**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. Data Preparation –

* 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)))

1. Modelling SARIMA using *auto-arima –*

* 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)

A screen shot of a computer

Description automatically generated

1. SARIMA Model Fitting –

* 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)

1. Predicting with SARIMA

* 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)

1. Truncating predictions to match length –

* 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]

1. Ensemble Combination –

* 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])

1. 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. 

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']

A graph showing the price of a stock market

Description automatically generated

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.



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,

A graph showing a price

Description automatically generated