# *Future Windsor Housing Price Forecasting*

*Ethan Aikman*
*University of Windsor*
*Mechatronics Eng Stream*
*Windsor, Ontario*
*aikmane@uwindsor.ca*

***Abstract*—This paper is a final project report written for the Artificial Intelligence and Machine Learning class at the University of Windsor. It dives into the current housing market of Canada, and specifically Windsor, Ontario, to forecast housing prices using machine learning. It finds that using an encoder decoder LSTM is an effective use on time series data. By using this model to predict four years ahead on Windsor's housing data, it finds that there will by an increase in housing price to the near peak saw in 2022. Reading this report will inform readers on the production of a time series prediction model and give insight into Windsor's housing market.**

***Keywords—Housing, Machine Learning, Linear Regression, Real Estate, Windsor, ML, Time Series, Neural Networks, AI, Time-series, LSTM***

## I. Introduction

The Canadian Property Bubble is the rapid increase of real estate price in Canada's housing market from 2002 to the present day. In January 2005, the average sold price for a house was $229,346, and in January 2022 the average was $748,439[4]. Subsequently, Windsor's housing market has considerably risen in the past two decades, with an extreme example being an enormous 226% average sale price increase from 2019-2022[23]. Windsor has also experienced a noticeable drop in vacancy rates, with an 8.8% in 2005, all the way down to a 1.9% in 2022[3]. Living in Windsor has made this issue prevalent, and exploring the impacts of the historically affordable city makes for an interesting topic of discussion. This paper aims to answer what we can expect for the future of Windsor's housing market by using several historical data points as our dataset. It is especially made for university aged students who are not yet in the housing market but may be interested in knowing what the future holds for housing in Windsor. Previous examples of related papers will be used to explore the findings, and machine learning (ML) models chosen.

## II. Literature Review

### *A. Related Housing Market Research*

The housing market is complex and can have hundreds if not thousands of parameters that can influence it. Determining which the most important ones are will be important to the goal of this paper. Immigration, interest rate, unemployment rate, income, housing size, and vacancy, were all variables for a paper written in 2018[12]. It studied what parameters most influenced the increase in price, using linear regression, over a 10-year period between 2001-2011. Interestingly, out of the $158,875 increase in that time, 54.57% was due to immigration, 11.32% was due to Income, 58.6 was due to interest rate, and -24.49% was due to unemployment [12]. This paper utilizes some key parameters that will be useful in the model developed for Windsor. The article only presents its findings of a 10-year period from 2001-2011 even though it is written in 2018. It would be even more interesting to see the results using more parameters and a greater length of time. Using a saturated dataset is important for the goal of predicting Windsor's housing prices.

Representing data in a way viewers can understand is just as important as finding the data itself. In 2021 Italian researchers did an assessment of Real Estate prices on 2 Italian cities using factors related to parameters of the house (Parking, Size, etc.…), rather than the macroeconomic variables of the previous paper [1]. The paper has an excellent usage of a heat map to show the correlation of housing parameters that it uses to conclude most of its findings. This will be useful to show how parameters correlate in Windsor's case.

Another paper written on factors that determine housing prices, is one that focuses primarily on sociodemographic and economic disparities across neighborhoods in Austin, Texas [6]. It uniquely uses 25 different factors to deduct its results. Notably, the paper found that Hispanics, and number of transit routes played a significant role in house value. Windsor and Austin have quite a different ethnicity spread, where Windsor has 65.5% White, 9.3% Arabic, 7.0% South Asian [10], and Austin has 47.7% White, 32.5% Hispanic, and 8.4% Black [2]. Austin has 2 large dominating populations, where comparisons can be made, whereas Windsor is still predominantly White. The paper mentions that the only noticeable parameters between ethnic differences are between White, and Hispanic neighborhoods. While there may be small correlations between ethnic demographics in Windsor, the paper suggests low findings in its study. Therefore, this area will be avoided.

To continue the idea of proximity, there is a relatively older paper written in 2004, that explores the impact of the newly opened at the time, casino, and how it brings crime and disorder, that affects housing sales in its neighborhood [5]. Their findings concluded that little to no pricing changes occurred over the studied time, likely due to increased police presence in the area in response to elevated crime. It will be interesting to use this idea of analyzing neighborhood's crime, by using police report data that is available.

Another question to ask is if fertility rate affects housing prices, and vice versa. In 2012, Chinese researchers studied China's unexpected fertility rate decline, as they were dealing with an emerging labor shortage[8]. Their findings concluded that a city's fertility rates have a strong effect when housing is spacious or cheaper, and that high living costs equate to lower fertility rates. In 2019, Windsor had an average fertility rate of 1.51[9], while from 1996-2001 it had an average of 1.56. Given the large pricing differences between those two periods, it is odd to see such a small change in fertility rate.

### *B. Model Review*

Time-series prediction uses previous sequences of data points that are indexed by time, used to predict future sequences in time. Exploring time-series prediction used in

other papers will help with selecting the best model for forecasting house prices in Windsor.

A common application of time-series prediction is stock price forecasting. Time-series Neural Network (TNN), Linear Regression, Decision Tree, Random Forest, Support Vector Machine (SVM), Recursive Neural Network (RNN), Long Short-Term Memory (LSTM) were all evaluated conducted on stock price prediction[21]. 250 sequences of stock prices were used in the experiment and the result metrics were calculated using Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Mean Absolute Percentage Error (MAPE), and the Coefficient of Determination. The results were compared over different time spans (1 day, 7 days, 30 days), and TNN performed the greatest in all time spans, and metrics. LSTM was the runner up, followed by RNN. It is clear that Neural Networks give the best results in timeseries prediction, beating all other models by a significant amount in all metrics. While the other models all performed worse as the time span increased, RNN and LNN had similar results. This may mean that both models are more efficient for long term predictions, which is important for a large time frame of housing data. The paper was an introduction to TNN and showed that it is highly accurate. However, it failed to show larger time frame predictions where TNN may have decreased in performance. It is also a relatively complex model to incorporate and is likely outside of the scope of this paper.

Another comparison studies the results of multilayer perceptron (MLP), autoregressive, and LSTM predictions[14]. The models were given a massive dataset containing 100,000 sequences with six data features. MLP had slightly better results on shorter term dataset predictions, but did worse as the forecasting dimensions increased. The LSTM model started increasing in performance as the forecasting increased. This is a useful feature for a dataset with a high density of sequences. The training was done univariately, and was said to not fully utilize the LSTM model, by not being able to show the advantages of recurrent learning. The house predictor will need to be multivariate, and research is done showing how to incorporate that in the following paragraphs.

Sequence to sequence learning is explored by comparing Deep Neural Networks (DNN) and LSTM[20]. DNNs are said to have difficulty when dealing with unknown input and output sequence sizes. LSTMs, unlike DNNs does not have fixed input and output requirements and is good at solving sequence to sequence problems. Interestingly, when doing their training on speech recognition, LSTM had increased in performance when the source sentence was reversed. The testing also found that deep LSTMs had an increase performance of 10% for each added layer. They conclude that their results suggest that LSTM based forecasting will do well in other sequential learning problems, as long as there is enough training data. There is mention of recent articles concluding LSTM models suffered when working on longer sequences, but is fixed when reversing the source input. Experimenting this in the house forecasting model will be implemented to see if similar results are achieved.

Real time prediction based on previous data is an important field of research, especially for fault detection. An attention-based bidirectional LSTM encode-decoder model is proposed, and evaluated showing improvement in real-time disturbance detection for satellite images[22]. The model consists of a LSTM encoder and LSTM decoder. The encoder reads a sequence and encodes it into a context vector. Then the decoder reads the context vector, and gives an output. A bidirectional LSTM creates a forward and backward sequence that are then concatenated. Bidirectional LSTM is said to be effective in related tasks of disturbance detection. There is a drawback however that the encoder is fixed with the fixed original length of the context vector. The attention-based part of the model is the component that solves this problem. Rather than a fixed context vector, the model is generated iteratively and its decoded outputs are fed back into the model. Then the previous outputs are used as inputs for the decoder to predict, creating a dynamic output size. When tested on a multivariate time-series forecast problem, using 4 different variations, unidirectional LSTM, unidirectional LSTM + attention-based, bidirectional LSTM, and bidirectional LSTM + attention-based. In all metrics, the change to bidirectional had a significant performance boost. Similarly, the change to including the attention mechanism was also effective. The ability of this model to deal with multivariate sequence to sequence problems will be effective with forecasting house prices. Using a similar adaptation of the disturbance detection model,

To summarize, data features were explored when forecasting housing prices for a better understanding of important features for model creation. Then comparisons of time-series model implementations were analyzed to determine advantages and use cases of them. The next section will outline the problem definition in detail.

## III. Problem Definition

Forecasting housing prices can present challenges due to the dynamic nature of the market being influenced by so many factors. This paper aims to implement a reliable, and accurate model that will aid buyers or sellers to make informed decisions about the future of Windsor's housing market. This study uses historical data trends and explores the implementation of prediction models. The research and implementation found will contribute to a better understanding of what drives the housing market in Windsor and to inform the reader how a forecasting model is implemented. Further reading will discuss how this concept was developed, and the methods used to answer our problem definition

## IV. Methodology

In this section, the design and implementation of the models is discussed in depth.

### *A. Gathering data*

The first step is to find a reliable source for historical datasets for Windsor. The Canadian Real Estate Association has exportable data of average housing prices in present day Windsor, back to 2005[7]. Thankfully, this dataset includes monthly data points from Jan-05 to Jan-24 adding up to a count of 229. This now sets the standard of datasets to find as

ones that are monthly timestamped. The unemployment rate and total population of working age(15 years and older) was found in a labor force census by metropolitan area[19]. This was the 229-count monthly dataset standard needed. Monthly inflation rates in the same period using consumer price index tool[16]. Canadian interest rates is also available and was similarly downloaded using the same timeframe[17]. Finally, as the last appended feature, vacancy rates were given, but only as a yearly period so data processing will be important here[15]. This data was put together in a csv file, making sure the datasets lined up correctly according to their starting date.

### *B. Preparing Data*

There are a few main issues with the current dataset: 1) There is NaN values that need to be filled 2) There is a time column that should be used as the index instead of a feature column 3) Models can over depend on features with large values[21]

The data's index was first set to the year-month column, and then dropped the unnecessary column leaving only the key features. Then we use isna().sum() to find the count of missing values. Vacancy was missing 211, and interest was missing 1. The data was filled by using pandas fillna() function with fill forward as its method. Then the index is resampled to a more readable YYYY-MM-DD format. Now that the format is correct and all the data is filled, its time to start setting the data up to be a more productive set of features. Normalization using MinMaxScaler: normalization is the process of adjusting values in a dataset to the same scale. This is typically between 0 and 1. The values are calculated by:

$$x_{norm} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

This will increase our model's learning performance and is essential to the process. It is important to note that an inverse scaler can bring the data points back to their original scale.

### *C. Choosing a Model and Training*

Previous forecasting model implementations differ greatly depending on the dataset and what is trying to be achieved. This dataset has six features per point in time and consists of 229 of those points. Usually, models are predicted using a univariate time series input, where there is only one variable per time slot. This model will need to deal with a multivariate time series input. The output is also important when choosing a model, in a similar way to the input. There is also univariate and multivariate time series forecasting options that change the way the model is structured. Let's see what a many to many time series model requires.

LSTM: is a RNN variant and at first glance looks perfect and can be implemented as is. Designed for sequential data processing, with a linear vector that holds the entire sequence as long term memory, and three gate mechanisms to regulate flow. The input gate controls the information stored in memory, forget gate controls what gets deleted from memory, and the output gate controls what gets outputted from memory. Doing this allows the model to process sequences step by step, while updating their internal data based on past data. LSTM can do well in time series forecasting applications. However, the applications end when not doing single step forecasting.

Encoder-Decoder: is needed if a many to many model is needed. It consists of a encoder LSTM, and a decoder LSTM. The encoder uses all of the historical time series data, which is then sent to the decoder to predict an output sequence. This allows for many to many but has a caveat. The memory that is initialized by the historical data is fixed and the model will start to lack the longer sequences needed.

To deal with this issue, an attention layer is introduced. The same initialization happens with the historical data and the encoder, now just getting sent to the activation layer. The decoder now again uses the first output of the encoder, but then appends its output to the activation layer, and uses the previous output as its input. This happens for as long as specified, creating a true variable range input and output. Firstly, the model needed to have its historical data to train with. Two sets of models, both of size 228 took the whole dataset minus one taking one less from the top of the data, and the other taking one less from the bottom of the dataset. This can be used to compile our encoder to be trained on the whole dataset. Each point will predict the point ahead of it.

### *D. Training*

As previously mentioned, two different LSTM models will be implemented for best results. The encoder layer is implemented with a (,6) input shape, a size 128 LSTM layer followed by a dropout layer to help with overfitting, and finally ended with a size 6 LSTM layer as our output. Compiles using the 'Adam' optimizer and is finally trained with 100 epochs of batch size 1. Then the decoder is implemented with the exact same layer setup, accept with parameter stateful=True, which uses the last state of each sample with be used as the initial state. This allows the reuse of states and storing of states, useful for many iterations of predicting. The decoder is compiled using the same parameters and is then initialized with the weights of the encoder model. At this point the model is now able to start its prediction step by step using a loop.

### *E. Evaluation*

Several evaluation metrics can be used to test specific factors of the model[21]. Root Mean Square Error (RMSE) is used to measure bias between prediction values and actual values. Mean Absolute Error (MAE) evaluates the average bias in the prediction and the actual values. Mean Squared Error (MSE) measures the average of the square of the distance between predicted and actual values. This is used as a quality model measurement. All these metrics indicate a good quality of the model when they are small. Using the sklearn library in python, we can get results from each of these metrics. The following was found:

TABLE I. Evaluation Metrics Pre Tune

| MAE | MSE | RMSE |
|---|---|---|
| 0.036432411468731 | 0.003647481836417502 | 0.060394385802338864 |

### F. *Hyperparameter Tuning*

A good way to maximize the model's results is by hyper tuning its parameters. This is done by continuously rebuilding the model with a range of different parameters, and tracking which combinations do best[18].

TensorFlow has a tuner library that helps streamline this for your models. This is done by first creating a builder function for your model encoder_model(hp) that returns a model. Then you can pick your tuner of choice, in this case, Hyperband is used, and can be modified by changing its input parameters, such as units count, or objective to set the focus metric.

The Hyperband implemented in this study, checked the range of learning rates from 0.0001 to 0.01, and a choice between the activation methods relu, or tanh. The objective search was set to loss, and it had a max_epochs of 30 meaning it has 30 epochs to train one model. After 251 completed trials and around 30 minutes of elapsed time, it was complete. The resulting metrics were:

TABLE II. Evaluation Metrics Post Tune

| MAE | MSE | RMSE |
|---|---|---|
| 0.0313842658660 44446 | 0.0313842658660 44446 | 0.04936533934 21628 |

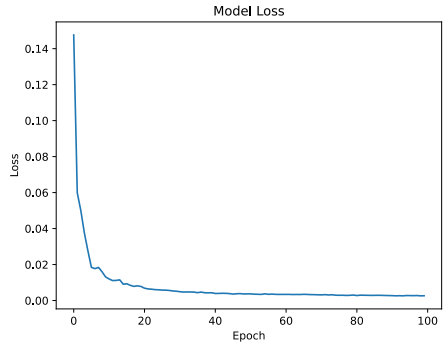


*Figure 1: Model Loss Graph*

This loss table was after the final optimization and is producing better results. The discovered parameters that best optimized loss, were tanh, 0.009023397373850085, and epochs=100. Conveniently, Hyperband saves progress of what it learns so it isn't always needed to have to relearn.

### G. *Prediction*

Now that the best parameters have been found, it's time to start forecasting. Predicting by our encoder, will give an equal sized prediction to the original data, just pushed in time one month.

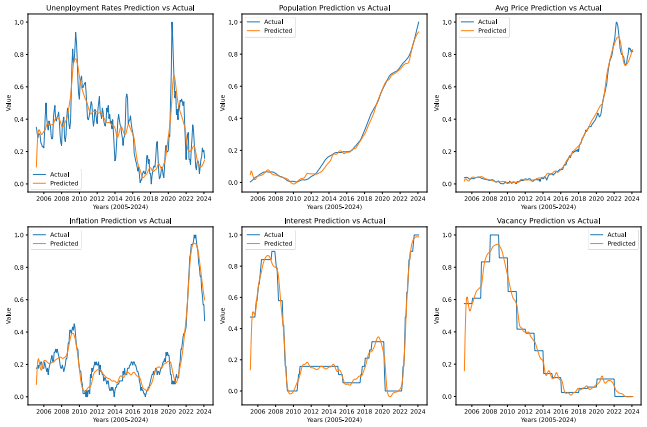


*Figure 2: All Predicted Data Graph*

The predictions look consistent with the actual data. This is only the first prediction in the iteration of predictions, however. The decoder model sets its weights to the weights of the encoders weights so that it can be initialized. Now the decoder can start iterating month by month to predict four years into the future, which is 48 time-steps. Each output is appended onto the memory of the system, and used for the next prediction. It's important to remember that the decoder layers have stateful=True so it can be reused each step. After the model completes its prediction, it is then inversely transformed using the scaler to bring back the original data's scale.

## V. Results and Discussions

The objective of this report is to create a model that is able to perform accurate average house price forecasting. Now that the model has completed its predictions let's analyze the data.

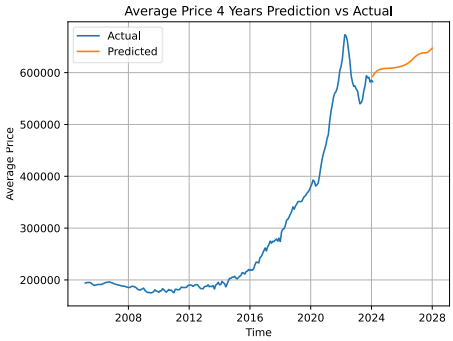


*Figure 3: Predicted House Prices Graph*

First, we can look at the housing price prediction. It shows an upwards trend nearly reaching the earlier peak in 2022 reaching a final $646,986.60. Not much explanation for this yet so let's look further into the other predictions to get some answers.

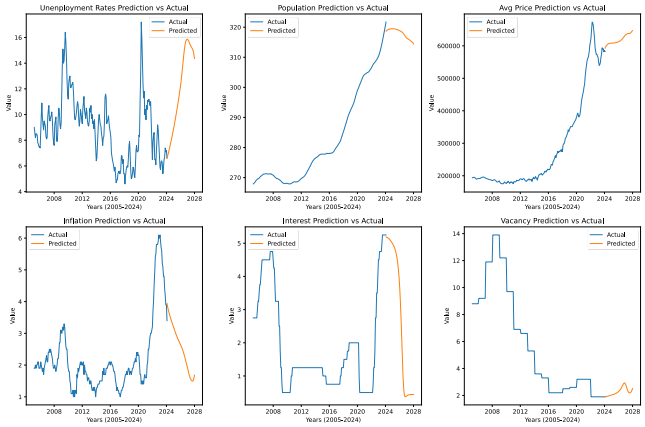


*Figure 4: All Predicted Future Data*

Interestingly, this shows that both interest rates and inflation are going to drop, while unemployment skyrockets. Population has a slight downward trend, and vacancy looks like it stays relatively the same. Clearly there is some kind of relationship between these features. Looking at the actual values for each feature it is clear that starting around 2018-2020, every feature aside from vacancy jumped. Unemployment rates lowered quickly after its rise along with inflation just a few years ahead. Vacancy is on an opposite trend compared to the other features, especially population and housing price. It is also interesting how there looks to be a shared trend between interest and inflation, and even to an extent unemployment rates. The last observation just from the graph is how similar the population and housing prices are, clearly showing it being the driving factor of these features.

During Covid-19s lockdowns from 2019-2022 there was a 226% average sale price increase for homes[23]. This is also the time when the other data points rise as well. As a possible insight to why we see these sudden opposite trends from what happened during Covid-19 is that this is a bounce back period. Everything is settling back to what it once was. Even the predicted housing data looks like it has an upwards trend like it did in 2016.

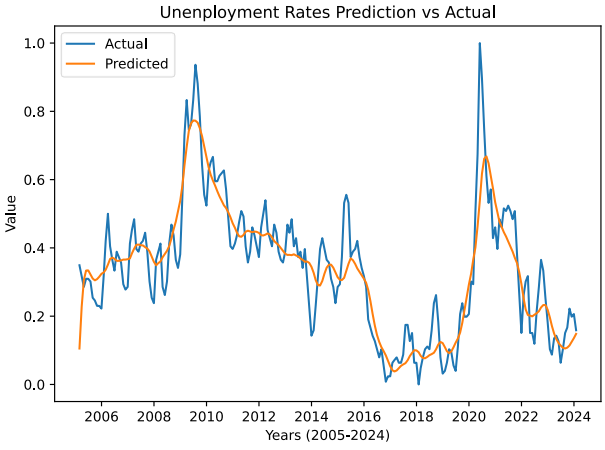


*Figure 5: Predicted Future Rate Data*

It's important to also note that it looks like the prediction for unemployment rates was slightly too sporadic for the model to handle. It still has the general trend shape but fails to hit those sharp jumps. For more accurate results greater but more time-consuming hyper tuning could be done.

## VI. Conclusion

Housing in Windsor has become increasingly less affordable in the past 20 years, seeing record low vacancy rates. Because of this, now more than ever it is important to be informed about the housing market. The paper explored different machine learning algorithms of previous studies and deployed those learnings to Windsor's housing market. Using an encoder decoder LSTM model, proved effective in timeseries forecasting. Using the model on data gathered from several monthly census's, pointed to an increase in the average housing price over the next four years (2024-2028). It also predicted a jump in unemployment, and a steep decrease in interest, and inflation rates. I believe future work can look deeper into the meaning of these results, and also create an even more accurate model. Housing markets should continue to be explored and analyzed in order to educate people.

## References

[1] L. Rampini and F. Re Cecconi, "Artificial intelligence algorithms to predict Italian real estate market prices," *JPIF*, vol. 40, no. 6, pp. 588–611, Sep. 2022, doi: 10.1108/JPIF-08-2021-0073.

[2] "Austin, Texas," *Wikipedia*. Feb. 05, 2024. Accessed: Feb. 15, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Austin,_Texas&oldid=1203707842

[3] S. C. Government of Canada, "Canada Mortgage and Housing Corporation, vacancy rates, apartment structures of six units and over, privately initiated in census metropolitan areas." Accessed: Feb. 15, 2024. [Online]. Available: https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=3410012701

[4] "Canadian Housing Market Report Jan. 17th, 2024 | Interactive Map - WOWA.ca." Accessed: Feb. 15, 2024. [Online]. Available: https://wowa.ca/reports/canada-housing-market

[5] A. G. Phipps, "Crime and disorder, and house sales and prices around the casino sites in Windsor, Ontario, Canada," *Canadian Geographies / Géographies canadiennes*, vol. 48, no. 4, pp. 403–432, Dec. 2004, doi: 10.1111/j.0008-3658.2004.00068.x.

[6] Y. Chen, J. Jiao, and A. Farahi, "Disparities in affecting factors of housing price: A machine learning approach to the effects of housing status, public transit, and density factors on single-family housing price," *Cities*, vol. 140, p. 104432, Sep. 2023, doi: 10.1016/j.cities.2023.104432

[7] "February 14 2024 News Release | CREA Statistics." Accessed: Feb. 15, 2024. [Online]. Available: https://stats.crea.ca/en-CA/

[8] L. Pan and J. Xu, "Housing price and fertility rate," *China Economic Journal*, vol. 5, no. 2–3, pp. 97–111, Jun. 2012, doi: 10.1080/17538963.2013.764675.

[9] "London's birth rate among big-city Ontario's lowest (even before pandemic)," lfpress. Accessed: Feb. 15, 2024. [Online]. Available: https://lfpress.com/news/local-news/londons-birth-rate-among-big-city-ontarios-lowest-even-before-pandemic

[10] "Windsor, Ontario," *Wikipedia*. Feb. 15, 2024. Accessed: Feb. 15, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Windsor,_Ontario&oldid=1207791599

[11] "Zillow Scraper - Extract Data from Zillow." Accessed: Feb. 15, 2024. [Online]. Available: https://chromewebstore.google.com/detail/zillow-scraper-extract-da/nlieamdebnjhijflpbkbaijnjpdpieoh?pli=1

[12] A. Nistor and D. Reianu, "Determinants of housing prices: evidence from Ontario cities, 2001-2011," *IJHMA*, vol. 11, no. 3, pp. 541–556, Jun. 2018, doi: 10.1108/IJHMA-08-2017-0078.

[13] "WINDSOR-ESSEX COUNTY ASSOCIATION OF REALTORS | RESIDENTIAL STATS President's." Accessed: Feb. 15, 2024. [Online]. Available: https://wecartech.com/wecfiles/stats_new/2022/dec/

[14] P. Lind, M. Ridhagen, and Y. Yang, "A comparative study of Neural Network Forecasting models on the M4 competition data".

[15] S. C. Government of Canada, "Canada Mortgage and Housing Corporation, vacancy rates, apartment structures of six units and over, privately initiated in census metropolitan areas." Accessed: Mar. 27, 2024. [Online]. Available: https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=3410012701

[16] S. C. Government of Canada, "Consumer Price Index Data Visualization Tool." Accessed: Mar. 27, 2024. [Online]. Available: https://www150.statcan.gc.ca/n1/pub/71-607-x/2018016/cpilg-ipcgl-eng.htm

[17] Organization for Economic Co-operation and Development, "Interest Rates: Immediate Rates (< 24 Hours): Central Bank Rates: Total for Canada," FRED, Federal Reserve Bank of St. Louis. Accessed: Mar. 27, 2024. [Online]. Available: https://fred.stlouisfed.org/series/IRSTCB01CAM156N

[18] "Introduction to the Keras Tuner | TensorFlow Core," TensorFlow. Accessed: Mar. 27, 2024. [Online]. Available: https://www.tensorflow.org/tutorials/keras/keras_tuner

[19] S. C. Government of Canada, "Labour force characteristics by census metropolitan area, three-month moving average, seasonally adjusted and unadjusted, last 5 months, inactive." Accessed: Mar. 27, 2024. [Online]. Available: https://www150.statcan.gc.ca/t1/tbl1/en/tv.action?pid=1410029401

[20] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks".

[21] L. Zhang *et al.*, "Time-Series Neural Network: A High-Accuracy Time-Series Forecasting Method Based on Kernel Filter and Time Attention," *Information*, vol. 14, no. 9, Art. no. 9, Sep. 2023, doi: 10.3390/info14090500.

[22] Y. Yuan *et al.*, "Using An Attention-Based LSTM Encoder–Decoder Network for Near Real-Time Disturbance Detection," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 13, pp. 1819–1832, 2020, doi: 10.1109/JSTARS.2020.2988324.

[23] "Windsor housing price hikes double the national average during COVID," windsorstar. Accessed: Mar. 27, 2024. [Online]. Available: https://windsorstar.com/news/local-news/windsor-housing-price-hikes-double-the-national-average-during-covid

APPENDICES

Appendix 1 Source data[15][16][17][19]

| Year | Unenployment Rates | Population | Avg Price | Inflation | Interest | Vacancy |
|---|---|---|---|---|---|---|
| 2005-01-01 | 9.8 | 267.7 | 193300 | 1.9 | 2.75 | 8.8 |
| 2005-02-01 | 9 | 267.9 | 193900 | 1.9 | 2.75 | |
| 2005-03-01 | 8.6 | 268.1 | 194800 | 1.9 | 2.75 | |
| 2005-04-01 | 8.2 | 268.3 | 195000 | 2 | 2.75 | |
| 2005-05-01 | 8.5 | 268.5 | 195100 | 1.9 | 2.75 | |
| 2005-06-01 | 8.5 | 268.8 | 194800 | 2 | 2.75 | |
| 2005-07-01 | 8.4 | 269.1 | 193000 | 2 | 2.75 | |
| 2005-08-01 | 7.8 | 269.3 | 191100 | 2.1 | 2.75 | |
| 2005-09-01 | 7.7 | 269.5 | 189400 | 2.1 | 3 | |
| 2005-10-01 | 7.5 | 269.6 | 190000 | 1.9 | 3.25 | |
| 2005-11-01 | 7.5 | 269.7 | 190700 | 1.9 | 3.25 | |
| 2005-12-01 | 7.4 | 269.8 | 191300 | 1.9 | 3.5 | |
| 2006-01-01 | 8.6 | 270.1 | 191300 | 2.1 | 3.75 | 9.2 |
| 2006-02-01 | 9.9 | 270.3 | 191200 | 2 | 3.75 | |
| 2006-03-01 | 10.9 | 270.5 | 191300 | 1.9 | 4 | |
| 2006-04-01 | 9.7 | 270.6 | 192700 | 1.8 | 4.25 | |
| 2006-05-01 | 9.2 | 270.8 | 193900 | 1.9 | 4.5 | |
| 2006-06-01 | 8.8 | 270.9 | 194800 | 1.7 | 4.5 | |
| 2006-07-01 | 9.5 | 271.1 | 195400 | 1.9 | 4.5 | |
| 2006-08-01 | 9.3 | 271.2 | 195600 | 2 | 4.5 | |
| 2006-09-01 | 9.1 | 271.2 | 196200 | 2.1 | 4.5 | |
| 2006-10-01 | 8.3 | 271.2 | 195300 | 2.2 | 4.5 | |
| 2006-11-01 | 8.1 | 271.3 | 194300 | 2.3 | 4.5 | |
| 2006-12-01 | 8.2 | 271.3 | 193600 | 2.1 | 4.5 | |
| 2007-01-01 | 9.8 | 271.3 | 192300 | 2.1 | 4.5 | 11.9 |
| 2007-02-01 | 10.3 | 271.2 | 191600 | 2.4 | 4.5 | |
| 2007-03-01 | 10.7 | 271.2 | 190700 | 2.4 | 4.5 | |
| 2007-04-01 | 9.6 | 271.2 | 190200 | 2.5 | 4.5 | |
| 2007-05-01 | 9.5 | 271.2 | 189900 | 2.4 | 4.5 | |
| 2007-06-01 | 9.8 | 271.2 | 188900 | 2.5 | 4.5 | |
| 2007-07-01 | 9.9 | 271.3 | 188500 | 2.4 | 4.75 | |
| 2007-08-01 | 10.2 | 271.2 | 187900 | 2.3 | 4.75 | |
| 2007-09-01 | 9.5 | 271.2 | 188000 | 2.2 | 4.75 | |
| 2007-10-01 | 8.4 | 271 | 186800 | 2 | 4.75 | |
| 2007-11-01 | 7.8 | 270.9 | 186200 | 1.9 | 4.75 | |
| 2007-12-01 | 7.6 | 270.8 | 185300 | 1.9 | 4.5 | |
| 2008-01-01 | 9.2 | 270.6 | 185500 | 2 | 4.25 | 13.9 |
| 2008-02-01 | 9.5 | 270.4 | 186800 | 1.9 | 4.25 | |
| 2008-03-01 | 9.8 | 270.2 | 187800 | 1.8 | 3.75 | |
| 2008-04-01 | 8.2 | 270 | 187400 | 1.9 | 3.25 | |
| 2008-05-01 | 7.9 | 269.9 | 186700 | 2 | 3.25 | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2008-06-01 | 8.4 | 269.7 | 185300 | 2.2 | 3.25 | |
| 2008-07-01 | 9.8 | 269.7 | 182700 | 2.2 | 3.25 | |
| 2008-08-01 | 10.5 | 269.6 | 181000 | 2.4 | 3.25 | |
| 2008-09-01 | 10.1 | 269.5 | 180400 | 2.7 | 3.25 | |
| 2008-10-01 | 9.2 | 269.4 | 181400 | 2.7 | 2.5 | |
| 2008-11-01 | 8.9 | 269.2 | 182700 | 3.1 | 2.5 | |
| 2008-12-01 | 9.4 | 269.1 | 184200 | 3.1 | 1.75 | |
| 2009-01-01 | 11.6 | 268.9 | 180000 | 3 | 1.25 | 12.2 |
| 2009-02-01 | 13.8 | 268.7 | 177900 | 3.1 | 1.25 | |
| 2009-03-01 | 15.1 | 268.5 | 176000 | 3.2 | 0.75 | |
| 2009-04-01 | 14 | 268.3 | 175700 | 3.1 | 0.5 | |
| 2009-05-01 | 14.2 | 268.2 | 175700 | 3.3 | 0.5 | |
| 2009-06-01 | 15.1 | 268.1 | 174800 | 3.2 | 0.5 | |
| 2009-07-01 | 16.4 | 268 | 175800 | 2.9 | 0.5 | |
| 2009-08-01 | 15.7 | 268 | 177600 | 2.6 | 0.5 | |
| 2009-09-01 | 14.5 | 268.1 | 181000 | 2.5 | 0.5 | |
| 2009-10-01 | 12.7 | 268.1 | 178600 | 2.5 | 0.5 | |
| 2009-11-01 | 11.6 | 268.1 | 177800 | 2.1 | 0.5 | |
| 2009-12-01 | 11.2 | 268.1 | 176100 | 1.9 | 0.5 | |
| 2010-01-01 | 12.5 | 268 | 179000 | 1.8 | 0.5 | 9.7 |
| 2010-02-01 | 12.8 | 268 | 178700 | 1.8 | 0.5 | |
| 2010-03-01 | 13 | 267.9 | 183200 | 1.5 | 0.5 | |
| 2010-04-01 | 12.1 | 267.9 | 181100 | 1.4 | 0.5 | |
| 2010-05-01 | 12.1 | 267.9 | 178500 | 1.1 | 0.5 | |
| 2010-06-01 | 12.3 | 268 | 176400 | 1.1 | 0.75 | |
| 2010-07-01 | 12.4 | 268.1 | 179000 | 1.1 | 1 | |
| 2010-08-01 | 12.5 | 268.2 | 181600 | 1.2 | 1 | |
| 2010-09-01 | 11.8 | 268.3 | 179600 | 1 | 1.25 | |
| 2010-10-01 | 10.7 | 268.5 | 180400 | 1.2 | 1.25 | |
| 2010-11-01 | 9.7 | 268.5 | 177000 | 1 | 1.25 | |
| 2010-12-01 | 9.6 | 268.6 | 175400 | 1.2 | 1.25 | |
| 2011-01-01 | 9.8 | 268.6 | 182000 | 1.2 | 1.25 | 6.9 |
| 2011-02-01 | 10.1 | 268.6 | 182000 | 1.1 | 1.25 | |
| 2011-03-01 | 10.6 | 268.5 | 180800 | 1.7 | 1.25 | |
| 2011-04-01 | 11 | 268.6 | 181200 | 1.6 | 1.25 | |
| 2011-05-01 | 10.8 | 268.6 | 182800 | 1.9 | 1.25 | |
| 2011-06-01 | 9.7 | 268.7 | 186400 | 1.8 | 1.25 | |
| 2011-07-01 | 9.1 | 268.8 | 185300 | 1.9 | 1.25 | |
| 2011-08-01 | 9.5 | 268.9 | 185500 | 2 | 1.25 | |
| 2011-09-01 | 10.4 | 269.1 | 185300 | 2.1 | 1.25 | |
| 2011-10-01 | 10.1 | 269.3 | 186000 | 2 | 1.25 | |
| 2011-11-01 | 9.7 | 269.5 | 189000 | 2.1 | 1.25 | |
| 2011-12-01 | 9.3 | 269.7 | 190100 | 2 | 1.25 | |
| 2012-01-01 | 10.4 | 269.8 | 190200 | 2 | 1.25 | 6.6 |
| 2012-02-01 | 10.9 | 270 | 189800 | 2.1 | 1.25 | |
| 2012-03-01 | 11.4 | 270.1 | 187500 | 1.7 | 1.25 | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2012-04-01 | 10.3 | 270.2 | 189100 | 2 | 1.25 | |
| 2012-05-01 | 10 | 270.4 | 190800 | 1.8 | 1.25 | |
| 2012-06-01 | 9.7 | 270.6 | 191200 | 1.8 | 1.25 | |
| 2012-07-01 | 10.5 | 270.9 | 189600 | 1.7 | 1.25 | |
| 2012-08-01 | 10.2 | 271.2 | 186100 | 1.7 | 1.25 | |
| 2012-09-01 | 9.5 | 271.6 | 183500 | 1.5 | 1.25 | |
| 2012-10-01 | 9.2 | 271.9 | 183300 | 1.6 | 1.25 | |
| 2012-11-01 | 9.1 | 272.3 | 182900 | 1.6 | 1.25 | |
| 2012-12-01 | 9.5 | 272.6 | 186800 | 1.5 | 1.25 | |
| 2013-01-01 | 10.5 | 273 | 187800 | 1.4 | 1.25 | 5.3 |
| 2013-02-01 | 10.2 | 273.4 | 187700 | 1.5 | 1.25 | |
| 2013-03-01 | 10.7 | 273.7 | 190600 | 1.5 | 1.25 | |
| 2013-04-01 | 9.7 | 274.1 | 186800 | 1.3 | 1.25 | |
| 2013-05-01 | 10 | 274.5 | 187700 | 1.2 | 1.25 | |
| 2013-06-01 | 9.3 | 274.9 | 187100 | 1.2 | 1.25 | |
| 2013-07-01 | 9.5 | 275.3 | 189100 | 1.3 | 1.25 | |
| 2013-08-01 | 8.9 | 275.6 | 182500 | 1.2 | 1.25 | |
| 2013-09-01 | 9.6 | 275.8 | 190000 | 1.3 | 1.25 | |
| 2013-10-01 | 8.5 | 276.1 | 192000 | 1.2 | 1.25 | |
| 2013-11-01 | 7.5 | 276.3 | 195400 | 1 | 1.25 | |
| 2013-12-01 | 6.4 | 276.4 | 190600 | 1.2 | 1.25 | |
| 2014-01-01 | 6.6 | 276.6 | 191100 | 1.3 | 1.25 | 3.6 |
| 2014-02-01 | 7.5 | 276.7 | 197300 | 1.3 | 1.25 | |
| 2014-03-01 | 8.6 | 276.8 | 194400 | 1.4 | 1.25 | |
| 2014-04-01 | 9.6 | 277 | 193100 | 1.4 | 1.25 | |
| 2014-05-01 | 10 | 277.1 | 186500 | 1.4 | 1.25 | |
| 2014-06-01 | 9.6 | 277.4 | 191800 | 1.5 | 1.25 | |
| 2014-07-01 | 9.2 | 277.5 | 198400 | 1.5 | 1.25 | |
| 2014-08-01 | 9.1 | 277.7 | 203100 | 1.5 | 1.25 | |
| 2014-09-01 | 8.5 | 277.7 | 203100 | 1.5 | 1.25 | |
| 2014-10-01 | 8.2 | 277.8 | 205300 | 1.5 | 1.25 | |
| 2014-11-01 | 7.6 | 277.8 | 205300 | 1.8 | 1.25 | |
| 2014-12-01 | 8.2 | 277.8 | 207100 | 1.7 | 1.25 | |
| 2015-01-01 | 8.3 | 277.8 | 203400 | 1.9 | 1 | 3.3 |
| 2015-02-01 | 9.3 | 277.8 | 202000 | 1.7 | 1 | |
| 2015-03-01 | 11.3 | 277.8 | 204900 | 1.7 | 1 | |
| 2015-04-01 | 11.6 | 277.8 | 206900 | 1.7 | 1 | |
| 2015-05-01 | 11.3 | 277.8 | 209100 | 1.9 | 1 | |
| 2015-06-01 | 9.3 | 277.9 | 214100 | 1.9 | 1 | |
| 2015-07-01 | 9.5 | 278 | 213500 | 1.9 | 0.75 | |
| 2015-08-01 | 9.6 | 278.1 | 211200 | 2 | 0.75 | |
| 2015-09-01 | 9.9 | 278 | 216100 | 2 | 0.75 | |
| 2015-10-01 | 9.3 | 278 | 216900 | 2.1 | 0.75 | |
| 2015-11-01 | 8.9 | 278 | 220200 | 2.1 | 0.75 | |
| 2015-12-01 | 8.6 | 278.1 | 218100 | 2 | 0.75 | |
| 2016-01-01 | 8.2 | 278 | 219800 | 2.1 | 0.75 | 2.2 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2016-02-01 | 7 | 278.1 | 218700 | 2.1 | 0.75 | |
| 2016-03-01 | 6.7 | 278.1 | 221900 | 2 | 0.75 | |
| 2016-04-01 | 6.4 | 278.2 | 229100 | 2 | 0.75 | |
| 2016-05-01 | 6.2 | 278.2 | 234500 | 1.9 | 0.75 | |
| 2016-06-01 | 5.9 | 278.4 | 233800 | 1.8 | 0.75 | |
| 2016-07-01 | 5.6 | 278.5 | 232800 | 1.8 | 0.75 | |
| 2016-08-01 | 5.9 | 278.8 | 243200 | 1.7 | 0.75 | |
| 2016-09-01 | 5.3 | 279.1 | 245200 | 1.5 | 0.75 | |
| 2016-10-01 | 4.7 | 279.5 | 251100 | 1.4 | 0.75 | |
| 2016-11-01 | 4.9 | 279.8 | 256300 | 1.2 | 0.75 | |
| 2016-12-01 | 4.9 | 280 | 261700 | 1.3 | 0.75 | |
| 2017-01-01 | 5.4 | 280.3 | 256000 | 1.2 | 0.75 | 2.2 |
| 2017-02-01 | 5.5 | 280.6 | 264000 | 1.2 | 0.75 | |
| 2017-03-01 | 5.6 | 280.9 | 267700 | 1.1 | 0.75 | |
| 2017-04-01 | 5.4 | 281.2 | 274900 | 1.1 | 0.75 | |
| 2017-05-01 | 5.4 | 281.5 | 270900 | 1 | 0.75 | |
| 2017-06-01 | 5.7 | 281.8 | 274600 | 1.1 | 0.75 | |
| 2017-07-01 | 6.8 | 282.2 | 273700 | 1.2 | 1 | |
| 2017-08-01 | 6.8 | 282.8 | 276700 | 1.2 | 1 | |
| 2017-09-01 | 6.2 | 283.4 | 279000 | 1.3 | 1.25 | |
| 2017-10-01 | 6.5 | 284 | 274900 | 1.3 | 1.25 | |
| 2017-11-01 | 5.4 | 284.6 | 280700 | 1.4 | 1.25 | |
| 2017-12-01 | 5.4 | 285.2 | 274200 | 1.5 | 1.25 | |
| 2018-01-01 | 4.6 | 285.8 | 293000 | 1.6 | 1.5 | 2.5 |
| 2018-02-01 | 5.2 | 286.4 | 297700 | 1.7 | 1.5 | |
| 2018-03-01 | 5.6 | 287.1 | 298900 | 1.7 | 1.5 | |
| 2018-04-01 | 5.9 | 287.7 | 304800 | 1.8 | 1.5 | |
| 2018-05-01 | 6 | 288.4 | 315300 | 1.7 | 1.5 | |
| 2018-06-01 | 5.9 | 289 | 317000 | 1.8 | 1.5 | |
| 2018-07-01 | 6.6 | 289.7 | 321600 | 1.9 | 1.75 | |
| 2018-08-01 | 7.6 | 290.3 | 327100 | 1.9 | 1.75 | |
| 2018-09-01 | 7.9 | 290.9 | 331800 | 1.7 | 1.75 | |
| 2018-10-01 | 6.9 | 291.3 | 341100 | 1.8 | 2 | |
| 2018-11-01 | 5.6 | 291.8 | 336300 | 1.8 | 2 | |
| 2018-12-01 | 5 | 292.2 | 342700 | 1.7 | 2 | |
| 2019-01-01 | 5.1 | 292.7 | 346400 | 1.8 | 2 | 2.6 |
| 2019-02-01 | 5.4 | 293.1 | 351200 | 1.9 | 2 | |
| 2019-03-01 | 5.9 | 293.5 | 351400 | 1.9 | 2 | |
| 2019-04-01 | 5.8 | 293.9 | 351000 | 2 | 2 | |
| 2019-05-01 | 5.3 | 294.4 | 351600 | 2.1 | 2 | |
| 2019-06-01 | 5.1 | 294.9 | 356700 | 2.1 | 2 | |
| 2019-07-01 | 6 | 295.6 | 361000 | 2.2 | 2 | |
| 2019-08-01 | 7.2 | 296.2 | 362700 | 2.1 | 2 | |
| 2019-09-01 | 7.6 | 297 | 367200 | 2.4 | 2 | |
| 2019-10-01 | 7.1 | 297.7 | 370000 | 2.3 | 2 | |
| 2019-11-01 | 7.1 | 298.3 | 373300 | 2.4 | 2 | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2019-12-01 | 7.2 | 298.9 | 379900 | 2.3 | 2 | |
| 2020-01-01 | 8.4 | 299.4 | 385700 | 2.3 | 2 | 3.2 |
| 2020-02-01 | 8.3 | 299.9 | 392800 | 2 | 2 | |
| 2020-03-01 | 11 | 300.4 | 389500 | 1.7 | 1 | |
| 2020-04-01 | 13.1 | 300.9 | 381000 | 1.7 | 0.5 | |
| 2020-05-01 | 17.2 | 301.4 | 383400 | 1.4 | 0.5 | |
| 2020-06-01 | 15.9 | 301.8 | 387400 | 1.5 | 0.5 | |
| 2020-07-01 | 14 | 302.3 | 401200 | 1.4 | 0.5 | |
| 2020-08-01 | 12.3 | 302.8 | 418100 | 1.5 | 0.5 | |
| 2020-09-01 | 11.3 | 303.3 | 432300 | 1.4 | 0.5 | |
| 2020-10-01 | 11.8 | 303.7 | 443500 | 1.6 | 0.5 | |
| 2020-11-01 | 10 | 304 | 452100 | 1.7 | 0.5 | |
| 2020-12-01 | 10.4 | 304.2 | 460300 | 1.6 | 0.5 | |
| 2021-01-01 | 9.6 | 304.4 | 473500 | 1.6 | 0.5 | 3.2 |
| 2021-02-01 | 10.7 | 304.5 | 480200 | 1.6 | 0.5 | |
| 2021-03-01 | 10.4 | 304.6 | 505200 | 2 | 0.5 | |
| 2021-04-01 | 11.1 | 304.8 | 523400 | 2.1 | 0.5 | |
| 2021-05-01 | 11 | 304.9 | 537800 | 2.5 | 0.5 | |
| 2021-06-01 | 11.2 | 305 | 551300 | 2.5 | 0.5 | |
| 2021-07-01 | 11 | 305.1 | 560300 | 2.7 | 0.5 | |
| 2021-08-01 | 10.7 | 305.3 | 563400 | 2.9 | 0.5 | |
| 2021-09-01 | 11 | 305.6 | 570600 | 3 | 0.5 | |
| 2021-10-01 | 9.4 | 305.9 | 584400 | 3 | 0.5 | |
| 2021-11-01 | 8 | 306.1 | 603400 | 3.1 | 0.5 | |
| 2021-12-01 | 6.5 | 306.5 | 612200 | 3.2 | 0.5 | |
| 2022-01-01 | 7.9 | 306.9 | 626100 | 3.7 | 0.5 | 1.9 |
| 2022-02-01 | 8.4 | 307.2 | 650600 | 4 | 0.5 | |
| 2022-03-01 | 8.6 | 307.6 | 673200 | 4.5 | 0.75 | |
| 2022-04-01 | 6.5 | 307.9 | 670800 | 4.9 | 1.25 | |
| 2022-05-01 | 6.5 | 308.1 | 661900 | 5.4 | 1.25 | |
| 2022-06-01 | 6.1 | 308.4 | 641900 | 5.5 | 1.75 | |
| 2022-07-01 | 7.3 | 308.6 | 622000 | 5.8 | 2.75 | |
| 2022-08-01 | 8.2 | 308.9 | 593800 | 5.8 | 2.75 | |
| 2022-09-01 | 9.2 | 309.2 | 582100 | 5.8 | 3.5 | |
| 2022-10-01 | 8.8 | 309.5 | 573500 | 5.9 | 3.5 | |
| 2022-11-01 | 7.8 | 309.9 | 574200 | 6.1 | 4 | |
| 2022-12-01 | 6.9 | 310.3 | 568200 | 6 | 4.5 | |
| 2023-01-01 | 5.9 | 310.8 | 564700 | 6.1 | 4.5 | |
| 2023-02-01 | 5.7 | 311.4 | 550900 | 5.9 | 4.75 | |
| 2023-03-01 | 6.3 | 312.1 | 539800 | 5.7 | 4.75 | |
| 2023-04-01 | 6.4 | 312.8 | 542000 | 5.6 | 4.75 | |
| 2023-05-01 | 6.2 | 313.7 | 548500 | 5.2 | 4.75 | |
| 2023-06-01 | 5.4 | 314.5 | 564600 | 5.1 | 5 | |
| 2023-07-01 | 5.9 | 315.5 | 574800 | 4.8 | 5.25 | |
| 2023-08-01 | 6.5 | 316.5 | 594000 | 4.8 | 5.25 | |
| 2023-09-01 | 6.7 | 317.5 | 590000 | 4.4 | 5.25 | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 2023-10-01 | 7.4 | 318.6 | 591000 | 4.2 | 5.25 | |
| 2023-11-01 | 7.1 | 319.6 | 581800 | 3.9 | 5.25 | |
| 2023-12-01 | 7.2 | 320.6 | 585400 | 3.9 | 5.25 | |
| 2024-01-01 | 6.6 | 321.7 | 582500 | 3.4 | | |
| 2024-02-01 | 7.3 | | | | | |

Appendix 2 Source Code

```
# -*- coding: utf-8 -*-
"""FinalFinalProject.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1OHRThWrJ28ODQngHuTPkNHMZzCMylhOD

#Importing Libraries
"""

!pip install --upgrade keras
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, accuracy_score
from sklearn.preprocessing import StandardScaler, MinMaxScaler
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold, GridSearchCV
from keras.models import Sequential
from keras.layers import Dense, LSTM, Dropout
import numpy as np
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
import tensorflow as tf
from keras import initializers, layers
import keras
from sklearn.datasets import make_classification
from tensorflow import keras
!pip install keras-tuner --upgrade
from keras_tuner import Hyperband

"""# Loading Dataset"""

#tf.random.set_seed(0)#Setting Random Seed for fixed random

keras.utils.set_random_seed(812)

dataset = pd.read_csv('MAIN.csv')

"""# Data Preprocessing"""

data = dataset.iloc[:300,:7]# Get rid of blank spaces
```

```
print(data.isna().sum())# Initial Summary
print(data.head)
#Get Rid of year column and make it index column to work better with model
data.index = pd.to_datetime(data.Year,errors='coerce')
data = data.drop('Year', axis=1)

# Find NaN value
data.isna().sum()
data = data.fillna(method='ffill')#Forward Fill missing values
data = data.resample('M').mean()# Format index to YYYY-MM-DD
data.isna().sum()
data = data.fillna(method='ffill')
data.isna().sum()

futureData = data

data = data.iloc[:229,:]# Get rid of blank spaces

"""# Feature Correlation"""

sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.show()

plt.plot(data.index, data.iloc[:,2], label='Actual')


plt.title('Average Housing Price Windsor, ON 2005-2024')
plt.xlabel('Year')
plt.ylabel('Average Price')
plt.legend()
plt.ylim(100000, 800000)
plt.grid(True)
plt.show()

"""#Scale Data"""

scaler = MinMaxScaler()
scaled_data = scaler.fit_transform(data)

"""# Create Sequence"""

seq_n = 1

X=[]
Y=[]
for i in range(len(scaled_data)-seq_n):
  X.append(scaled_data[i:i+seq_n, :])
  Y.append(scaled_data[i+seq_n, :])

X = np.array(X)
```

```
Y = np.array(Y)


print(Y)
X = X.reshape(1, 228, 6)
Y = Y.reshape(1, 228, 6)

"""# Create First LSTM Model"""

def encoder_model(hp):
  #Training Model
  model = Sequential()
  model.add(tf.keras.Input((None,6)))
  model.add(LSTM(128, return_sequences=True, activation=hp.Choice("activation",
['relu','tanh']) ) )#
  model.add(Dropout(0.2))
  model.add(LSTM(6, return_sequences=True))
  #model.add(Dense(6))
  learning_rate = hp.Float("lr", min_value=1e-4, max_value=1e-2, sampling="log")
  model.compile(optimizer=keras.optimizers.Adam(learning_rate=learning_rate),
loss='mean_squared_error', metrics=['accuracy'])#
  return model

"""#Create Tuner and Find best parameters for loss"""

tuner = Hyperband(
    encoder_model,
    objective='loss',
    max_epochs=100,
    directory='my_dir',
    project_name='intro_to_kt')

tuner.search(X, Y, epochs=10, batch_size = 1)

"""#Train Model"""

# Get the best hyperparameters
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]

# Build the best model
best_model = tuner.hypermodel.build(best_hps)

# Train the best model
history = best_model.fit(X, Y, epochs=100, batch_size=1)

"""# Plot Loss

"""

plt.plot(history.history['loss'])
```

```
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')


best_model.summary()
print(best_hps.values)
plt.savefig("test.svg", format="svg", dpi=1200)
print()
plt.show()

"""# Evaluation Metrics

"""

from sklearn.metrics import mean_absolute_error, mean_squared_error

predictions = best_model.predict(X)

# Calculate evaluation metrics
mae = mean_absolute_error(Y.reshape(228, 6), predictions.reshape(228, 6))
mse = mean_squared_error(Y.reshape(228, 6), predictions.reshape(228, 6))
rmse = np.sqrt(mse)

print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)

"""# Plotting All predicted values subplots"""

fig, axes = plt.subplots(2, 3, figsize=(15, 10))
for i,ax in enumerate(axes.flat):
    ax.plot(data.index[-228:], Y.reshape(228, 6)[:,i], label='Actual')
    ax.plot(data.index[-228:], predictions.reshape(228, 6)[:,i], label='Predicted')
    ax.set_title(f'{data.columns.tolist()[i]} Prediction vs Actual')
    ax.set_ylabel('Value')
    ax.set_xlabel('Years (2005-2024)')
    ax.legend()

plt.tight_layout()
plt.show()

best_model.summary()
print(best_hps.values)

image_format = 'svg' # e.g .png, .svg, etc.
image_name = 'featurePred.svg'

fig.savefig(image_name, format=image_format, dpi=1200)
```

```
"""# PLotting just unemployment pred"""

plt.plot(data.index[-228:], Y.reshape(228, 6)[:,0], label='Actual')
plt.plot(data.index[-228:], predictions.reshape(228, 6)[:,0], label='Predicted')
plt.title(f'{data.columns.tolist()[0]} Prediction vs Actual')
plt.ylabel('Value')
plt.xlabel('Years (2005-2024)')
plt.legend()

plt.tight_layout()


best_model.summary()
print(best_hps.values)

image_format = 'svg' # e.g .png, .svg, etc.
image_name = 'UnempfeaturePred.svg'

plt.savefig(image_name, format=image_format, dpi=1200)
plt.show()

"""# Create Prediction Model (Decoder)"""

newModel = Sequential()
newModel.add(tf.keras.Input(batch_size=1, shape=(None,6)))
newModel.add(LSTM(128, return_sequences=True, stateful=True))
newModel.add(Dropout(0.2))
newModel.add(LSTM(6, return_sequences=False, stateful=True))

#newModel.add(Dense(6,))

newModel.compile(optimizer='adam', loss='mean_squared_error', metrics=['accuracy'])

newModel.set_weights(best_model.get_weights())

"""# Predict N timesteps in the future"""

#newModel.reset_states()
newModel.layers[0].reset_states()
#first = newModel.predict(X).reshape(1,1,6)

future_predictions = []


forward_n =  48# Time Steps
nextpred = newModel.predict(X)#.reshape(1,1,6)


for i in range(forward_n):
  X = np.append( X[:, 1:, :] , nextpred.reshape(1,1,6), axis=1)
```

```
  nextpred = newModel.predict(X, verbose=0)
  future_predictions.append(nextpred)

#newModel.layers[0].reset_states()

future_predictions = np.array(future_predictions)

future_predictions = future_predictions.reshape(forward_n, 6)
final = scaler.inverse_transform(future_predictions)#[:,2]

"""# Plot Forecasted Data"""

plt.plot(data.index[-228:], data.iloc[1:,2], label='Actual')
plt.plot(futureData.index[228:228+forward_n], final[:,2], label='Predicted')

plt.title('Average Price 4 Years Prediction vs Actual')
plt.xlabel('Time')
plt.ylabel('Average Price')
plt.legend()
#plt.ylim(250, 400)
plt.grid(True)

image_format = 'svg' # e.g .png, .svg, etc.
image_name = 'Future.svg'

plt.savefig(image_name, format=image_format, dpi=1200)



plt.show()

"""# PLot all forecasted data (subplots"""

fig, axes = plt.subplots(2, 3, figsize=(15, 10))
for i,ax in enumerate(axes.flat):
    ax.plot(data.index[-228:], data.iloc[1:,i], label='Actual')
    ax.plot(futureData.index[228:228+forward_n], final[:,i], label='Predicted')
    ax.set_title(f'{data.columns.tolist()[i]} Prediction vs Actual')
    ax.set_ylabel('Value')
    ax.set_xlabel('Years (2005-2024)')
    ax.legend()

plt.tight_layout()
plt.show()

best_model.summary()
print(best_hps.values)

image_format = 'svg' # e.g .png, .svg, etc.
image_name = 'futurefeaturePred.svg'
```

```
fig.savefig(image_name, format=image_format, dpi=1200)

final[:,2]
```