Tesla Stock Analysis & Prediction App

This project is a **Streamlit-based web application** that provides **real-time Tesla stock analysis and predictions** using **machine learning** and RAG-LLM-**powered insights**.

Key Features

1. **Real-time Tesla Stock Data**
    - Fetches Tesla stock prices (Open, Close, High, Low, and Volume) using **Yahoo Finance (yfinance)**.
    - Displays stock metrics in a visually appealing way.
2. **Stock Price Prediction**
    - Uses a **pre-trained deep learning model (TensorFlow Keras)** to predict Tesla's stock prices for the next **7 days**.
    - Visualizes **historical and predicted** stock prices using **Plotly** interactive charts.
3. **Market News Analysis with LLM**
    - Scrapes Tesla-related news/articles from **Yahoo Finance or other sources**.
    - Uses a **Hugging Face LLM model** to generate **insights on market trends, sentiment, and predictions**.
4. **User-Friendly Interface**
    - Interactive **sidebar controls** for user input.
    - Expanding sections for **trend analysis, technical indicators, and market sentiment**.

1. Setup & Imports

The script imports **various libraries** for web development, data fetching, machine learning, and AI processing.

- **`streamlit`** → Builds the web app.
- **`yfinance`** → Fetches real-time stock data.
- **`tensorflow`** → Loads and runs the stock prediction model.
- **`BeautifulSoup`** & **`requests`** → Scrapes news articles for AI analysis.
- **`HuggingFaceEndpoint`** → Uses a Hugging Face **LLM** for AI-driven insights.
- **`plotly`** → Creates interactive charts for stock data visualization.

Configuration

```
warnings.filterwarnings('ignore')
st.set_page_config(page_title="Tesla Stock Analysis", layout='wide')
MODEL_PATH = "Tesla_model.h5"
TICKER_SYMBOL = "TSLA"
FORECAST_DAYS = 7
```

- **Suppresses warnings** for a cleaner output.

- **Sets the Streamlit page title & layout**.
- **Defines constants** like the stock ticker (**TSLA**) and the number of forecasted days (**7**).

2. AI & Model Initialization

Language Model (LLM) Setup

```
api = LargeLanguageModel()
api_key = api.get_Key()

llm = HuggingFaceEndpoint(
    name="WEB_PILOT",
    huggingfacehub_api_token=api_key,
    repo_id= 'mistralai/Mistral-7B-Instruct-v0.3',
    task="text-generation",
    max_new_tokens=1000,
    temperature=0.1
)
```

- Loads an **AI language model (Mistral-7B-Instruct)** from **Hugging Face**.
- This model will analyze scraped stock news and generate **market insights**.

Stock Prediction Model

```
def load_stock_model():
    """Load and compile the stock prediction model"""
    model = tf.keras.models.load_model(MODEL_PATH, compile=False)
    model.compile(loss=tf.keras.losses.MeanSquaredError())
    return model
```

- Loads a **pre-trained deep learning model** for stock price prediction.

3. Fetching & Processing Stock Data

Get Historical Stock Prices

```
def fetch_stock_data(days=20):
    """Fetch historical stock data"""
    ticker = yf.Ticker(TICKER_SYMBOL)
    hist = ticker.history(period=f"{days}d")["Close"]
    return hist
```

- Fetches Tesla's stock price **for the last 20 days** from Yahoo Finance.

Generate Future Predictions

```
python
CopyEdit
def generate_predictions(model, historical_data, forecast_days=7):
    """Generate stock price predictions"""
    current_prices = list(historical_data)
    for _ in range(forecast_days):
        input_data = np.array(current_prices[-7:]).reshape(1, 7, 1)
        prediction = model.predict(input_data, verbose=0)
```

```
        current_prices.append(prediction[0][0])
    return current_prices
```

- Uses the **deep learning model** to predict the next **7 days of stock prices**.

4. Visualization with Plotly

```
def create_stock_chart(historical_data, predictions, hist_dates):
    """Create interactive Plotly chart"""

    last_date = hist_dates[-1]
    pred_dates = pd.date_range(start=last_date + pd.Timedelta(days=1),
periods=FORECAST_DAYS)

    fig = go.Figure()
    fig.add_trace(go.Scatter(x=hist_dates, y=historical_data,
mode='lines+markers', name='Historical Data'))
    fig.add_trace(go.Scatter(x=pred_dates, y=predictions, mode='lines+markers',
name='Predicted Data'))

    fig.update_layout(title="Tesla Stock Price Forecast", xaxis_title="Date",
yaxis_title="Price (USD)", height=600)

    return fig
```

- Uses **Plotly** to create an **interactive stock price chart**.
- Displays **historical vs. predicted** stock prices.

5. Scraping Market News

```
def scrape_website(url):
    """Scrape website content"""
    try:
        headers = {"User-Agent": "Mozilla/5.0"}
        response = requests.get(url, headers=headers, timeout=10)
        soup = BeautifulSoup(response.text, 'html.parser')
        scraped_text = soup.get_text(separator=' ', strip=True)
        return scraped_text[4000:-2000]
    except Exception as e:
        st.error(f"Error scraping website: {str(e)}")
        return None
```

- Fetches **Tesla news articles** from finance websites.
- Extracts and **cleans text** for AI-based market analysis.

6. AI-Powered Market Insights

Generate Insights with LLM

```
def generate_insights(stock_data):
    """Generate market insights using LLM"""
    prompt_template = PromptTemplate.from_template("""
        You are an AI Stock Assistant specializing in Tesla market insights...
        {data}
    """)
```

```
    chain = LLMChain(prompt=prompt_template, llm=llm)
    response = chain.invoke({"data": stock_data})
    return parse_json_response(response['text'])
```

- Uses **AI (LLM)** to analyze market data and generate insights in **JSON format**.

Extract AI Output

```
python
CopyEdit
def parse_json_response(text):
    """Parse JSON response from AI output"""
    try:
        json_match = re.search(r"\{.*\}", text, re.DOTALL)
        return json.loads(json_match.group()) if json_match else None
    except json.JSONDecodeError:
        return None
```

- Ensures the AI **returns structured JSON data** for further processing.

7. Streamlit Dashboard

Main Application

```
def main():
    st.title("📈 Tesla Stock Analysis & Prediction")
    st.write(f"#### {datetime.today().strftime('%A, %B %d, %Y')}")

    # Fetch today's stock data
    ticker = yf.Ticker(TICKER_SYMBOL)
    today_data = ticker.history(period="1d").iloc[0]

    # Sidebar input
    analysis_url = st.text_input("News Source URL:",
"https://finance.yahoo.com/quote/TSLA/analysis/")
    analyze_btn = st.button("Analyze Stock")

    if analyze_btn:
        with st.spinner("Analyzing market data..."):
            model = load_stock_model()
            hist_data = fetch_stock_data()
            predictions = generate_predictions(model, hist_data.values)
            st.plotly_chart(create_stock_chart(hist_data.values, predictions,
hist_data.index))

            # Scrape and analyze market news
            scraped_data = scrape_website(analysis_url)
            if scraped_data:
                insights = generate_insights(scraped_data)
                display_insights(insights)
```

- **Displays stock metrics, predictions, and AI insights in Streamlit.**