## TC-FIAP

## Aluno: Eric Lopes Mello

Vídeo de apresentação:

https://drive.google.com/file/d/1La8PQcD7atCoy9RvxpM9pMmqUPrgg5Eo/view

Repositório GIT:

https://github.com/ericlmello/previsao\_bolsa

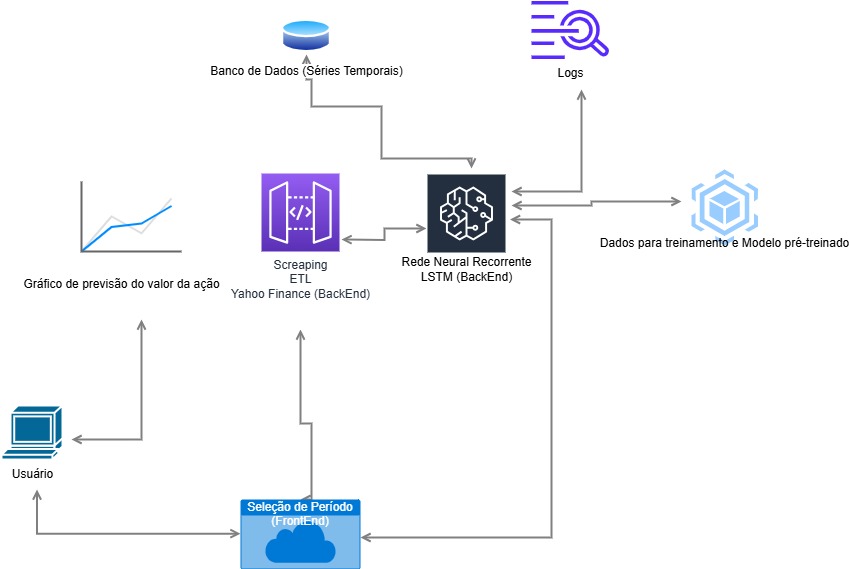
## Sistema de previsão de valores de ações na bolsa

Modelo de Rede Neural: LSTM

## Sumário

1. [Visão Geral do Projeto](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "1-visão-geral-do-projeto)
2. [Instalação e Configuração](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "2-instalação-e-configuração)
   * [2.1 Requisitos do Sistema](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "21-requisitos-do-sistema)
   * [2.2 Dependências](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "22-dependências)
   * [2.3 Instalação](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "23-instalação)
   * [2.4 Configuração Inicial](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "24-configuração-inicial)
3. [Estrutura do Projeto](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "3-estrutura-do-projeto)
4. [Funcionalidades Principais](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "4-funcionalidades-principais)
   * [4.1 Download de Dados de Ações](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "41-download-de-dados-de-ações)
   * [4.2 Processamento de Dados](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "42-processamento-de-dados)
   * [4.3 Modelo LSTM](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "43-modelo-lstm)
   * [4.4 Treinamento do Modelo](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "44-treinamento-do-modelo)
   * [4.5 Previsão de Preços](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "45-previsão-de-preços)
   * [4.6 Armazenamento em Banco de Dados](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "46-armazenamento-em-banco-de-dados)
   * [4.7 Visualizações](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "47-visualizações)
5. [Uso da Aplicação](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "5-uso-da-aplicação)
   * [5.1 Interface Web](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "51-interface-web)
   * [5.2 Fluxo de Uso](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "52-fluxo-de-uso)
   * [5.3 Endpoint de Treinamento](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "53-endpoint-de-treinamento)
6. [Monitoramento e Métricas](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "6-monitoramento-e-métricas)
   * [6.1 MLflow](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "61-mlflow)
   * [6.2 Prometheus](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "62-prometheus)
7. [Testes](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "7-testes)
   * [7.1 Testes Unitários](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "71-testes-unitários)
   * [7.2 Testes de Integração](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "72-testes-de-integração)
   * [7.3 Testes de Performance](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "73-testes-de-performance)
   * [7.4 Executando os Testes](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "74-executando-os-testes)
8. [Deploy e Produção](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "8-deploy-e-produção)
   * [8.1 Containerização com Docker](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "81-containerização-com-docker)
   * [8.2 Deploy em Cloud](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "82-deploy-em-cloud)
   * [8.3 Configurações de Produção](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "83-configurações-de-produção)
   * [8.4 Backup e Recuperação](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "84-backup-e-recuperação)
9. [Troubleshooting](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "9-troubleshooting)
   * [9.1 Problemas Comuns](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "91-problemas-comuns)
   * [9.2 Logs e Debugging](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "92-logs-e-debugging)
   * [9.3 Performance Issues](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "93-performance-issues)
10. [Informações Adicionais](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "10-informações-adicionais)
    * [10.1 Considerações de Segurança](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "101-considerações-de-segurança)
    * [10.2 Tratamento de Erros](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "102-tratamento-de-erros)
11. [Limitações e Melhorias Futuras](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "11-limitações-e-melhorias-futuras)
12. [Apêndice: Referência de API](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "12-apêndice-referência-de-api)
13. [Contribuição](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "13-contribuição)
14. [Licença](https://claude.ai/chat/459284e4-e454-42e5-9fc0-a74b160bfcd1" \l "14-licença)

## 1. Visão Geral do Projeto



### ****Objetivo****

O objetivo deste sistema é fornecer uma previsão do preço das ações de uma empresa usando o modelo LSTM (Long Short-Term Memory) para séries temporais. O usuário pode inserir um intervalo de datas para coletar os dados de ações, visualizar gráficos e obter previsões para o próximo dia e para os próximos 5 dias.

**LSTM** (Long Short-Term Memory) é um tipo de rede neural recorrente (RNN, do inglês Recurrent Neural Network) que foi projetada para permitir captura de dependências de longo prazo em dados sequenciais. É composta por **células de memória** que podem manter informações ao longo do tempo e são controladas por três portas principais:

* **Porta de entrada (input gate)**: Decide quais informações serão armazenadas na célula de memória
* **Porta de esquecimento (forget gate)**: Determina quais informações da célula de memória serão descartadas
* **Porta de saída (output gate)**: Controla o que será produzido como saída da célula

### ****Arquitetura do Sistema****

O sistema é composto por:

* **Frontend Web**: Interface Flask para interação com usuário
* **Backend de ML**: Modelos PyTorch LSTM para previsão
* **Banco de Dados**: SQLite para persistência de dados
* **Monitoramento**: MLflow e Prometheus para tracking
* **APIs Externas**: Yahoo Finance para dados de mercado

## 2. Instalação e Configuração

### 2.1 Requisitos do Sistema

* Python 3.8+ (recomendado Python 3.9 ou superior)
* Acesso à internet para download de dados de ações
* Mínimo de 4GB de RAM (8GB recomendado para datasets maiores)
* Processador multi-core (recomendado para treinamento de modelo)
* GPU compatível com CUDA (opcional, para treinamento acelerado)
* Espaço em disco: mínimo 2GB livres

### 2.2 Dependências

O sistema requer as seguintes bibliotecas Python:

| Biblioteca | Versão Mínima | Uso no Sistema |
| --- | --- | --- |
| Flask | 2.0.0+ | Framework de aplicação web |
| NumPy | 1.20.0+ | Computação numérica |
| Pandas | 1.3.0+ | Manipulação de dados |
| yfinance | 0.1.70+ | Download de dados de ações |
| Matplotlib | 3.4.0+ | Visualização de dados |
| scikit-learn | 1.0.0+ | Pré-processamento de dados |
| PyTorch | 1.9.0+ | Framework de deep learning |
| MLflow | 1.20.0+ | Rastreamento de experimentos |
| Prometheus | 0.9.0+ | Monitoramento de métricas |

### 2.3 Instalação

# Clonar o repositório

git clone https://github.com/ericlmello/previsao\_bolsa

cd previsao\_bolsa

# Criar ambiente virtual (recomendado)

python -m venv venv

source venv/bin/activate # No Windows: venv\Scripts\activate

# Instalar dependências

pip install -r requirements.txt

### 2.4 Configuração Inicial

Antes de executar o aplicativo, o sistema inicializa automaticamente o banco de dados SQLite:

# Iniciar o servidor web

python app.py

O servidor estará disponível em http://localhost:5000.

Configurações adicionais podem ser ajustadas através de variáveis de ambiente:

export FLASK\_ENV=development # Para desenvolvimento

export MLFLOW\_TRACKING\_URI=sqlite:///mlflow.db

export MODEL\_PATH=./models

export DATA\_PATH=./data

## 3. Estrutura do Projeto

O projeto é organizado nos seguintes componentes principais:

stock-prediction-app/

├── app.py # Aplicativo principal (Flask + lógica de negócios)

├── model\_utils.py # Utilitários do modelo (importado pelo app.py)

├── requirements.txt # Dependências do projeto

├── static/ # Arquivos estáticos (CSS, JS, imagens)

├── templates/ # Templates HTML para interface web

│ ├── form.html # Formulário para entrada de dados

│ ├── result.html # Página de resultados da previsão

│ └── history.html # Página de histórico

└── stock\_prediction.db # Banco de dados SQLite (criado automaticamente)

## 4. Funcionalidades Principais

**Inserção de período para treinamento do modelo:**

### 4.1 Download de Dados de Ações

O sistema utiliza a biblioteca yfinance para baixar dados históricos de ações:

def download\_with\_timeout(symbol, start\_date, end\_date, timeout=180):

"""

Baixa dados históricos de ações usando yfinance.

Parâmetros:

- symbol: Símbolo da ação (ex: "QQQ", "AAPL")

- start\_date: Data inicial no formato YYYY-MM-DD

- end\_date: Data final no formato YYYY-MM-DD

- timeout: Tempo limite em segundos

Retorna:

- DataFrame do pandas com dados históricos de ações

"""

**Recursos de Download:**

* Retry automático em caso de falha
* Cache de dados para evitar downloads desnecessários
* Timeout configurável
* Validação de dados recebidos

### 4.2 Processamento de Dados

As séries temporais são normalizadas e transformadas:

def create\_sequences(data, seq\_length):

"""

Cria sequências temporais para o modelo LSTM.

Parâmetros:

- data: Dados normalizados

- seq\_length: Tamanho da sequência (número de dias anteriores)

Retorna:

- xs: Sequências de entrada

- ys: Valores alvo (próximo dia)

"""

**Etapas do Processamento:**

1. Limpeza de dados (remoção de outliers)
2. Normalização com MinMaxScaler
3. Criação de sequências temporais
4. Divisão treino/validação

### 4.3 Modelo LSTM

Implementação com PyTorch:

class LSTMModel(nn.Module):

def \_\_init\_\_(self, input\_size=1, hidden\_layer\_size=150,

output\_size=1, num\_layers=1):

"""

Modelo LSTM para previsão de preços de ações.

Arquitetura:

- Camada LSTM com hidden\_layer\_size neurônios

- Camada Linear para saída

- Dropout para regularização

"""

### 4.4 Treinamento do Modelo

def train\_lstm(model, X, y, num\_epochs=55, batch\_size=32):

"""

Treina o modelo LSTM com monitoramento de métricas.

Features:

- Early stopping para evitar overfitting

- Learning rate scheduling

- Validação cruzada

- Tracking de métricas com MLflow

"""

### 4.5 Previsão de Preços

def predict\_prices(data, model, scaler, sequence\_length, days\_ahead=1):

"""

Gera previsões para dias futuros.

Tipos de Previsão:

- Próximo dia: Previsão single-step

- Múltiplos dias: Previsão multi-step recursiva

"""

### 

### 

### 4.6 Armazenamento em Banco de Dados

**Estrutura do Banco:**

1. **raw\_stock\_data**: Dados históricos das ações
2. **predictions**: Previsões geradas pelo modelo
3. **model\_metrics**: Métricas de performance dos modelos
4. **saved\_models**: Informações sobre modelos salvos

### 4.7 Visualizações

O sistema gera cinco tipos de gráficos:

1. **Série temporal**: Preços históricos
2. **Histograma**: Distribuição de preços
3. **Previsões futuras**: Próximos 5 dias
4. **Curva de aprendizado**: Loss durante treinamento
5. **Métricas sigmoid**: Indicador de qualidade do modelo

## 5. Uso da Aplicação

### 5.1 Interface Web

**Páginas Principais:**

1. **Página inicial (/)**: Formulário de entrada de dados
2. **Resultados**: Exibição de previsões e gráficos
3. **Histórico (/history)**: Visualização de previsões anteriores
4. **Treinamento (/train\_model)**: Interface para treinar novos modelos

### 5.2 Fluxo de Uso

**Fluxo Básico:**

1. Acesse http://localhost:5000
2. Preencha o formulário com símbolo da ação e datas
3. Escolha entre usar modelo salvo ou treinar novo
4. Visualize resultados e métricas
5. Consulte histórico de previsões

### 5.3 Endpoint de Treinamento

O endpoint /train\_model permite treinar modelos sem gerar previsões:

@app.route('/train\_model', methods=['GET', 'POST'])

def train\_model\_route():

"""

Treina e salva modelo para uso posterior.

Parâmetros configuráveis:

- Número de épocas

- Tamanho da sequência

- Batch size

"""

## 6. Monitoramento e Métricas

### 6.1 MLflow

**Tracking de Experimentos:**

* Parâmetros do modelo (epochs, batch\_size, sequence\_length)
* Métricas de performance (MSE, MAE, RMSE)
* Artefatos do modelo (modelo treinado)
* Comparação entre experimentos

**Inicialização:**

mlflow ui --backend-store-uri sqlite:///mlflow.db

### 6.2 Prometheus

**Métricas Coletadas:**

* model\_training\_duration\_seconds: Tempo de treinamento
* model\_loss\_count: Função de perda
* model\_sigmoid\_value: Valor sigmoid baseado na perda

## 7. Testes

### 7.1 Testes Unitários

**Estrutura de Testes:**

# tests/test\_model.py

import pytest

from model\_utils import create\_sequences, build\_lstm\_model

def test\_create\_sequences():

"""Testa criação de sequências temporais."""

pass

def test\_lstm\_model\_creation():

"""Testa criação do modelo LSTM."""

pass

def test\_model\_prediction():

"""Testa geração de previsões."""

pass

**Testes Implementados:**

* Criação de sequências temporais
* Inicialização do modelo LSTM
* Processo de treinamento
* Geração de previsões
* Funções de utilidade

### 7.2 Testes de Integração

# tests/test\_api.py

import pytest

from app import app

@pytest.fixture

def client():

app.config['TESTING'] = True

return app.test\_client()

def test\_home\_page(client):

"""Testa página inicial."""

response = client.get('/')

assert response.status\_code == 200

def test\_prediction\_endpoint(client):

"""Testa endpoint de previsão."""

pass

**Cenários Testados:**

* Endpoints da API Flask
* Integração com banco de dados
* Download de dados de ações
* Fluxo completo de previsão

### 7.3 Testes de Performance

# tests/test\_performance.py

import time

import pytest

def test\_training\_performance():

"""Testa performance do treinamento."""

pass

def test\_prediction\_speed():

"""Testa velocidade de previsão."""

pass

def test\_memory\_usage():

"""Monitora uso de memória."""

pass

### 7.4 Executando os Testes

# Executar todos os testes

pytest

# Executar testes específicos

pytest tests/test\_model.py

# Executar com coverage

pytest --cov=app --cov-report=html

# Executar testes de performance

pytest tests/test\_performance.py -v

## 8. Deploy e Produção

### 8.1 Containerização com Docker

**Dockerfile:**

FROM python:3.9-slim

WORKDIR /app

COPY requirements.txt .

RUN pip install --no-cache-dir -r requirements.txt

COPY . .

EXPOSE 5000

CMD ["gunicorn", "--bind", "0.0.0.0:5000", "app:app"]

**Docker Compose:**

version: '3.8'

services:

web:

build: .

ports:

- "5000:5000"

environment:

- FLASK\_ENV=production

volumes:

- ./data:/app/data

- ./models:/app/models

prometheus:

image: prom/prometheus

ports:

- "9090:9090"

volumes:

- ./prometheus.yml:/etc/prometheus/prometheus.yml

### 8.2 Deploy em Cloud

**AWS Deployment:**

# Deploy usando AWS ECS

aws ecs create-cluster --cluster-name stock-prediction

# Deploy usando AWS Lambda (para previsões)

serverless deploy

**Google Cloud Deployment:**

# Deploy no Google Cloud Run

gcloud run deploy stock-prediction \

--image gcr.io/PROJECT-ID/stock-prediction \

--platform managed

### 8.3 Configurações de Produção

**Variáveis de Ambiente:**

FLASK\_ENV=production

SECRET\_KEY=your-secret-key

DATABASE\_URL=postgresql://user:pass@host:port/db

REDIS\_URL=redis://host:port/0

MLFLOW\_TRACKING\_URI=postgresql://...

**Configurações de Segurança:**

* HTTPS obrigatório
* Rate limiting
* Autenticação JWT
* Validação de entrada
* Logs de auditoria

### 8.4 Backup e Recuperação

**Estratégia de Backup:**

# Backup do banco de dados

sqlite3 stock\_prediction.db ".backup backup.db"

# Backup dos modelos

tar -czf models\_backup.tar.gz models/

# Backup automatizado (cron)

0 2 \* \* \* /path/to/backup\_script.sh

## 9. Troubleshooting

### 9.1 Problemas Comuns

**Erro de Download de Dados:**

Sintoma: "Erro ao coletar dados"

Causa: Conexão com Yahoo Finance falhou

Solução: Verificar conectividade e símbolo da ação

**Erro de Memória:**

Sintoma: "RuntimeError: CUDA out of memory"

Causa: Batch size muito grande para GPU

Solução: Reduzir batch\_size ou usar CPU

**Modelo não Converge:**

Sintoma: Loss não diminui

Causa: Learning rate inadequado

Solução: Ajustar hiperparâmetros

### 9.2 Logs e Debugging

**Configuração de Logs:**

import logging

logging.basicConfig(

level=logging.INFO,

format='%(asctime)s - %(name)s - %(levelname)s - %(message)s',

handlers=[

logging.FileHandler('logs/app.log'),

logging.StreamHandler()

]

)

**Debug Mode:**

# Executar em modo debug

FLASK\_DEBUG=1 python app.py

# Logs detalhados

export LOG\_LEVEL=DEBUG

### 9.3 Performance Issues

**Otimizações:**

* Usar GPU quando disponível
* Implementar cache de dados
* Otimizar queries do banco
* Usar batch processing
* Implementar lazy loading

## 10. Informações Adicionais

### 10.1 Considerações de Segurança

**Medidas Implementadas:**

* Desabilitação de verificação SSL (apenas desenvolvimento)
* Sanitização de entrada de dados
* Rate limiting por IP
* Logs de auditoria

**Recomendações para Produção:**

* Implementar HTTPS
* Usar certificados SSL válidos
* Implementar autenticação
* Criptografar dados sensíveis
* Monitorar tentativas de ataque

### 10.2 Tratamento de Erros

**Categorias de Erro:**

1. **Erros de Rede**: Timeout, conexão perdida
2. **Erros de Dados**: Dados inválidos, faltando
3. **Erros de Modelo**: Falha no treinamento
4. **Erros de Sistema**: Memória, disco

**Estratégias de Recovery:**

* Retry automático com backoff
* Fallback para dados cached
* Notificação de administradores
* Graceful degradation

## 11. Limitações e Melhorias Futuras

### Limitações Atuais:

* Modelo LSTM simples pode não capturar todas as nuances do mercado
* Não considera fatores externos (notícias, eventos econômicos)
* Utiliza apenas o preço de fechamento como entrada
* Limitado a previsões de curto prazo

### Melhorias Futuras:

**Aprimoramentos do Modelo:**

* Implementação de Transformers
* Ensemble de múltiplos modelos
* Incorporação de análise de sentimento
* Uso de indicadores técnicos

**Melhorias na Interface:**

* Dashboard em tempo real
* Mobile app
* Alertas personalizáveis
* Comparação de múltiplas ações

**Funcionalidades Avançadas:**

* Backtesting automatizado
* Portfolio optimization
* Risk assessment
* API REST completa

## 12. Apêndice: Referência de API

### Endpoints da Aplicação Web

| Endpoint | Método | Descrição | Parâmetros |
| --- | --- | --- | --- |
| / | GET | Exibe formulário | Nenhum |
| / | POST | Gera previsões | symbol, start\_date, end\_date, use\_saved\_model |
| /history | GET | Histórico de previsões | symbol (opcional) |
| /train\_model | GET/POST | Treina novos modelos | symbol, start\_date, end\_date, num\_epochs |

### Funções Principais

| Função | Descrição | Parâmetros |
| --- | --- | --- |
| download\_with\_timeout() | Download de dados | symbol, start\_date, end\_date, timeout |
| create\_sequences() | Criação de sequências | data, seq\_length |
| build\_lstm\_model() | Construção do modelo | sequence\_length, hidden\_size |
| train\_lstm() | Treinamento | model, X, y, num\_epochs, batch\_size |
| predict\_prices() | Geração de previsões | data, model, scaler, sequence\_length, days\_ahead |

## 13. Contribuição

### Como Contribuir

1. **Fork** o repositório
2. **Crie** uma branch para sua feature
3. **Implemente** suas mudanças
4. **Adicione** testes
5. **Execute** os testes existentes
6. **Submeta** um Pull Request

### Guidelines

* Siga PEP 8 para estilo de código
* Adicione docstrings para todas as funções
* Mantenha cobertura de testes acima de 80%
* Atualize a documentação

### Reportar Bugs

Use o template de issue do GitHub:

* Descrição do problema
* Passos para reproduzir
* Comportamento esperado
* Screenshots se aplicável