Project B
Task 7 – Extension

The task this week is to extend the v0.6 file to incorporate a sentiment analysis of tweets pertaining to the selected company.

Potential Approaches

To improve the effectiveness of the stock analysis tool, I explored three different sentiment analysis models for evaluating tweets about Amazon.

SentiBERT is variant of the BERT (Bidirectional Encoder Representations from Transformers) model that has been fine-tuned specifically for general sentiment analysis tasks. It is trained on diverse datasets encompassing various domains, enabling it to accurately capture a wide range of emotional tones from positive and negative to neutral. This versatility makes SentiBERT effective in analysing different types of tweets, providing reliable sentiment insights across multiple contexts.

RoBERTa (Robustly Optimized BERT Approach) is an advanced version of BERT that has been trained on a larger dataset with longer training sequences to enhance its performance. RoBERTa offers superior accuracy in sentiment detection by capturing deep contextual relationships within the text. Its flexible architecture allows for fine-tuning to meet specific requirements, making it a robust choice for comparison sentiment analysis tasks.

FinBERT is a specialised BERT model designed exclusively for financial sentiment analysis. It has been fine-tuned using a substantial number of financial texts, enabling it to accurately interpret sentiments unique to the financial sector. FinBERT excels at identifying subtle emotional cues and nuances in financial discussions, providing highly precise insights into market-related sentiments.

The best choice for v0.7 is FinBERT.

Twitter Dataset

Now that a model has been chosen, I needed to find a dataset for it to analyse. The dataset I chose is this one:
https://www.kaggle.com/datasets/omermetinn/tweets-about-the-top-companies-from-2015-to-2020

It provides tweets for Google, Apple, Amazon, Tesla, and Microsoft between 2015 and 2020.

Approach Taken and Implementation

The first step to implementing FinBERT is to load and pre-process the twitter data.

```
# Paths to data files
DATA_DIR = os.path.join(SCRIPT_DIR, 'SentimentData')
os.makedirs(DATA_DIR, exist_ok=True)

COMPANY_CSV = os.path.join(DATA_DIR, 'Company.csv')
COMPANY_TWEET_CSV = os.path.join(DATA_DIR, 'Company_Tweet.csv')
TWEET_CSV = os.path.join(DATA_DIR, 'Tweet.csv')

# Load raw sentiment data
company_df = pd.read_csv(COMPANY_CSV)
company_tweet_df = pd.read_csv(COMPANY_TWEET_CSV)
tweet_df = pd.read_csv(TWEET_CSV)
```

This code retrieves the 3 files from the Kaggle dataset.

- Company.csv contains a list of companies represented in the dataset and their tickers.
- Company_Tweet.csv contains a list of tweet_id's and the ticker they are associated with.
- Tweet.csv contains a list of tweets sorted by their tweet_id.

Now that the dataset was loaded, it needs to be filtered and merged to only contain tweets related to Amazon.

```
# Filter for the desired ticker symbol
desired_ticker = COMPANY_TICKER
company_tweet_filtered = company_tweet_df[company_tweet_df['ticker_symbol'] ==
desired_ticker]

# Merge filtered tweets with tweet details
tweets_for_stock = pd.merge(company_tweet_filtered, tweet_df, on='tweet_id')
```

The dataframe now ONLY contains tweets pertaining to Amazon, however the content of those tweets may not be valid. We have to clean it.

```
def clean_tweet(text):
    text = re.sub(r'http\S+', '', str(text))
    text = re.sub(r'@\w+|#\w+', '', text)
    text = re.sub(r'[^a-zA-Z\s]', '', text)
    text = re.sub(r'\s+', ' ', text).strip()
    return text

tweets_for_stock['cleaned_body'] = tweets_for_stock['body'].apply(clean_tweet)
```

This method removes URLs, mentions, hashtags, non-alphabetic characters, and extra whitespace from the tweets.

We now adjust the START_DATE and END_DATE parameters to be the earliest and latest date of the processed tweet dataset.

```
# Determine START_DATE and END_DATE based on tweet data
START_DATE = tweets_for_stock['date'].min().strftime('%Y-%m-%d')
END_DATE = tweets_for_stock['date'].max().strftime('%Y-%m-%d')
```

Before analysing the data with FinBERT, we first check if it already exists locally.

```
# Update the sentiment data filename to include the ticker and date range
PROCESSED_SENTIMENT_FILENAME =
f"processed_tweets_{COMPANY_TICKER}_{START_DATE}_to_{END_DATE}.csv"
PROCESSED_SENTIMENT_PATH = os.path.join(DATA_DIR, PROCESSED_SENTIMENT_FILENAME)

# Check if processed sentiment data exists
if os.path.exists(PROCESSED_SENTIMENT_PATH):
    print(f"Processed sentiment data found: {PROCESSED_SENTIMENT_PATH}.
Loading...")
    tweets_for_stock = pd.read_csv(PROCESSED_SENTIMENT_PATH,
parse_dates=['post_date'])
    tweets_for_stock['date'] = tweets_for_stock['post_date'].dt.date
```

This is to reduce inference time as sentiment analysis can be a very time-consuming task.

If we do NOT have an already analysed set of tweets, we can proceed to the FinBERT analysis.

```
    tokenizer = AutoTokenizer.from_pretrained("ProsusAI/finbert")
    model = AutoModelForSequenceClassification.from_pretrained("ProsusAI/finbert")

    def get_sentiment(text):
        inputs = tokenizer(text, return_tensors="pt", truncation=True,
max_length=512)
        outputs = model(**inputs)
        scores = outputs[0][0].detach().numpy()
        scores = softmax(scores)
        sentiment = np.argmax(scores)
        sentiment_label = ['positive', 'negative', 'neutral'][sentiment]
        return sentiment_label
```

This code loads the FinBERT model and classifies each tweet as positive, negative, or neutral.

```
    sentiment_mapping = {'positive': 1, 'neutral': 0, 'negative': -1}
    tweets_for_stock['sentiment_score'] =
tweets_for_stock['sentiment'].map(sentiment_mapping)
```

Next, we map sentiment labels to numerical scores so we can aggregate a daily score later.

```
daily_sentiment =
tweets_for_stock.groupby('date')['sentiment_score'].mean().reset_index()
```

Here is where the sentiment score is aggregated by date. We now have the fully processed sentiment data, but we need to allow the LSTM, RNN, and Random Forest models to train on this data. To do this, the sentiment data must be merged with the stock price data.

```
# Merge stock data with sentiment
data.reset_index(inplace=True)
data['date'] = data['Date'].dt.date

data = pd.merge(data, daily_sentiment, on='date', how='left')
data['sentiment_score'].fillna(0, inplace=True)

FEATURE_COLUMN = ["Open", "Adj Close", "Volume", "High", "Close", "Low",
"sentiment_score"]
```

The sentiment data is merged with the stock data and a new column is added to the FEATURE_COLUMN. The rest of the model training and analysis is the same as v0.6. On the next page is some of the training and testing data.

Training and Testing Data

For all models the N_STEPS = 14, and LOOKUP_STEP = 7.

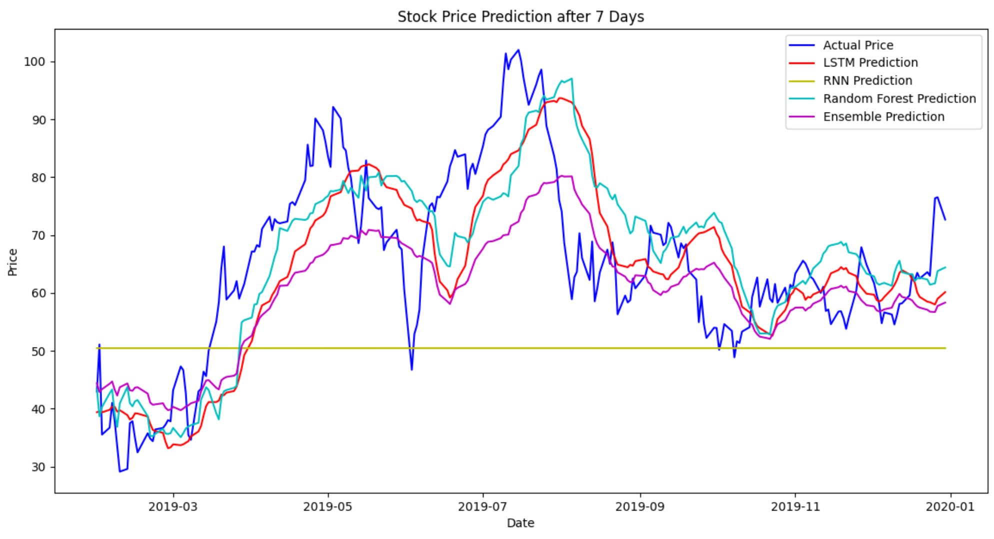
LSTM and RNN:
- Units = 256
- Dropout = 0.4
- Batch Size = 64
- N_Layers = 8
- Epochs = 50

LSTM:
- Bidirectional = True

RNN:
- Bidirectional = False

Random Forest:
- N_ESTIMATORS = 100
- RANDOM_STATE = 42

LSTM MAE: 9.174431620510493
RNN MAE: 17.788378122491977
Random Forest MAE: 9.09163664168928
Ensemble MAE: 9.552910514189273
Accuracy Score: 0.5043103448275862
Total Profit: 101.67136192088392
Profit Per Trade: 0.43823862896932725
Predicted future price after 7 days using Ensemble: 58.34$
Predicted future price after 7 days using Random Forest: 64.40$
Predicted future price after 7 days using LSTM: 60.15$
Predicted future price after 7 days using RNN: 50.47$



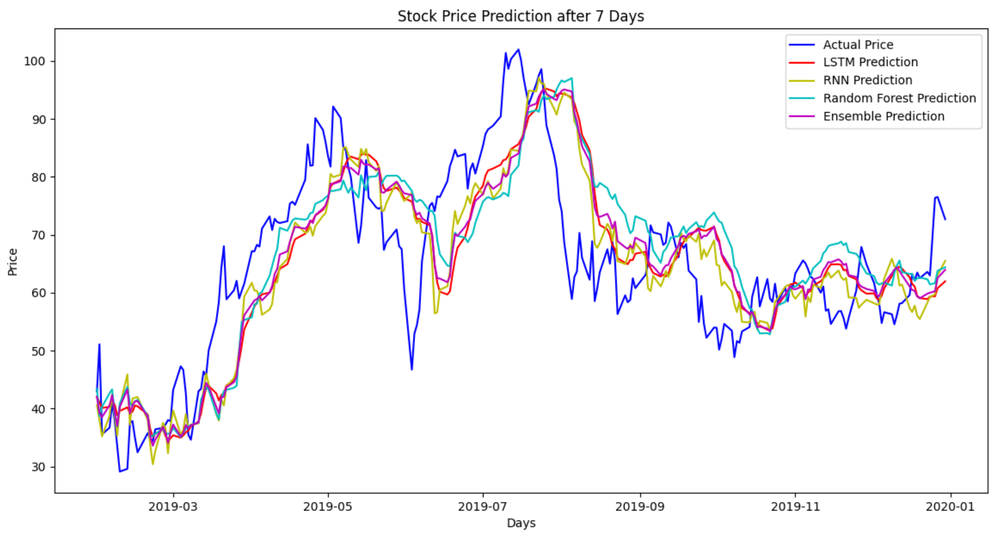
LSTM and RNN:
- Units = 256
- Dropout = 0.4
- Batch Size = 32
- N_Layers = 4
- Epochs = 75

LSTM:
- Bidirectional = True

RNN:
- Bidirectional = False

Random Forest:
- N_ESTIMATORS = 100
- RANDOM_STATE = 42

LSTM MAE: 8.69026627375414
RNN MAE: 9.125317869521586
Random Forest MAE: 9.09163664168928
Ensemble MAE: 8.84266653061835
Accuracy Score: 0.5
Total Profit: 64.42800054401275
Profit Per Trade: 0.27770689889660666



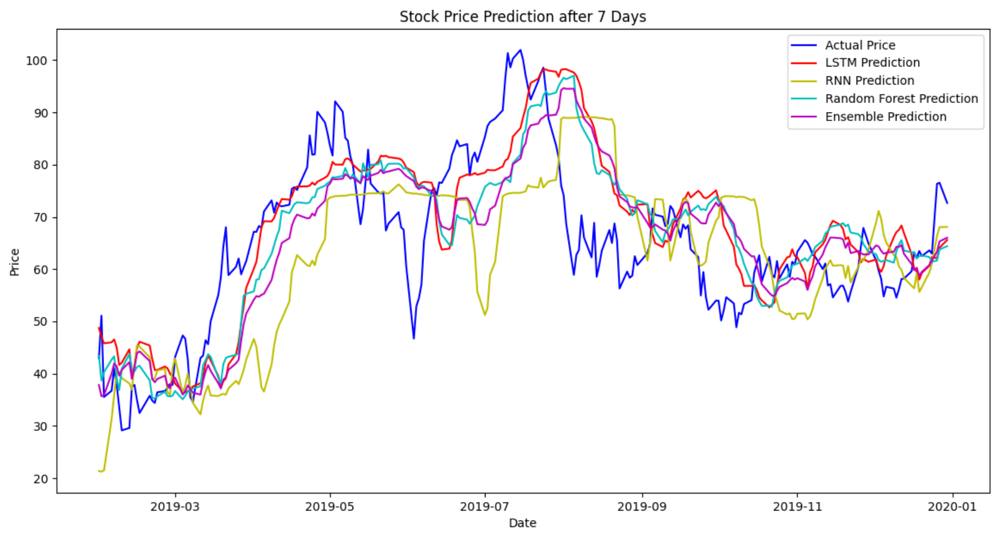
LSTM and RNN:
- Units = 256
- Dropout = 0.4
- Batch Size = 64
- N_Layers = 8
- Epochs = 400

LSTM:
- Bidirectional = True

RNN:
- Bidirectional = False

Random Forest:
- N_ESTIMATORS = 100
- RANDOM_STATE = 42

LSTM MAE: 9.002830216824027
RNN MAE: 12.38007123143237
Random Forest MAE: 9.09163664168928
Ensemble MAE: 9.766259535510851
Accuracy Score: 0.5431034482758621
Total Profit: 91.41096197135207
Profit Per Trade: 0.39401276711789684
Predicted future price after 7 days using Ensemble: 66.03$
Predicted future price after 7 days using Random Forest: 64.40$
Predicted future price after 7 days using LSTM: 65.59$
Predicted future price after 7 days using RNN: 68.09$


Stock Price Prediction after 7 Days
Actual Price
LSTM Prediction
RNN Prediction
Random Forest Prediction
Ensemble Prediction
Price
Date