7. HVDSentimentAnalysis

September 6, 2023

[ ]:
```
import pandas as pd
import os
```

[ ]:
```
# Load CSV Data

# Stocks :- AAPL, MSFT, AMZN, NVDA, TSLA, GOOGL,
# Sector Indices :- SSINFT (^SP500-45)

ticker = "SSINFT"
method = "HVD"

if method != "DistilBERT":
  indirectory = "MergedDataset"
else:
  indirectory = "MergedContextDataset"

# Load the merged dataset file
df = pd.read_csv(f"{indirectory}/{ticker}_agg_news_stock_trend_output.csv")
```

[ ]:
```
# 2. Load Harvard IV-4 sentiment dictionary from previous step stored in two␣
 ↪separate text files as positive and negative words.
with open(f"SentimentAnalysis/{method}/{method}_positive.txt", 'r') as f:
    positive_words = set(f.read().splitlines())

with open(f"SentimentAnalysis/{method}/{method}_negative.txt", 'r') as f:
    negative_words = set(f.read().splitlines())


# 3. Sentiment Analysis
def analyze_sentiment(text):
    text_words = str(text).split()
    positive_count = sum(1 for word in text_words if word in positive_words)
    negative_count = sum(1 for word in text_words if word in negative_words)
    # return sentiment score
    return positive_count - negative_count

df['polarity'] = df['Headline'].apply(analyze_sentiment)
```

[ ]:
```
df
```

[ ]:
```
# 5. Output Sentiment Results with stock price trend
df.to_csv(f"SentimentAnalysis/{method}/{ticker}sentiment_agg_stock_trend_output.
 ↪csv", index=False)
```

[ ]:
```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Explanation about polarity and subjectivity
print("\nPolarity is a float which lies in the range from -1 to 1. -1 means a␣
 ↪negative statement and 1 means a positive statement.")
print("Subjectivity is a float which lies in the range of 0 to 1. 0 being␣
 ↪objective and 1 being subjective.\n")

# Function to get sentiment intensity analyzer scores
def get_sia(text):
    sia = SentimentIntensityAnalyzer()
    sentiment = sia.polarity_scores(text)
    return sentiment

compound = []
neg = []
neu = []
pos = []

for i in range(0, len(df['Headline'])):
    sia = get_sia(df['Headline'][i])
    compound.append(sia['compound'])
    neg.append(sia['neg'])
    neu.append(sia['neu'])
    pos.append(sia['pos'])

# Storing sentiment scores in the merged dataset
df['compound'] = compound
df['negative'] = neg
df['neutral'] = neu
df['positive'] = pos

# Columns to keep
keep_columns = ['Open', 'High', 'Low', 'Volume', 'polarity', 'price_trend']
# keep_columns = ['Open', 'High', 'Low', 'Close', 'Volume', 'polarity',␣
 ↪'next_day_price_trend']
df = df[keep_columns]
```

```
print(df)
```

[ ]:
```
# Creating the feature dataset
x = np.array(merged_df.drop(columns=['price_trend']))
# x = np.array(merged_df.drop(columns=['next_day_price_trend']))
# Creating the target dataset
y = np.array(merged_df['price_trend'])
# y = np.array(merged_df['next_day_price_trend'])

# Splitting the data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2,␣
 ↪random_state=0)

# Creating and training the model
model = LinearDiscriminantAnalysis().fit(x_train, y_train)

# Model's predictions
predictions = model.predict(x_test)
print(predictions)

print(y_test)

# Model metrics
print(classification_report(y_test, predictions))
```

[ ]:
```

```