**Projeto de Previsão de Preços de Ações com LSTM (Apple - AAPL)**

**1. Introdução e Objetivo**

Este projeto tem como objetivo aplicar Redes Neurais Recorrentes do tipo LSTM (Long Short-Term Memory) para prever os preços de fechamento das ações da Apple Inc. (AAPL). O projeto é dividido em três etapas principais: coleta e preparação dos dados, construção e treinamento do modelo, e análise de desempenho.

**2. Etapa 1: Coleta e Preparação dos Dados**

A base de dados foi obtida através da biblioteca yfinance, utilizando cotações históricas da Apple entre 01/01/2015 e 01/01/2025.

symbol = 'AAPL'

start\_date = '2015-01-01'

end\_date = '2025-01-01'

df = yf.download(symbol, start=start\_date, end=end\_date, auto\_adjust=False)

Apenas a coluna "Close" (preço de fechamento) foi considerada para a análise, por ser um dos indicadores mais representativos em análises técnicas.

Após a coleta, os dados foram normalizados com MinMaxScaler, que transforma os valores para o intervalo entre 0 e 1, facilitando o aprendizado do modelo.

scaler = MinMaxScaler(feature\_range=(0, 1))

scaled\_data = scaler.fit\_transform(df[['Close']])

Os dados foram divididos em 80% para treino e 20% para teste. Em seguida, foi criada uma função create\_dataset que constrói janelas de observação com 60 timesteps para entrada e a próxima posição como saída esperada.

def create\_dataset(data, time\_step=1):

X, Y = [], []

for i in range(len(data)-time\_step-1):

a = data[i:(i+time\_step), 0]

X.append(a)

Y.append(data[i + time\_step, 0])

return np.array(X), np.array(Y)

**Por que usar 60 timesteps?** O uso de 60 pontos passados (cerca de 3 meses de preços diários) permite capturar padrões de curto prazo no comportamento da série temporal.

**3. Etapa 2: Construção do Modelo LSTM**

Foi construída uma rede sequencial composta por:

* Duas camadas LSTM com 50 neurônios cada;
* Uma camada densa intermediária com 25 neurônios;
* Uma camada densa final com 1 neurônio (previsão).

model = Sequential()

model.add(LSTM(50, return\_sequences=True, input\_shape=(time\_step, 1)))

model.add(LSTM(50, return\_sequences=False))

model.add(Dense(25))

model.add(Dense(1))

**Racional para a arquitetura escolhida:**

* As camadas LSTM são eficazes para lidar com dependências temporais em séries históricas;
* return\_sequences=True na primeira LSTM é necessário para alimentar a segunda camada LSTM;
* A camada densa com 25 neurônios serve como camada de ativação intermediarária, ajudando na modelagem da complexidade;
* A camada final com um único neurônio produz a previsão do próximo valor da série.

O modelo foi compilado com otimizador Adam e perda mean\_squared\_error, adequada para problemas de regressão:

model.compile(optimizer='adam', loss='mean\_squared\_error')

model.fit(X\_train, y\_train, batch\_size=64, epochs=30, validation\_data=(X\_test, y\_test))

**4. Etapa 3: Previsão e Avaliação**

As previsões foram feitas sobre o conjunto de teste, e os valores foram revertidos à escala original:

predictions = model.predict(X\_test)

predictions = scaler.inverse\_transform(predictions)

y\_test\_actual = scaler.inverse\_transform(y\_test.reshape(-1, 1))

A avaliação foi feita com duas métricas:

* **MAE (Erro Absoluto Médio)**: mede o erro médio absoluto entre valores reais e previstos.
* **RMSE (Raiz do Erro Quadrático Médio)**: penaliza mais erros maiores, sendo sensível a outliers.

mae = mean\_absolute\_error(y\_test\_actual, predictions)

rmse = math.sqrt(mean\_squared\_error(y\_test\_actual, predictions))

Foram feitos testes com diferentes quantidades de épocas para analisar impacto na performance:

|  |  |  |
| --- | --- | --- |
| **Epochs** | **MAE** | **RMSE** |
| 30 | 6.07 | 7.24 |
| 50 | 5.07 | 6.20 |
| 100 | 2.07 | 2.74 |

**Conclusão:** Treinar por mais épocas proporcionou uma redução significativa nos erros. O modelo com 100 épocas teve melhor capacidade de generalização

model.fit(X\_train, y\_train, batch\_size=64, epochs=100, validation\_data=(X\_test, y\_test))

**5. Visualização**

Um gráfico foi gerado para comparar os valores reais e previstos:

plt.figure(figsize=(14,6))

plt.plot(y\_test\_actual, label='Real')

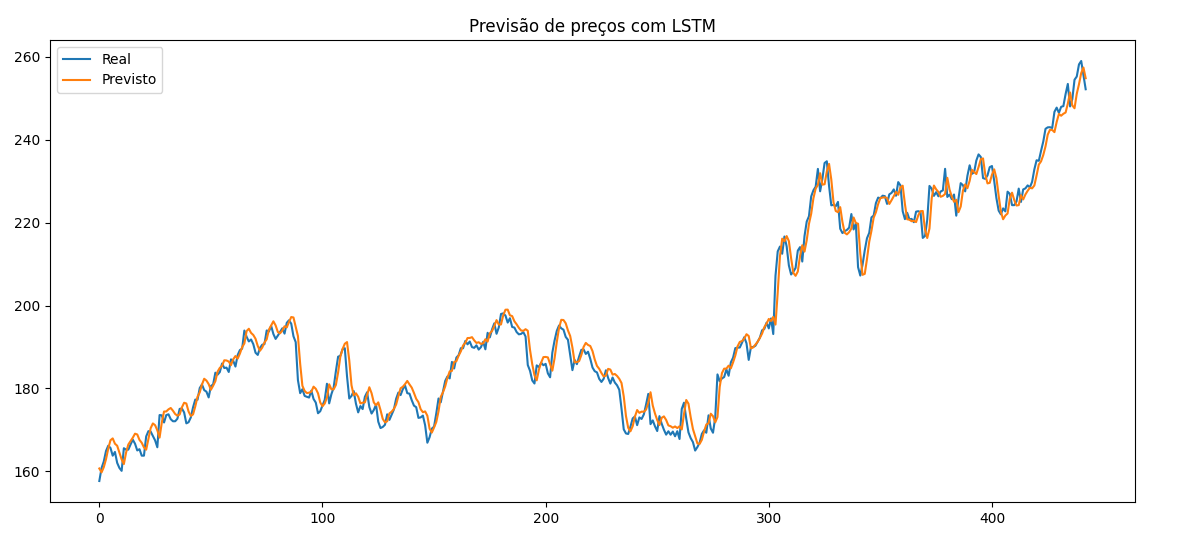
plt.plot(predictions, label='Previsto')

plt.title("Previsão de preços com LSTM")

plt.legend()

plt.show()

Este tipo de visualização é essencial para verificar qualitativamente a performance do modelo.



**6. Considerações Finais**

O projeto demonstra a eficácia de LSTM na previsão de séries temporais financeiras. A escolha de 100 épocas e pré-processamento adequado foram cruciais para alcançar bons resultados. O pipeline é replicável e pode ser adaptado para outras ações com ajustes mínimos.