GRADUATE CERTIFICATE

PATTERN REASONING SYSTEM (PRS)

PRACTICE MODULE

---------- **PROJECT REPORT** ----------

iCARe – An Intelligent Car Budgeting Assistant

GROUP 4

Muniyandi Vanitha (A0249302L)

Sim Yuh Fan (A0249251E)

Tan Gek Teng (A0030151E)

Teoh Jeng Wei (A0093899B)

Table of Contents

Background

Following the rapid development of Singapore, we cannot limit our daily life into a fixed region. We always travel out of the current region to explore this world. Even with the aid of public transportation, our travel is still being constraint due to the inflexibility of public transportation such as fixed station location, crowd level, and limited operating hour. If we plan to travel to a farther place such as Malaysia, transportation arrangement becomes another headache.

Vehicle has progressively become an important part of our daily life. As of year 2018, it is estimated that approximately 16% to 17% of the global population are drivers. [1] In Singapore, there is about 11% of adult own a car as of year 2019. [2] There is a significant 5% below global average which are caused by many factors. The most discussed factor is Certificate of Entitlement (COE).

Introduction

The Certificate of Entitlement (COE) is the quota licence for owning a vehicle in the city-state of Singapore. The licence is obtained from a successful winning bid in an open bid uniform price auction which grants the legal right of the holder to register, own and use a vehicle in Singapore for a period of 10 years.

COE was introduced in year 1990 by Singapore government. 30 years later, COE is grouped into 5 categories – Category A, B, C, D and E. Each category cover different range of vehicle specifications. Furthermore, COE pricing is refreshed twice monthly by the Singapore government agency in transportation. These two pricing announcement dates are known as “Bidding Exercise”.

COE pricing in each bidding exercise varies based on multiple factors. When demand is high, the cost of a COE can exceed the value of the car itself. In general, it is significantly affected by these 3 factors: [3]

1. Actual number of vehicles taken off the roads (i.e. number of vehicles de-registered)
2. Allowable growth in vehicle population
3. Adjustments arising from temporary COEs that have expired or were cancelled.

Problem Statement

In recent years, the COE prices in Singapore have been skyrocketing due to many unprecedented factors such as high inflation, low COE quotas and high demand for private vehicles. Predicting future COE prices in this environment is not an easy task and the fear of increasing prices induces the average consumer to make potentially rash financial decisions.

Objective

To predict future COE prices and provide recommendations based on users' preferences and budget to help users make informed financial decisions through a web application named "iCARe" - an intelligent car budgeting assistant.

Proposed Solution

In iCARe system, a web application is presented which consists of 3 main components:

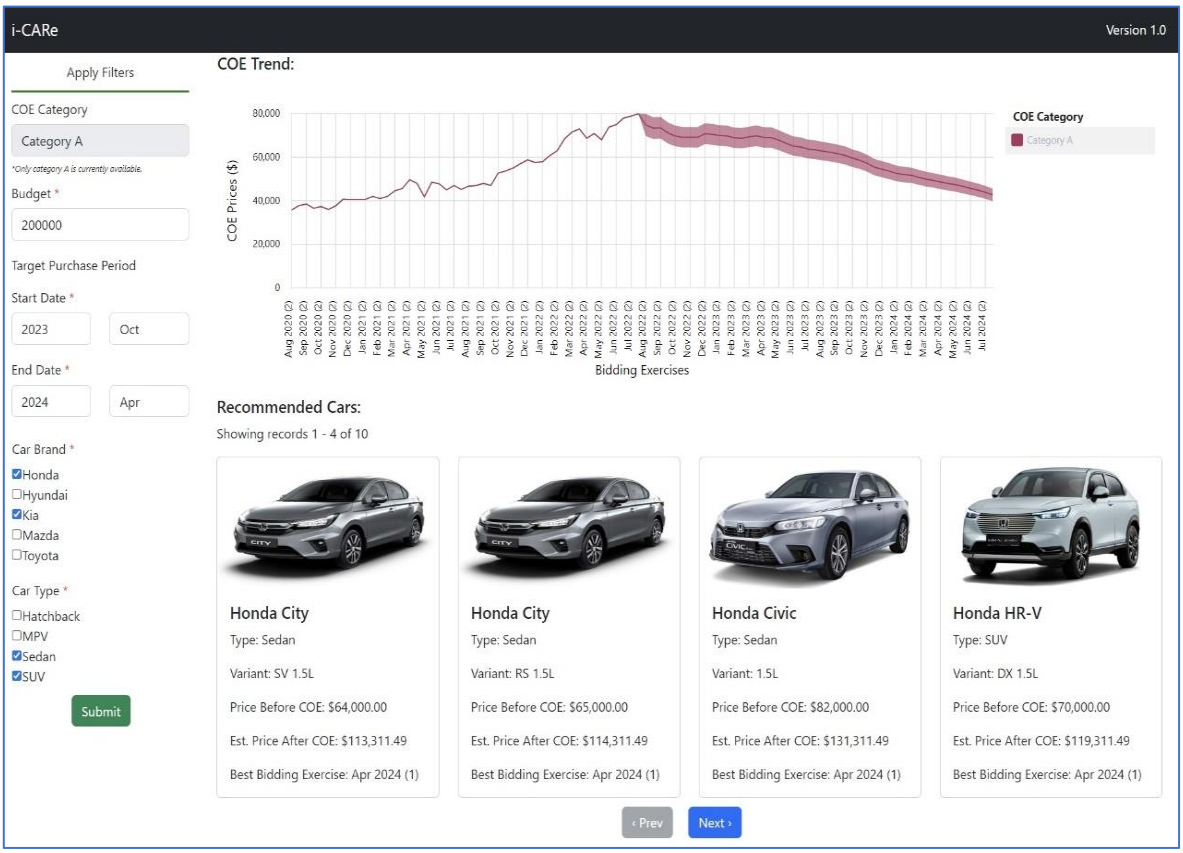

Figure 1 Screenshot of web application iCARe

First component is the "COE Trend". This component shows a graph plotting the COE price along the time domain. It indicates the historical COE price and forecasts future COE price.

Second component is the filter sidebar. It allows user to input their desired requirements and condition such as budget, target purchase time, interested car brand etc. Based on this user customization, system will sort and present final data in the last component.

Last component shows a list of recommended cars. It outputs the optimal solution based on COE price forecasting result and user filter from previous two components.

Methodology

The methodology utilised in this project can be broken down into the following sections:

- System Architecture
- Time Series Forecasting Models
- Model Performance Metrics
- Univariate Time Series Forecasting
- Multivariate Time Series Forecasting

System Architecture

The system architecture designed for the web application is shown below:

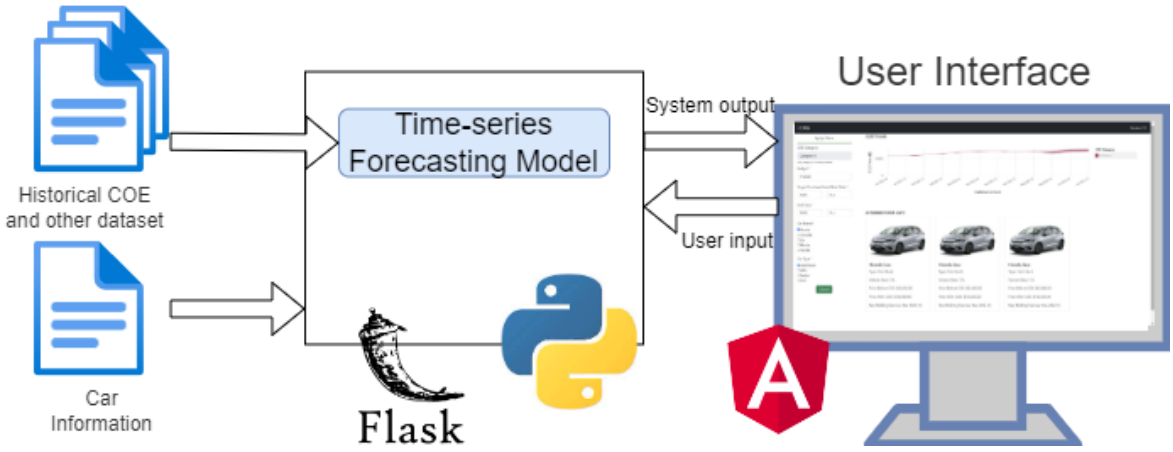

Figure 2 High-level system architecture design

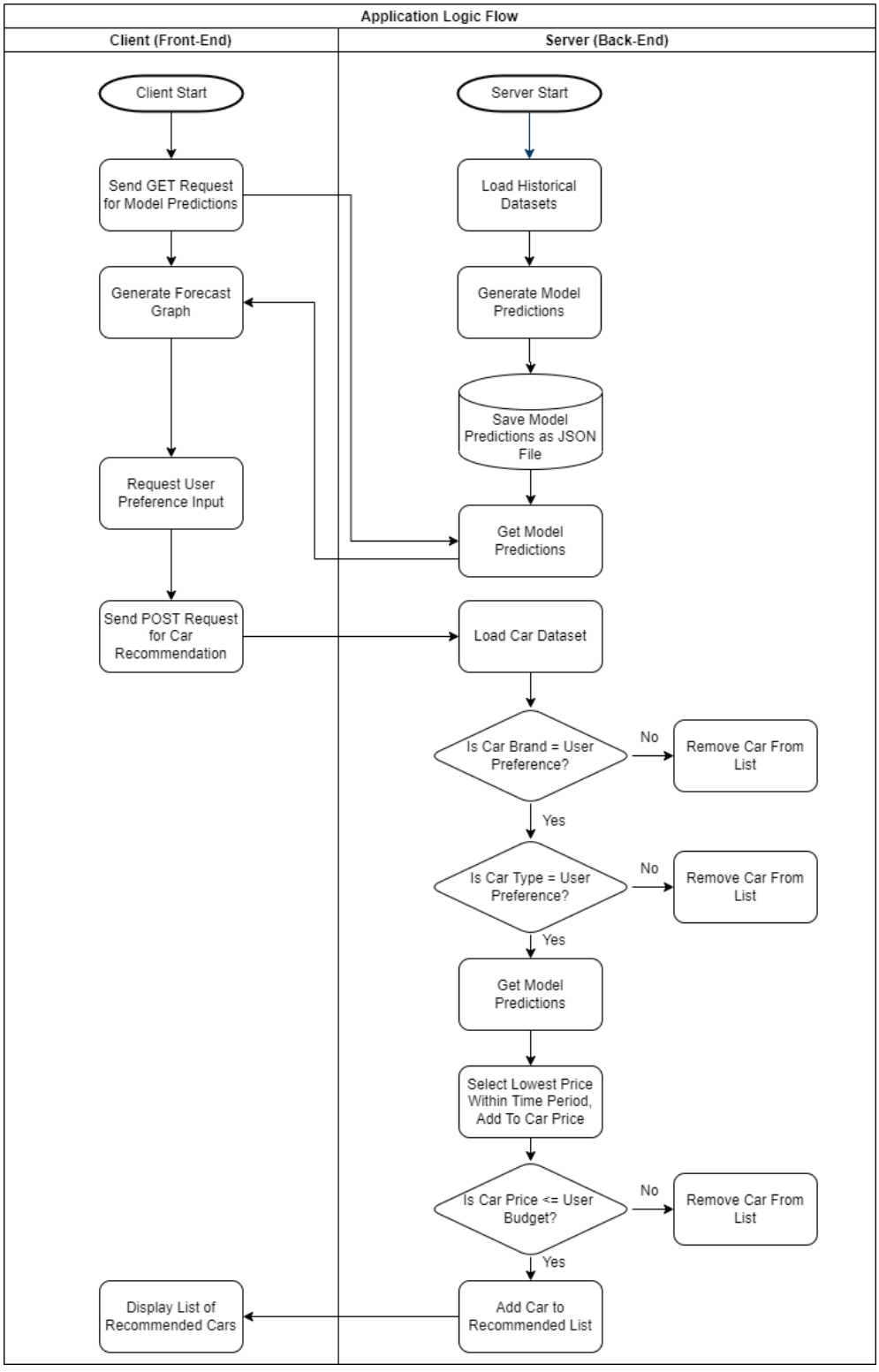

Figure 3 Application logic flow diagram

To run the application, the back-end system is first initialized. During which, the back-end system will load all relevant historical datasets such as COE, Gross Domestic Product (GDP) and Consumer Price Index (CPI), and subsequently get the trained forecasting model to generate predictions based on the historical values as inputs. The back-end system then saves the model predictions into a JSON file and

initializes a server endpoint for the front-end system to interact with. The frontend application is built using Angular 14, NG Bootstrap and Ngx-Charts framework. The backend application is built using Flask and consists of 2 REST APIs:

- Get COE price prediction
- Get list of recommended cars based on user preferences

When the application is accessed via the user's browser, the front-end system will send a GET request to the back-end system to get the model predictions. The front-end system will then use the model predictions to plot and display a graph of COE prices over time.

Subsequently, the front-end system will receive a set of user preferences including brand, car type, purchase period and budget via the set of filters that are present on the left-hand panel of the web application. Upon clicking the "Submit" button, the front-end system will send a POST request with the user preferences as the payload to the back-end system to get a list of recommended cars for the user. The back-end system will then generate a list of cars by loading the car dataset, which consists of information such as brand, type, variant, price before COE and image. This list is subsequently filtered to only include cars that matches the user preferences. It will also use the generated model predictions and the user preferred purchase period to extract the bidding exercise corresponding to the lowest predicted COE price. This predicted COE price is then added to the car price (excluding COE) to get the total car price. Finally, the list of cars is filtered again to remove cars that have prices that exceeded the user's stated budget. The list of cars with all the associated information is then passed from the back-end system to the front-end system which is then responsible for rendering a list of cars to be displayed on the application page.

Time Series Forecasting Models

Time series forecasting is the process of analysing time series data using statistics or machine learning to make predictions to make informed strategic decisions. However, an exact prediction of a future value is unlikely, and likelihood of forecasts can vary significantly, especially when time series data that is highly volatile and influenced by multiple factors that are beyond our control. [4] Although the set of forecasted values are never likely to be exact, a good model can make good approximations of future values, and this allows key insights to be generated to realise business value.

In this project, we are using various time series forecasting algorithms to analyse historical data points from COE dataset and various economic datasets, and train statistical and machine learning models to learn patterns which is then used to forecast future COE prices.

The performance of time series forecasting models is highly dependent on the nature of the time series dataset that the forecasting model is trained on and there is currently no reliable method of determining the best model to use for a given time series dataset. Thus, it is paramount that we train different forecasting models on both univariate and multivariate datasets and evaluate the respective models' performance to identify the best model to use for our application.

Models	Univariate	Multivariate
Autoregressive Integrated Moving Average (ARIMA)	✓	
Seasonal Autoregressive Integrated Moving Average (SARIMA)	✓	
Vector Autoregressive Moving Average (VARMA)		✓
Holt-Winters	✓	
Recurrent Neural Network (RNN)	✓	✓
Long Short-Term Memory Network (LSTM)	✓	✓

Table 1 List of time series forecasting models trained

Autoregressive Integrated Moving Average (ARIMA)

The Autoregressive Integrated Moving Average (ARIMA) model is a statistical model which is combination of the differenced autoregressive model with the moving average model. It is a class of models that 'explains' a given time series based on its own past values (lags) and the lagged forecast errors. An ARIMA model can be characterized by 3 terms: p, d, q, where p corresponds to the order of the auto-regressive (AR) term, q corresponds to the order of the moving average (MA) term and d is the order of differencing (I) required to make the time series stationary. [5] [6]

Seasonal Autoregressive Integrated Moving Average (SARIMA)

Seasonal Autoregressive Integrated Moving Average (Seasonal ARIMA) is an extension of ARIMA that models the univariate time series data with a seasonal component, which should yield an improvement over ARIMA if the time series data follows an inherent seasonal pattern. SARIMA has 4 additional parameters – P, D and Q which specifies the autoregression (AR), differencing (I) and moving average (MA) for the seasonal component of the time series, and s which is an additional parameter which specifies the period of the seasonality. [7]

Vector Autoregressive Moving Average (VARMA)

Vector Autoregressive Moving Average (VARMA) is a statistical model similar to both ARIMA and SARIMA, but it is used for multivariate time series forecasting, unlike the first two statistical models. VARMA is a combination of the vector auto regression (VAR) model and vector moving average (VMA) model. The VAR model allows VARMA to understand and use the relationship between several variables, unlike AR used in ARIMA and SARIMA. In a VAR model, each variable is a linear function of the past values of itself and the past values of all the other variables:

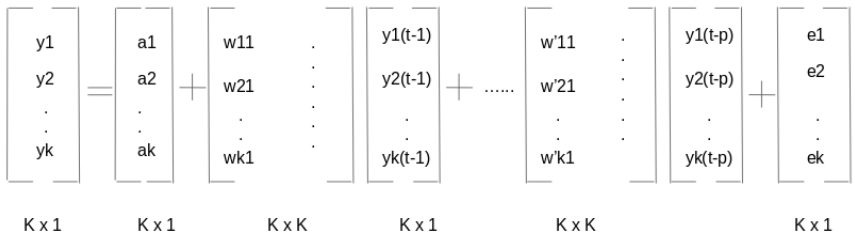

Figure 4 Vector auto regression model

The VMA model is a generalization of the Moving Average (MA) Model for multivariate time series where the time series is stationary. [8]

Holt-Winters

Holt-Winters is a model of time series behaviour, and it attempts to model three different aspects of the time series: a typical value (average), a slope (trend) over time, and a cyclical repeating pattern (seasonality). [9]

Recurrent Neural Network (RNN)

When it comes to sequential or time series data, the conventional feedforward neural networks are not effective for learning and prediction. Recurrent Neural Networks (RNNs) are a variant of the feedforward artificial neural networks that can deal with sequential data by making use of a concept

of “memory” that helps them store the states or information of previous inputs to generate the next output of the sequence. [10] [11]

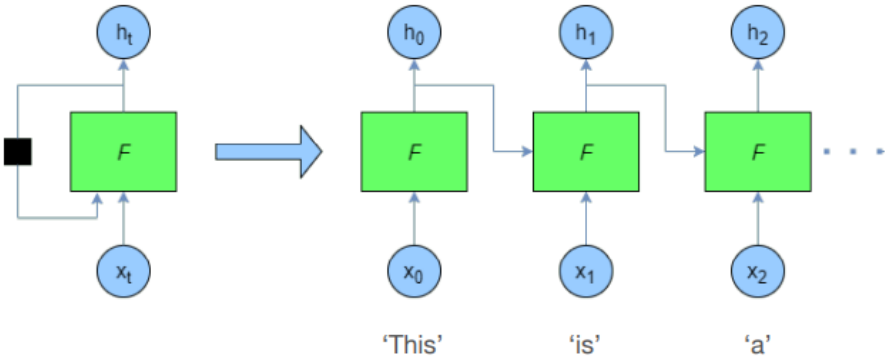

Figure 5 Recurrent neural network architecture

Long Short-Term Memory Network (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network that is capable of learning order dependence in sequence prediction problems. LSTM networks combat the RNN's vanishing gradients by ignoring useless information in the network through the implementation of a ‘forget’ gate. [11]

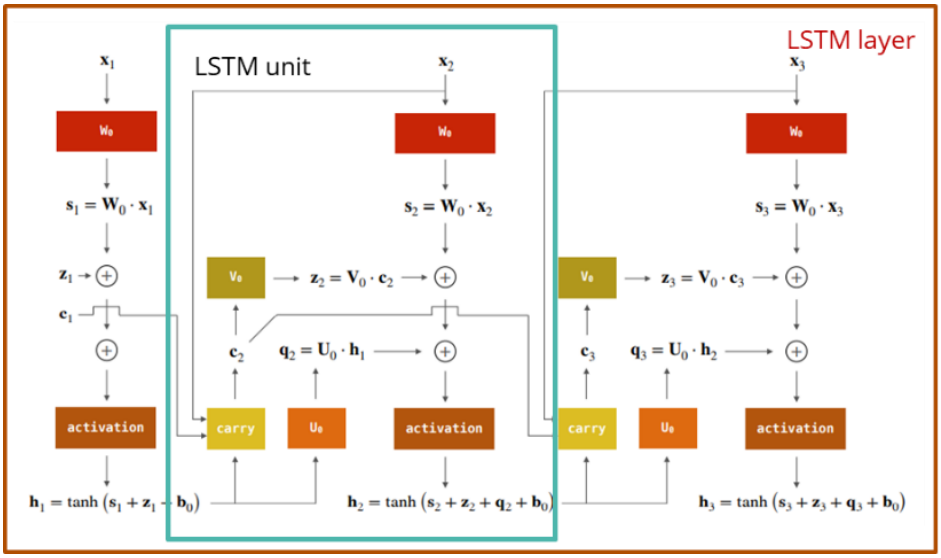

Figure 6 LSTM network architecture

The basic model architecture that we are training comprises of 3 different layers: LSTM, Dropout and Dense. The LSTM Layer(s) processes the sequential time-series data (COE CAT A historical prices) and passes the output values to the Dense Layer which only has one hidden unit as we are training the

model to predict the COE price for the next time step only. The Dropout Layer is used to 'deactivate' random nodes in the LSTM Layer to reduce model overfitting.

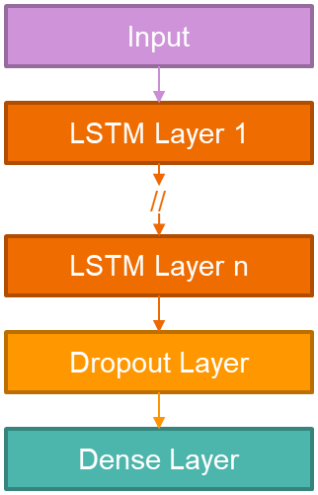

Figure 7 Simplified model architecture with LSTM, Dropout and Dense layers

Bi-directional LSTM

Bi-directional LSTM is a variant of LSTM which consists of a forward and a backward layer. The model learns the input sequence in both the forward and backward direction which allows information from the past and the future to be preserved. [12] This could potentially add important context to the prediction making process.

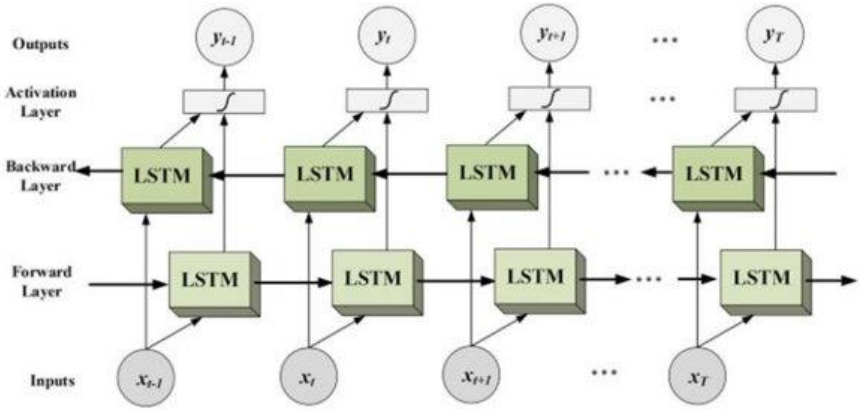

Figure 8 Bi-directional LSTM model architecture

Model Performance Metrics

To select the time series forecasting model which can deliver the most accurate forecast of COE prices for our web application, we need to employ metrics which are able to quantify model performance. Some popular performance metrics for time-series forecasting models include: [13] [13]

- Mean Absolute Error (MAE)
- Root Mean Square Error (RMSE)
- Mean Absolute Percentage Error (MAPE)

Mean Absolute Error (MAE)

The Mean Absolute Error (MAE) is calculated by taking the mean of the absolute differences between the actual values (y) and the predicted values (y-hat):

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Equation 1 Mean absolute error formula

The advantages of using MAE as a performance metric is that it is easy to understand and to compute. It is recommended for assessing accuracy on a single time series, but it is not suitable to use MAE if we are comparing the model performance across different time series as it is scale dependent. [13] Another issue with MAE is that it does not penalize outliers which makes it unsuitable for use cases which cannot afford for the forecasted values to deviate greatly from the actual values. [13]

Root Mean Square Error (RMSE)

To give a higher weightage and penalty to the outliers, we can use RMSE which takes the mean of the squared differences between y and y-hat and taking a square root of the mean squared differences:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Equation 2 Root mean square error formula

However, it has the same disadvantage as MAE metric in that it is not suitable if we are comparing the model performance across different time series due to its scale dependency. [13]

Mean Absolute Percentage Error (MAPE)

As scale dependent metrics such as MAE and RMSE are not suitable for comparing different time series, percentage error metrics can be used as they are scale independent. One popular percentage error metric is the Mean Absolute Percentage Error, which is calculated by taking the average of the absolute difference between actual and predicted values, divided by the actual values and converting it to a percentage by multiplying by 100: [13]

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i} * 100\right|$$

Equation 3 Mean absolute percentage error formula

The advantages of MAPE are that it is easily interpretable and makes it possible to compare model performance across different time series. In our future work, we would like to extend our time series forecasting models to other COE categories – B, C, D and E. Thus, MAPE will be the primary performance metric used in this project. However, MAPE generates infinite or undefined values for zero or close-to-zero actual values. To mitigate this limitation, MAE and RMSE will be used as secondary performance metrics for this project.

Univariate Time Series Forecasting

Data Preparation

The COE master dataset consists of information, which includes quota, premium, total number of bids and total number of successful bids, for all 5 COE categories from year 2002 to 2022. In this dataset, there are a total of 483 data points, each representing the result of a COE bidding exercise, which is held on the first and third week of every month.

Dataset	Description	Frequency	Year	Source
Certificate of Entitlement (COE)	COE bidding results for vehicle Category A, B, C, D and E	Bi-monthly	2002 – 2022	Land Transport Authority

Table 2 Description of dataset(s) used for Univariate Time Series Forecasting

For this project, we have decided to narrow the scope to only look at data points corresponding to the Category A as a Proof-of-Concept for the Minimum Viable Product.

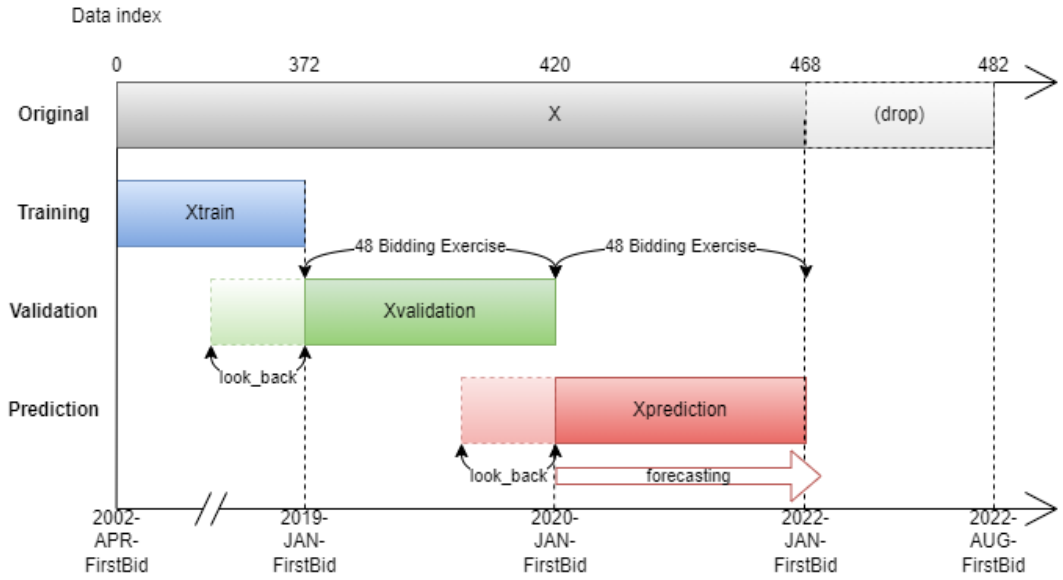

Figure 9 Truncation of master dataset into training, validation and test sets

The original data consists of 483 data points, but we have decided to drop the data points that correspond to the bidding exercises held in 2022 as we will be using additional data such as Gross Domestic Product (GDP), Consumer Price Index (CPI) in the multivariate analysis and modelling segment of this project and these additional data are mostly provided only at the end of the given year or quarter which results in missing data in some of the columns. As this project is focussed on the pattern recognition aspect of time series forecasting models, we decided to keep the master dataset

as clean as possible, which means that we had to drop the 2022 data points. As a result, the original dataset size is reduced from 483 rows to 468 rows.

Next, we truncate the dataset into training, validation and test datasets to best evaluate model performance. As the primary objective of our pattern recognition system is forecast the future COE prices for the next 48 bidding exercises, which is equivalent to 2 years, we decided to allocate 48 data points to the validation dataset and the test dataset each. The preceding 372 data points are used as the training dataset. As we are dealing with time series forecasting, it is important to maintain the sequential nature of the dataset and thus, data points are truncated sequentially as illustrated in the diagram above.

Data Exploration

As stated in the earlier section, the original dataset size has been reduced from 483 to 468 rows. The master dataset for univariate time series forecasting modelling consists of 468 data points of only 1 variable which is the target variable "Quota Premium A". The descriptive statistics for "Quota Premium A" are calculated in the following table.

	count	mean	std	min	25%	50%	75%	max
Quota Premium A	468.0	36975.893162	20008.386179	2.0	18799.75	33004.5	51506.25	92100.0

Figure 10 Descriptive statistics for univariate dataset

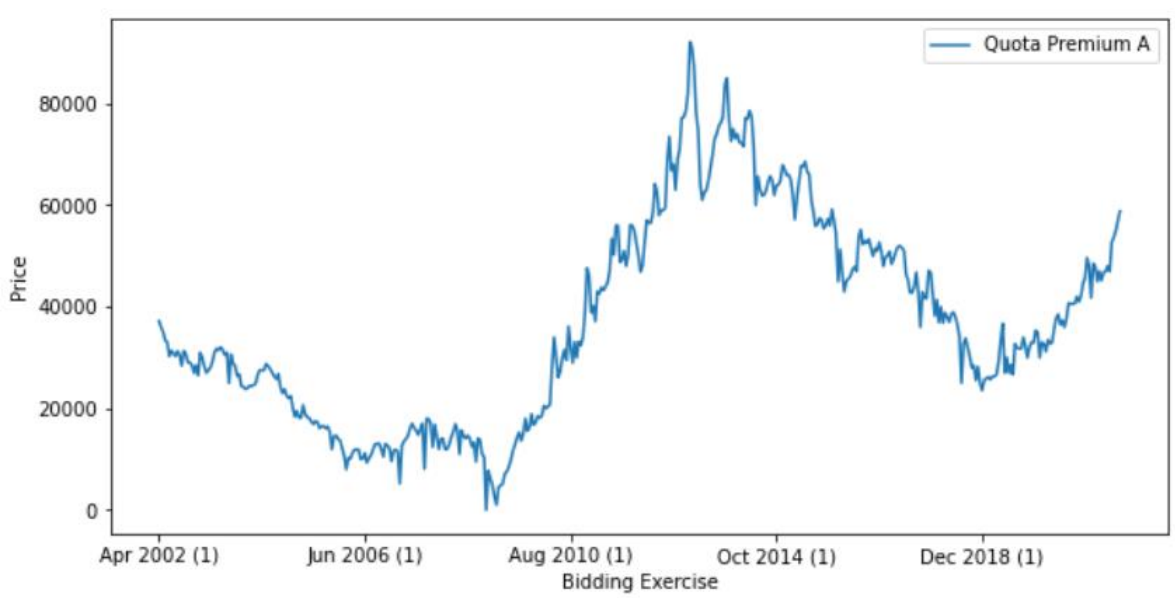

Figure 11 Graph of historical values for quota premium A

Model Training and Evaluation

These are the time-series forecasting models we used for model training on the univariate dataset, which consists of only the historical COE prices for Category A:

- Autoregressive Integrated Moving Average (ARIMA)
- Seasonal Autoregressive Integrated Moving Average (SARIMA)
- Holt-Winters
- Simple Recurrent Neural Network (RNN)
- Long Short-Term Memory Network (LSTM)

Autoregressive Integrated Moving Average (ARIMA)

Firstly, we need to determine a set of values for p, d and q to create the baseline ARIMA model.

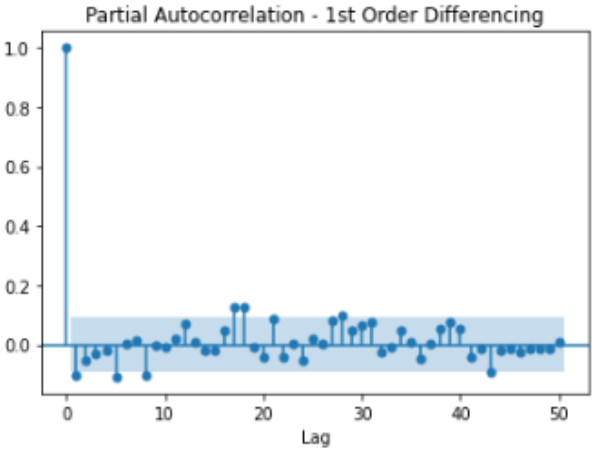

Figure 12 Partial autocorrelation graphical plot

As observed from the Partial Autocorrelation graph, PACF lag 1 is significant since is below the significance line. Therefore, **p=1** is selected.

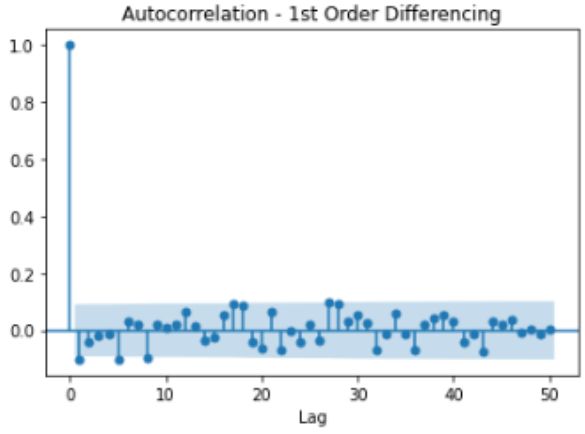

Figure 13 Autocorrelation graphical plot

As observed from the Autocorrelation graph, 2 of the ACF lags exceeded the significance line. Hence, **q=2** is selected.

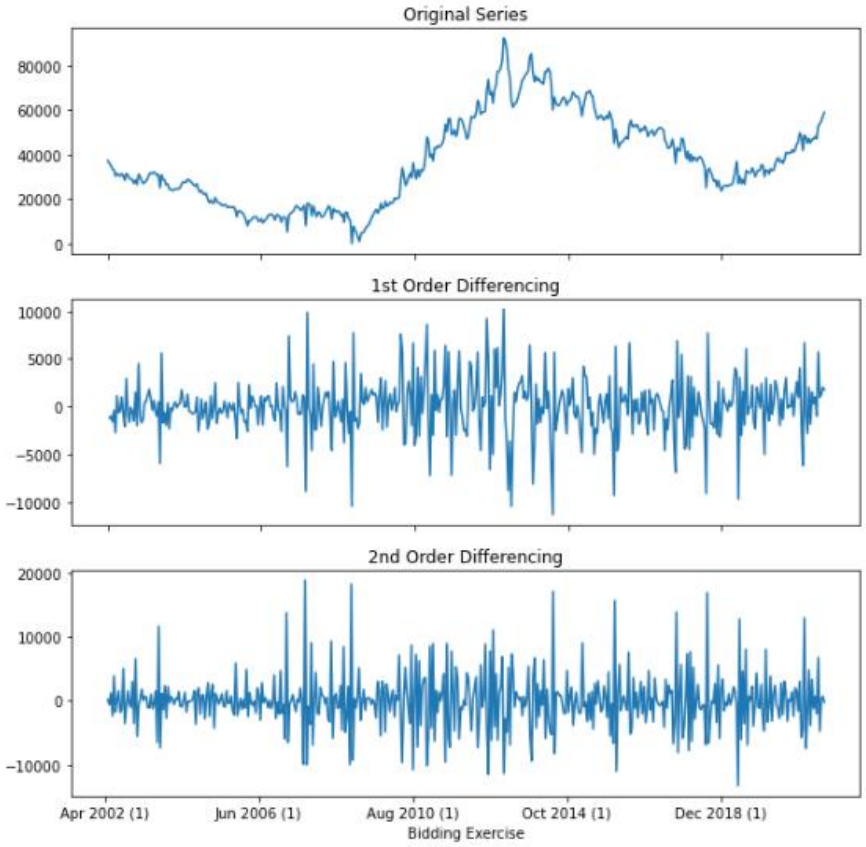

Figure 14 Graphs of Original Series, 1st Order Differencing and 2nd Order Differencing of Quota Premium A

As observed from the plots shown above, the original time-series data for COE CAT A prices only becomes stationary after 2 orders of differencing, which is why we have selected **d=2**

ARIMA Model Results

Dep. Variable:	D2.Quota Premium A	**No. Observations:**	433
Model:	ARIMA(1, 2, 2)	**Log Likelihood**	-4077.979
Method:	css-mle	**S.D. of innovations**	2964.423
Date:	Sat, 24 Sep 2022	**AIC**	8165.958
Time:	10:48:56	**BIC**	8186.311
Sample:	2	**HQIC**	8173.993

	coef	std err	z	P>|z|	[0.025	0.975]
const	0.8515	3.483	0.244	0.807	-5.976	7.679
ar.L1.D2.Quota Premium A	0.7680	0.071	10.882	0.000	0.630	0.906
ma.L1.D2.Quota Premium A	-1.8756	0.050	-37.334	0.000	-1.974	-1.777
ma.L2.D2.Quota Premium A	0.8810	0.050	17.778	0.000	0.784	0.978

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.3020	+0.0000j	1.3020	0.0000
MA.1	1.0645	-0.0444j	1.0654	-0.0066
MA.2	1.0645	+0.0444j	1.0654	0.0066

Figure 15 ARIMA model parameters

Fitting the ARIMA model using the p, d and q values determined from the previous section, the coefficients for the respective Auto Regression and Moving Average terms are significant and the P z value for all three terms are significantly smaller than 0.05 which indicates a theoretical good fit.

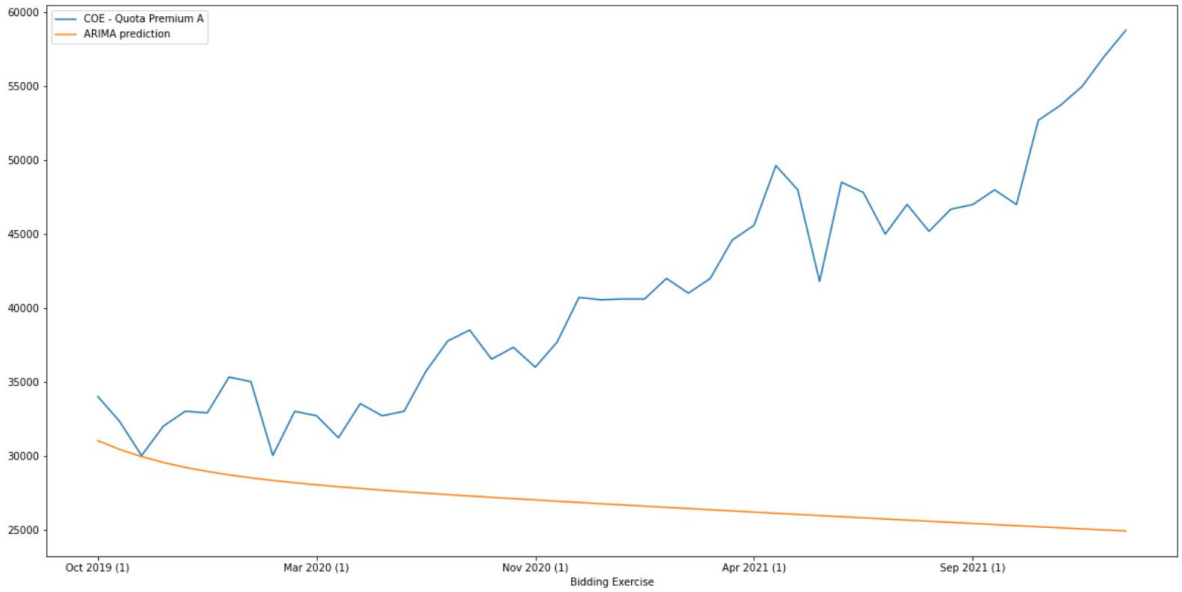

Figure 16 Graph of actual values vs ARIMA-predicted values

Metric	Value
MAE	21174.30
RMSE	17583.81
MAPE	29.56%

Table 3 Performance evaluation metrics for ARIMA model

As observed from the graph and the MAPE of 29.56%, the model did not perform well. Attempts to improve on the ARIMA model involves trying different combinations of p,d and q but that did not yield any improvements.

Seasonal Autoregressive Integrated Moving Average (SARIMA)

To improve on the base ARIMA model, we tried to introduce seasonal terms by using the SARIMA model, short for Seasonal ARIMA. For SARIMA, there are 4 additional terms: seasonal P, Q, D and s. By default, we set the seasonal components P, Q and D to be the same value as p, q and d. s can be estimated by determining the interval between the ACF lags that are significant. Since there is a significant negative lag at 1 and 4, we selected s=4.

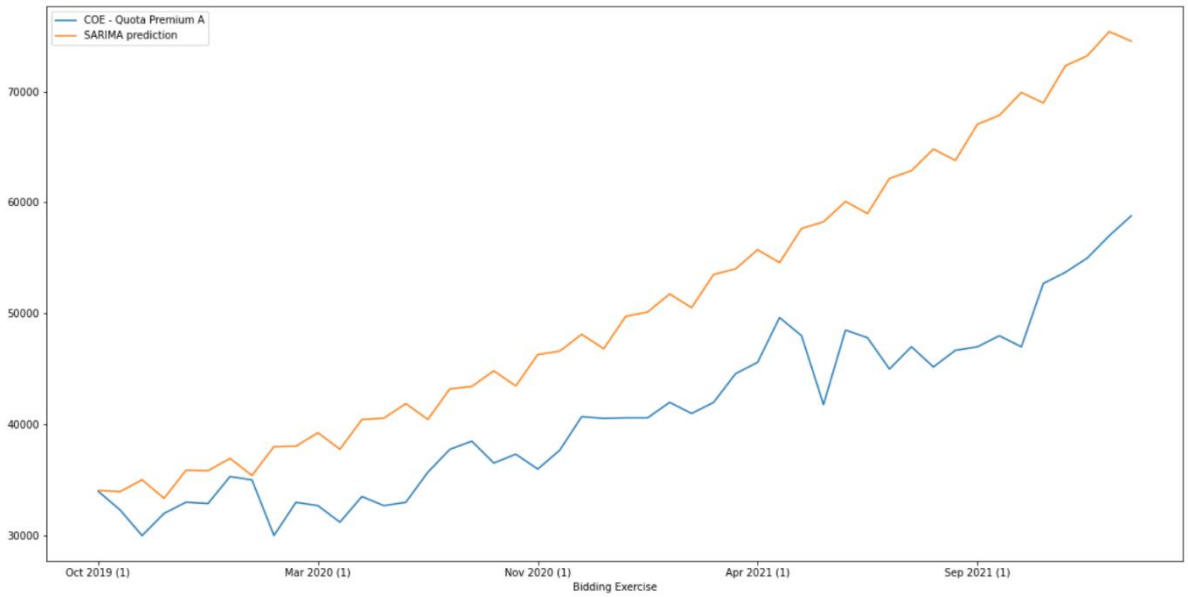

Figure 17 Graph of actual values vs SARIMA-predicted values

Metric	Value
MAE	11439.19
RMSE	9810.83
MAPE	22.76%

Table 4 Performance evaluation metrics for SARIMA model

Looking at the performance metrics of the SARIMA model, we observe that the SARIMA model has a significantly lower MAPE of 22.76% than ARIMA model which has a MAPE of 29.56%, which indicates that SARIMA is significantly better than ARIMA. This is also corroborated by the forecast vs actual graph, where the forecast trend is now in the correct direction and the gap between the forecasted values and actual values are noticeably smaller.

Holt-Winters

As introduced in the earlier section, the Holt-Winters model attempts to model three different aspects of the time series: a typical value (average), a slope (trend) over time, and a cyclical repeating pattern (seasonality). Hence, the first step is to decompose the average, trend, and seasonality from the master dataset:

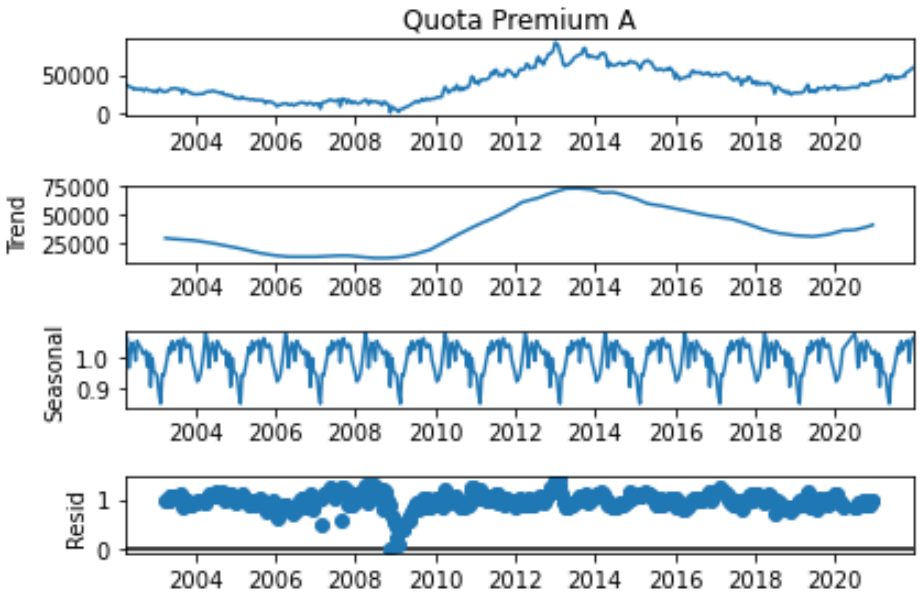

Figure 18 Graphs of the original time series, trend, seasonal and residuals for Quota Premium A

Secondly, we define the value of the average, trend, and seasonality based on chart above. In this case, we set seasonal_period='24'.

Thirdly, we build 4 Holt-winter models across four different orders with the training data. These 4 Holt-winter models are single Holt-winter exponential smoothing (Single-HWES), double Holt-winter exponential smoothing with addition (Double-HWES-1), double Holt-winter exponential smoothing with multiplication (Double-HWES-2), and triple Holt-winter exponential smoothing with multiplication (Triple-HWES).

Forecast data of these 4 models are shown:

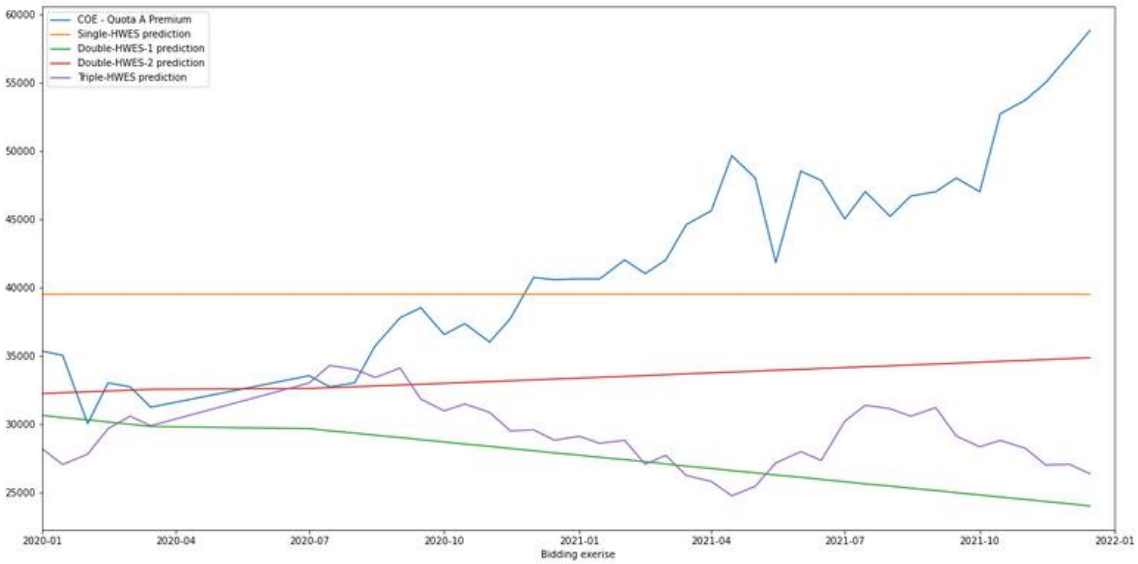

Figure 19 Graph of actual values vs Holt-Winters-predicted values

Metric	Single HWES	Double HWES-1	Double HWES-2	Triple HWES
MAE	**6433.00**	**13258.89**	**7889.79**	**11718.44**
RMSE	**7685.80**	**16369.56**	**10183.34**	**14634.92**
MAPE	**15.44%**	**29.21%**	**17.13%**	**25.79%**

Table 5 Performance evaluation metrics for different Holt-Winters models

In the result above, we obtain both Single HWES, Double HWES-1, and Double HWES-1 are forecasting a linear equation. Only the Triple HWES show the data variation across the time stamp. However, none of the HWES models correctly predict the data trend of the actual data. The accuracy is getting larger follow by the increment of HWES order.

HWES1 has the best MAPE with 15.44% but still under expectation because it is a straight line and error is very large with more than 10%.

Recurrent Neural Network (RNN)

On top of statistical models, we also try to forecast COE result with deep learning model such as RNN with the following details:

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 Input (InputLayer)          [(None, 1, 48)]           0

 simple_rnn (SimpleRNN)      (None, 32)                2592

 dropout (Dropout)           (None, 32)                0

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 2,625
Trainable params: 2,625
Non-trainable params: 0
```

Figure 20 Model summary of the final RNN model

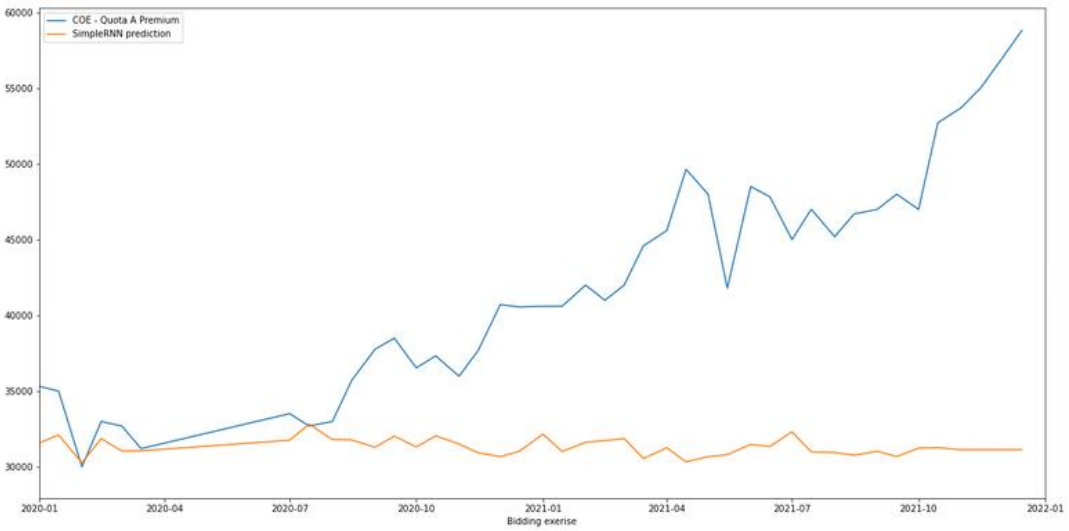

Figure 21 Graph of actual values vs RNN-predicted values

Metric	Value
MAE	10009.39
RMSE	12074.98
MAPE	21.37%

Table 6 Performance evaluation metrics for RNN model

A single-layer SimpleRNN model gives a 21.37% of MAPE.

We obtain the turning points which indicates the data direction. However, the loss increases along the timestamp. On top of this, the data trend is also moving towards opposite direction from the actual COE data.

Long Short-Term Memory Network (LSTM)

There are multiple parameters to tune for LSTM, which includes, segment length, segment distance, segment width, number of LSTM layers, number of hidden units per layer, activation function used and batch size. To establish a baseline, we decided to start with a single-layer LSTM model architecture, and we trained several models using different combinations of parameter values (refer to Appendix A) to optimise the MAPE metric. After several iterations, the following model architecture and parameters yielded the best model performance:

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_10 (LSTM)              (None, 16)                3392

 dropout_10 (Dropout)        (None, 16)                0

 dense_10 (Dense)            (None, 1)                 17

=================================================================
Total params: 3,409
Trainable params: 3,409
Non-trainable params: 0
_________________________________________________________________
```

Figure 22 Model summary of the final single-layer LSTM model

Parameters	Value
Segment Length	36
Segment Distance	36
Segment Width	36
LSTM Layers	1
Number of Hidden Units per Layer	16
Dropout ratio	0.2
Batch Size	4

Table 7 Parameters selected for final single-layer LSTM model

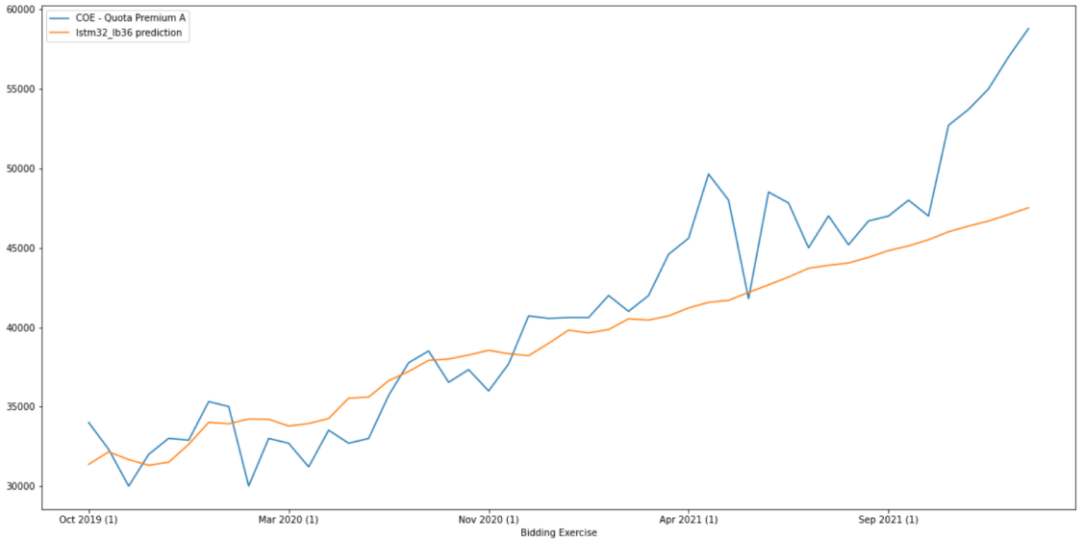

Figure 23 Graph of actual values vs single-layer LSTM-predicted values

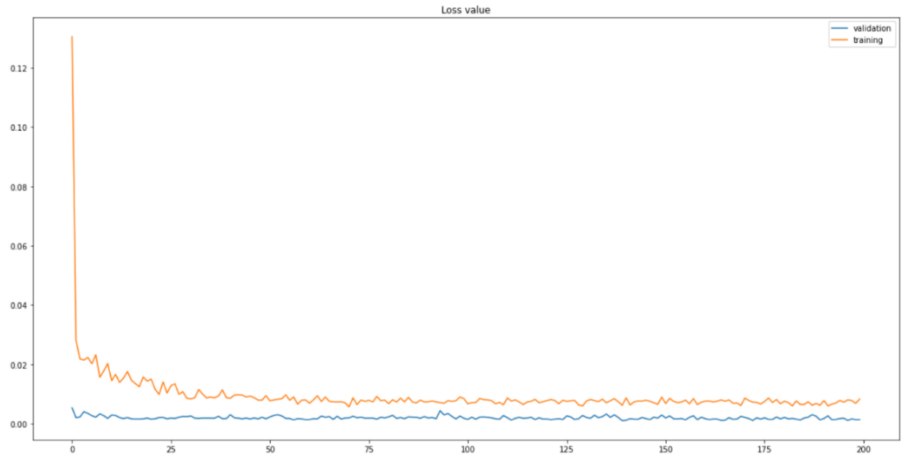

Figure 24 Graph of loss vs epochs for single-layer LSTM model

Metric	Value
MAE	3844.37
RMSE	2787.80
MAPE	6.27%

Table 8 Performance evaluation metrics for single-layer LSTM model

This single-layer LSTM model significantly outperforms all previous statistical and deep learning models, with a low MAPE of 6.27%, which is very apparent by looking at the relatively small differences between the forecasted values versus the actual values in the graphical plot above. It is also interesting to note that the training loss is higher than the validation loss, which can be attributed to the usage of Dropout layers, where a partial network is used to predict values during the training stage but the fully trained network is used to predict values during the validation stage.

In an attempt to improve on the baseline single-layer LSTM model, we added an additional layer of LSTM to the existing model architecture. After multiple iterations with different combinations of parameters (refer to Appendix A), the following model architecture and parameters yielded the best model performance for a stacked LSTM model architecture:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 1, 16)             3392

dropout_2 (Dropout)          (None, 1, 16)             0

lstm_3 (LSTM)                (None, 1, 8)              800

dropout_3 (Dropout)          (None, 1, 8)              0

dense_2 (Dense)              (None, 1, 1)              9

=================================================================
Total params: 4,201
Trainable params: 4,201
Non-trainable params: 0
_________________________________________________________________
```

Figure 25 Model summary of the final stacked LSTM model

Parameters	Value
Segment Length	36
Segment Distance	36
Segment Width	36
LSTM Layers	2
Number of Hidden Units per Layer	16, 8
Dropout ratio	0.2
Batch Size	16

Table 9 Parameters selected for final stacked LSTM model

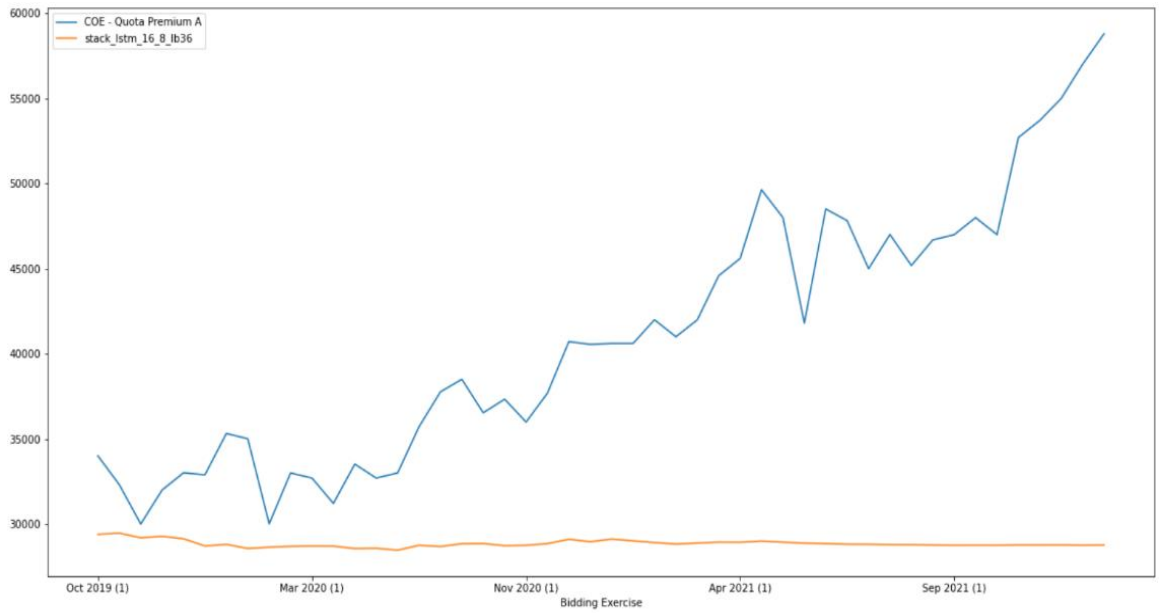

Figure 26 Graph of actual values vs stacked LSTM-predicted values

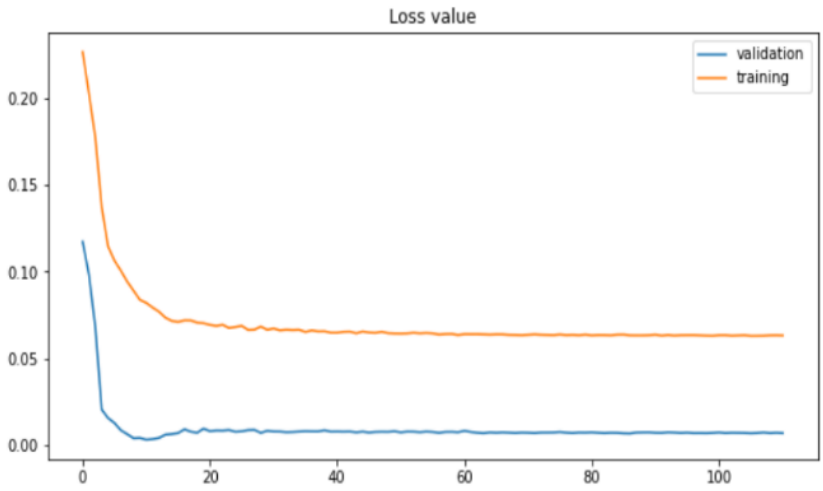

Figure 27 Graph of loss vs epochs for stacked LSTM model

Metric	Value
MAE	**14294.1**
RMSE	**12126.93**
MAPE	**27.25%**

Table 10 Performance evaluation metrics for stacked LSTM model

It is interesting to observe that the model performance of the best stacked LSTM model (MAPE =27.25%) is significantly worse than the baseline single-layer LSTM model (MAPE=6.27%). This indicates that the increase in model complexity is causing the model to be overfitted on the training dataset and does not generalize well, leading to a drop in model performance.

Summary of Model Performance Evaluation (Univariate)

	Models								
Metric	**ARIMA**	**SARIMA**	**Holt-Winters (Single HWES)**	**Holt-Winters (Double HWES-1)**	**Holt-Winters (Double HWES-2)**	**Holt-Winters (Triple HWES)**	**RNN**	**Single-Layer LSTM**	**Stacked LSTM**
MAE	21174.30	11439.19	6433.00	13258.89	7889.79	11718.44	9779.10	3844.37	14294.10
RMSE	17583.81	9810.83	7685.80	16369.56	10183.34	14634.92	12074.98	2787.80	12126.93
MAPE	29.56%	22.76%	15.44%	29.21%	17.13%	25.79%	21.37%	6.27%	27.25%

Table 11 Summary of model performance evaluation metrics for univariate time series forecasting

As observed from the summary table, for univariate time series forecasting on COE Category A prices, the single-layer LSTM model significantly outperformed all the other models, as evident in the huge improvement in terms of MAE, RMSE and MAPE over the other models trained.

Multivariate Time Series Forecasting

After our study on univariate machine learning model, we extended our dataset to include other datasets which may potentially affect COE prices (refer to table below).

Dataset	Description	Frequency	Year	Source
Certificate of Entitlement (COE)	COE bidding results for vehicle Category A, B, C, D and E	Bi-monthly	2002 – 2022	Land Transport Authority
Household income	Average and Median monthly household income from work among resident and resident employed households (per household and per household member)	Yearly	2000 – 2022	Singapore Department of Statistics
Gross domestic product (GDP)	Gross domestic product at current market prices and by industry	Yearly	1960 – 2022	Singapore Department of Statistics
Public transport ridership	Public transportation operation and ridership (Bus, MRT, LRT)	Yearly	1990 – 2022	Singapore Department of Statistics
Personal Disposable income (PDI)	Personal disposable income and personal saving	Quarterly	1980 – 2022	Singapore Department of Statistics
Household Net Worth	Household net worth, assets and liabilities	Quarterly	1995 – 2022	Singapore Department of Statistics
Consumer price index (CPI)	Consumer price index and inflation rates	Monthly	1990 – 2022	Singapore Department of Statistics
Oil Price	Crude oil West Texas Intermediate (WTI) futures data	Weekly	2002 – 2022	Investing.com

Table 12 Datasets collected for COE forecasting

Data Preparation

As the datasets are of different frequencies, we pre-process all the datasets to use the same frequency (bi-monthly) as the COE dataset. Missing values are handled by either filling in with the same value within the same period (i.e. year, quarter or month) or extracting values which are closest to the 1st or 15th of each month. The pre-processed dataset is then merged into a single dataset.

This master dataset consists of 468 rows and 70 columns (excluding features specific to other COE categories) with following summary statistics:

	count	mean	std	min	25%	50%	75%	max
Quota A	468.0	1424.450855	701.987701	333.0	706.00	1442.0	2025.50	2858.0
Quota Premium A	468.0	36975.893162	20008.386179	2.0	18799.75	33004.5	51506.25	92100.0
Total Bids Received A	468.0	1986.976496	977.418103	418.0	1126.25	2021.5	2610.25	5927.0
Number of Successful Bids A	468.0	1413.764957	700.735576	324.0	703.75	1434.0	1982.50	2858.0
Resident Households Average Per Pax	468.0	2689.987179	673.138213	1713.0	2136.00	2752.0	3289.00	3676.0
...	...	...	...	...	...	...	...	...
LRT km Operated (Thousand Car-Kilometres)	468.0	4739.000000	1669.509061	3303.0	3303.00	3751.5	6620.00	7566.0
Average Daily Ridership - MRT (Thousand Passenger-Trips)	468.0	2206.871795	708.991310	1321.0	1527.00	2100.0	2871.00	3384.0
Average Daily Ridership - LRT (Thousand Passenger-Trips)	468.0	122.000000	45.976905	69.0	79.00	117.5	153.00	208.0
Average Daily Ridership - Bus (Thousand Passenger-Trips)	468.0	3324.192308	475.345421	2779.0	2878.00	3199.0	3891.00	4099.0
Average Daily Trip - Point-To-Point (P2P) Transport (Taxis And Private Hire Cars) (Thousand Daily-Trips)	468.0	736.371795	62.809615	516.0	755.00	755.0	755.00	772.0

70 rows × 8 columns

Figure 28 Summary Statistics of master dataset

Data Exploration

A pairwise correlation heatmap is plotted and the result shows that the dataset contains highly correlated features. This suggests that we could apply dimension reduction as our next step.

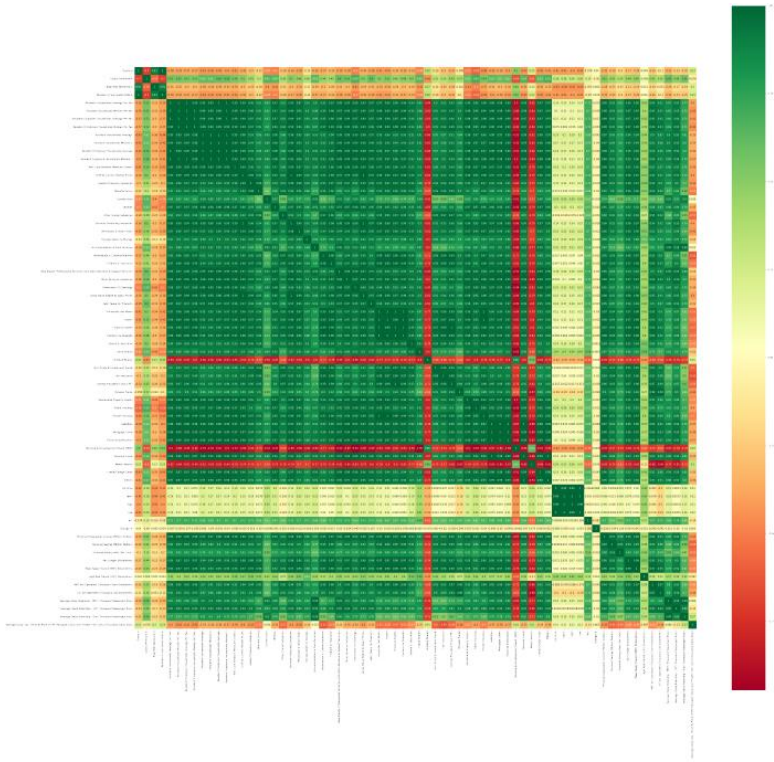

Figure 29 Correlation heatmap of master dataset

Dimension Reduction – Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a technique which transforms data from a high dimension space into a lower dimension space while preserving as much information as the original data as possible. This is achieved by finding a new axis that captures the maximum variance within the data once it is projected onto the new axis. [14]

Bartlett's Test of Sphericity (BS) and Kaiser-Meyer-Olkin (KMO) test are commonly used to determine whether a dataset is suitable to undergo dimension reduction. In Bartlett's Test of Sphericity, the null hypothesis is that the correlation matrix in an identity matrix (i.e. the variables are not related) and thus not suitable for dimension reduction. Therefore, we would want to reject this null hypothesis if p-value < 0.05. KMO test is a measure of sampling adequacy. As a rule of thumb, $0 \leq KMO < 0.5$ indicates that the sampling is not adequate while $0.5 \leq KMO \leq 1$ indicates that the sampling is adequate. [14]

Both tests are conducted to find out whether the dataset is suitable for PCA and the test results are shown below:

Dataset includes target column 'Quota Premium A':

```
BS test passed for category A with p_value = 0.0
KMO test fail for category A with value = nan
```

Dataset excludes target column 'Quota Premium A':

```
BS test passed for category A with p_value = 0.0
KMO test passed for category A with value = 0.756159735266312
```

Since the dataset excluding 'Quota Premium A' passes both tests, we'll perform PCA on the dataset excluding 'Quota Premium A'.

According to Kaiser Rule, principal component eigenvalue should be greater than 1 and this suggest that the number of principal components we could use should be less than or equals to 5.

Since 1 or 2 principal components can explain around 78% - 86% of the variance, we decided to try out with 1 to 2 principal components.

```
Cat A Eigenvalues         : [5.4015e+01 5.9070e+00 2.8410e+00 2.4420e+00 1.0150e+00 6.3800e-01
 4.9300e-01 4.7000e-01 2.6400e-01 2.4400e-01 1.7700e-01 1.3500e-01
 1.0900e-01 8.1000e-02 6.2000e-02 5.0000e-02 4.4000e-02 2.7000e-02
 2.6000e-02 2.1000e-02]
Cat A % Variance Explained: [7.8116e+01 8.5420e+00 4.1080e+00 3.5320e+00 1.4680e+00 9.2300e-01
 7.1300e-01 6.7900e-01 3.8100e-01 3.5300e-01 2.5600e-01 1.9500e-01
 1.5800e-01 1.1800e-01 8.9000e-02 7.2000e-02 6.4000e-02 4.0000e-02
 3.7000e-02 3.1000e-02]
Cat A % Cumulative Var Explained: [78.116 86.658 90.766 94.298 95.766 96.689 97.402 98.081 98.462 98.815
 99.071 99.266 99.424 99.542 99.631 99.703 99.767 99.807 99.844 99.875]
```

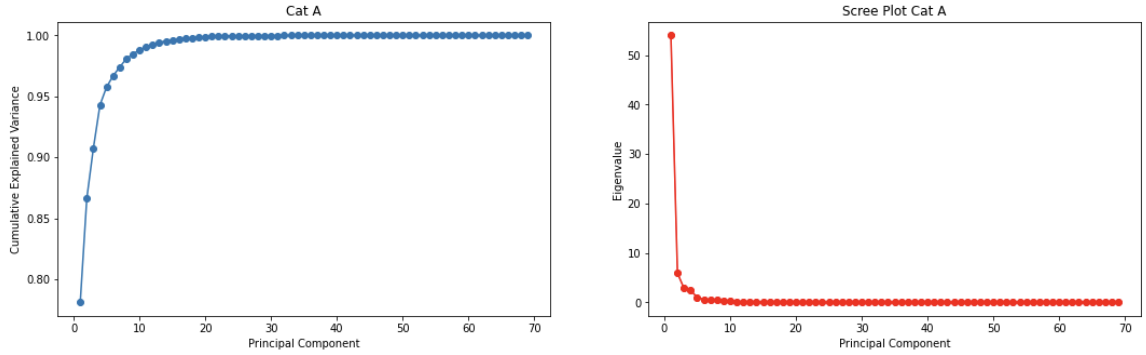

Figure 30 Graph of cumulative explained variance vs principal component and eigenvalues vs principal component

Inclusion of historical 'Quota Premium A' feature

As the historical 'Quota Premium A' data could also provide useful contextual information, we also experiment with adding the standardised 'Quota Premium A' data as one of the lag features when training the model.

PC_1	...	PC_n	Lag feature Quota Premium A (standardised)
x.xxx	...	x.xxx	x.xxx
⋮	⋮	⋮	⋮

Figure 31 Adding 'Quota Premium A' to the feature set

Segment preparation

The feature set is split into segments for input into the model for training. Each segment having shape (*b*, *n*), where b is the "lookback" period and *n* is the number of features.

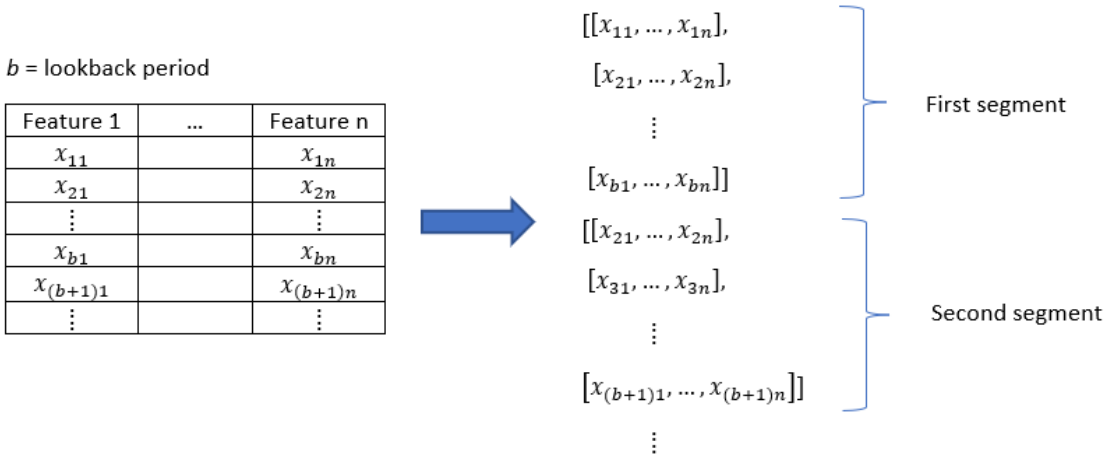

Figure 32 Splitting feature set into segments

Model Training and Evaluation

As we did for univariate time series forecasting, we also experimented with both statistical and machine learning models for multivariate time series forecasting:

- Vector Autoregressive Moving Average (VARMA)
- Long Short-Term Memory Network (LSTM)

Vector Autoregressive Moving Average (VARMA)

For the baseline VARMA model, we decided to select two PCs as the feature set which resulted in the following input dataset:

Bidding Exercise	PC1	PC2	COE - Quota Premium A
Apr 2002 (1)	-11.384867	-2.906990	37201
Apr 2002 (2)	-11.382702	-2.652822	36000
May 2002 (1)	-11.356144	-2.566344	35000
May 2002 (2)	-11.333255	-2.484609	33401
Jun 2002 (1)	-11.352310	-2.622939	33009
...	...	...	...
Jul 2019 (2)	9.675561	-1.681277	26667
Aug 2019 (1)	9.922495	-1.589211	32725
Aug 2019 (2)	9.763148	-1.366921	31917
Sep 2019 (1)	9.844759	-1.336438	31783
Sep 2019 (2)	9.834322	-0.965025	31759

Figure 33 Train dataset with 2 principal components for VARMA

The dataset shown above is used to train the VARMA model and the trained VARMA model is used to predict future values against the test set.

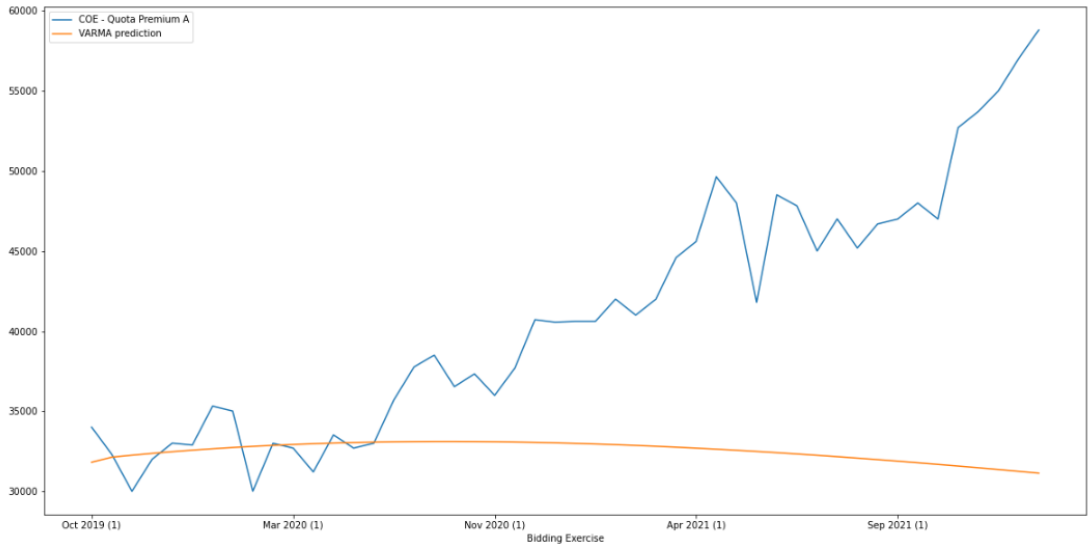

Figure 34 Graph of actual values vs VARMA predicted values

Metric	Value
MAE	11603.73
RMSE	8790.22
MAPE	18.94%

Table 13 Performance evaluation metrics for VARMA

Model predictions are very close to the actual values for the first one-third of the test set but starts to diverge as the bidding exercises progresses. MAPE of the trained VARMA model is also quite significant at 18.94%, which indicates that it is performing relatively poorly.

Therefore, we tried to improve on the VARMA model by tweaking the input dataset. Instead of using the principal components that are derived from the PCA analysis, we decided to do a manual feature selection exercise. After trying out different combinations of features selected, it was found that VARMA model performs the best when features 'Quota A' and 'Total Bids Received A' are used in the input dataset.

Bidding Exercise	Quota Premium A	Quota A	Total Bids Received A
Apr 2002 (1)	37201	1110	2484
Apr 2002 (2)	36000	1149	1890
May 2002 (1)	35000	1128	1705
May 2002 (2)	33401	1111	1320
Jun 2002 (1)	33009	1111	1382
...	...	...	...
Jul 2019 (2)	26667	1623	1902
Aug 2019 (1)	32725	1059	2023
Aug 2019 (2)	31917	1062	1725
Sep 2019 (1)	31783	1056	1656
Sep 2019 (2)	31759	1056	1453

Figure 35 Train dataset with 2 manually selected features for VARMA

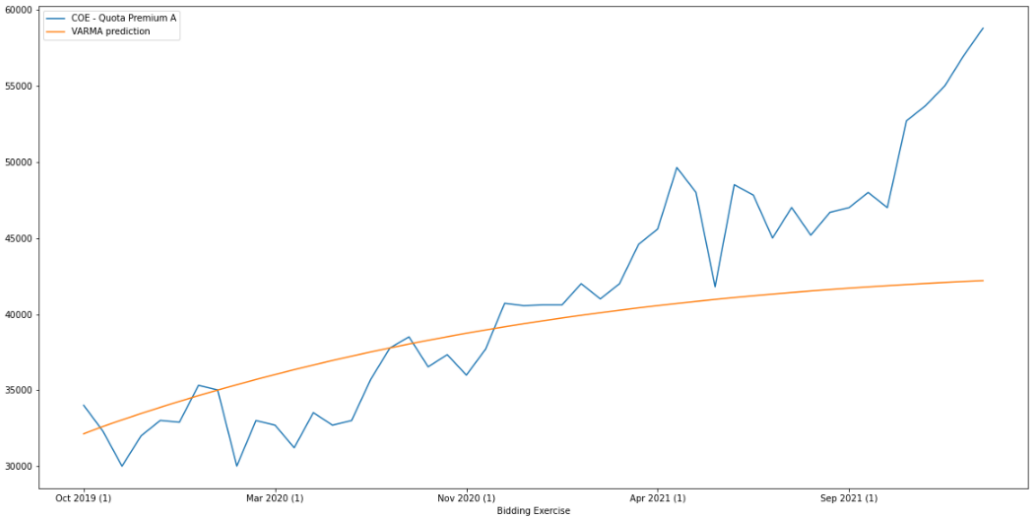

Figure 36 Graph of actual values vs VARMA predicted values

Metric	Value
MAE	5655.44
RMSE	4125.50
MAPE	9.26%

Table 14 Performance evaluation metrics for VARMA

As observed by the graphical plots and the significantly lower MAPE value of 9.26%, the VARMA model performance has improved significantly after the set of features used in the input dataset has changed to manually selected features.

Long Short-Term Memory Network (LSTM)

We experimented with different variant of LSTM (namely single-layer LSTM, stacked LSTM and bi-directional LSTM) together with different combinations of parameters with our multivariate dataset (refer to appendix B for the different combinations tried and their respective performance). The following model naming convention is used to indicate the different parameters used to train the model.

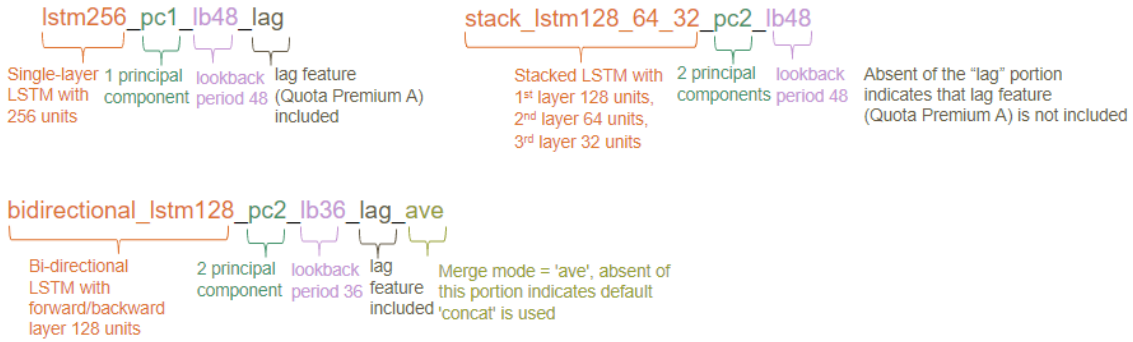

Figure 37 Model naming convention

Multi-step (single-shot) approach with batch size = 32, dropout rate = 0.2, activation function = tanh is used to train the model and the best performing model (in terms of MAPE) is shown below.

LSTM (single layer)

Model name: lstm256_pc1_lb48_lag

Total number of features: 2

```
Model: "lstm256_pc1_lb48_lag"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_9 (LSTM)               (None, 256)               265216

 dropout_9 (Dropout)         (None, 256)               0

 dense_6 (Dense)             (None, 48)                12336

=================================================================
Total params: 277,552
Trainable params: 277,552
Non-trainable params: 0
_________________________________________________________________
```

Figure 38 Model summary of single-layer LSTM (multivariate)

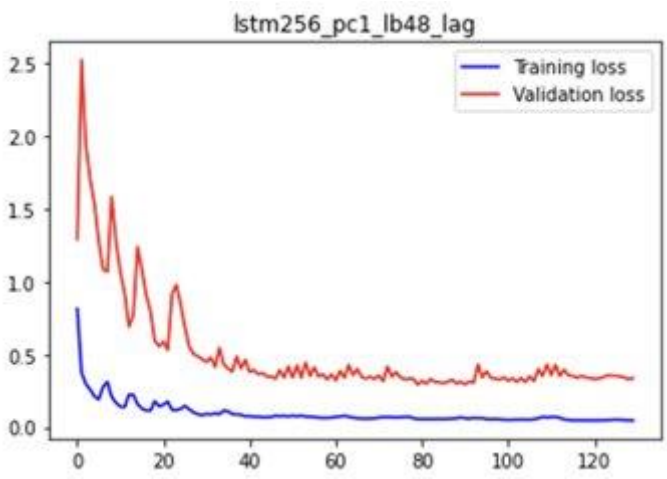

Figure 39 Graph of training vs validation loss for single-layer LSTM (multivariate)

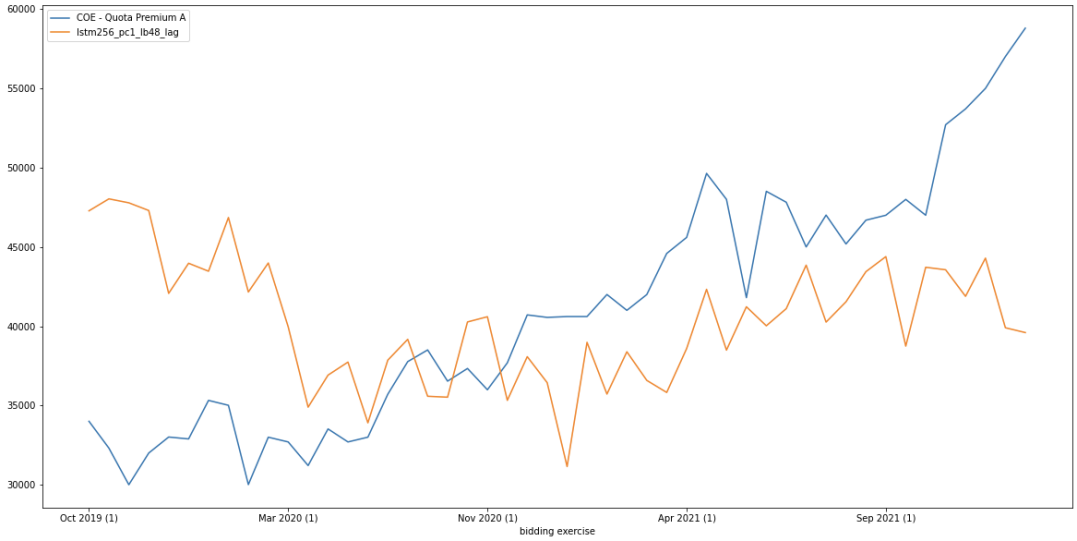

Figure 40 Graph of actual values vs single-layer LSTM predicted values (multivariate)

Metric	Value
MAE	7114.61
RMSE	8614.47
MAPE	17.16%

Table 15 Performance evaluation metrics for single-layer LSTM (multivariate)

From the graph, we can see that the difference between predicted value and the actual value is quite large at first but subsequently the predicted values are getting closer to the actual value before diverging near the end again.

From the performance result, multivariate single-layer LSTM (MAPE 17.16%) did not manage to outperform univariate single-layer LSTM (MAPE 6.27%). Thus, we try adding more layers to see if the performance can be improved.

Stacked LSTM

Model name: stack_lstm128_64_32_pc2_lb48

Total number of features = 2

```
Model: "stack_lstm128_64_32_pc2_lb48"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_3 (LSTM)               (None, 48, 128)           67072

 dropout_3 (Dropout)         (None, 48, 128)           0

 lstm_4 (LSTM)               (None, 48, 64)            49408

 dropout_4 (Dropout)         (None, 48, 64)            0

 lstm_5 (LSTM)               (None, 32)                12416

 dropout_5 (Dropout)         (None, 32)                0

 dense_3 (Dense)             (None, 48)                1584

=================================================================
Total params: 130,480
Trainable params: 130,480
Non-trainable params: 0
_________________________________________________________________
```

Figure 41 Model summary of stacked LSTM (multivariate)

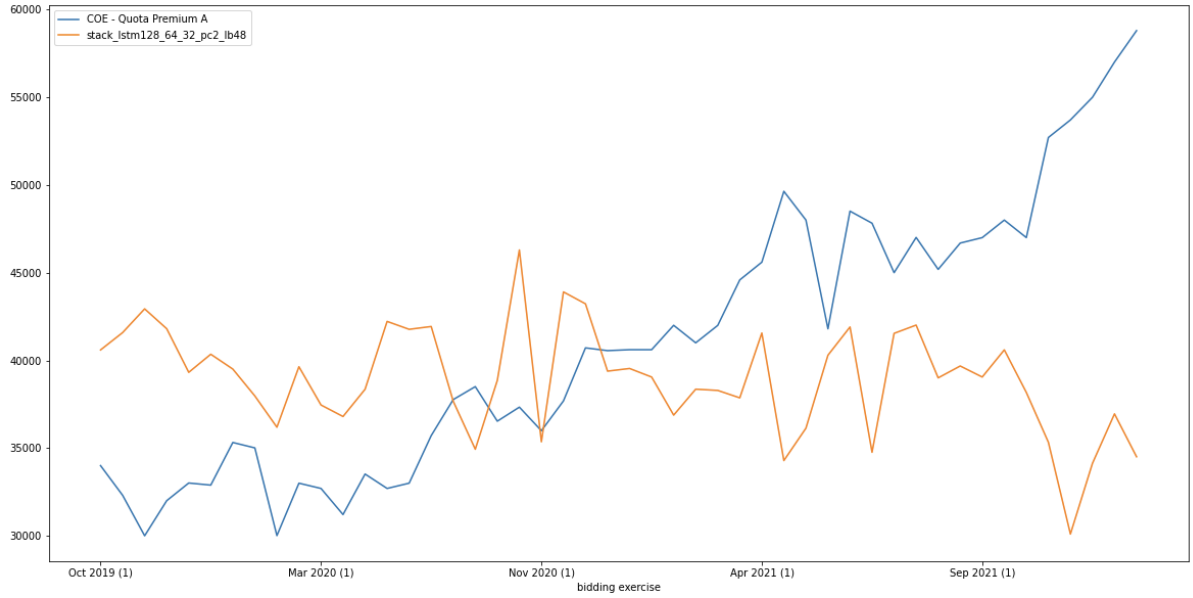

Figure 42 Graph of actual values vs stacked LSTM predicted values (multivariate)

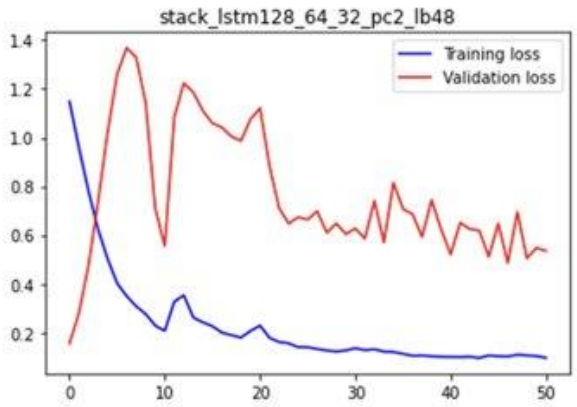

Figure 43 Graph of training vs validation loss for stacked LSTM (multivariate)

Metric	Value
MAE	7558.98
RMSE	9507.52
MAPE	16.22%

Table 16 Performance evaluation metrics for stacked LSTM (multivariate)

From the graph, we can see that the stacked LSTM predicted values for the first half is closer to the actual value than single-layer LSTM but it subsequently diverges from the actual value.

From the performance results, adding additional layers does not seem to have significant improvement in the model performance. Lastly, we will try a variant of LSTM, the bi-directional LSTM to see if there could be further performance improvement.

Bi-directional LSTM

Model name: bidirectional_lstm128_pc2_lb36_lag_ave

Total number of features: 3

Merge mode = 'ave'

```
Model: "bidirectional_lstm128_pc2_lb36_lag_ave"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bidirectional (Bidirectiona  (None, 128)              135168
 l)

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 48)                6192

=================================================================
Total params: 141,360
Trainable params: 141,360
Non-trainable params: 0
_________________________________________________________________
```

Figure 44 Model summary of bi-directional LSTM (multivariate)

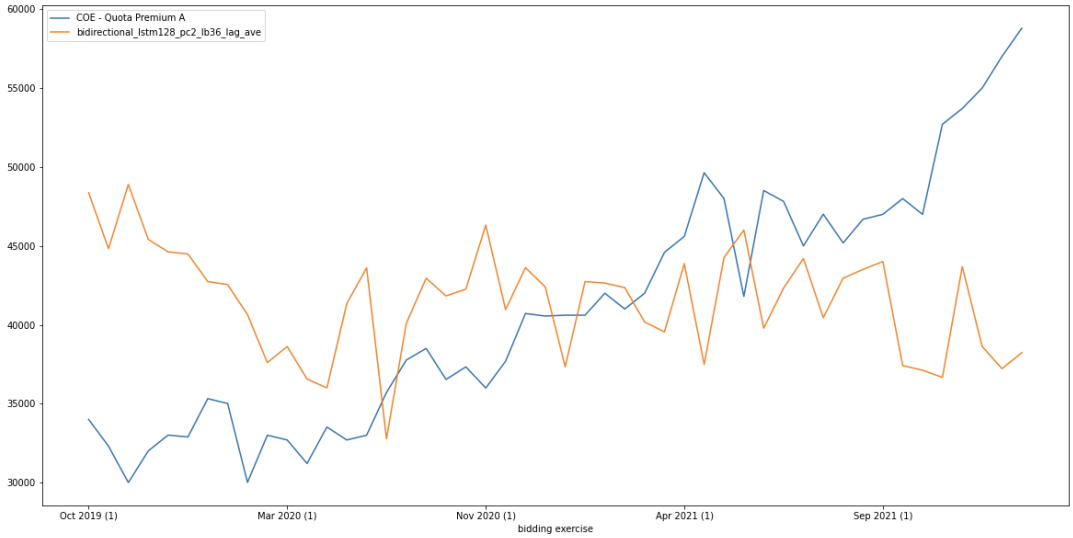

Figure 45 Graph of actual values vs bi-directional LSTM predicted values (multivariate)

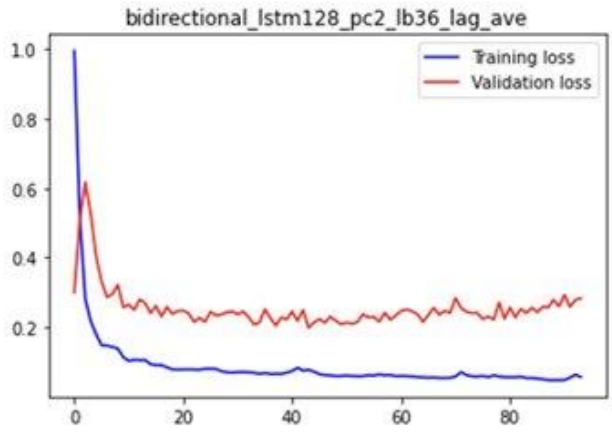

Figure 46 Graph of training vs validation loss for bi-directional LSTM (multivariate)

Metric	Value
MAE	7358.82
RMSE	9061.10
MAPE	17.37%

Table 17 Performance evaluation metrics for bi-directional LSTM (multivariate)

From the graph, we can see that the bi-directional LSTM has a similar trend as the single-layer LSTM (i.e. the difference between the predicted value and the actual value is also large at first subsequently the predicted values are getting closer to the actual value and starts to diverge near the end again). This shows that the additional backward layer did not really help to improve the performance in this case.

From the performance results, bi-directional LSTM does not perform better than single-layer or stacked LSTM.

Summary of Model Performance Evaluation (Multivariate)

	Models			
Metric	**VARMA**	**Single-Layer LSTM**	**Stacked LSTM**	**Bi-directional LSTM**
MAE	5655.44	7114.61	7558.98	7358.82
RMSE	4125.50	8614.47	9507.52	9061.10
MAPE	9.26%	17.16%	16.22%	17.37%

Table 18 Summary of model performance evaluation metrics for multivariate time series forecasting

For multivariate time series forecasting, the VARMA model has outperformed the LSTM models with the lowest MAPE of 9.26%.

Summary of Model Performance Evaluation (Univariate and Multivariate)

	Models												
	Univariate									Multivariate			
Metric	ARIMA	SARIMA	Holt-Winters (Single HWES)	Holt-Winters (Double HWES-1)	Holt-Winters (Double HWES-2)	Holt-Winters (Triple HWES)	RNN	Single-Layer LSTM	Stacked LSTM	VARMA	Single-Layer LSTM	Stacked LSTM	Bi-directional LSTM
MAE	21174.30	11439.19	6433.00	13258.89	7889.79	11718.44	9779.10	3844.37	14294.10	5655.44	7114.61	7558.98	7358.82
RMSE	17583.81	9810.83	7685.80	16369.56	10183.34	14634.92	12074.98	2787.80	12126.93	4125.50	8614.47	9507.52	9061.10
MAPE	29.56%	22.76%	15.44%	29.21%	17.13%	25.79%	21.37%	6.27%	27.25%	9.26%	17.16%	16.22%	17.37%

Table 19 Summary of model performance evaluation metrics for all models trained

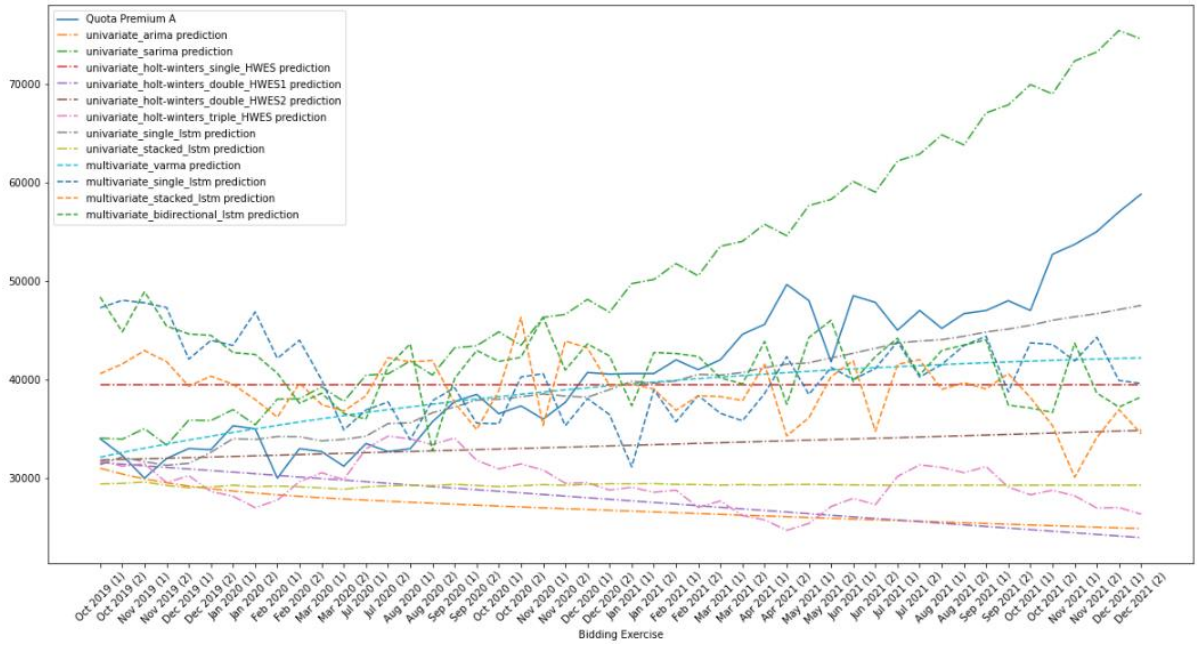

Figure 47 Actual vs forecasted values for all models

Amongst all the models trained, including both univariate and multivariate, the univariate single-layer LSTM has performed the best, with a lowest MAPE or 6.27%. Therefore, this univariate single-layer LSTM model is used in the backend application to do the forecasting of COE Category A prices for the next 48 bidding exercises.

Test Results and Evaluation

The developed Intelligent system was subjected to testing from end user/intended audience perspective. One cycle of User Acceptance Test (UAT) had been performed to look for unidentified problems in previous rounds of testing. Test scenarios for UAT testing covered as many functional cases as possible that end users would face. No bugs or issues were reported at the end of UAT testing and the system developed is tested to have met the end users' requirements.

The following are the sample of testcases performed on the application for evaluation:

No.	Test Description	Expected result	Test status
1	Input the following test data in the Budget, start date, end date, car brand and car type fields. Budget =**200000**, Start date =**Jan 2023**, End date =**Apr 2023** Car brand= **Honda**, Car type = **Hatch back**	List of recommended cars (3) with picture and price information which correctly matches the input appears on the application screen.	Pass
2	Input the following test data in the Budget, start date, end date, car brand and car type Budget =**180000**, Start date =**Feb 2024**, End date =**Mar 2024** Car brand= **Kia** Car type = **SUV** and **Sedan**	List of recommended cars (5) with picture and price information which correctly matches the input appears on the application screen.	Pass
3	Input the following test data in the budget, start date, end date, car brand and car type Budget =**160000**, Start date =**Nov 2024**, End date =**Dec 2024** Car brand= **Hyundai** Car type = **Hatchback**	Message: 'No cars matching your filter criteria found' is displayed on screen	Pass
4	Input the following test data in the budget, start date, end date, car brand and car type Budget =**120000**, Start date =**Nov 2024**, End date =**Dec 2024** Car brand= **Honda** and **Toyota** Car type = **Hatchback**	List of recommended cars (3) with picture and price information which correctly matches the input appears on the application screen.	Pass
5	Input the following test data in the budget, start date, end date, car brand and car type Budget = **0**, Start date =**Aug 2023**, End date =**Dec 2023** Car brand= **Honda** and **Toyota** Car type = **Hatchback**	Message: 'Budget should be positive' is displayed	Pass

Table 20 User Acceptance Testing Results

Future Work

To extend the functionality of the web application, the time series forecasting models used for the forecasting of COE Category A prices can be extended and optimised to do forecasting for all the other COE categories (B, C, D and E).

Secondly, model performance is also a potential area of improvement for future versions of our web application. This can be accomplished by experimenting with even more combinations of model hyperparameters which would require more time and computing resources. This can also be accomplished by experimenting with more advanced deep learning models such as attention mechanism models.

The web application functionality can also be expanded to embed links to car dealers' websites for users to explore and purchase the cars that are recommended to them. We can also collaborate with the car dealers to give exclusive promotions to users of the web application in exchange for promoting their websites.

Finally, to ensure that the web application is well-maintained and up to date, we can implement a web crawler that automatically search the web for the latest car models and prices and automatically updates the car database. Also, the forecasting models should also be re-trained on new data using an automated script after every COE bidding exercise.

Conclusion

In conclusion, we have successfully built a web application which is able to predict future COE prices and provide recommendations based on users' preferences and budget to help users make informed financial decisions when purchasing a new car as predicting future COE prices in the current high-inflationary environment is not an easy task and the fear of increasing prices induces the average consumer to make potentially rash financial decisions.

After training several models and optimizing the model performance by tweaking the models' hyperparameters and input feature set, the final forecasting model selected and embedded in the application's backend is capable of predicting future prices with relatively high confidence, having a reasonably low mean absolute percentage error (MAPE) of 6.27%. The backend system was also interfaced seamlessly with the frontend system, rendering the model predictions and list of recommended cars on the web interface with minimal latency.

References

[1] Hedges Company, “HOW MANY LICENSED DRIVERS ARE THERE IN THE US?,” [Online]. Available: https://hedgescompany.com/blog/2018/10/number-of-licensed-drivers-usa/. [Accessed 4 November 2022].

[2] Budget Direct Insurance, “A comprehensive guide to car ownership in Singapore,” [Online]. Available: https://www.budgetdirect.com.sg/car-insurance/research/car-ownership-singapore. [Accessed 4 November 2022].

[3] Wikipedia, “Certificate of Entitlement,” [Online]. Available: https://en.wikipedia.org/wiki/Certificate_of_Entitlement. [Accessed 3 November 2022].

[4] Tableau, “Time Series Forecasting: Definition, Applications, and Examples,” [Online]. Available: https://www.tableau.com/learn/articles/time-series-forecasting#:~:text=Time%20series%20forecasting%20occurs%20when,drive%20future%20strategic%20decision%2Dmaking.. [Accessed 2 November 2022].

[5] ScienceDirect, “Autoregressive Integrated Moving Average,” [Online]. Available: https://www.sciencedirect.com/topics/mathematics/autoregressive-integrated-moving-average. [Accessed 15 September 2022].

[6] Machine Learning +, “ARIMA Model – Complete Guide to Time Series Forecasting in Python,” [Online]. Available: https://www.machinelearningplus.com/time-series/arima-model-time-series-forecasting-python/. [Accessed 15 September 2022].

[7] Machine Learning Mastery, “A Gentle Introduction to SARIMA for Time Series Forecasting in Python,” [Online]. Available: https://machinelearningmastery.com/sarima-for-time-series-forecasting-in-python/. [Accessed 3 November 2022].

[8] Analytics Vidhya, “A Multivariate Time Series Guide to Forecasting and Modeling (with Python codes),” [Online]. Available: https://www.analyticsvidhya.com/blog/2018/09/multivariate-time-series-guide-forecasting-modeling-python-codes/. [Accessed 14 October 2022].

[9] SolarWinds, “Holt-Winters Forecasting and Exponential Smoothing Simplified,” [Online]. Available: https://orangematter.solarwinds.com/2019/12/15/holt-winters-forecasting-simplified/. [Accessed 3 November 2022].

[10] Machine Learning Mastery, “An Introduction to Recurrent Neural Networks and the Math That Powers Them,” [Online]. Available: https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/. [Accessed 4 November 2022].

[11] T. Jen Hong, “Module 7 - Solving temporal sequential problems using recurrent neural networks,” National University of Singapore, Singapore, 2022.

[12] AIM, “Complete Guide To Bidirectional LSTM,” [Online]. Available: https://analyticsindiamag.com/complete-guide-to-bidirectional-lstm-with-python-codes/. [Accessed 1 November 2022].

[13] Towards Data Science, “Time Series Forecast Error Metrics You Should Know,” [Online]. Available: https://towardsdatascience.com/time-series-forecast-error-metrics-you-should-know-cc88b8c67f27. [Accessed 3 November 2022].

[14] C. Pang, “Day 3: Dimensionality Reduction,” National University of Singapore, Singapore, 2022.

[15] J. Brownlee, “How to Develop LSTM Models for Time Series Forecasting,” [Online]. Available: https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/. [Accessed 10 October 2022].

Appendices

Appendix A – Univariate Model Results

Model	Name	Other Parameters	MAPE (%)
LSTM	lstm4_lb48	dropout=0, batch_size=8	17.80
	Lstm16_lb48	dropout=0.1, batch_size=8	41.23
	Lstm16_lb48	Dropout=0.2, batch_size=8	27.23
	Lstm16_lb48	Dropout=0.2, batch_size=4	11.00
	Lstm16_lb48	Dropout=0.2, batch_size=2	24.58
	Lstm32_lb48	Dropout=0.2, batch_size=4	33.68
	Lstm32_lb24	Dropout=0.2, batch_size=4	20.70
	Lstm16_lb24	Dropout=0.2, batch_size=4	23.23
	Lstm16_lb36	Dropout=0.2, batch_size=4	6.27
	Lstm32_lb36	Dropout=0.2, batch_size=4	14.50
Stack LSTM	Stack_lstm_16_16_lb36	Dropout=0.2, batch_size=4	27.82
	Stack_lstm_16_8_lb36	Dropout=0.2, batch_size=4	23.74
	Stack_lstm_16_16_lb36	Dropout=0.5, batch_size=4	30.12
	Stack_lstm_16_16_lb48	Dropout=0.2, batch_size=4	29.53
	Stack_lstm_8_8_lb48	Dropout=0, batch_size=4	34.78
	Stack_lstm_8_8_lb48	Dropout=0.1, batch_size=4	29.19
	Stack_lstm_16_16_lb48	Dropout=0.25, batch_size=4	27.40
	Stack_lstm_16_16_lb36	Dropout=0.25, batch_size=4	31.17
	Stack_lstm_16_8_lb24	Dropout=0.2, batch_size=8	27.79
	Stack_lstm_16_8_lb36	Dropout=0.2, batch_size=16	27.25

Appendix B – Multivariate Model Results

The table below shows the model performance using different combinations of parameters:

Model	Name	RMSE	MAE	MAPE (%)
LSTM	lstm128_pc2_lb24	11708.57	9298.09	24.25
	lstm64_pc2_lb36	13208.85	10983.7	26.13
	lstm128_pc2_lb36	10403.17	8753.81	22.44
	lstm128_pc2_lb48	10659.11	8586.93	24.53
	lstm128_pc2_lb48_lag	10713.1	8615.95	23.32
	lstm64_pc2_lb48_lag	11464.7	9655.59	23.9
	lstm128_pc2_lb36_lag	13283.46	10793.33	26.61
	lstm64_pc2_lb36_lag	13185.66	10920.59	26.03
	lstm128_pc1_lb48_lag	7689.41	6003.71	18.15
	lstm64_pc1_lb48_lag	7824.79	6074.94	17.96
	lstm256_pc1_lb48_lag	8614.47	7114.61	17.16
	lstm128_pc1_lb36_lag	8807.58	6558.1	18.33
	lstm64_pc1_lb36_lag	13080.51	10693.61	25.2
Stack LSTM	stack_lstm128_64_pc2_lb24	10775.46	8776.12	20.06
	stack_lstm128_64_pc2_lb36	11252.46	9152.26	19.5
	stack_lstm128_64_pc2_lb48	10709.87	8839.87	19.92
	stack_lstm128_64_32_pc2_lb48	9507.52	7558.98	16.22
	stack_lstm128_64_32_pc2_lb24	8998.82	7465.95	18.06
	stack_lstm128_64_32_pc2_lb36	9653.09	7939.07	16.98
	stack_lstm256_128_64_32_pc2_lb48	9469.32	8136.2	17.41
	stack_lstm128_64_pc2_lb48_lag	10848.4	8854.04	19.87
	stack_lstm128_64_pc2_lb36_lag	11126.55	9045.1	20.39
	stack_lstm128_64_32_pc2_lb48_lag	8925.54	7281.39	16.62
	stack_lstm128_64_pc1_lb48_lag	10675.39	8618.43	20.59
	stack_lstm128_64_32_pc1_lb48_lag	8831.56	6980.78	17.77
	stack_lstm256_128_64_pc1_lb48_lag	10068.64	8774.37	21.73
	stack_lstm128_64_32_pc1_lb36_lag	9030.1	7827.88	17.44
	stack_lstm256_128_64_pc1_lb36_lag	12367.91	10512.27	24.46
	stack_lstm128_64_32_pc2_lb36_lag	8406.61	7106.92	17.17
Bi-directional LSTM	bidirectional_lstm128_pc2_lb24	12502.62	10231.02	22.61
	bidirectional_lstm128_pc2_lb36	11748.58	9694.46	19.07
	bidirectional_lstm128_pc2_lb48	13844.31	11567.46	23.18
	bidirectional_lstm64_pc2_lb24	11950.56	9477.92	24.26
	bidirectional_lstm64_pc2_lb36	11939.37	9686.22	21.64
	bidirectional_lstm64_pc2_lb48	13395.23	11206.99	21.14
	bidirectional_lstm64_pc2_lb48_lag	11945.33	9360.86	22.08
	bidirectional_lstm128_pc2_lb48_lag	11136.16	8099.82	22.26
	bidirectional_lstm128_pc1_lb48_lag	10624.52	9091.86	25.34
	bidirectional_lstm64_pc1_lb48_lag	14197.82	11453.68	30.54
	bidirectional_lstm256_pc1_lb48_lag	13438.01	10273.69	24.03
	bidirectional_lstm128_pc1_lb36_lag	10211.11	7752.31	20.42
	bidirectional_lstm64_pc1_lb36_lag	9733.04	7880.14	19.29

	bidirectional_lstm128_pc2_lb36_lag	9462.08	8214.35	18.03
	bidirectional_lstm64_pc2_lb36_lag	12354.18	9754.02	22.21
	bidirectional_lstm128_pc2_lb36_lag_ave	9061.1	7358.82	17.37
	bidirectional_lstm64_pc2_lb36_lag_ave	9909.85	8394.41	19.45
	bidirectional_lstm256_pc2_lb36_lag_ave	10733.81	8555.3	19.4
	bidirectional_lstm128_pc2_lb48_lag_ave	10612.11	8897.04	19.46
	bidirectional_lstm64_pc2_lb48_lag_ave	10017.16	8228.4	19.96
	bidirectional_lstm64_pc2_lb24_lag_ave	10584.35	8023.11	19.41
	bidirectional_lstm128_pc1_lb36_lag_ave	9352.75	7171.49	17.74
	bidirectional_lstm64_pc1_lb36_lag_ave	11876.1	9942.03	22.41
	bidirectional_lstm64_pc2_lb36_ave	9485.29	7675.87	19.02
	bidirectional_lstm128_pc2_lb36_ave	10894.19	8820.41	19.4
	bidirectional_lstm128_pc2_lb48_ave	11055.91	9041.75	20.6
	bidirectional_lstm64_pc2_lb48_ave	9062.12	7134.01	18.9