Assignment B Task 5

Aidan Grimmett 103606838

Task 1:

For the first task, I implemented a new method `predict_next_k_days()`. It is fed the following parameters:

model: a layered trained keras model

model_inputs: the data we will use to base our predictions on

scaler: the scaler to use to normalise the data

prediction_prev_days: the amount of previous days we look at to make our prediction

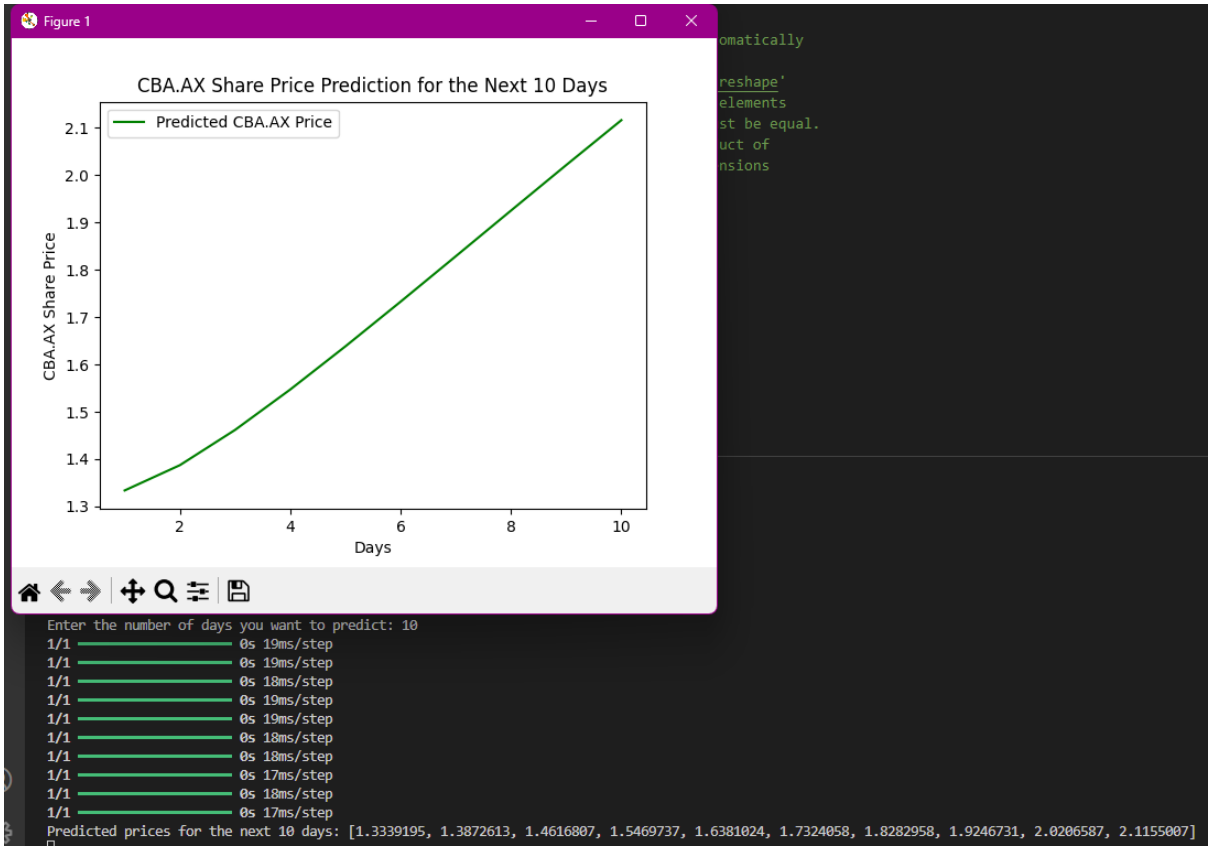
k: the amount of days into the future we will predict the closing price for

```
def predict_next_k_days(model, model_inputs, scaler, prediction_prev_days, k):
    predictions = []
    real_data = model_inputs[len(model_inputs) - prediction_prev_days:, 0]

    for _ in range(k):
        real_data = np.reshape(real_data, (1, real_data.shape[0], 1))
        prediction = model.predict(real_data)
        prediction = scaler.inverse_transform(prediction)
        predictions.append(prediction[0][0])
        # Update the real_data with the new prediction for the next step
        real_data = np.append(real_data[0, 1:], prediction[0][0])

    return predictions
```

First we create a new list to store our predictions in, then `real_data = model_inputs[len(model_inputs) - prediction_prev_days:, 0] ` extracts the last prediction_prev_days closing values from the model inputs data.

After this we set up a loop that will run k times, where will will make a prediction and add it to the predictions list. The reshape line ensures that the data is in the correct form for the model (A 3D input) where 1 is the batch size, real_data.shape[0] is the number of previous days used and 1 is the feature, closing price.

After shaping the data we make a prediction using the model, and unscale the data back to the original price scale and add it to the predictions list. Finally this value is also appended to the real data list so that it can be used to predict the next day on the next iteration.



This is what the output graph looks like.

Task 2:

The second task was much more challenging.

I had to change the x_train shape to include the extra features with

```
` x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], x_train.shape[2]))` instead of ` x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], 1))`
```

The input shape for the model training was also adjusted to include the extra features, `input_shape=(x_train.shape[1], x_train.shape[2])`, as well as the actual prices having to be reshaped and scaled differently:
Scale the actual test prices correctly for plotting

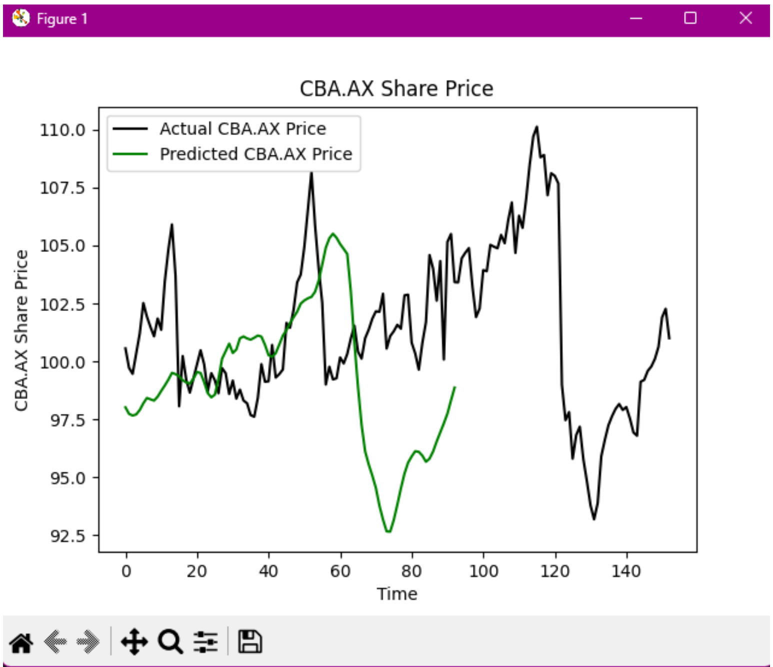
actual_prices = test_data[PRICE_VALUE].values.reshape(-1, 1)

actual_prices = scalers['Close'].inverse_transform(actual_prices)

The prediction code had to be adjusted in similar ways to suit the new features.

```
def predict_next_k_days(model, model_inputs, scaler, prediction_prev_days, k):
    real_data = [model_inputs[-prediction_prev_days:]]  # Extract the last prediction_prev_days of model inputs

    real_data = np.array(real_data)  # Convert to np array
    real_data = np.reshape(real_data, (real_data.shape[0], real_data.shape[1], real_data.shape[2]))  # Ensure 3D shape with extra features

    predictions = []

    for _ in range(k):
        prediction = model.predict(real_data)  # Predict the next day
        predictions.append(prediction) # Add new prediction
        # Update the real_data with the new prediction (to predict the next day after that)
        new_data = np.append(real_data[0][1:], prediction, axis=0)  # Remove the first timestep and append the new prediction
        real_data = np.array([new_data])

    predictions = np.array(predictions)
    predictions = scaler.inverse_transform(predictions.reshape(-1, 1))

    return predictions
```

However this was not successful, and I was unable to get this code to run correctly. The predictions would not run as there were shape mismatches between the data and what the model expected. The main code prediction also suffered at some point, with it becoming much less accurate and cutting off part way through the prediction period. Outputs and error messages can be seen here:




```
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 85ms/step - loss: 0.0677 - mean_absolute_error: 0.0677
Epoch 15/20
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 84ms/step - loss: 0.0589 - mean_absolute_error: 0.0589
Epoch 16/20
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 84ms/step - loss: 0.0779 - mean_absolute_error: 0.0779
Epoch 17/20
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 85ms/step - loss: 0.0611 - mean_absolute_error: 0.0611
Epoch 18/20
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 85ms/step - loss: 0.0632 - mean_absolute_error: 0.0632
Epoch 19/20
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 85ms/step - loss: 0.0574 - mean_absolute_error: 0.0574
Epoch 20/20
10/10 ━━━━━━━━━━━━━━━━━━━━ 1s 85ms/step - loss: 0.0472 - mean_absolute_error: 0.0472
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step
Input:
1 for standard graph
```

```
Enter the number of days you want to predict: 1
2024-09-22 23:32:07.186021: I tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: INVALID_ARGUMENT: Matrix size-incompatible: In[0]: [1,1], In[1]: [6,1024]
         [[{{node sequential_1/lstm_1/while/body/_1/sequential_1/lstm_1/while/lstm_cell_1/MatMul}}]]
Traceback (most recent call last):
  File "c:\Users\aidsg\Documents\Repos\StockPrediction\StockPrediction_AidanGrimmett_COS30018\B.5\stock_prediction.py", line 194, in <module>
    predictions = predict_next_k_days(model, model_inputs, scalers['Close'], PREDICTION_DAYS, k)
                  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "c:\Users\aidsg\Documents\Repos\StockPrediction\StockPrediction_AidanGrimmett_COS30018\B.5\predict_k_days.py", line 15, in predict_next_k_days
    prediction = model.predict(real_data)
                 ^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Users\aidsg\AppData\Roaming\Python\Python312\site-packages\keras\src\utils\traceback_utils.py", line 122, in error_handler
    raise e.with_traceback(filtered_tb) from None
  File "C:\Users\aidsg\AppData\Roaming\Python\Python312\site-packages\tensorflow\python\eager\execute.py", line 53, in quick_execute
    tensors = pywrap_tfe.TFE_Py_Execute(ctx._handle, device_name, op_name,
  ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
tensorflow.python.framework.errors_impl.InvalidArgumentError: Graph execution error:

Detected at node sequential_1/lstm_1/while/body/_1/sequential_1/lstm_1/while/lstm_cell_1/MatMul defined at (most recent call last):
<stack traces unavailable>
Matrix size-incompatible: In[0]: [1,1], In[1]: [6,1024]
         [[{{node sequential_1/lstm_1/while/body/_1/sequential_1/lstm_1/while/lstm_cell_1/MatMul}}]] [Op:__inference_one_step_on_data_distributed_7336]
```