COS30018 Assignment B – Task 7 Report

Aidan Grimmett: 103606838 – Friday 12:30 class

For this simple extension task, I tried to integrate the daily twitter mentions of the company as another feature in the dataset. This would allow most of the code to remain the same, but with a new element in the data for the model to train and predict on. I set up a twitter developer account through the X Developer Portal, and made a new app, generating all the API access tokens. Once this was done I was able to make a connection with the Twitter API through the tweepy package.

```python
import tweepy
import datetime

# super secret information (please don't steal)
API_KEY = 'DnzKfdItAT7OENyBmQCTU3s9t'
API_SECRET_KEY = 'QWeDZvBzXU1bPyZmeOvV2VB0PYqN68c9CQ2RZMlytjXgYIq4uw'
ACCESS_TOKEN = '1571766349915095040-kKcNiZuq6DsvAuusGovTawFs2YAODo'
ACCESS_TOKEN_SECRET = 'Rbx8oSJLX1K2LPMrMnuIWTDbUzuOh7BSYVQcWuJ7j5tUX'
BEARER_TOKEN = 'AAAAAAAAAAAAAAAAAAAAAKKtwgEAAAAABIx5ASCHFouTkyf21ITPNBE9%2BtU%3Ds9UlN30WMD59eJsjARuuAymqR4Ydjsf18uuuaBmaI3r6CjluKj'

# Set up Tweepy authentication
auth = tweepy.OAuthHandler(API_KEY, API_SECRET_KEY)
auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)

# Create an API object to interact with Twitter
api = tweepy.API(auth)

# Verify credentials
try:
    api.verify_credentials()
    print("Authentication successful")
except:
    print("Authentication failed")
```

This code defines all of the necessary access tokens, then creates a new OAuthHandler and sets the access tokens to my personal tokens. Then an API object can be created, and to test whether I can successfully connect or not, I use api.verify_credentials(), which works successfully and “Authentication successful” is printed to the console.

My idea for implementing twitter mentions into the stock predictor was to get the amount of mentions for each day in the time frame we are training/predicting in.

```python
def get_twitter_mentions(queryIn, start_date, end_date):
    # authenticate with credentials
    client = tweepy.Client(
        bearer_token=BEARER_TOKEN,
        consumer_key=API_KEY,
        consumer_secret=API_SECRET_KEY,
        access_token=ACCESS_TOKEN,
        access_token_secret=ACCESS_TOKEN_SECRET
    )

    # Initialize list of daily mention counts
    daily_mentions = []

    # convert start and end dates to datetime objects
    current_date = datetime.datetime.strptime(start_date, "%d-%m-%Y")
    end_date = datetime.datetime.strptime(end_date, "%d-%m-%Y")

    # loop through each day in the range
    while current_date <= end_date:
        # search tweets mentioning the company for this day using Twitter API
        tweets = client.search_all_tweets(
            query=queryIn,
            start_time=current_date.isoformat() + "Z",
            end_time=next_day.isoformat() + "Z",
            max_results=100
        )

        # count the number of tweets for the day
        tweet_count = len(tweets.data) if tweets.data else 0
        # append to data object
        daily_mentions.append(tweet_count)

        # move to the next day
        next_day = current_date + datetime.timedelta(days=1)
        current_date = next_day

    return daily_mentions

get_twitter_mentions("Commonwealth Bank", "01-01-2020", "01-01-2022")
```

This code will set up and authenticate a client which we can interact with the API through. Then a list is initialised which will hold the amount of mentions the company gets for each day throughout the time period. The start/current date and end date are converted into date time formats for easier date incrementing. Then finally we can loop through the time period, and access the amount of mentions through a search_all_tweets() function. This will provide us with tweet data, which we only need to take the length of, and this is appended to the daily_mentions list.

Changes also had to be made to the load_process_data() function. Namely, a new parameter hd to be added:

```
def load_and_process_data(
    company,
    multivariate,
    twitterMentions,
    scale_features,
    save_data,
    data_dir
```

twitterMentions acts as a Boolean flag for which the function knows to call the get twitter mentions or not.

If twitterMentions is true, then they will be added to the data as a new column. If there is a mismatch between the length of the twitter mentions list and the length of the stock data, then an error message is shown and the twitter data will not be added.

```
if os.path.exists(file_path):
    print(f"Loading data from {file_path}")
    # Read data from local file
    data = pd.read_csv(file_path, index_col=0, parse_dates=True)
else:
    # Download and save data
    print(f"Downloading data for {company} from {start_date} to {end_date}")
    data = yf.download(company, start_date, end_date)
    if save_data:
        os.makedirs(data_dir, exist_ok=True)
        data.to_csv(file_path)

if twitterMentions:
    print("Fetching Twitter mentions...")
    mentions = get_twitter_mentions(f"${company}", start_date, end_date)
    if len(mentions) != len(data):
        print("Mismatch between Twitter mentions and stock data length.")
    else:
        data["TwitterMentions"] = mentions
```

And lastly, when the scalers are being set up, we also need a new scaler for the twitter data:

```
# Step 4: Scale the Features
# Create a dictionary of the different scalers we will store
scalers = {}
if multivariate:
    features = ['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume']
else:
    features = ['Close']

if twitterMentions:
    features.append('TwitterMentions')
```

Unfortunately, this project was doomed to fail, because it seems like the free tier of the twitter API does not allow for historical data access like this. Free developer accounts are only permitted to access tweets from the last seven days, and “full-archive search endpoint is only available to Projects with Pro access and Enterprise access” which cost thousands of dollars per month (source: documentation). So when I run the program I encounter the following error:

`tweepy.errors.Forbidden: 403 Forbidden

When authenticating requests to the Twitter API v2 endpoints, you must use keys and tokens from a Twitter developer App that is attached to a Project. You can create a project via the developer portal.`

This means that I am unable to fully test my code, and I am unable to complete this project at this time. Hopefully I was able to demonstrate some knowledge of how this type of system should work, if Twitter hadn’t started dramatically limiting its API usage.

If the API did allow me to access the data I need, then I believe I would need to change the way the data is shaped for training and prediction, but other than that this code should function just like the multivariate data training.