Task 6: Machine Learning 3

Muhammad Ali, 109604337, COS30018, 06/10/2024, Tutor: Ru Jia.

Requirements / Deliverables –

- Develop an ensemble modelling approach consisting of at least two models ARIMA (or SARIMA) and our existing DL model (starting with the LSTM one).
- Experiment with different ensemble models (e.g. ARIMA/SARIMA/Random Forrest/LSTM/RNN/GRU, etc.) and with different hyperparameter configurations.

Result –

Summary of Implementation of ensemble model (SARIMA) –

1. <u>Data Preparation –</u>

- Two time series objects, train_series and test_series are created from the training and testing datasets (y_train and y_test), respectively. These are indexed by date, making them compatible with time series analysis.

```
train_series = pd.Series(y_train.flatten(),
index=pd.date_range(start=TRAIN_START, periods=len(y_train)))
test_series = pd.Series(y_test.flatten(),
index=pd.date_range(start=TRAIN_END, periods=len(y_test)))
```

2. <u>Modelling SARIMA using *auto-arima* –</u>

- The auto_arima() function automatically finds the best parameters for the SARIMA model by trying different combinations of orders and seasonal orders. It uses metrics like the **Akaike Information Criterion (AIC)** to determine the optimal parameters.
- The best parameters are stored in best_order and best_seasonal_order.

```
print("Finding the best SARIMA parameters using auto_arima...")
auto_sarima_model = auto_arima(train_series, seasonal=True, m=12,
trace=True, error_action='ignore', suppress_warnings=True, stepwise=True)
```

```
Finding the best SARIMA parameters using auto_arima...
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=3170.495, Time=2.29 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=3184.100, Time=0.09 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=3176.324, Time=0.26 sec
```

3. <u>SARIMA Model Fitting –</u>

- The **SARIMA** model is created using the SARIMAX function from the statsmodels library with the optimal parameters obtained from auto_arima(). It's then fit to the training series.

```
sarima_model = SARIMAX(train_series, order=best_order,
seasonal_order=best_seasonal_order)
sarima_result = sarima_model.fit(disp=False)
```

4. <u>Predicting with SARIMA</u>

- Once the model is fitted, it predicts future values over the test set period. These predictions are stored in sarima_predictions.

```
sarima_predictions = sarima_result.predict(start=len(train_series),
end=len(train_series) + len(test_series) - 1)
```

5. <u>Truncating predictions to match length –</u>

- Since predictions from the DL model (predicted_prices_inv) and the SARIMA model (sarima_predictions) may differ in length, they are truncated to the same length. This step ensures compatibility when combining predictions.

```
min_length = min(len(predicted_prices_inv), len(sarima_predictions))
sarima_predictions = sarima_predictions[:min_length]
predicted_prices_inv = predicted_prices_inv[:min_length]
actual_prices_inv = actual_prices_inv[:min_length]
```

6. <u>Ensemble Combination –</u>

- The ensemble model combines the DL and SARIMA predictions using a weighted average.
- Weights are set to 0.7 for the DL model and 0.3 for the SARIMA model, meaning the DL model's predictions are given more importance.

```
weight_dl = 0.7
weight_sarima = 0.3
```

```
ensemble_predictions = (weight_dl * predicted_prices_inv) + (weight_sarima
* sarima_predictions[:, None])
```

7. Evaluation –

The ensemble predictions and actual values are flattened, making them suitable for comparison or evaluation using metrics like RMSE or MAE.

```
ensemble_predictions_flat = ensemble_predictions.flatten()
actual_prices_flat = actual_prices_inv.flatten()

mae_ensemble = mean_absolute_error(actual_prices_flat,
ensemble_predictions_flat)
rmse_ensemble = np.sqrt(mean_squared_error(actual_prices_flat,
ensemble_predictions_flat))
```

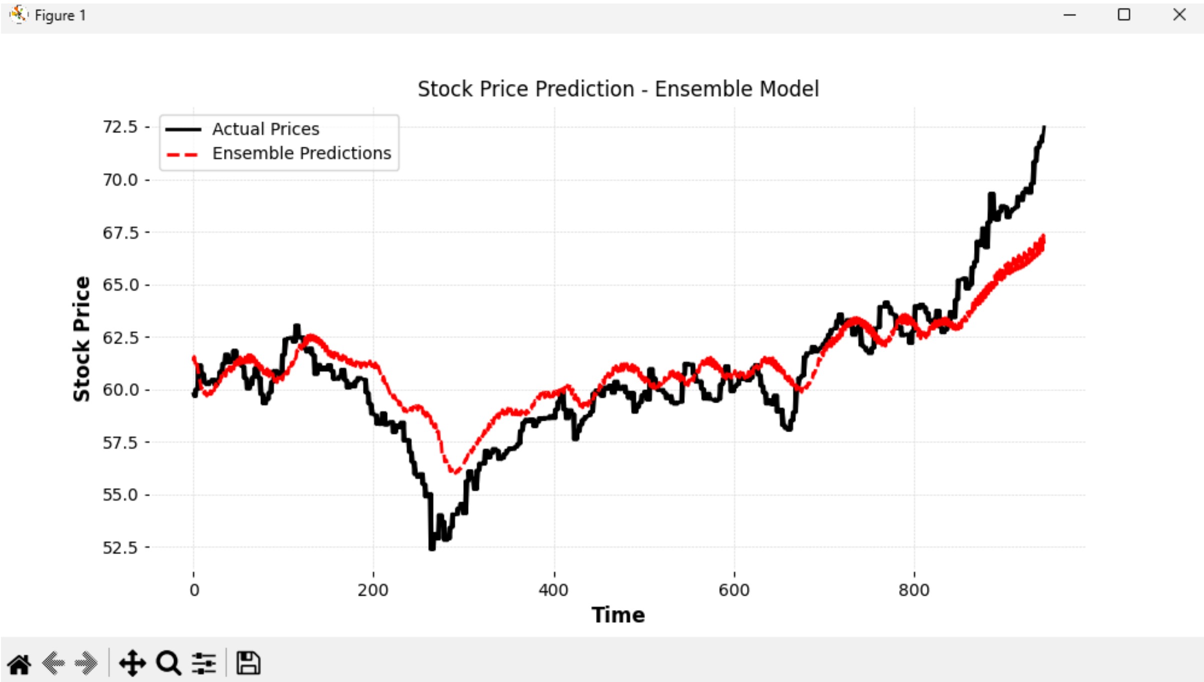
Summaries of results of different configurations of ensemble models and training –

Configuration 1 –

Multi layered LSTM only approach - an ensemble approach using the following hyperparameters paired with SARIMA. This configuration provided impressive prediction results, picking up on seasonality very accurately and acquiring a MAE (Mean absolute Error) score of 1.3.

```
Ensemble Model - Mean Absolute Error (MAE): 1.3142334143352878
```

```
layer_types = ['LSTM', 'LSTM', 'LSTM']
layer_sizes = [150, 100, 50]
dropout_rates = [0.2, 0.2, 0.2]
return_sequences = [True, True, False]
activation_functions = ['tanh', 'tanh', 'relu']
```



Configuration 2 –

LSTM, RNN, and GRU multi layered approach – An ensembled approach using the following LSTM, RNN, GRU layers and their accompanying hyperparameters. This configuration provided slightly better results over the previous configuration with a MAE (Mean Absolute Error) score of 1.09.

```
Ensemble Model - Mean Absolute Error (MAE): 1.093792915664244
```

```
layer_types = ['LSTM', 'RNN', 'GRU']
layer_sizes = [200, 150, 100]
dropout_rates = [0.3, 0.3, 0.2]
return_sequences = [True, True, False]
activation_functions = ['tanh', 'tanh', 'relu']
```

And the following batch size (decreased to 16 from 32) to improve overfitting results –

```
epochs=100, batch_size=16, verbose=1,
```

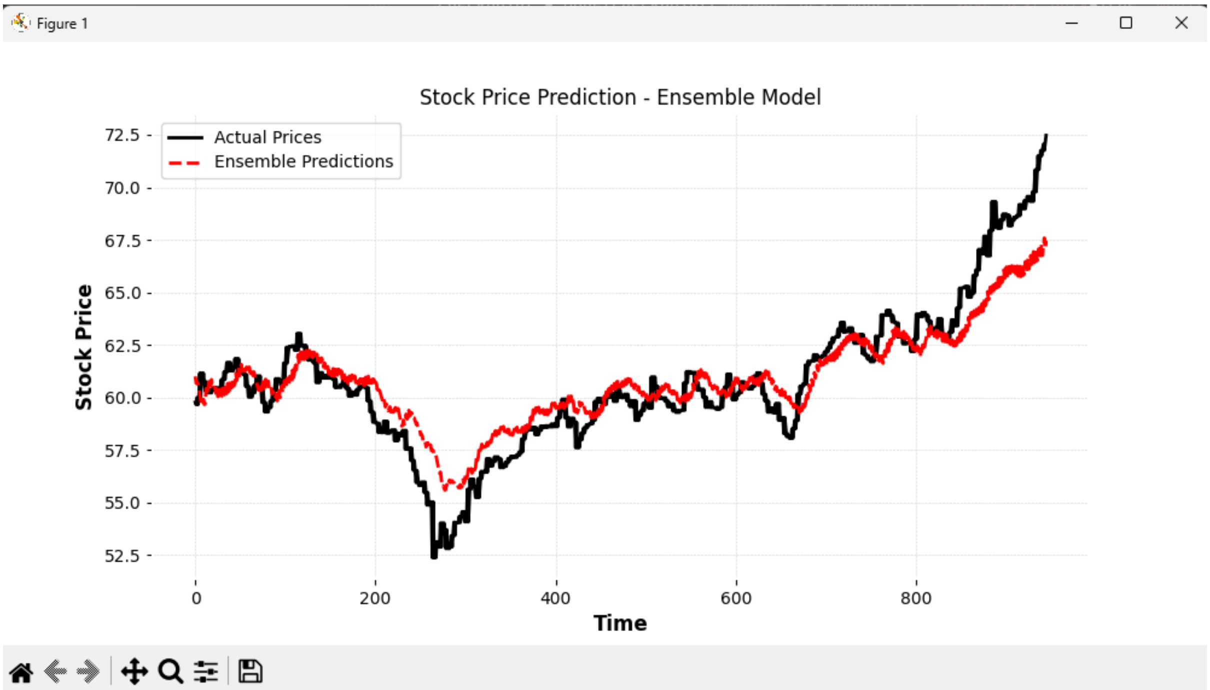


Figure 1
Stock Price Prediction - Ensemble Model
Actual Prices
Ensemble Predictions
72.5
70.0
67.5
65.0
62.5
60.0
57.5
55.0
52.5
Stock Price
0
200
400
600
800
Time