Relatório de Atividades do Projeto

Mateus Mendes Caminha

Contribuição no Código:

Responsabilidades:

1. **Importação dos Dados:**
   - Mateus foi responsável pela importação dos dados necessários para o projeto utilizando a biblioteca `yfinance`. Ele escreveu código para baixar dados históricos do mercado financeiro de fontes confiáveis.
   - Garantiu que os dados fossem acessíveis e organizados adequadamente para as etapas subsequentes de processamento e análise.

```
import yfinance as yf


# Baixando os dados históricos
df = yf.download('AAPL', start='2010-01-01', end='2020-01-01')
```

2. **Preparo dos Dados:**
   - Mateus realizou a limpeza dos dados, removendo valores ausentes e outliers que poderiam afetar negativamente o desempenho dos modelos.
   - Aplicou técnicas de normalização utilizando o `MinMaxScaler` da biblioteca `sklearn` para escalar os dados entre 0 e 1, facilitando o treinamento eficiente dos modelos LSTM e GRU.
   - Dividiu os dados em conjuntos de treinamento e teste, garantindo que os modelos pudessem ser treinados e validados

adequadamente.

```
from sklearn.preprocessing import MinMaxScaler
import numpy as np


# Normalização dos dados
scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(df['Close'].values.reshape(-1, 1))


# Dividindo os dados em conjuntos de treinamento e teste
train_data_len = int(np.ceil(len(scaled_data) * 0.95))
train_data = scaled_data[0:int(train_data_len), :]
```

3. **Análise Pós-Treinamento:**
    - Após o treinamento dos modelos LSTM e GRU, Mateus conduziu análises detalhadas para avaliar a eficiência dos modelos. Ele utilizou métricas como o erro quadrático médio (MSE) para medir o desempenho.
    - Gerou gráficos de comparação entre os valores previstos e os valores reais utilizando `matplotlib` e `seaborn`, facilitando a visualização dos resultados e identificação de áreas de melhoria.
    - Documentou as descobertas e insights, fornecendo uma base para ajustes e otimizações futuras.

```
import matplotlib.pyplot as plt
import seaborn as sns


# Gerando gráficos de comparação
plt.figure(figsize=(14, 5))
plt.plot(df['Close'], label='Real')
plt.plot(predictions, label='Previsto')
plt.legend()
plt.show()
```

Importância das Contribuições:

- **Importação e Preparo dos Dados:**
  - A qualidade dos dados é crucial para o sucesso de qualquer modelo de aprendizado de máquina. Dados bem preparados e limpos garantem que os modelos possam aprender padrões úteis e generalizáveis.
  - A normalização e a divisão adequada dos dados são etapas fundamentais para garantir que os modelos não sejam tendenciosos e possam generalizar bem em dados não vistos.
- **Análise Pós-Treinamento:**
  - A análise detalhada permite identificar pontos fortes e fracos dos modelos, fornecendo insights valiosos para ajustes e otimizações.
  - A documentação das análises facilita a comunicação dos resultados e a tomada de decisões informadas para melhorias futuras.

Paulo Vinicius Alves Melo

Contribuição no Código:

Responsabilidades:

1. **Montagem do Modelo LSTM:**
   - Paulo foi responsável pela construção do modelo LSTM. Ele definiu a arquitetura do modelo utilizando a biblioteca `tensorflow.keras`.
   - Selecionou as camadas apropriadas, incluindo camadas LSTM e camadas `Dense`, ajustando o número de unidades em cada camada para otimizar o desempenho.

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense


# Construção do modelo LSTM
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(time_step, 1)))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dense(units=25))
model.add(Dense(units=1))
```

2. **Treinamento do Modelo:**
    - Conduziu o treinamento do modelo LSTM, ajustando hiperparâmetros como o tamanho do batch e o número de épocas para garantir que o modelo pudesse aprender de forma eficiente.
    - Utilizou o otimizador `adam` e a função de perda `mean_squared_error` para treinar o modelo, minimizando o erro entre as previsões do modelo e os valores reais.

```
model.compile(optimizer='adam', loss='mean_squared_error')
model.fit(X_train, Y_train, batch_size=1, epochs=1)
```

3. **Testes e Melhoria dos Parâmetros:**
    - Realizou testes detalhados para avaliar o desempenho do modelo em dados de teste. Analisou métricas de desempenho e ajustou os hiperparâmetros conforme necessário para melhorar a precisão.
    - Aplicou técnicas de validação cruzada para garantir que o modelo pudesse generalizar bem em diferentes subconjuntos de dados.

Importância das Contribuições:

- **Construção e Treinamento do Modelo:**
    - Um modelo bem construído e treinado é fundamental para obter previsões precisas e confiáveis. A arquitetura do modelo e a escolha dos hiperparâmetros impactam diretamente o desempenho.
    - O treinamento eficiente garante que o modelo possa aprender padrões nos dados de forma eficaz, minimizando o erro e melhorando a capacidade preditiva.

- **Otimização dos Parâmetros:**

- A melhoria contínua dos parâmetros do modelo é vital para maximizar o desempenho e garantir que o modelo opere de forma eficiente em dados reais.
- A aplicação de técnicas de validação cruzada ajuda a prevenir overfitting, garantindo que o modelo possa generalizar bem em diferentes conjuntos de dados.

Carlos Mariano de Souza Rocha Neto

Contribuição no Código:

Responsabilidades:

1. **Instanciação do Modelo Prophet e GRU:**
   - Carlos foi responsável pela configuração e instanciação dos modelos Prophet e GRU, utilizando bibliotecas como `fbprophet` e `tensorflow.keras`.
   - Configurou os parâmetros iniciais dos modelos, garantindo que estivessem prontos para o treinamento.

```
from fbprophet import Prophet


# Instanciação do modelo Prophet
prophet_model = Prophet()
from tensorflow.keras.layers import GRU


# Construção do modelo GRU
gru_model = Sequential()
gru_model.add(GRU(units=50, return_sequences=True, input_shape=(time_step, 1)))
gru_model.add(GRU(units=50, return_sequences=False))
gru_model.add(Dense(units=25))
gru_model.add(Dense(units=1))
```

2. **Treinamento dos Modelos:**
   - Conduziu o treinamento dos modelos Prophet e GRU, aplicando técnicas de ajuste fino para otimizar os parâmetros e melhorar o desempenho.

- Utilizou dados de treino para ajustar os modelos, garantindo que pudessem aprender padrões temporais e realizar previsões precisas.

```
# Treinamento do modelo Prophet
prophet_model.fit(train_data)


# Treinamento do modelo GRU
gru_model.compile(optimizer='adam', loss='mean_squared_error')
gru_model.fit(X_train, Y_train, batch_size=1, epochs=1)
```

3. **Testes para Melhoria dos Parâmetros:**
   - Realizou testes rigorosos para avaliar o desempenho dos modelos em dados de teste. Analisou métricas de desempenho e ajustou os hiperparâmetros conforme necessário para melhorar a precisão.
   - Aplicou técnicas de validação cruzada e análise de resíduos para identificar e corrigir possíveis erros nos modelos.

Importância das Contribuições:

- **Instanciação e Treinamento dos Modelos:**
  - A configuração correta dos modelos Prophet e GRU é fundamental para garantir que estejam prontos para o treinamento e possam aprender de forma eficaz a partir dos dados.
  - O treinamento eficiente dos modelos é crucial para garantir que possam realizar previsões precisas e confiáveis.
- **Ajuste de Parâmetros:**
  - A otimização contínua dos parâmetros dos modelos ajuda a garantir que eles possam operar de forma eficiente e com alto desempenho, proporcionando previsões precisas e confiáveis.
  - A aplicação de técnicas de validação cruzada e análise de resíduos é vital para garantir que os modelos possam generalizar bem e não sejam tendenciosos.