Created function

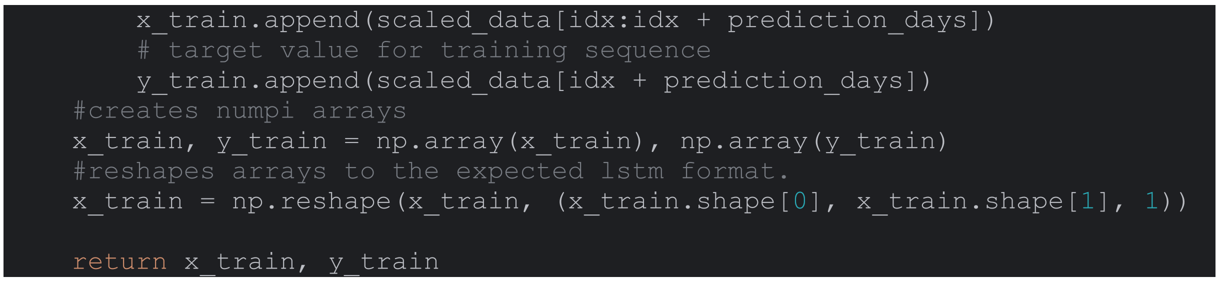
```
def prepare_stock_data(company, start_date, end_date, price_column="Close",
prediction_days=60, test_ratio=0.2, random_split=True):
    # Requirement a: Specifying the start and end dates
    results_folder = "results"
    # Checks if folder exists and makes one if it isnt
    if not os.path.exists(results_folder):
        os.makedirs(results_folder)
    # Directory of data file
    scaled_data_file = os.path.join(results_folder,
f"{company}_scaled_data.pkl")
    # If file exists import it
    if os.path.exists(scaled_data_file):
        # Requirement d: Load data from a local file if it exists
        scaled_data = pd.read_pickle(scaled_data_file)
        scaler = scaled_data["scaler"]
        scaled_data = scaled_data["data"]
    # Downloads data if scaled_data_file doesnt exist
    else:
        #download from yahoo finance and save file
        data = yf.download(company, start=start_date, end=end_date,
progress=False)
        data.to_csv(os.path.join(results_folder, f"{company}_data.csv"))

        # Requirement b: Dealing with NaN issues
        # dropping data is the most common resolution of bad data
        data = data.dropna()

        # Requirement e: Scaling feature columns and storing scaler
        # scaled 0,1 unsure if there are better scaling for this data set
but 0,1 seems to be "standard"
        scaler = MinMaxScaler(feature_range=(0, 1))
        # transforming the data to the required format
        scaled_data =
scaler.fit_transform(data[price_column].values.reshape(-1, 1))

        # Requirement d: Save data to a local file
        # saves scaled data to a pickle file
        pd.to_pickle({"scaler": scaler, "data": scaled_data},
scaled_data_file)
    # Calculate the number of data points that will be used for training
the model
    train_size = int(len(scaled_data) * (1 - test_ratio))

    if random_split:
        # Requirement c: Different methods for splitting data
        # Randomly select data to assign data for training and testing at
the specified train_size ratio
        train_indices = np.random.choice(len(scaled_data) -
prediction_days, train_size, replace=False)
    else:
        # Sequentially select the initial train_size% of indices for
training and the remaining for testing
        train_indices = range(train_size)
    # training arrays predicts the next point of sequence x and stored in y
    x_train = []
    y_train = []
    # loops through training data
    for idx in train_indices:
        # creates training sqeuence
```



```
        x_train.append(scaled_data[idx:idx + prediction_days])
        # target value for training sequence
        y_train.append(scaled_data[idx + prediction_days])
    #creates numpi arrays
    x_train, y_train = np.array(x_train), np.array(y_train)
    #reshapes arrays to the expected lstm format.
    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))

    return x_train, y_train
```

used os import to handle the files that store data (saves exception handling on my end) checks and creates a folder results (might change to data later) which holds all downloaded data.

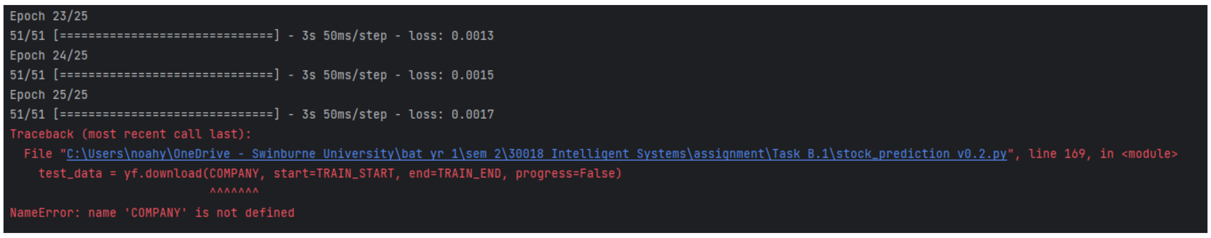
NaN issue is solved by dropping missing data, other methods exist but is most common for its ease of use (and we have a fairly large data set so losing some volume isn’t a huge worry)

scaling 0,1 simply because this is the defacto and nothing I read seemed to imply other scaling would help, saved scaled data to a pickle file (used by pandas and sklearn

splits data randomly at a .2 ratio by default but can change that in the function call using test_ratio and random_split (prediction days can also be changed too)

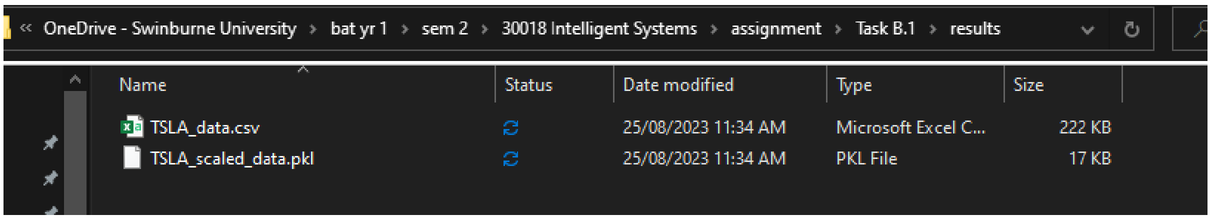
returns the x and y training arrays (unsure if other data needs to be returned but not hard to fix.

function runs and trains a model but due to change in variable names and some data being encapsulated inside the function the testing of the model does not work.



```
Epoch 23/25
51/51 [==============================] - 3s 50ms/step - loss: 0.0013
Epoch 24/25
51/51 [==============================] - 3s 50ms/step - loss: 0.0015
Epoch 25/25
51/51 [==============================] - 3s 50ms/step - loss: 0.0017
Traceback (most recent call last):
  File "C:\Users\noahy\OneDrive - Swinburne University\bat yr 1\sem 2\30018 Intelligent Systems\assignment\Task B.1\stock_prediction v0.2.py", line 169, in <module>
    test_data = yf.download(COMPANY, start=TRAIN_START, end=TRAIN_END, progress=False)
                            ^^^^^^^
NameError: name 'COMPANY' is not defined
```

Saved data in results folder



« OneDrive - Swinburne University › bat yr 1 › sem 2 › 30018 Intelligent Systems › assignment › Task B.1 › results

Name	Status	Date modified	Type	Size
TSLA_data.csv		25/08/2023 11:34 AM	Microsoft Excel C...	222 KB
TSLA_scaled_data.pkl		25/08/2023 11:34 AM	PKL File	17 KB