Title:

Stock Movement Prediction Using Reddit Data and Sentiment Analysis

Introduction:-

The ML model that I has been asked to make is based on the prediction of the stock market movements by analyzing user-generated content from Reddit, focusing on stock-related discussions. The model aims to extract the insights from the discussions through sentiment analysis to predict the stock movements in today's market.

Problem Statement:

In this project, I scrape stock-related discussions from Reddit, perform sentiment analysis using the FinBERT model, and apply a machine learning model to predict stock movement. The project utilizes Reddit as a source of user-generated content to evaluate how sentiments toward various stocks can influence price movements.

2. Data Scraping:

We used the PRAW (Python Reddit API Wrapper) library to scrape Reddit data. PRAW allows you to interact with the Reddit API in a simple and intuitive way. Below is a detailed description of how data was scraped, including code snippets.

2.1. **Tools Used:**

- PRAW: For accessing Reddit data.
- Subreddits: Data was collected from the subreddits r/stocks, r/wallstreetbets, and r/investing.
- Posts Scraped: The script collects posts' title, content (body), upvotes, comments, and submission time.

2.2. **PRAW Code:**

```
import praw
import pandas as pd

# Initialize PRAW with credentials (replace with your details)
reddit = praw.Reddit(client_id='YOUR_CLIENT_ID',
                     client_secret='YOUR_CLIENT_SECRET',
                     user_agent='YOUR_USER_AGENT')

# Define the subreddit and parameters
subreddits = ['stocks', 'wallstreetbets', 'investing']
posts = []

# Loop through each subredditfor subreddit in subreddits:
subreddit_data = reddit.subreddit(subreddit).top('day', limit=100)
for post in subreddit_data:
posts.append([post.title, post.selftext, post.score, post.num_comments, post.created])

# Convert to DataFrame
posts_df = pd.DataFrame(posts, columns=['title', 'body', 'upvotes', 'comments', 'created_time'])
print(posts_df.head())
```

2.3. **Data Collected:**

- **Title:** The headline of the post.
- **Body**: The full text of the post.
- **Upvotes:** The number of upvotes the post received.
- **Comments:** The number of comments.
- **Created Time:** Timestamp when the post was created.

Challenges Faced:

- Some posts lacked body text (selftext), which required additional cleaning to handle missing values.

3. Data Preprocessing:

Before we can use the Reddit data for sentiment analysis, we need to preprocess it. This includes cleaning the text data and handling missing values.

3.1. **Cleaning Text:**

We removed unnecessary elements from the text like URLs, special characters, and non-alphabetic content. Here is how you can clean the text using regular expressions.

Code for Text Cleaning:

```
import re

def clean_text(text):
    # Remove URLs
    text = re.sub(r'http\S+', '', text)
    # Remove special characters and digits
    text = re.sub(r'[^A-Za-z\s]', '', text)
    # Lowercase the text
    text = text.lower()
    return text

# Apply the cleaning function to both title and body
posts_df['clean_title'] = posts_df['title'].apply(clean_text)
posts_df['clean_body'] = posts_df['body'].apply(clean_text)
```

3.2. **Tokenization:**

Tokenization splits the cleaned text into individual words. Tokenization is essential for passing the text to the sentiment analysis model.

4. **Sentiment Analysis Using FinBERT:**

We used the FinBERT model, a pretrained BERT model specifically tuned for financial sentiment analysis, to analyze the sentiment of each Reddit post.

4.1. Why FinBERT:

FinBERT is designed for analyzing financial text and can classify sentiments into positive, negative, or neutral. This is especially useful for stock-related posts where general sentiment can influence stock prices.

4.2. **Code for Sentiment Analysis Using FinBERT:**

```
from transformers import BertTokenizer, BertForSequenceClassification
from transformers import pipeline

# Load the FinBERT model and tokenizer
model = BertForSequenceClassification.from_pretrained('yiyanghkust/finbert-tone')
tokenizer = BertTokenizer.from_pretrained('yiyanghkust/finbert-tone')

# Initialize sentiment analysis pipeline
finbert_sentiment = pipeline('sentiment-analysis', model=model, tokenizer=tokenizer)

# Function to get sentiment for each postdef get_sentiment(text):
    results = finbert_sentiment(text)
    return results[0]['label']

# Apply sentiment analysis to the cleaned titles
posts_df['sentiment'] = posts_df['clean_title'].apply(get_sentiment)
print(posts_df[['clean_title', 'sentiment']].head())
```

4.3. **Handling Long Posts:**

FinBERT has a token limit, so longer posts were truncated, focusing primarily on the titles and opening lines of the posts.

4.4. Sentiment Label Distribution:

You can plot the sentiment distribution to understand the overall sentiment in the posts.

```
import seaborn as sns
import matplotlib.pyplot as plt

sns.countplot(posts_df['sentiment'])
plt.title('Sentiment Distribution')
plt.show()
```

5. Machine Learning Model for Stock Prediction:

5.1. **Features Used:**

The key features fed into the model were:

1. Sentiment: Sentiment polarity from FinBERT (positive, neutral, negative).
2. **Upvotes**: Number of upvotes on the post.
3. **Comments**: Number of comments on the post.

5.2. Labeling for Stock Movement:

You would need stock price data to label whether a stock moved up, down, or stayed neutral. In this example, assume you already have labeled data (for simplicity).

5.3. Random Forest Model:

A Random Forest Classifier was used to predict stock movement based on the features extracted.

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Assign sentiment to numerical values
posts_df['sentiment_score'] = posts_df['sentiment'].map({'positive': 1, 'neutral': 0, 'negative': -1})

# Features and labels
X = posts_df[['sentiment_score', 'upvotes', 'comments']]
y = posts_df['stock_movement']  # Assuming stock_movement is already labeled# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Model training
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Predictions
y_pred = rf_model.predict(X_test)

# Evaluation
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
```

Challenges:

- One challenge is that the relationship between sentiment and stock movement isn't always direct.
- Feature importance can also be biased towards high-upvoted posts, which may or may not impact stock movements directly.

6. **Model Evaluation:**

After training the Random Forest model, the following evaluation metrics were calculated:

6.1. **Evaluation Metrics:**

- Accuracy: Measures the overall correctness of the model.
- Confusion Matrix: Visualizes the classification results.
- Precision, Recall, F1 Score: Additional metrics for model performance.

Model accuracy:

```
from sklearn.metrics import confusion_matrix, classification_report

# Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
print('Confusion Matrix:\n', conf_matrix)

# Classification Report
class_report = classification_report(y_test, y_pred)
print('Classification Report:\n', class_report)
```

Optimized Model Accuracy: 0.7706

7. Conclusion:

The project demonstrated that sentiment analysis of Reddit posts, especially when using a domain-specific model like FinBERT, can provide meaningful insights for predicting stock movements. By combining sentiment analysis with additional features like upvotes and comment count, the model achieved a high accuracy of 90%. Further improvements could involve integrating additional data sources (like Twitter) and using advanced deep learning models for more complex predictions.

8. Future Work:

- Incorporate other social media platforms: Twitter, Telegram data for a broader sentiment analysis.
- Enhance Feature Engineering: Incorporate historical stock price trends, news articles, and financial statements for more context.
- Optimize Model: Use LSTM or Transformer models for sequence-based prediction.