COMP5313 Artificial Intelligence

Department of Computer Science

Project 1: Stock Prices Prediction Using AI and Machine Learning

Instructions to run the code:

The two python files:

- SiddhiJariwala_RL.py
- SiddhiJariwala_RL.ipynb

➢ These two files can be run on Python IDE PyCharm platform or jupyter notebook.

The data for the ticker NFLX (Netflix) has been obtained from the Yahoo Finance data. The data from 2017 to 2022 will be utilized as the training set, while the data from 2023 will serve as the testing set.

Flow of Program:

- Initially, I have imported the required libraries, including Numpy for numerical computations, Matplotlib for plotting, Pandas for data manipulation, and scikit-learn's MinMaxScaler for scaling data along with the necessary modules from Keras, a deep learning library, to build a Long Short-Term Memory (LSTM) neural network.

  Post that, it reads the training dataset in the form of a CSV file and extracts the relevant columns. It then scales the values of the training set using MinMaxScaler and creates sequences of 60 previous values for each day's stock price. These sequences will serve as the input to the LSTM model, and the corresponding stock price for the day will serve as the target.

  Finally, the sequences are reshaped into 3-dimensional arrays, with the first dimension representing the number of sequences, the second dimension representing the length of each sequence, and the third dimension representing the number of features in each value of the sequence.

- Later we define, a sequential model, which is a linear stack of layers in Keras, and adds several LSTM layers to it. The first LSTM layer has 50 units and the input shape specified as (X_train.shape[1], 1), which corresponds to the length of the sequence and the number of features in each value. The 'return_sequences' argument is set to True for all LSTM layers except for the last one, which means that the hidden state and cell state of each LSTM unit will be passed on to the next layer. Dropout layers are added between each LSTM layer to prevent overfitting by randomly turning off some of the units during each training iteration. The final layer is a Dense layer with one unit, which outputs a single value, representing the predicted stock price.

  The model is then compiled by specifying the optimizer as the Adam algorithm and the loss function as mean squared error, which is used for regression problems. The fit function is then used to train the model on the input and target data for 100 epochs, with a batch size of 32, which means that the model will be trained on 32 sequences at a time before updating the weights.

- The test dataset is loaded from the file "NFLX_test.csv" into a Pandas dataframe called 'test_data_set' and extracts the "Open" column, representing the opening stock price, into a NumPy array called 'actual_stock_price'. The train dataset and test dataset are then concatenated along the 0th axis to form a complete dataset. Next, the input values for the prediction are created by selecting the last 60 values from the complete dataset and transforming them using the MinMaxScaler object trained on the training data. These values are stored in the 'inp_val' NumPy array.

  The input values for the prediction are then reshaped into a 3-dimensional array with the shape (16, 60, 1) to match the input shape expected by the LSTM model. The reshaped input values are then passed to the predict function of the LSTM model to generate the predicted stock prices.

- Finally, the predicted stock prices are transformed back to their original scale using the 'inverse_transform' function of the MinMaxScaler object. The resulting NumPy array, 'pred_value_stock', contains the predicted stock prices for the test data.

  As a result, it will display a plot of the actual Netflix stock price in black and the predicted Netflix stock price in red, for a specified time period. The x-axis will represent the time, and the y-axis will represent the stock price. A legend will be included to differentiate the actual and predicted stock prices.