# COS30018 Assignment B – Task 2

Aidan Grimmett: 103606838 – Friday 12:30 class

I created a new Python file with the following new function:

```
def load_and_process_data(
    company,
    scale_features,
    save_data,
    data_dir
):
    start_date = input("Please enter data start date (yyyy-mm-dd): ")     # Start date to read
    end_date = input("Please enter data end date (yyyy-mm-dd): ")         # End date to read
```

Firstly the user inputs the start and end dates for the sampling period

Then it will create a file path that will be used to either retrieve or save a data file

```
# Step 1: Load Data
# Either load or create a new file path for the data we want to use
# Come up with a file path based on the company and time span
file_path = os.path.join(data_dir, f"{company}_{start_date}_{end_date}.csv")
# Check if it exists and if not we will create it
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)
```

To deal with any NaN records, we drop those values:

```
# Step 2: Handle NaN values
# Data often contains missing or incorrect data. So here, we must drop all NaN data
data = data.dropna()
```

Then the user can choose how the data will be split out of ratio, random or from a specific date into the training and test data sets

```python
# Step 3: Split the Data

# Get the user to choose a split method
split_method = input("What training/test data split method would you like to use? (ratio, date or random): ")

# Split the data accordingly
if split_method == 'ratio':
    # Get a custom ratio from the user
    train_ratio = float(input("What ratio do you want to use? (0.1 - 1)"))
    split_index = int(len(data) * train_ratio)
    # Store all data values up to the split index as train_data, and the rest as test_data
    train_data = data.iloc[:split_index]
    test_data = data.iloc[split_index:]
elif split_method == 'date':
    # Get a date to split from
    test_start_date = input("Testing start date (yyyy-mm-dd): ")
    train_data = data.loc[:test_start_date]
    test_data = data.loc[test_start_date:end_date]
elif split_method == 'random':
    train_ratio = float(input("What ratio do you want to use? (0.1 - 1)"))
    # Randomly sample the data into the two training or test sets. Uses a constant seed to ensure reproducability
    train_data = data.sample(frac=train_ratio, random_state=22)
    test_data = data.drop(train_data.index)
```

Then we scale the data so that all features fit within a specific range. This should enhance the performance of the model in recognising and being able to predict patterns in the data.

```python
# Step 4: Scale the Features
# Create a dictionary of the different scalers we will store
scalers = {}
if scale_features:
    # Normalise all of the features to fit within a range of 0-1 (ie. the highest value will be 1 and the minimum value will be 0)
    for column in train_data.columns:
        scaler = MinMaxScaler(feature_range=(0, 1))
        # Scale train and test data independently
        train_data[column] = scaler.fit_transform(train_data[column].values.reshape(-1, 1))
        test_data[column] = scaler.transform(test_data[column].values.reshape(-1, 1))
        scalers[column] = scaler
    # Save the scaled data
    if save_data:
        joblib.dump(scalers, os.path.join(data_dir, f"{company}_scalers.pkl"))

return train_data, test_data, scalers
```

Here is the function being used in the main script:

```python
41  #------------------------------------------------------------------------------
42  # Load Data
43  ## TO DO:
44  # 1) Check if data has been saved before.
45  # If so, load the saved data
46  # If not, save the data into a directory
47  #------------------------------------------------------------------------------
48  # DATA_SOURCE = "yahoo"
49  COMPANY = 'CBA.AX'
50
51  data, test_data, scalers  = load_and_process_data(COMPANY, True, True, '/Data')
52  #------------------------------------------------------------------------------
```

Proof that the code still executes and works: