

```

# -*- coding: utf-8 -*- """
STREAMLIT LSTM.ipynb Automatically generated by Colab. Original file
is located at https://colab.research.google.com/drive/1eoN80OO2qR9Eyhovyf9Kc39c7xjxeHiS
"""
# streamlit_app.py import streamlit as st import yfinance as yf import pandas as pd import
numpy as np from sklearn.linear_model import LinearRegression from sklearn.metrics import
mean_squared_error import matplotlib.pyplot as plt from datetime import datetime, timedelta #
Set page config st.set_page_config( page_title="Stock Price Forecast", page_icon="📈",
layout="wide" ) # App title st.title("📈 Stock Price Forecast App") st.markdown("Predict future
stock prices using Linear Regression") # Sidebar for inputs st.sidebar.header("Settings") ticker =
st.sidebar.text_input("Stock Ticker", "AAPL").upper() days = st.sidebar.slider("Days to Analyze",
30, 365, 180) # Calculate date range end_date = datetime.now() start_date = end_date -
timedelta(days=days) # Main app st.header(f"Analysis for {ticker}") if st.sidebar.button("Run
Analysis"): try: # Download data with st.spinner(f'Downloading {ticker} data...'): data =
yf.download(ticker, start=start_date, end=end_date, progress=False) if data.empty or len(data) <
10: st.error(f"No data found for {ticker}. Please check the ticker symbol.") st.stop() # Use High
prices high_prices = data['High'] # Display basic info - FIXED: Ensure values are properly
formatted col1, col2, col3 = st.columns(3) with col1: current_price = float(high_prices.iloc[-1])
st.metric("Current Price", f"${current_price:.2f}") with col2: st.metric("Data Points",
len(high_prices)) with col3: st.metric("Analysis Period", f"{days} days") # Prepare data for ML
dates_numeric = np.array([d.toordinal() for d in high_prices.index]).reshape(-1, 1) prices =
high_prices.values.astype(float) # Ensure float type # Create and train model model =
LinearRegression() model.fit(dates_numeric, prices) predictions = model.predict(dates_numeric)
# Calculate metrics rmse = np.sqrt(mean_squared_error(prices, predictions)) r2 =
model.score(dates_numeric, prices) # Display metrics - FIXED: Ensure proper formatting
st.subheader("Model Performance") col1, col2 = st.columns(2) with col1: st.metric("RMSE",
f"${rmse:.2f}") with col2: st.metric("R² Score", f'{r2:.4f}') # Create plot st.subheader("Price
Prediction Chart") fig, ax = plt.subplots(figsize=(10, 6)) ax.plot(high_prices.index, prices,
label='Actual Price', linewidth=2, color='blue') ax.plot(high_prices.index, predictions,
label='Predicted Price', linewidth=2, color='red', linestyle='--') ax.set_title(f'{ticker} - Price
Prediction', fontweight='bold') ax.set_xlabel('Date') ax.set_ylabel('Price ($)') ax.legend()
ax.grid(True, alpha=0.3) plt.xticks(rotation=45) plt.tight_layout() st.pyplot(fig) # Forecast next
day - FIXED: Ensure proper float conversion st.subheader("Next Day Forecast") last_date =
high_prices.index[-1] next_day_num = last_date.toordinal() + 1 next_day_pred =
float(model.predict([[next_day_num]])[0]) current_price = float(high_prices.iloc[-1]) change =
next_day_pred - current_price change_pct = (change / current_price) * 100 col1, col2 =
st.columns(2) with col1: st.metric( "Next Day Prediction", f"${next_day_pred:.2f}", delta=f"
{change:.2f} ({change_pct:.2f}%)", delta_color="normal" ) with col2: st.metric("Current Price",
f"${current_price:.2f}") # Show recent data with st.expander("View Recent Data"): # Format the
display dataframe properly display_df = pd.DataFrame({ 'Date':
high_prices.tail(10).index.strftime('%Y-%m-%d'), 'High Price': high_prices.tail(10).values })
display_df['High Price'] = display_df['High Price'].apply(lambda x: f"${x:.2f}")
st.dataframe(display_df) except Exception as e: st.error(f"An error occurred: {str(e)}")
st.info("Please try a different ticker symbol or date range") # Footer st.markdown("---")
st.markdown("Built with Streamlit • Data from Yahoo Finance")

```