Project B

Task 5 – Machine Learning 2

Tasks:

1. Implement a function that solves the multistep prediction problem to allow the prediction to be made for a sequence of closing prices of k days into the future.

2. Implement a function that solve the simple multivariate prediction problem to that takes into account the other features, for the same company (including opening price, highest price, lowest price, closing price, adjusted closing price, trading volume) as the input for predicting the closing price of the company for a specified day in the future.

3. Combine the above two function to solve the multivariate, multistep prediction problem.

To implement multistep prediction, I defined a variable called LOOKUP_STEP. This variable will be used to determine how many days ahead the model will make a prediction.

```
N_LAYERS = 2
# Type of recurrent unit
CELL = SimpleRNN
# Amount of neurons per laye
UNITS = 256
# 40% dropout
DROPOUT = 0.4
# whether to use bidirectional
BIDIRECTIONAL = False

LOSS = "mae"
OPTIMIZER = "adam"
BATCH_SIZE = 64
EPOCHS = 200

N_STEPS = 120 # How many days of past data the model uses to make a prediction
LOOKUP_STEP = 120 # How far into the future the model is predicting
```

In the snippet above, LOOKUP_STEP is set to 120. This means that the model will predict the prices 120 in the future.

The LOOKUP_STEP was used in the following code:

```
y_train = train_data[FEATURE_COLUMN[0]].shift(-LOOKUP_STEP).dropna().values
```

```
y_test = test_data[FEATURE_COLUMN[0]].shift(-LOOKUP_STEP).dropna().values
```

This snippet shows how I shifted the target values using LOOKUP_STEP to train the model to predict X number of days in advance.

Then, to incorporate multivariate predictions for this notebook, the features that are going to be used first had to be defined.

```
COMPANY_TICKER = "AMZN"

# Specification of start and end date for both train and test sets
# This means that at some point we need to split the dataset
START_DATE = '2019-01-01'
END_DATE = '2024-01-01'

NAN_HANDLER = "drop" # Determines how NaN data is handled. Can be "drop",
"fill_mean", "fill_median", or "fill_ffill"

TEST_SIZE = 0.20 # e.g. this would be that 75% of the available data is being used
for the train set and 25% is being used for the test set.

# If false the data will be randomly split
SPLIT_BY_DATE = True

# ["Open", "Adj Close", "Volume", "High", "Close", "Low"]

FEATURE_COLUMN = ["Open", "Adj Close", "Volume", "High", "Close", "Low"]  # Can be
set to "Adj Close", "Volume", "Open", "High", or "Low"

# Scalar is used to normalize data to a specific range (In this code, it normalizes
the data to between 0 and 1)
SCALE = True

DATE_NOW = time.strftime("%Y-%m-%d")
```

I defined them as the FEATURE_COLUMN. I then used this feature column for the data in combination with a sequence creation method to create sequences that include all of the determined columns.

```
def create_sequences(data, n_steps):
    sequences = deque(maxlen=n_steps)
    sequence_data = []

    for entry in data:
        sequences.append(entry)
```

```
        if len(sequences) == n_steps:
            sequence_data.append(np.array(sequences))

    return np.array(sequence_data)
```

This method is used to create the sequences that the model will use to make its prediction. E.g., if N_STEPS is set to 50, then the model will use the last 50 data points (in this case days) to make its stock price prediction.

Here is where the feature columns are used in conjunction with the sequence method.

```
X_train = create_sequences(train_data[FEATURE_COLUMN].values, N_STEPS)
X_test = create_sequences(test_data[FEATURE_COLUMN].values, N_STEPS)
```

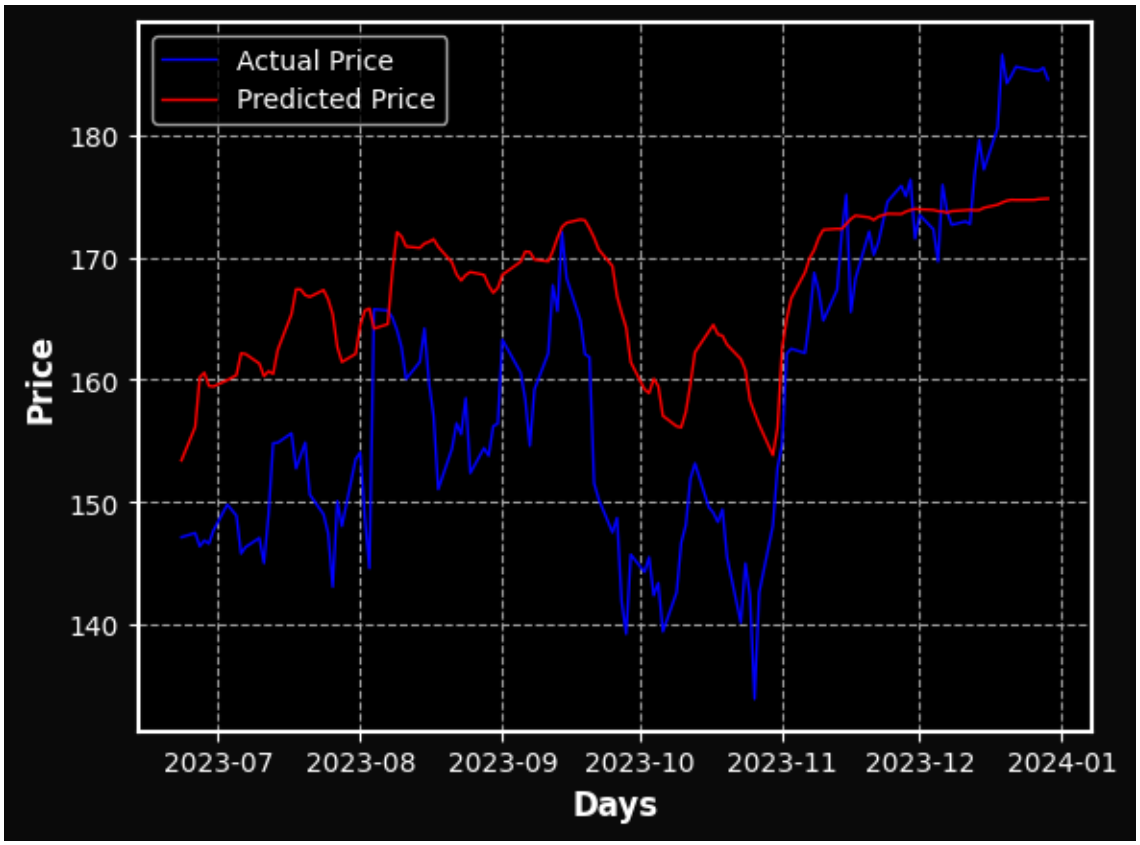
To solve the multivariate, multistep prediction problem I used FEATURE_COLUMNS, and N_STEPS

The feature columns are also used when creating the model architecture.

```
model = create_model(N_STEPS, len(FEATURE_COLUMN), loss=LOSS, units=UNITS,
cell=CELL, n_layers=N_LAYERS,
                    dropout=DROPOUT, optimizer=OPTIMIZER,
bidirectional=BIDIRECTIONAL)
```

By passing in the length of the feature columns, the model can accept datasets with the same number of variables as the feature column has.

Here is an example of the model predicting more than 1 day into the future using multiple variables.




```
Loss: 0.0910729169845581
Mean Absolute Error: 0.0910729169845581
Accuracy Score: 0.5303030303030303
Total Buy Profit: -7.97847766635428
Total Sell Profit: 40.80331337790088
Total Profit: 32.8248357115466
Profit Per Trade: 0.24867299781474694
Predicted future price after 120 days: 174.80$
```