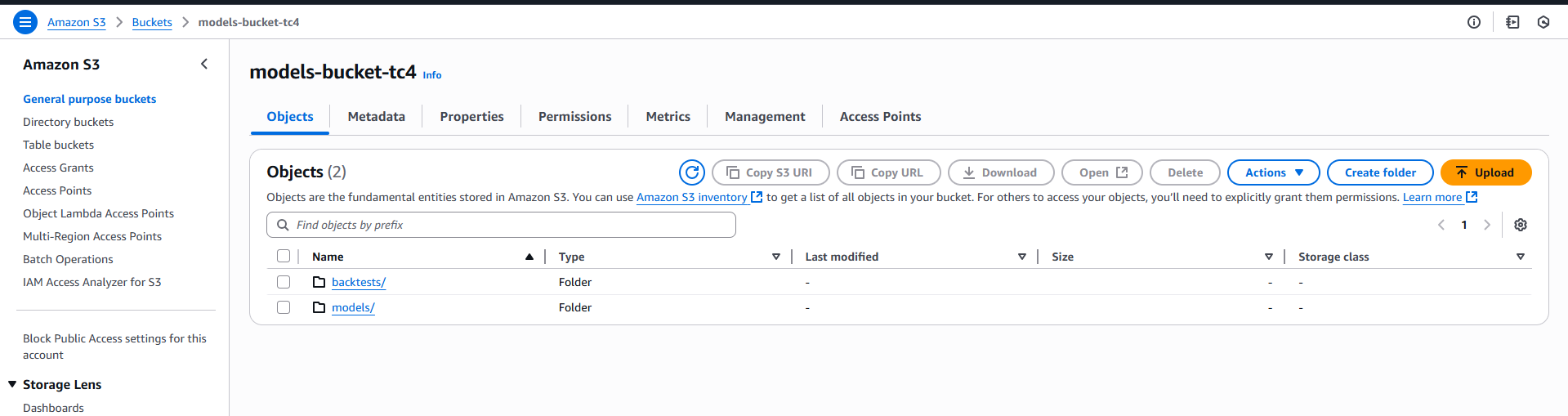
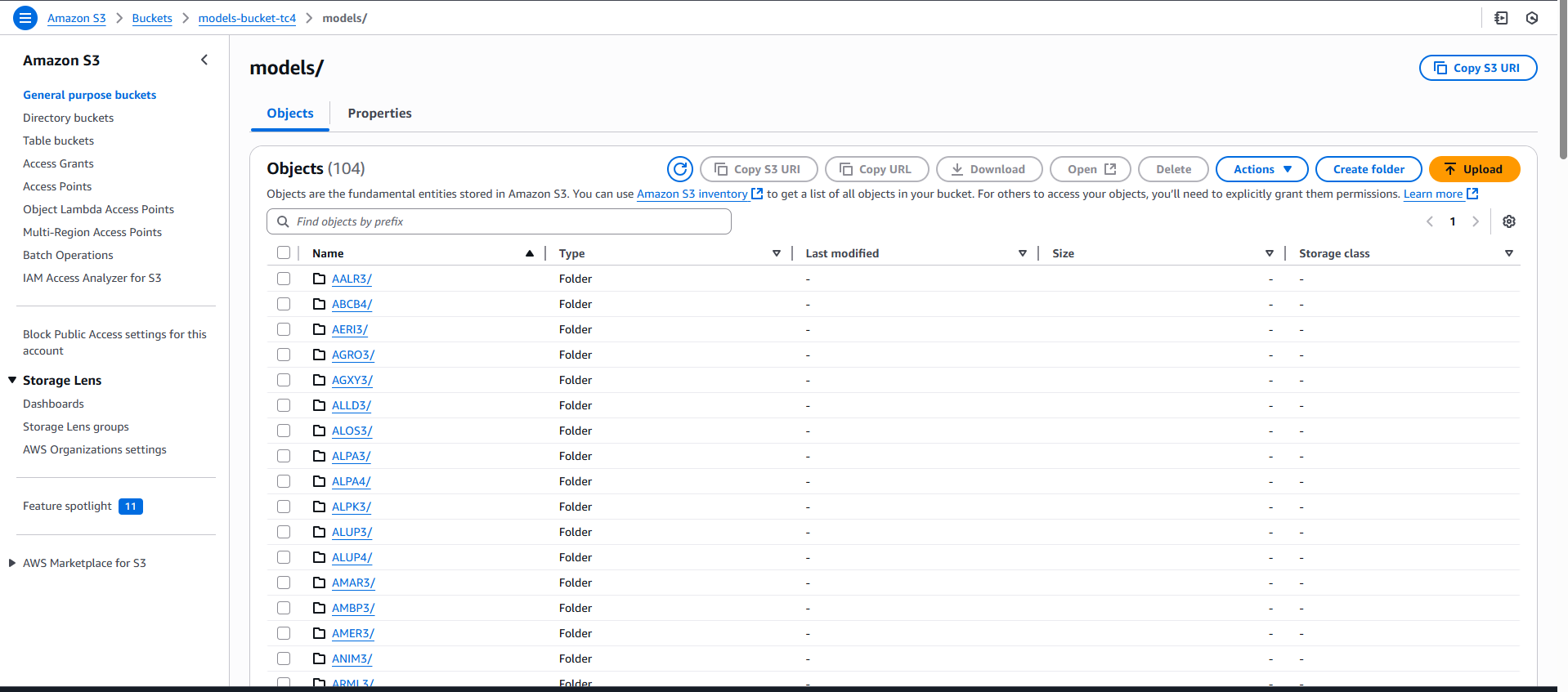
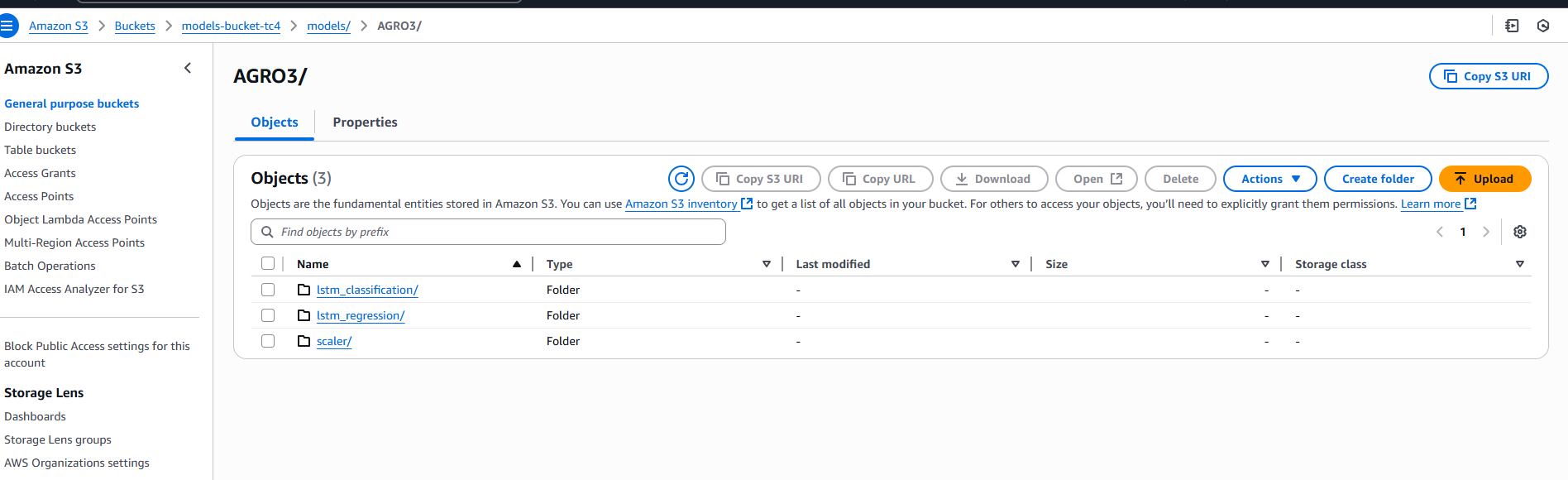


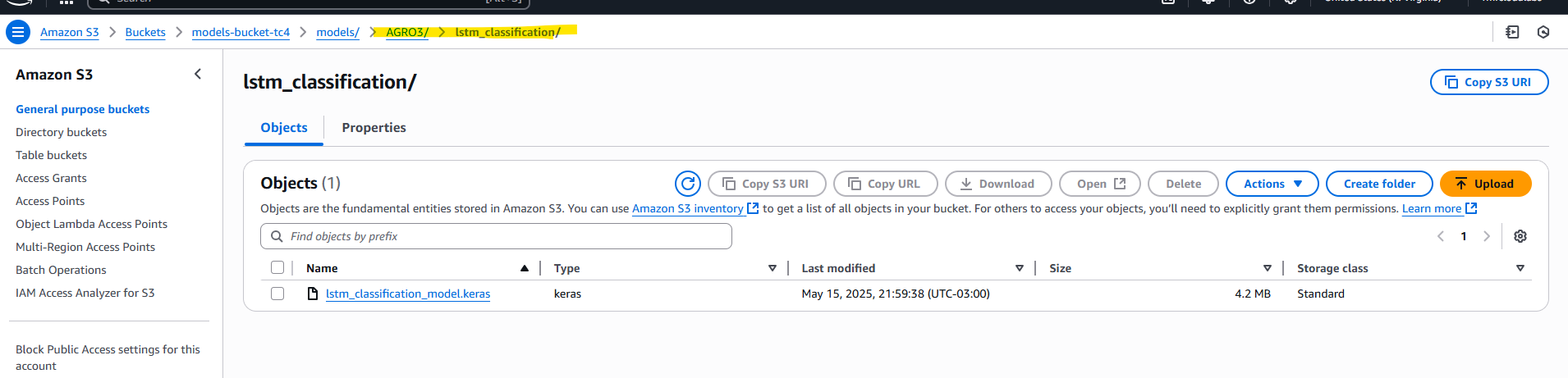
Imagem do data-Lake com indicadores “datalake-tc4”:



Estrutura do bucket que guarda as informações dos modelos dividos por ticker  






  
  
Backtests salvos para apresentação em tela :

