Machine Learning for Real Estate Time Series Prediction

Fatim Z. Habbab[1] and Michael Kampouridis[2,3]

School of Computer Science and Electronic Engineering,
University of Essex,
United Kingdom
{fh20175, mkampo}@essex.ac.uk

Abstract. Several researchers have demonstrated that real estate investments have improved the risk-adjusted performance of mixed-asset portfolios belonging to institutional investors. In order for these portfolio strategies to be more effective, one could use price predictions (instead of historical data) to optimize weights. The goal of this paper is to investigate the predictive performance on price time series of REITs (real estate investment trusts), stocks and bonds, of five different machine learning (ML) algorithms. These algorithms are: linear regression; support vector regression; gradient boosting; long short-term memory neural networks; and k-nearest neighbour. We run experiments on 90 datasets and compare the ML results to those of an ARIMA model, which is a popular econometric benchmark used in financial time series predictions. Our results show that machine learning algorithms statistically outperform ARIMA. In addition, we find that all machine learning algorithms are able to produce very low root mean square errors, with linear regression and long short-term memory obtaining the lowest error values.

Keywords: machine learning REITs financial time series ARIMA.

1 Introduction

The main reason why investors in financial markets select their investment weights in a portfolio is to maximize their return and/or to reduce the risk associated to their portfolio [3]. In order to reduce the risk associated to their portfolios, many investors have decided to gain exposure to real estate markets that proved to be a stable and resilient market [1, 12, 10]. However, since the direct real estate investment alternative (i.e., purchasing real estate assets in exchange of a given price) is generally an expensive route, many investors have decided to invest in real estate indirectly, i.e. by purchasing shares in listed or non-listed companies that own and manage real estate. Some of these companies are known as REITs (real estate investment trusts) and are listed on major exchanges.

Several authors demonstrated that adding real estate to a mixed-asset portfolio (i.e., already composed of other asset classes such as stocks and bonds)

increases the return and/or reduces the risk associated to that portfolio [2, 21, 19, 9]. Many works in the literature perform portfolio optimization by calculating the optimal weights in a training set and then applying those weights to an unseen test set [13, 6, 19]. A potential disadvantage of this approach is that prices in the test set might be significantly different than prices in the training set. As a result, weights computed using the training set might not fit the test set very well and thus lead to worse portfolio performance (i.e., increased risk and/or reduced return).

To alleviate the above issue, an alternative approach is to try and predict prices in the test set (step 1), and then perform the portfolio optimization task (i.e. calculating the optimal weights) directly in the test set (step 2) [7, 17, 18, 5]. The advantage of this approach is that we focus only on the data period we're interested in (i.e., the test set); as a result, accurate predictions would closely reflect the prices in the test set, and thus lead to a more efficient portfolio selection. However, the quality of the results is very much dependent on the effectiveness of the price predictions.

While the above methodology of price prediction has been used before in mixed-asset portfolios, it has never been used before in portfolios that include REITs. To the best of our knowledge, portfolio optimization with REITs has only taken place by calculating the optimal sets of weights in the training set [22, 8]. This is an important finding since the real estate market has attracted billions of investors worldwide due to its capacity to provide low risk investments.

This work thus focuses on step 1 of the above process, i.e. price prediction. We investigate the performance of five different machine learning algorithms, namely linear regression (LR), support vector regression (SVR), extreme gradient boosting (XGBoost), long-short term memory neural networks (LSTM), and k-nearest neighbors (KNN). We are interested in identifying the algorithm that can best fit the price series data, as these predictions can then be applied to the portfolio optimization task. The above five algorithms are also compared with an auto-regressive integrated moving average (ARIMA) model, which is a common econometric approach used in financial time series prediction.

The rest of this paper is organized as follows. Section 2 presents a brief background on REITs, and Section 3 discusses the methodology of this paper. Our experimental setup is presented in Section 4. Section 5 provides a detailed discussion of the experimental results we obtained by applying machine learning and ARIMA models to our data. Finally, Section 6 concludes the paper.

2 Background

REITs are companies that own and manage real estate. They hold considerable amounts of investments in real estate. Example of REITs include Realty Income Corporation (O), Digital Realty Trust, Inc (DLR), Simon Property Group, Inc (SPG), and so on. REITs provide an investment opportunity that makes it possible for many investors (such as everyday investors, banks, and hedge funds) to

benefit from real estate investments, dividend-based income and steady returns without needing to spend large sums of money to buy a real estate unit.

REITs historically have provided investors with several advantages, including competitive return levels, high, steady dividends, and long-term capital appreciation. This is due to their low correlation with other asset classes that enhances portfolio diversification. This is the main reason why many investors decide to invest in REITs.

There has been some limited attempts to predict REITs prices using machine learning algorithms. For instance, [14] applied a neural network algorithm to predict stock and REIT prices and showed that such algorithm outperformed ARIMA in the prediction accuracy. Other authors compared machine learning algorithms to ARIMA in the prediction of REIT returns ([15, 11, 16]). Such works focused mainly on artificial neural networks relying on multiple variables. In conclusion, while there have been a few works on REITs price prediction, the majority of them have focused on neural networks. This work will consider more algorithms (five in total) to allow a more in-depth investigation of the applicability of machine learning to REITs prices predictions.

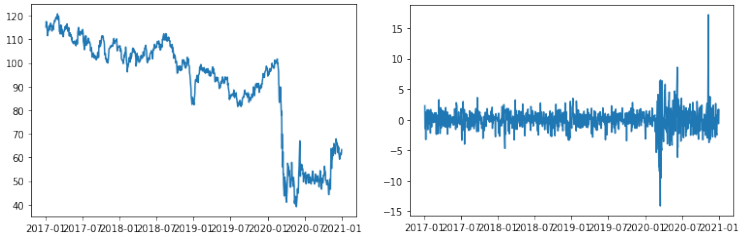
3 Methodology

Before applying the five machine learning algorithms, we first needed to take several data pre-processing steps, which are presented in Section 3.1. We then present the loss function, which is the same across all algorithms, in Section 3.2. Lastly, we briefly present some information on how we used the machine learning algorithms in Section 3.3.

3.1 Data preprocessing

Before being used for price prediction, each time series data is differenced and scaled. Differencing consists of calculating one-step lag for each time point in such a way that $D_t = P_t - P_{t-1}$. For instance, price at time $t1$ will be transformed into $D_{t2} = P_{t2} - P_{t1}$. Figure 1a shows the original SL Green Realty Corp REIT (SLG) time series (before differencing) and Figure 1b shows the differenced time series. As we can observe, the differencing process makes the time series stationary, eliminating its upward trend and making the average constant over time. Stationarity is important in time series analysis as several models (including ARIMA) assume that data are independent of each other. Since market price time series are often featured by time dependency (i.e., each time point depends on the past ones), it is necessary to remove such dependency in order to apply our prediction models.

Once D_t has been obtained, its values are then scaled to be in the range of 0 and 1, according to the following transformation, presented in Equations 1 and 2:



(a) SLG original time series (b) SLG differenced time series

Fig. 1: Differencing example

$$D_{std} = \frac{(D - D_{min})}{(D_{max} - D_{min})} \tag{1}$$

$$N_t = D_{std} * (max - min) + min \tag{2}$$

where D_{std} is the standardized value of each variable (in this case the differenced price D), D_{min} is the minimum value for D, D_{max} is the maximum value for D, and min and max are the lower and upper range limits, in this case 0 and 1 respectively. So in the end, D_t takes the value of N_t.

We present an example of the differencing and scaling processes in Table 1, which presents sample data for the SLG time series between the time period between 01 January 2021 and 15 January 2021. The first column presents the different time steps, the second column the price P_t of the given security, the third column the one-lag value of P_t, the fourth column the differenced D_t value, the fifth column the scaled N_t variable, and the last two columns the lagged values for N_t. As we can observe, at time $t2$, D_t is equal to the difference between P_t and P_{t-1}, which is equal to $61.70 - 63.88 = -2.19$. Similarly, the first-order difference at $t3$ is equal to -2.50, and so on. The fifth column contains the D_t values after scaling, i.e., N_t. As we can observe the independent variable is scaled to the range between 0 and 1. For instance, after normalization at time step $t2$, the value of -2.19 becomes 0.07.

The dependent variable is N_t, while the independent variables are past observations of N_t, i.e., N_{t-1}, N_{t-2}, N_{t-3}, ..., N_{t-n}. The n value is decided on the basis of the Akaike Information Criteria (AIC) optimization. AIC is a metric widely used for model selection [20, 23]. Each dataset has a different value for n, thus a different number of features.

After having obtained the predicted time series, we revert the differencing and scaling process to calculate the loss function described in Section 3.2.

3.2 Loss function

The machine learning models used in this paper are evaluated by using one-day-ahead predictions, rather than out-of-sample predictions. The latter is when

Table 1: Example of time series differencing and feature selection.

t	P_t	P_{t-1}	D_t	N_t	N_{t-1}	N_{t-2}
t1	63.88	-	-	-	-	-
t2	61.70	63.88	-2.19	0.07	-	-
t3	59.20	61.70	-2.50	0	0.07	-
t4	59.40	59.20	0.20	0.64	0	0.07
t5	60.21	59.40	0.80	0.79	0.64	0
t6	60.24	60.21	0.03	0.60	0.79	0.64
t7	61.93	60.24	1.69	1	0.60	0.79
t8	61.26	61.93	-0.67	0.44	1	0.60
t9	61.62	61.26	0.36	0.68	0.44	1
t10	63.26	61.62	1.64	0.99	0.68	0.44

today's N_t value (time-step 0) is known and is used to forecast the value of tomorrow (time-step 1). However, tomorrow's value is unknown and cannot be used to forecast the value two days ahead. Hence, this method uses the value forecast at time-step 1 to forecast the value at time-step 2, and so on. This method is commonly used in many forecasting problems, but is not appropriate in this article's task. Its main disadvantage is that it uses forecast values as features, which causes the regression error to propagate. In addition, in the real world we would have the option of retraining the machine learning algorithms on a daily basis; thus we would have access to tomorrow's *real* price, rather then the forecast price, as it happens with the out-of-sample predictions. Hence, this paper uses the one-day-ahead forecast. In this method, the price today (time-step 0) is known, and is used to forecast tomorrow's price (time-step 1). Then tomorrow's *real* price is used to forecast the price at time-step 2, and so on. This second method is expected to be more accurate, because we are using the actual values as features, instead of predictions.

For our problem, we use the *root mean square error* as loss function, which is presented in Equation 3:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(P_t - \hat{P}_t)^2}{N}}, \tag{3}$$

where P_t refers to the actual value of the price, $\hat{P}_t$ is its predicted value, and N is the number of observations. Please note that as it was explained in Section 3.1, the differenced and scaled values are reverted back to their original price values, so that the loss function can be calculated. That is why in the above equation we use P_t and not D_t or N_t.

3.3 Machine learning algorithms

To apply our machine learning algorithms, we used the following python libraries: *sklearn*, *xbgoost*, and *keras*. The functions used to fit the algorithm to the training data include `sklearn.linear_model.LinearRegression`, `sklearn.svm.SVR`,

`xgboost`, `Sequential`, and `sklearn.neighbors.KNeighborsRegressor` (KNN). The parameters included in such functions were determined using a grid search method which is described in Section 4. Once the algorithms were fit to the training data, they were then applied to the test set by using the `predict` attribute of the relevant model.

4 Experimental setup

The goal of these experiments is to predict REITs prices through five machine learning algorithms: linear regression (LR), support vector regression (SVR), XGBoost, LSTM, and KNN. Since our eventual goal in future research is to optimize a multi-asset portfolio made of stocks and bonds too, we will apply our models to those asset classes, as well. Moreover, we will compare the results obtained to those of ARIMA models, which is a common econometric benchmark used in financial time series predictions. As explained in Section 1, it is important to have accurate price predictions, as this can lead to improved portfolio optimization results.

4.1 Data

For our experiments, we use daily prices downloaded from *Yahoo!Finance* for stocks and REITs and *Investing.com* for bonds, referred to the period between January 2017 and January 2021 of financial instruments belonging to three asset classes (i.e., stocks, bonds, and real estate), and to three countries (i.e., US, UK, and Australia). For the each of the three markets, we use prices for ten stocks, ten bonds, and ten REITs. Thus, we run our experiments on 90 datasets.

4.2 Experimental parameters

The data was split into three sets: training (January 2017 - June 2019), validation (July 2019 - December 2019), and test (January 2020 - December 2020). The validation set was used to decide on the experimental parameters through a grid search tuning phase. We performed tuning for each individual dataset, thus each dataset has its own tailored experimental parameters. Next we present the range of values that were considered during grid search for each algorithm.

For linear regression, we did not perform parameter tuning as the the library provided in *Python* did not have parameters to be tuned. With regards to the support vector regression algorithm, we selected the kernel function which provided the best results among the other functions (*Linear*, *Gaussian*, *Radial Basis Function*). Regarding the XGBoost algorithm, the main parameter was the number of estimators (or number of trees) whose value was chosen between 100, 200, 500, and 1000. The parameters for the LSTM algorithm were the number of epochs (selected on the basis of the early stop callback method), the batch size (which might be 8, 16, or 32), and the number of neurons (which might be 1, 2, 3, or 4). For the KNN algorithm, the number of neighbors might be 5, 8, or 10.

4.3 Benchmark: ARIMA

The *Autoregressive Integrated Moving Average* (ARIMA) class of models is used to analyze a time series structure. It predicts the value of a variable (e.g., current market price) using the historical values of the same variable and its error distribution. It is used as a benchmark, as it is commonly used in finance with time series prediction.

Given time series X_t, an ARIMA model of order (p, d, q) contains three components, the autoregression model of order p, differencing of order d, and the moving average model of order q. Equation 4 shows the mathematical form of ARIMA.

$$x_t = c + \sum_{i=1}^{p} \phi_i x_{t-i} + \epsilon_t + \sum_{i=0}^{q} \theta_i \epsilon_{t-1} \tag{4}$$

where ϕ denotes the autoregression coefficient, θ refers to the moving average coefficient, and ϵ refers to the error rate of the autoregression model at each time point.

In order to fit the ARIMA model for each of the training datasets, we select the p, d, and q order based on the AIC criterion, as previously mentioned in Section 3.1. In other words, the best ARIMA model is found through a search of the minimum AIC value.

5 Results

5.1 Summary statistics

In Table 2 we present summary statistics of RMSE distributions for the REITs. As we can observe, there is a large improvement from ARIMA to machine learning models in terms of average, standard deviation, and minimum-maximum range. While the average from ARIMA is 67.91, the average RMSE value obtained from machine learning models is 4.97 (LR), 5.25 (SVR), 5.92 (XGBoost), 5.06 (LSTM), and 5.12 (KNN). The standard deviation from ARIMA is 183.50, while that of machine learning models is 13.06 (LR), 13.52 (SVR), 14.61 (XGBoost), 13.15 (LSTM), and 13.44 (KNN). The RMSE distribution obtained from ARIMA presents higher skewness and kurtosis values than that resulting from machine learning models, meaning that in the first case it is more likely to observe higher than the mean values and extreme values.

In order to determine whether there are any significant differences between our regression algorithms, we perform the Friedman test, which is a non parametric test based on the mean rank of all algorithms across all data sets. The null hypothesis is that all algorithms perform similarly at the 95% confidence level. The mean ranks and the Bonferroni post-hoc test, which compares the control (best) algorithm against each of the others, are presented in Table 3. As we can observe, LR has the best rank (1.57), followed by LSTM (2.43). There is

Table 2: RMSE values for REITs.

Algorithm	Mean	Standard Deviation	Skewness	Kurtosis	Max - min
ARIMA	67.91	183.50	4.28	19.84	967.57
LR	**4.97**	**13.06**	4.24	19.71	**68.95**
SVR	5.25	13.52	4.17	19.08	70.96
XGBoost	5.92	14.61	**3.99**	**17.61**	75.67
LSTM	5.06	13.15	4.17	19.09	69.01
KNN	5.12	13.44	4.23	19.67	70.93

Table 3: Statistical test results according to the non-parametric Friedman test with the Bonferroni's post-hoc test for REITs. Values in bold represent a statistically significant difference.

Algorithm	Average Rank	p_{Bonf}
LR (c)	1.57	-
LSTM	2.43	0.36
KNN	3.17	**4.60e-03**
SVR	3.30	**3.33e-04**
XGBoost	4.53	**4.09e-09**
ARIMA	6.00	**2.20e-19**

no statistical difference between LR and LSTM, but LR statistically outperforms KNN, SVR, XGBoost, and ARIMA.

Table 4 shows summary statistics of RMSE distributions for stocks. There is again a large improvement from ARIMA to machine learning models in terms of average, standard deviation, and minimum-maximum range. While the average from ARIMA is 206.48, the average RMSE value obtained from machine learning models is 15.79 (linear regression), 16.08 (support vector regression), 19.50 (XGBoost), 15.82 (LSTM), and 16.36 (KNN). The standard deviation from ARIMA is 405.08, while that of machine learning models is 27.77 (linear regression), 28.08 (support vector regression), 33.43 (XGBoost), 27.84 (LSTM), and 28.84 (KNN). According to the skewness and kurtosis values, the RMSE distribution obtained from ARIMA presents higher values than that resulting from machine learning models, meaning that in the first case it is more likely to observe higher than the mean values and outliers. Results from the Friedman test are reported in Table 5. Linear regression again ranks first (1.73), but only statistically outperforms XGBoost and ARIMA.

Table 6 presents summary RMSE distribution statistics for bonds. In this case, machine learning appear to perform better than ARIMA with regards to the RMSE average values, standard deviation, and maximum-minimum range. As we could observe, the average RMSE value for ARIMA is 1.07, while for machine learning models we observe an average value of 0.13 (linear regression), 0.24 (support vector regression), 0.27 (XGBoost), 0.13 (LSTM), and 0.13 (KNN). The standard deviation from ARIMA is 1.75, while that of machine learning models is 0.19 (linear regression), 0.58 (support vector regression), 0.64 (XGBoost), 0.18

Table 4: RMSE summary statistics for stocks.

Algorithm	Mean	Standard Deviation	Skewness	Kurtosis	Max - min
ARIMA	206.48	405.08	3.94	18.20	2159.11
LR	**15.79**	**27.77**	**3.26**	**12.53**	**138.84**
SVR	16.08	28.08	3.27	12.66	140.81
XGBoost	19.50	33.43	3.14	11.81	166.03
LSTM	15.82	27.84	**3.26**	12.54	139.19
KNN	16.36	28.84	3.27	12.75	144.74

Table 5: Statistical test results according to the non-parametric Friedman test with the Bonferroni's post-hoc test for stocks. Values in bold represent a statistically significant difference.

Algorithm	Average Rank	p_{Bonf}
LR (c)	1.73	-
LSTM	2.43	0.74
KNN	2.90	0.08
SVR	2.93	0.06
XGBoost	5.00	**6.77e-11**
ARIMA	6.00	**5.10e-18**

(LSTM), and 0.20 (KNN). Similarly to the RMSE distribution for stocks, the RMSE distribution for bonds obtained from ARIMA presents higher skewness and kurtosis values than that resulting from machine learning models which means that there is a lower likelihood of observing larger error rates than the average, and a lower probability of observing outliers in the case of machine learning models. It is worth mentioning that all models' RMSE values are quite low (including ARIMA's), when compared to the earlier performance in REITs and stocks. This is mainly due of the fact that bond time series tend to be less volatile than stock and REIT time series, and thus more predictable. In addition, Table 7 presents results from the Friedman test, where LSTM ranks first with an average rank of 1.87 and statistically outperforms SVR, XGBoost, and ARIMA at the 5% significance level.

Table 6: RMSE values for bonds.

Algorithm	Mean	Standard Deviation	Skewness	Kurtosis	Max - min
ARIMA	1.07	1.75	3.44	13.83	8.98
LR	**0.13**	0.19	4.25	20.73	1.09
SVR	0.24	0.58	5.22	28.04	3.32
XGBoost	0.27	0.64	5.20	27.87	3.67
LSTM	**0.13**	**0.18**	**4.08**	**19.47**	**0.99**
KNN	**0.13**	0.20	4.13	19.71	1.09

Table 7: Statistical test results according to the non-parametric Friedman test with the Bonferroni's post-hoc test for bonds. Values in bold represent a statistically significant difference.

Algorithm	Average Rank	p_{Bonf}
LSTM (c)	1.87	-
LR	2.40	1.35
KNN	2.53	0.84
SVR	3.80	**3.14e-04**
XGBoost	4.80	**6.29e-09**
ARIMA	5.60	**5.43e-14**

5.2 Discussion

Through our experimental results, we have demonstrated that machine learning algorithms have statistically outperformed ARIMA. In fact, we observed lower average RMSE values in all three cases of stocks, bonds and REITs price prediction. In addition to this, we noticed that the RMSE distributions obtained from machine learning algorithms was less volatile than that obtained from ARIMA, less right-tailed, and light-tailed, indicating a lower chance of observing high error rates.

These results might be explained by the limited ability of ARIMA models to predict outliers. In fact, we observed some extreme error levels for the ARIMA results, especially in the case of stocks and REITs that are more volatile than bonds, and thus more difficult to predict at some points. This is mainly due to the fact that the ARIMA model order has been decided on the whole training set once before being applied to the test set.

With regards to the machine learning algorithms, linear regression and LSTM are the ones that have consistently shown the best performance across the three asset classes. More specifically, LR ranks first in both REIT and stock cases, making them the most preferable algorithms for these categories. This good performance is likely because of the fact that prices are linearly correlated to past prices, which is something LR can leverage on. With regards to the less volatile class of bonds, LSTM ranks first, although it does not statistically outperform LR and KNN.

5.3 Computational times

The computational times for the majority of algorithms are similar. On average, ARIMA took 0.168 minutes to run, while LR, SVR and KNN took between 0.2 and 0.3 minutes. LSTM was the most computationally expensive at 1.818 minutes. But this difference in runtime is not of concern, as usually such algorithms are ran offline, and only their models are run in real time. Besides, such algorithms' computational times can be reduced by parallelization processes [4].

6 Conclusion

This work contributed to the current literature about multi-asset portfolio optimization including REITs. Instead of using historical prices, as usually happens in the literature, we introduced the price prediction by using machine learning algorithms. Our experimental analysis indicates that machine learning models perform better than ARIMA models when applied to one-step-ahead price prediction. This is explained by the lower average RMSE values, with lower volatility, skewness and kurtosis. Among the machine learning models, LSTM and linear regression are the best performing ones with a statistically significant difference with respect to ARIMA, as well as other ML algorithms.

Future work will focus on enhancing the machine learning predictive models by including additional features, and considering more algorithms. As this work only considered lagged features (e.g. normalized prices at time step $t-1$), it would be worth investigating the effect of other variables, such as technical analysis indicators.

References

1. Akinsomi, O.: How resilient are reits to a pandemic? the covid-19 effect. Journal of Property Investment & Finance (2020)
2. Andrew, M., Glenn, M.: Public and private real estate in a mixed-asset portfolio. Journal of Real Estate Portfolio Management **9**(3), 193–203 (2003)
3. Brabazon, A., Kampouridis, M., O'Neill, M.: Applications of genetic programming to finance and economics: past, present, future. Genetic Programming and Evolvable Machines **21**(1), 33–53 (2020)
4. Brookhouse, J., Otero, F.E., Kampouridis, M.: Working with OpenCL to speed up a genetic programming financial forecasting algorithm: Initial results pp. 1117–1124 (2014)
5. Chen, W., Zhang, H., Mehlawat, M.K., Jia, L.: Mean–variance portfolio optimization using machine learning-based stock price prediction. Applied Soft Computing **100**, 106943 (2021)
6. Fisher, J.D., Sirmans, C.: The role of commercial real estate in a multi-asset portfolio. Journal of property management **59**(1), 54–9 (1994)
7. Freitas, F.D., De Souza, A.F., De Almeida, A.R.: Prediction-based portfolio optimization model using neural networks. Neurocomputing **72**(10-12), 2155–2170 (2009)
8. Geiger, P., Cajias, M., Fuerst, F.: A class of its own: the role of sustainable real estate in a multi-asset portfolio. Journal of Sustainable Real Estate **8**(1), 190–218 (2016)
9. Habbab, F., Kampouridis, M., Voudouris, A.: Optimizing mixed-asset portfolios involving reits (2022)
10. Jain, P.: J-reit market quality: impact of high frequency trading and the financial crisis. Available at SSRN 2972092 (2017)
11. Jain, S., Mandal, P., Singh, B., Kulkarni, P.V., Sayed, M.: Prediction of stock indices, gold index, and real estate index using deep neural networks. In: Cybernetics, Cognition and Machine Learning Applications, pp. 333–339. Springer (2021)

12. Jayaraman, B.: Building Wealth Through REITS. Marshall Cavendish International Asia Pte Ltd (2021)
13. Lekander, J.R.: Real estate portfolio construction for a multi-asset portfolio. Journal of Property Investment & Finance (2015)
14. Li, R.Y.M., Fong, S., Chong, K.W.S.: Forecasting the reits and stock indices: group method of data handling neural network approach. Pacific Rim Property Research Journal **23**(2), 123–160 (2017)
15. Lian, Y.M., Li, C.H., Wei, Y.H.: Machine learning and time series models for vnq market predictions. Journal of Applied Finance and Banking **11**(5), 29–44 (2021)
16. Loo, W.K.: Predictability of hk-reits returns using artificial neural network. Journal of Property Investment & Finance (2019)
17. Ma, Y., Han, R., Wang, W.: Prediction-based portfolio optimization models using deep neural networks. IEEE Access **8**, 115393–115405 (2020)
18. Mishra, S.K., Panda, G., Majhi, B., Majhi, R.: Improved portfolio optimization combining multiobjective evolutionary computing algorithm and prediction strategy. In: World Congress on Engineering. vol. 1 (2012)
19. Stephen, L.: The return due to diversification of real estate to the us mixed-asset portfolio. Journal of Real Estate Portfolio Management **11**(1), 19–28 (2005)
20. Vrieze, S.I.: Model selection and psychological theory: a discussion of the differences between the akaike information criterion (aic) and the bayesian information criterion (bic). Psychological methods **17**(2), 228 (2012)
21. Webb, J.R., Curcio, R.J., Rubens, J.H.: Diversification gains from including real estate in mixed-asset portfolios. Decision Sciences **19**(2), 434–452 (1988)
22. Wiklund, E., Flood, J.H., Lunde, J.: Why include real estate and especially reits in multi-asset portfolios? (2020)
23. Yamaoka, K., Nakagawa, T., Uno, T.: Application of akaike's information criterion (aic) in the evaluation of linear pharmacokinetic equations. Journal of pharmacokinetics and biopharmaceutics **6**(2), 165–175 (1978)