- Importer les bibliothèques/modules nécessaires : requests, BeautifulSoup, RandomForestClassifier, SentimentIntensityAnalyzer, yfinance.
- Définir une fonction scrape_financial_news pour extraire les données textuelles à partir d'une URL donnée.
- Définir l'URL du site web d'actualités financières et le nombre de mots à extraire.
- Extraire les données textuelles des actualités financières en utilisant la fonction définie scrape_financial_news.
- Effectuer une analyse de sentiment sur le texte extrait des actualités financières en utilisant VADER de NLTK.
- Récupérer les données boursières en utilisant yfinance pour un symbole d'action spécifié (par exemple, "AMZN").
- Préparer les données pour l'entraînement en créant des étiquettes basées sur le mouvement boursier et les scores de sentiment.
- Entraîner un modèle RandomForestClassifier en utilisant les données préparées.
- Prédire le mouvement boursier en fonction du dernier score de sentiment.
- Afficher le texte extrait, les scores d'analyse de sentiment et le mouvement boursier prédit.

```
Commencez à coder ou à générer avec l'IA.
```

```
#This library is used to send HTTP requests to a specified URL.
import requests
```

```
#This line imports the BeautifulSoup class from the bs4 library.
from bs4 import BeautifulSoup
```

```
#This line imports the RandomForestClassifier class from the sklearn.ensemble module.
from sklearn.ensemble import RandomForestClassifier
```

```
#This line imports the SentimentIntensityAnalyzer class from the nltk.sentiment.vader
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

```
# This line imports the yfinance library and aliases it as yf.
import yfinance as yf
```

```
import nltk
nltk.download('vader_lexicon')
```

```
[nltk_data] Downloading package vader_lexicon to /root/nltk_data...
True
```

```
# This function extracts financial news text data from a given URL up to a specified number of words
def scrape_financial_news(url, num_words):
    response = requests.get(url)
    soup = BeautifulSoup(response.content, 'html.parser')
    words = []

    if soup:
        paragraphs = soup.find_all('p')
        for paragraph in paragraphs:
            text = paragraph.get_text(strip=True)
            if text and len(words) < num_words:
                words.extend(text.split())
            else:
                break
    return words
```

```
# URL of the financial news website (e.g., BBC)
url = "https://www.bbc.co.uk/"
num_words = 100  # Number of words to extract
```

```
# Scrape financial news text data
financial_news = scrape_financial_news(url, num_words)
```

```
# Perform sentiment analysis using NLTK's VADER
sia = SentimentIntensityAnalyzer()
sentiment_scores = [sia.polarity_scores(word)['compound'] for word in financial_news]
```

```
# Get stock data for demonstration (e.g., Apple)
stock_ticker = "AMZN"
stock_data = yf.download(stock_ticker, period="1d")
```

```
[**********100%%**********]  1 of 1 completed
```

```
# Feature engineering: Creating labels based on stock movement
stock_data['Movement'] = (stock_data['Close'] - stock_data['Open']).apply(lambda x: 1 if x > 0 else 0)
```

```
# Prepare data for training
X = sentiment_scores[:len(stock_data)]
y = stock_data['Movement']
```

```
# Train RandomForestClassifier
model = RandomForestClassifier()
model.fit([X], y)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
# Predict stock movement
prediction = model.predict([[sentiment_scores[-1]]])  # Providing a 2D array for prediction
```

```
#Display extracted text
print("Extracted Text:")
for word in financial_news[:100]:  # Displaying the first 10 words for demonstration
    print(word)
```

```
Extracted Text:
The
price
of
peace
between
Israelis
and
Palestinians
'Absolutely
no':
Israel's
ambassador
to
UK
rejects
two-state
solution
One
unaccounted
for
and
injuries
after
explosion
Lobbying
scandal
MP
faces
Commons
suspension
Can
you
solve
GCHQ's
codebreaker
challenge
for
kids?
Search
continues
for
missing
Gaynor
Lord
Circus
performer
falls
from
height
during
show
Buyers
put
off
new
homes
over
```

```
#Display sentiment analysis Scores
print("\nSentiment Analysis Scores:")
print(sentiment_scores[:100])  # Displaying sentiment scores for the first 10 words
```

```
Sentiment Analysis Scores:
[0.0, 0.0, 0.0, 0.5423, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, -0.4939, 0.0, 0.3182, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0,
```

```
#Predicted stock name
print("\nPredicted Stock:")
if prediction == 1:
    print("Prediction: The stock may rise.")
    print("Predicted Stock Name:", stock_ticker)
else:
    print("Prediction: The stock may fall.")
    print("Predicted Stock Name:", stock_ticker)
```

```
Predicted Stock:
Prediction: The stock may rise.
Predicted Stock Name: AMZN
```