Step 1: Data Scraping from Reddit

Objective: Collect stock-related discussions from Reddit.

Steps:

Install Required Libraries:

- pip install praw pandas

Set Up Reddit API:

- Create a Reddit Developer account at Reddit API.
- Create an application and get the following credentials:
  - client_id
  - client_secret
  - user_agent

Scrape Data:

Use the PRAW (Python Reddit API Wrapper) library to fetch posts.

Code:

```
python

Copy code

import praw

import pandas as pd


# Authenticate with Reddit API

reddit = praw.Reddit(

   client_id='YOUR_CLIENT_ID',

   client_secret='YOUR_CLIENT_SECRET',

   user_agent='StockScraper'

)


# List of subreddits to scrape

subreddits = ['stocks', 'wallstreetbets']

data = []
```

```
# Scrape Reddit posts
for subreddit_name in subreddits:
    subreddit = reddit.subreddit(subreddit_name)
    for post in subreddit.hot(limit=100):  # Adjust limit as needed
        post_data = {
            'title': post.title,
            'body': post.selftext,
            'score': post.score,
            'comments': [comment.body for comment in post.comments if hasattr(comment, 'body')],
            'created': post.created_utc
        }
        data.append(post_data)

# Save raw data to CSV
df = pd.DataFrame(data)
df.to_csv('reddit_stock_data.csv', index=False)
print("Data saved to reddit_stock_data.csv")
```

Step 2: Data Cleaning

Objective: Prepare the data for analysis by removing unnecessary elements (e.g., links, special characters).

Steps:

Install Required Libraries:

- pip install re

Clean Text:

Remove links, special characters, and convert text to lowercase.

Code:

```
import re

# Function to clean text
def clean_text(text):
```

```
    return re.sub(r'http\S+|www\S+|[^a-zA-Z\s]', '', text).lower()

# Apply cleaning
df['cleaned_body'] = df['body'].apply(clean_text)
df.to_csv('cleaned_reddit_stock_data.csv', index=False)
print("Cleaned data saved to cleaned_reddit_stock_data.csv")
```

Step 3: Sentiment Analysis

Objective: Analyze the sentiment (positive, negative, or neutral) of each post.

Steps:

Install TextBlob:

- pip install textblob

Calculate Sentiment:

Use TextBlob to compute sentiment polarity (ranges from -1 to 1).

Code:

```
from textblob import TextBlob

# Function for sentiment analysis
def get_sentiment(text):
    analysis = TextBlob(text)
    return analysis.sentiment.polarity

# Apply sentiment analysis
df['sentiment'] = df['cleaned_body'].apply(get_sentiment)
df.to_csv('sentiment_reddit_stock_data.csv', index=False)
print("Sentiment data saved to sentiment_reddit_stock_data.csv")
```

Step 4: Stock Mentions

Objective: Count the frequency of specific stock mentions in the posts.

Steps:

List Stock Symbols:

Create a list of stock symbols to monitor, e.g., ['AAPL', 'TSLA', 'AMZN'].

Count Mentions:

Count how many times each symbol appears in the cleaned text.

```
stock_symbols = ['AAPL', 'TSLA', 'AMZN']  # Add more symbols as needed

# Count stock mentions
for stock in stock_symbols:
    df[f'mention_{stock}'] = df['cleaned_body'].apply(
        lambda x: x.lower().count(stock.lower()) if isinstance(x, str) else 0
    )

df.to_csv('stock_mentions_reddit.csv', index=False)
print("Stock mention data saved to stock_mentions_reddit.csv")
```

Step 5: Prediction Model

Objective: Predict stock price movements using the sentiment and mention data.

Steps:

Prepare the Dataset:

Features: Sentiment and stock mentions.

Target: Stock movement (e.g., 1 for up, 0 for down).

Train-Test Split:

Split the data into training and testing sets.

Train Model:

Use a Random Forest Classifier for prediction.

Evaluate Model:

Use metrics like accuracy, precision, recall, and F1-score.

Code:

```
python

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification_report, accuracy_score, confusion_matrix


# Features and target

X = df[['sentiment'] + [f'mention_{stock}' for stock in stock_symbols]]

y = [1, 0, 1, 0, 1]  # Example target values; replace with real data.


# Train-test split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)


# Train model

model = RandomForestClassifier(n_estimators=100, random_state=42)

model.fit(X_train, y_train)


# Predict and evaluate

y_pred = model.predict(X_test)

print(f"Accuracy: {accuracy_score(y_test, y_pred)}")

print("Classification Report:\n", classification_report(y_test, y_pred))

print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```