COS30018 Intelligent Systems - Task B.5

Ryan Chessum

This week I incorporated multistep and multivariate prediction into the project.

First, I made a basic function for multistep prediction. The function takes in data and a model for parameters as well as the number of days into the future the program watches to forecast. The function works by first using model to make a prediction on the next day using the last sequence data. Then the last sequence data is modified by removing the first day in the sequence and adding the prediction to the end of the sequence. The function will do this k times, so the previous prediction is used to predict the next day as if it was the price of the previous day. Once the model has made all the predictions, the function will return them.

Next, I added multivariate predictions to the model. To do this, I added the other columns to the input data for the model. We could easily add more variables for the model to make predictions on by adding another feature column with the data.

```
for entry, target in zip(data_frame[feature_columns + ["date"]].values, data_frame['future'].values):
    sequences.append(entry)
    if len(sequences) == n_steps:
        sequence_data.append([np.array(sequences), target])
```

After adding multivariate predictions to the model, the multistep prediction function will no longer work as the prediction made will only give the closing price of the company, so we can't add it as the last sequence as we are only using 1 input when 6 are now required. To solve this, I made the function take in 6 models, 1 to predict the stock price of each feature column. Each model makes its prediction then the data is added the last sequence so predictions can be made for future days like before.

```
[[137.17459]]
[[136.56473]]
[[136.1949]]
[[136.04036]]
[[136.04912]]
[[136.17075]]
[[136.36774]]
[[136.6174]]
[[136.90813]]
[[137.23515]]
```

```python
def multistep_prediction(adjclose, close, volume, op, high, low, data, n_steps, k = 1, scale=True, target='adjclose'):
    #list of predictions
    results = []
    # retrieve the last sequence from data
    last_sequence = data["last_sequence"][-n_steps:]
    # expand dimension
    last_sequence = np.expand_dims(last_sequence, axis=0)

    i = 0

    while i < k:
        predictions = []
        #make a prediction
        ac_p = adjclose.predict(last_sequence)
        c_p = close.predict(last_sequence)
        v_p = volume.predict(last_sequence)
        o_p = op.predict(last_sequence)
        h_p = high.predict(last_sequence)
        l_p = low.predict(last_sequence)
        predictions.append(ac_p[0][0])
        predictions.append(c_p[0][0])
        predictions.append(v_p[0][0])
        predictions.append(o_p[0][0])
        predictions.append(h_p[0][0])
        predictions.append(l_p[0][0])

        preds = np.array(predictions).astype(np.float32)
        preds = np.expand_dims(preds, axis=0)

        #remove first item in last sequence
        ls = np.delete(last_sequence[0], 0, axis=0)
        #add predictions
        ls = np.append(ls, preds, axis=0)
        last_sequence[0] = ls

        #add desired preds to results

        if scale:
            ac_p = data["column_scaler"]["adjclose"].inverse_transform(ac_p)
            c_p = data["column_scaler"]["close"].inverse_transform(c_p)
            v_p = data["column_scaler"]["volume"].inverse_transform(v_p)
            o_p = data["column_scaler"]["open"].inverse_transform(o_p)
            h_p = data["column_scaler"]["high"].inverse_transform(h_p)
            l_p = data["column_scaler"]["low"].inverse_transform(l_p)

        if target == "adjclose":
            results.append(ac_p)
        if target == "close":
            results.append(c_p)
        if target == "volume":
            results.append(v_p)
        if target == "open":
            results.append(o_p)
        if target == "high":
            results.append(h_p)
        if target == "low":
            results.append(l_p)

        i += 1

    return results
```