UNIVERSITY OF ECONOMICS AND LAW

FACULTY OF INFORMATION SYSTEMS

FINAL REPORT

Time Series Analysis and Forecasting

TOPIC: **Comparison of Closing Stock Price Prediction Performance Using Deep Learning Methods**

Lecturer: Le Hoanh Su, Ph.D.
Group: ***lan***

Ho Chi Minh City, May 03rd, 2023

Members of Group lan

No.	**NAME**	**STUDENT ID**	**ROLE**	**Task**	**Contribution rate**
1	Nguyen Hoang Minh Nhat	K204161996	Leader	Evaluation, Data Processing	25%
2	Bui Van Toan	K204162002	Member	Building Model	25%
3	Nguyen Quang Tuan	K204162005	Member	Feature Selection: L1 regularization (Lasso)	25%
4	Do Quang Huy	K204160662	Member	Feature Selection: Backward Feature Selection	25%

Table Catalog

Image Catalog

Chapter I: Theoretical Overview

1.1 Theory of Deep Learning Algorithms

1.1.1 LSTM Algorithm Theory

1) LSTM Neural Network

The Long Short-Term Memory Neural Network, also commonly known as "LSTM", is a special type of RNN, capable of studying dependent dragons. It was introduced by Hochreiter & Schmidhuber in 1997. They work very well in various problems and are now widely used. LSTM is explicitly designed to avoid long-term dependency problems. Remembering information for a long time is actually the default behavior of this algorithm, so it doesn't have a hard time in our learning problem.

All RNN neural networks take the form of a series of repeating modules of neural networks. In standard RNNs, this repeater module will have a very simple structure, such as a single fishy layer.

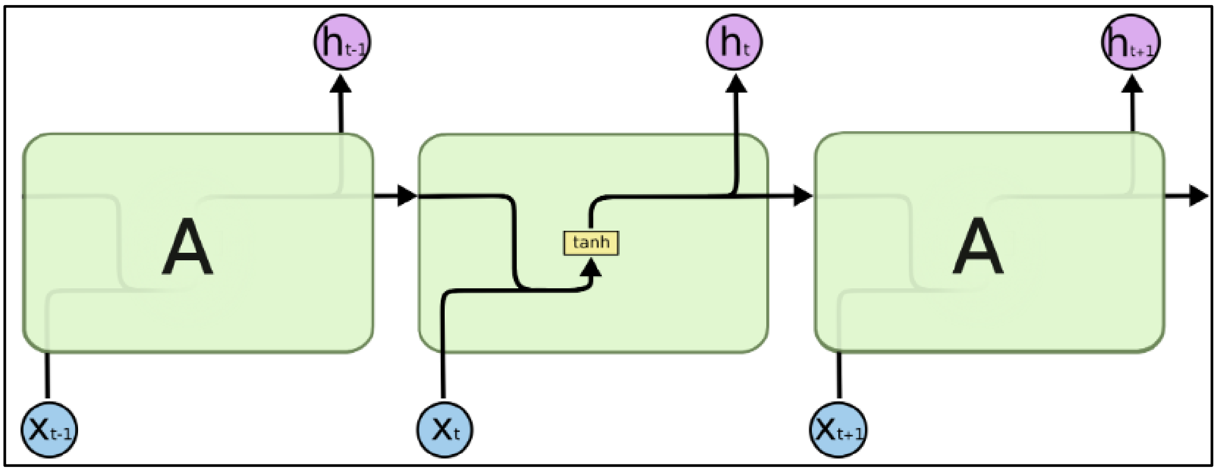


Figure 1. 1: The repeater module in a standard RNN contains a single class

LSTM networks also have the same structure as this chain, but the repeater module has a different structure. Instead of having a single layer of neural networks, there are four layers, which interact in a very special way.

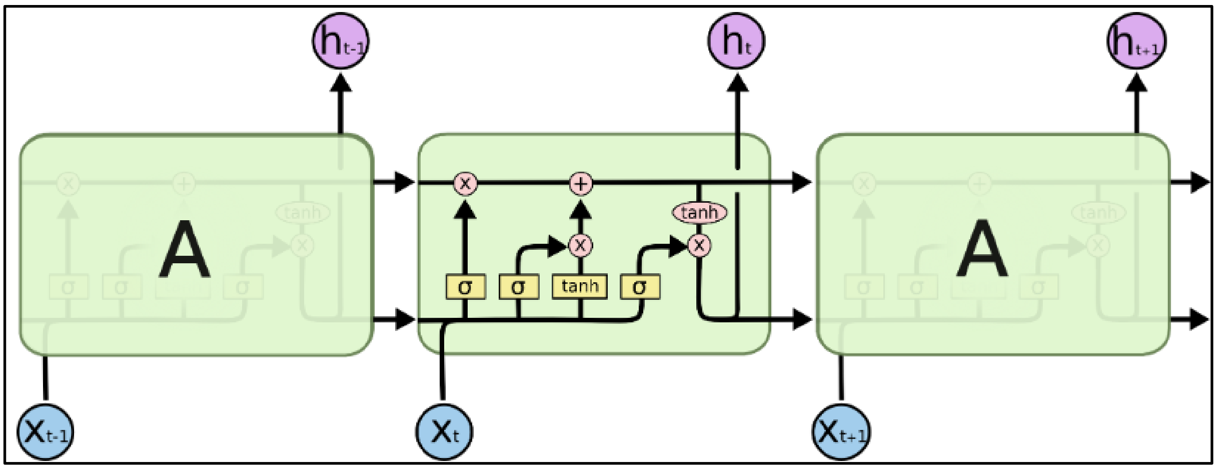


Figure 1. 2: The repeatable module in LSTM contains four interactive layers

An overview of the LSTM conversion equations is as follows:

i_t = σ(+)W_i . [h_{t-1}, x_t]b_i	(1)
f_t = σ(+)W_f . [h_{t-1}, x_t]b_f	(2)
Ot = σ(Wf .[ht-1,xt] + bf)	(3)
Ct = fish(WC. [ht-1,xt] +WC)	(4)
Ct = ft* Ct-1+ it * Ct	(5)
ht = Ot *tanh(Ct)	(6)

Table: LSTM conversion equations

2) Detailed structure of LSTM neural network

a) Forgotten port layer

The first step in LSTM is to decide what information will be removed from the cellular state. This decision is made by a sigmoid layer known as the "forget-and-forget gate layer". Input and , and output a number that is between 0 from 1, for each number in the cell state. The number 1 represents "completely holding this," while 0 represents "completely getting rid of this."$h_{t-1}x_tC_{t-1}$

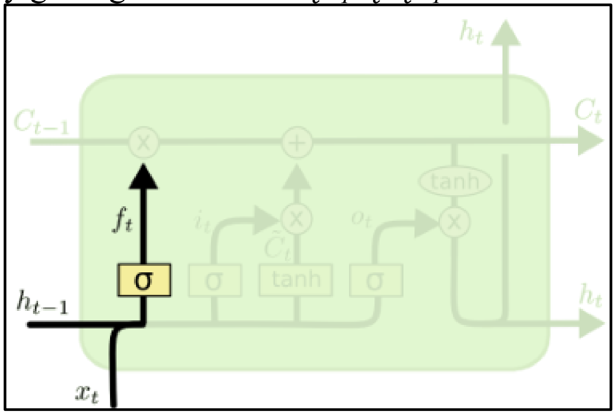


Figure 1. 3: Structure of the forget-and-forget gate layer

Output formula of the forget-gate layer:

f_t = σ(+) (2)W_f . [h_{t-1}, x_t]b_f

b) Input port layer

To decide which new information will be stored in the memory cell state. This has two main parts. First, a sigmoid class called the "input port layer" decides which value will be updated. Next a fishy layer creates a vector of values called new candidates, which can be added to the memory cell state. In the next step, combine both of these to create an update to the memory cell state.$\tilde{C}_t$

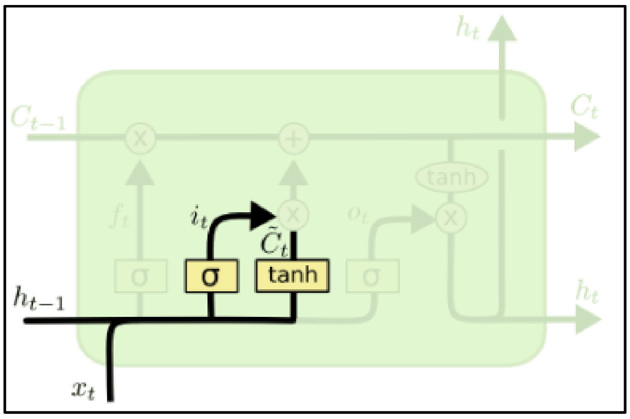


Figure 1. 4: Structure of the input port layer

Operation formula of the input port layer:

i_t = σ(+) (1)W_i . [h_{t-1}, x_t]b_i

= tanh(. +) (4)$\tilde{C}_t$ W_C[h_{t-1}, x_t]W_C

c) Updated port layer

The steps to update the old memory cell state, to the new memory cell state. By proceeding to multiply the old state equally, forgetting the things that have been decided to be forgotten before. Then add *. This is the new candidate value, which is proportional to the degree to which each status value is updated.$C_{t-1}C_t f_t i_t \tilde{C}_t$

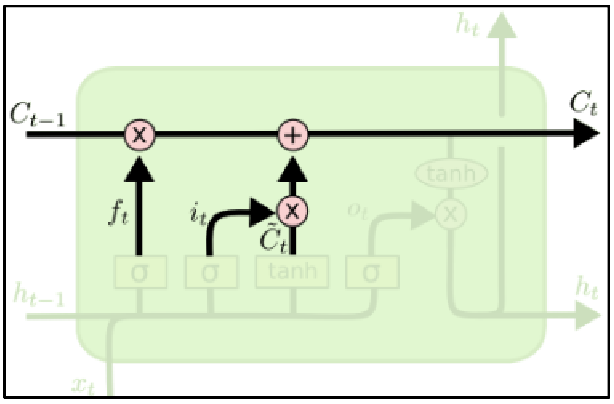


Figure 1. 5: Updated port layer structure

The working formula of the updated port layer:

$$C_t = f_t * C_{t-1} + * (5) i_t \tilde{C}_t$$

d) Output port layer

The output port result will be based on the state of the memory cell, but this is the version of the memory cell that has been selected. First, a sigmoid class is run to decide which part of the memory cell state will be output. The through-memory cell state is then placed in the fishy layer (the values will be between -1 and 1), multiply it by the output of the sigmoid function, so that the output values are the parts that want to make the decision.

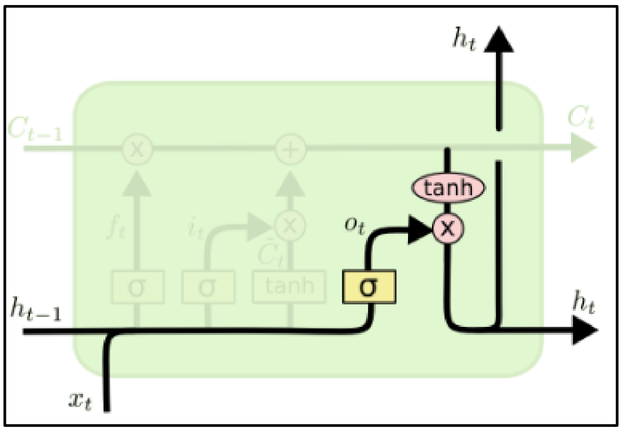


Figure 1. 6: Output port layer structure

Operation formula of the output port layer:

$$O_t = \sigma(\ +\ ) \quad (3) W_f \,.\, [h_{t-1}, x_t] b_f$$

$$= * \tanh() \ (6) \ h_t \ O_t C_t$$

1.1.2 GRU Algorithm Theory

1) GRU Neural Networks

As RNN networks and especially LSTM structures (Section 2.2.1) quickly became popular in the 2010s, several papers began experimenting with simplified architectures in hopes of retaining the main idea of RNNs combining internal state and memory gate mechanisms, but with the aim of speeding up faster calculations. A new variant of the RNN has been created, which is slightly more impressive than the LSTM, but the error column platform is still based on the structure of the LSTM, called the Gated Recurrent Unit, or GRU, introduced by Cho, et al. (2014). It combines the input layer ports and the forgotten port layer, into a single port layer called the "update port layer". It also merges the cell state and the hidden state into one, and makes some other changes. The GRU model is simpler than standard LSTM models and is becoming increasingly popular.

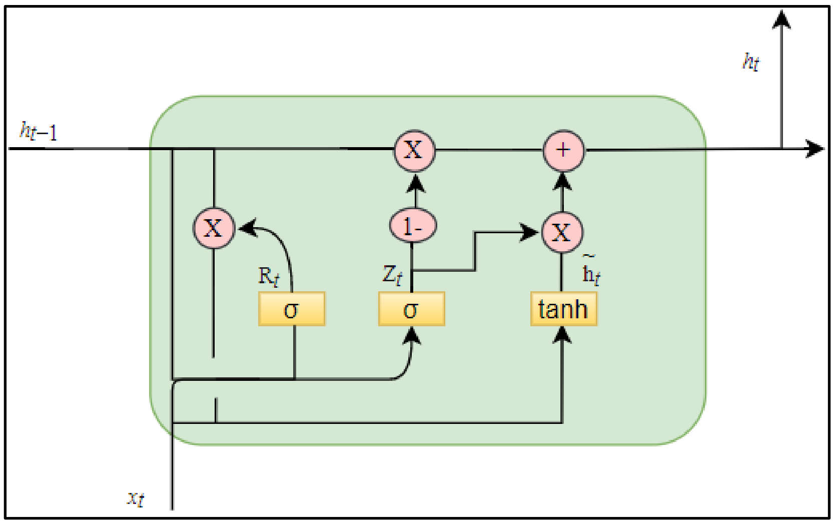


Figure 1. 7: GRU Neural Network Structure

An overview of the structural equations of the GRU is as follows:

= σ(.   (1)Z_t $W_Z[\,h_{t-1}, x_t]$

= σ(.)   (2)R_t $W_R[\,h_{t-1}, x_t]$

= tanh(W.)   (3) $\tilde{h}_t[R_t * \ h_{t-1}, x_t]$

= (1 -)* + * (4)h_t $Z_t h_{t-1} Z_t \tilde{h}_t$

2) Detailed Structure of GRU Neural Networks

a) Updated and reset port layers

Here the three port layers of the LSTM are replaced by two port layers: the update port layer and the reset port. As with LSTM, this gate is triggered by a sigmoid function, forcing their return values to be between 0 and 1. By its mechanism, the reset port controls all previous memory cell states that we might still want to remember. Likewise, an updated port will allow us to control how many new memory cell states are just a memory cell copy of the old state that was regenerated. (Figure 8) illustrates the inputs for both the reset and update ports in the GRU, with the inputs of the current time step and the hidden state of the previous time step. The output of the two ports is provided by two layers that are fully connected to the sigmoid activation function.

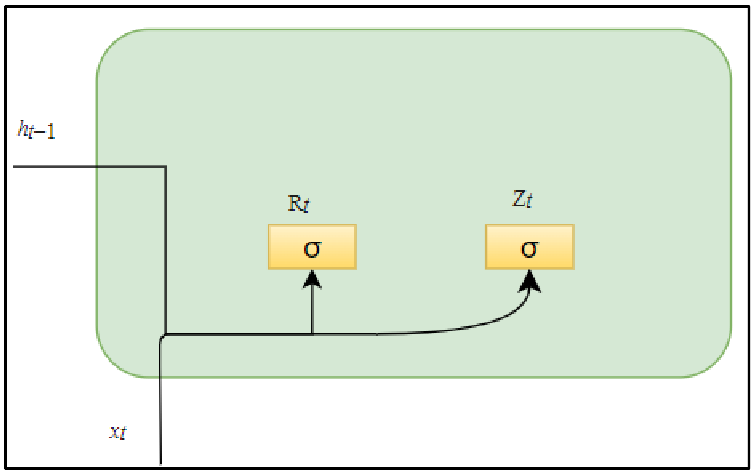


Figure 1. 8: Illustration of the GRU's reset port and update port different from LSTM

The operation formula of the reset and update ports:

= σ(.   (1)Z_t W_Z[h_{t-1}, x_t]

= σ(.)   (2)R_t W_R[h_{t-1}, x_t]

b) Restart port operation

To combine the reset port with the regular update mechanism, which results in the candidate memory cell state at the period t, mathematically we use the following formula:R_t

= tanh(W.)   (3) $\tilde{h}_t$[R_t ∗ h_{t-1}, x_t]

The result is that a candidate memory cell is selected, and in order for them to be operational, we need to update them based on a combination of the operation of the update port and the re-arrival port. Regardless of when the reset port returns a value equal to 1, the update happens like a regular RNN. If the reset port returns a value of 0, then the node at the input port will be a hidden candidate memory cell state. There, any previously existing hidden candidate memory state is set to the default state.$R_t x_t$

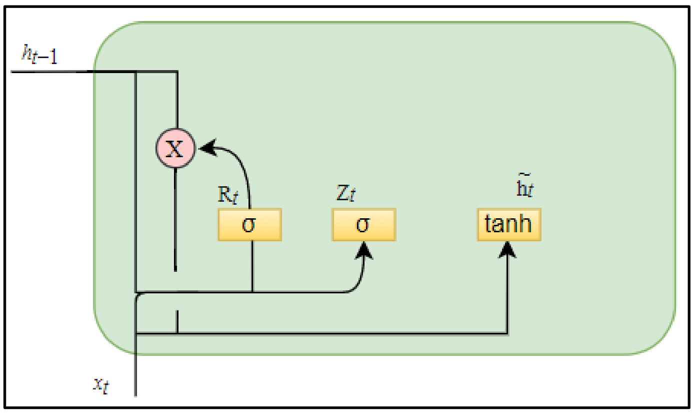


Figure 1. 9: Reset Gateway Operation Flow Procedure

c) Update portal operation

To determine whether the new hidden memory cell state matches the old hidden memory cell state or compared to the new candidate memory cell state. We use the update port. The update portal works by taking the convex combination of each element and . This leads to the final update equation for the GRU:$h_t\tilde{h}_t Z_t Z_t h_{t-1}\tilde{h}_t$

= (1 -)* + * (4)h_t $Z_t h_{t-1} Z_t \tilde{h}_t$

If the update port returns a value of 1, it retains the old hidden state. The information obtained from the discarded inlet port. Conversely, when a value of 0 is returned, the new hidden state is selected as the candidate hidden state.$Z_t x_t Z_t h_t \tilde{h}_t$

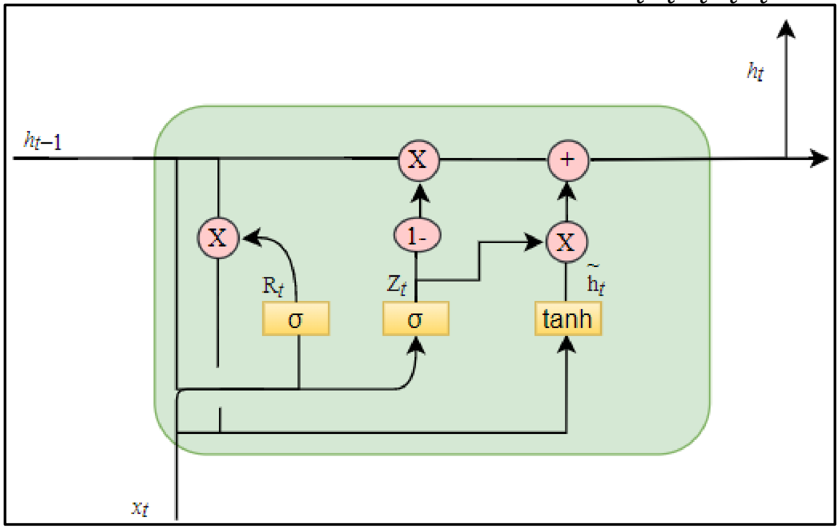


Figure 1. 10: Update gateway workflow

1.2 Theory of performance evaluation methods

1.2.1 Theory of MSE

The Root Mean Square Error (RMSE) is the square root of the average, squared value of all errors. RMSE is considered a general-purpose measure of error, which is a measurement of error that works well in checking numerical predictions. RMSE is a good measure of accuracy, but only for comparing the prediction error of models or division operations, adjusting the model differently for a particular variable, not between variables because it depends on the scale. It is a measure of how well a regression line fits data points. The formula for calculating RMSE is:

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

1.2.2 Theory of RMSE

The Mean Square Error (MSE) is an estimator that measures the average of the squares of error, i.e., the mean squared difference between the predicted and actual values. It is a risk function, which corresponds to the expected value of the squared error loss. It is always non-negative and values close to 0 are better. The formula for calculating MSE is:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n} (\hat{y_i} - y_i)$$

1.2.3 Theory of MAE

Mean Absolute Error (MAE) is a common method used to evaluate errors in predictive activity. In MAE, the different errors are not weighted more or less, but the score increases linearly with the increase in errors. The MAE score is measured by the average of the absolute error values. Absolute is a mathematical function that makes a positive number. Therefore, the difference between the expected value and the predicted value can be positive or negative and will necessarily be positive when calculating the MAE. The MAE value can be calculated as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |(\hat{y_i} - y_i)|$$

1.2.4 Theory of R2

R^2 Adjusted is a measure of the fit (accuracy of the model) that has been adjusted in linear models. It determines the percentage of variance in the target field that is explained by the inputs or inputs.

R^2 adjusted is calculated by dividing the remaining mean squared error by the sum of the mean squared error (which is the sample variance of the target field). The result is then subtracted by 1. adjusted is always less than or equal to . A value of 1 indicates a model that perfectly predicts the values in the target field. A value less than or equal to 0 indicates that the model has no predictive value. In practice, adjusted is usually somewhere between these values.R^2 R^2 R^2

$$R^2 = 1() - \frac{n-1}{n-i} 1 - \frac{\sum_{i=1}^{n} (\hat{y_i} - \underline{y_i})^2}{\sum_{i=1}^{n} (y_i - \underline{y_i})^2}$$

In which:

$\hat{y_i}$: is the predicted value for the ith observation
y_i: is the actual value for the ith observation
$\underline{y_i}$: is the average price for the second observation y_i
n: total number of observations

Chapter II: Experiments

2.1 Data preparation and processing

a) Data collection

Data is an extremely important part of using Artificial Intelligence to make certain predictions, which includes predicting fluctuations in trading prices in the market. The timeline fixed by us is from 2018-04-19 to 2021-12-31 through the library supported by the Python programming language, because in order to ensure that a large enough amount of data is obtained for the model training process, we have agreed to select the data in the timeline of about 3 years as above$pandas$.

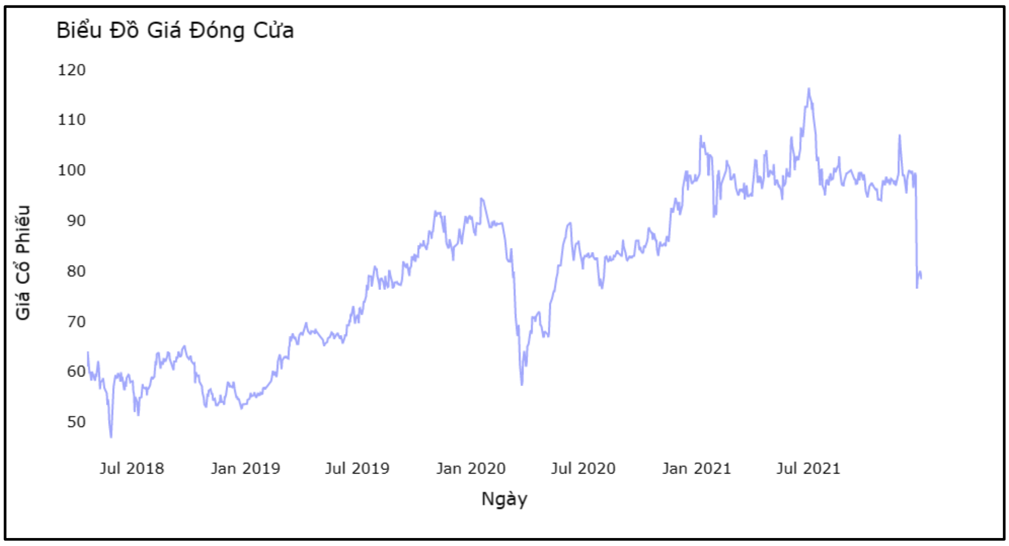


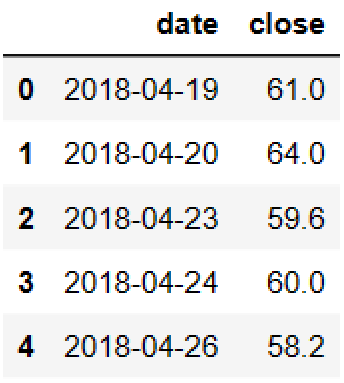
Figure 2. 1: Closing price chart of VCB Bank

b) Setting up the environment

The programming language we use here is with the . For data analysis and restructuring purposes, we use with version, which is a BSD-licensed open source library. The processing of calculations for large arrays, multi-dimensional matrices we use the version library. The data we use is historical data of the Joint Stock Commercial Bank for Foreign Trade of Vietnam. To evaluate the performance of the Deep Learning models, the library named with the version was used in this study. In building two neural networks, GRU and LSTM, we used the library that supports the construction of neural networks that is with the version written in . The visualization of the results to compare the brands of the models in the clearest way, the version libraries, version libraries and version institutes have been used by us. $Python3.9.13Pandas1.5.3Numpy1.23.5\ Scikit - learning1.2.1Tensorflow2.11.0Pythoncufflinks0.17.3plotly5.9.0seaborn0.12.2$

c) Prepare data and divide training files and test files

Data processing is a very important step when you want to get information from data files that help you make predictions. Since the initial data can have a lot of noise, it is necessary to reduce them so that the results obtained are not affected by these types of data. Besides, we should also handle some non-meaningful attributes of the data file, in order to help improve the performance of the training model. In this data file, we take two main representative variables: Date and Close.



	date	close
0	2018-04-19	61.0
1	2018-04-20	64.0
2	2018-04-23	59.6
3	2018-04-24	60.0
4	2018-04-26	58.2

Figure 2. 2: Variables selected in the Deep Learning model

To ensure the best performance of the model, we have divided the data file into two sub-files: the training file and the test file at 65% and 35% respectively. The statistics tables below provide an overview of the variables in the data file.

X_train	X_test	y_train	y_test
588	310	588	310

Table 2. 1: Split the training and test data files.

d) Handling of foreign values

Extraneous values are a data item or object that is significantly different from the rest of the data file known as normal data values. Analysis to detect foreign values is known as the foreign value pool. To detect foreign values, we can use mathematical formulas on the data file or use statistical methods.

Looking at the graph below, we can see that the distribution of the data is really not good, so we decided to do a transformation to bring the data back to a more standardized form.

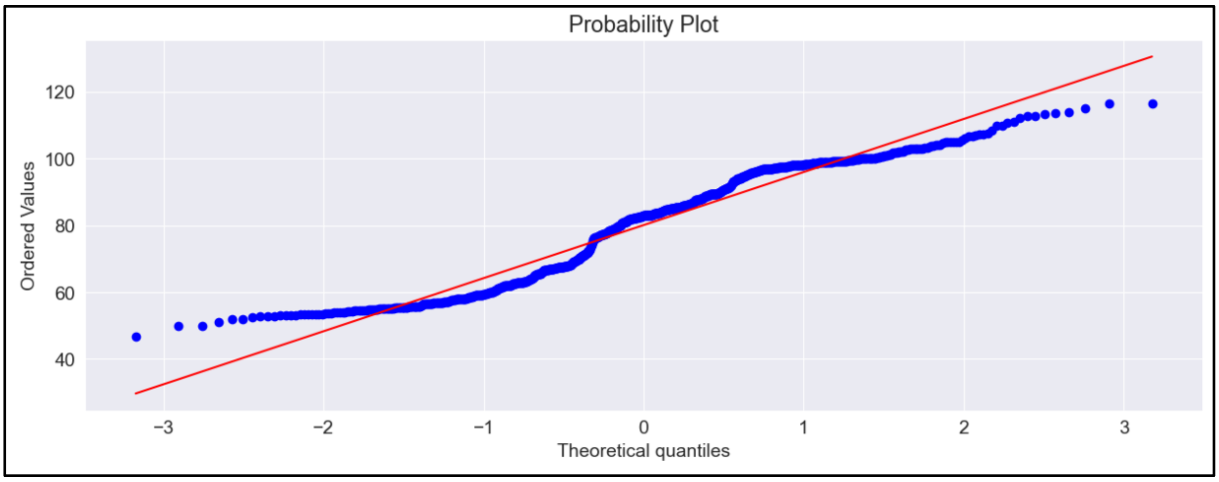


Figure 2. 3: The QQ-Plot chart shows the distribution of the data file

In this project, we use the Min-MaxScaler method with the support of the Sklearn Institute. To process the data to the most appropriate form before proceeding to run the Deep Learning models. The formula is as follows:

$$M = \frac{X - X_{min}}{X_{max} - X_{min}}$$

- M is the new value
- X is the original value
- X_{min} is the smallest value in the data file
- X_{max} is the largest value in the data file

2.2 Feature engineering

2.2.1 Overview of Feature Engineering

2.2.1.1 What is Feature Engineering?

Feature engineering is the process of selecting, manipulating, and transforming raw data into features that can be used as inputs in machine learning models.

In order for machine learning models to forecast better, we need to design and build better input data variables. Input data variables can be units of measurement, such as: closing price, opening price, intraday trading volume, etc.

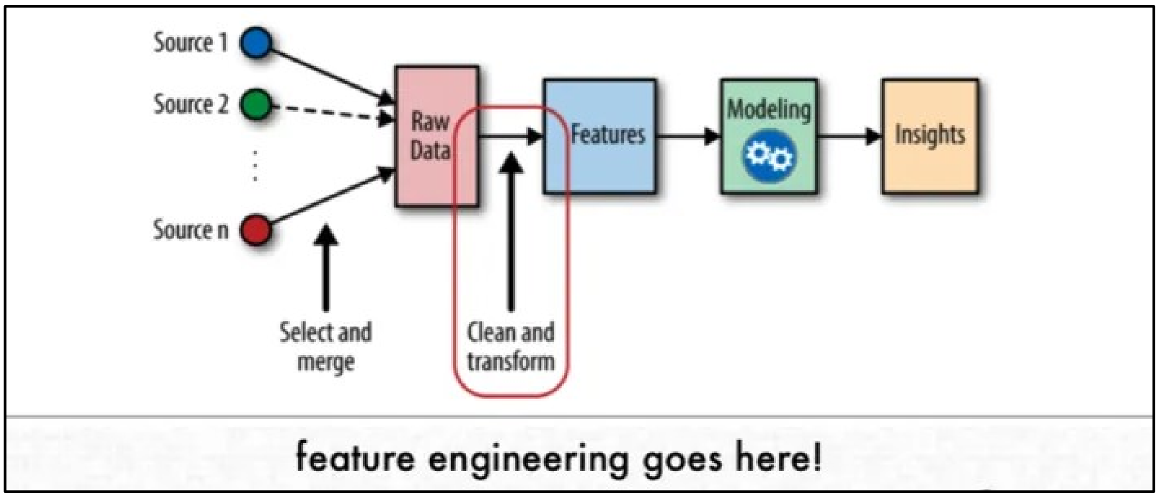


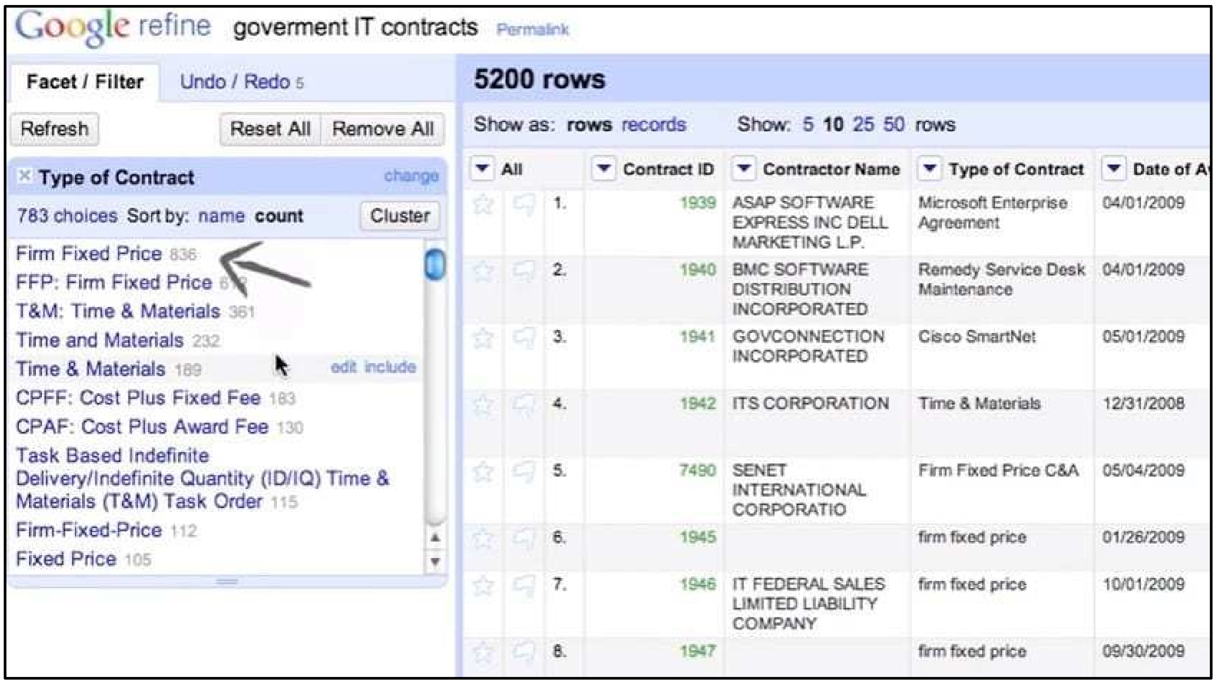
Figure 2. 4: The goal of feature engineeringrring

2.2.2 Techniques in Feature Engineering

There are many techniques in Feature Engineering, depending on the problem and data set we have to apply these techniques, here are some common techniques in Feature Engineering that are commonly used:

Data Cleaning

Data cleaning is the process of dealing with errors or inconsistencies in a data set. It includes steps such as identifying inaccurate data, missing data, duplicate data, and irrelevant data.



5200 rows

Type of Contract

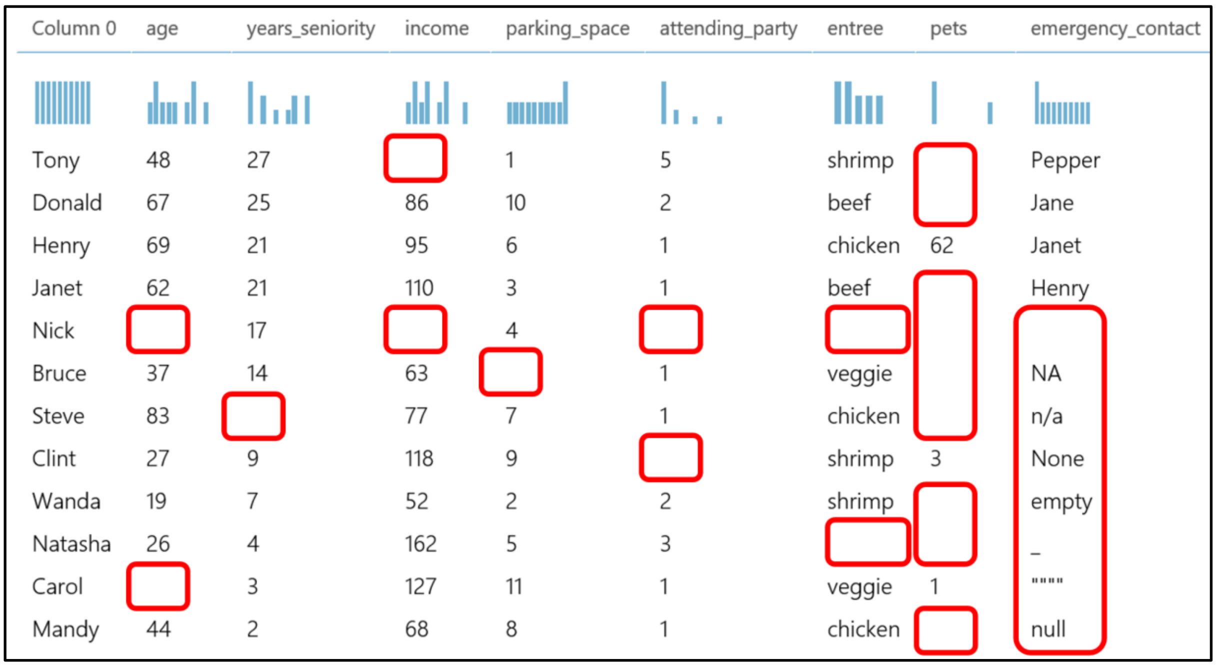
Firm Fixed Price	836
FFP: Firm Fixed Price	6[illegible]
T&M: Time & Materials	361
Time and Materials	232
Time & Materials	189
CPFF: Cost Plus Fixed Fee	183
CPAF: Cost Plus Award Fee	130
Task Based Indefinite Delivery/Indefinite Quantity (ID/IQ) Time & Materials (T&M) Task Order	115
Firm-Fixed-Price	112
Fixed Price	105

All	Contract ID	Contractor Name	Type of Contract	Date of A
1.	1939	ASAP SOFTWARE EXPRESS INC DELL MARKETING L.P.	Microsoft Enterprise Agreement	04/01/2009
2.	1940	BMC SOFTWARE DISTRIBUTION INCORPORATED	Remedy Service Desk Maintenance	04/01/2009
3.	1941	GOVCONNECTION INCORPORATED	Cisco SmartNet	05/01/2009
4.	1942	ITS CORPORATION	Time & Materials	12/31/2008
5.	7490	SENET INTERNATIONAL CORPORATIO	Firm Fixed Price C&A	05/04/2009
6.	1945		firm fixed price	01/26/2009
7.	1946	IT FEDERAL SALES LIMITED LIABILITY COMPANY	firm fixed price	10/01/2009
8.	1947		firm fixed price	09/30/2009

Figure 2. 5: Examples of data cleaning

Handling missing values

Actual data often has a lot of missing values. The cause of the missing values can be data corruption or data failure to be written. Processing missing data is critical in dataset preprocessing because many machine learning algorithms don't support missing values. Some methods to handle missing values such as: deleting rows with missing values, replacing them with values such as mean, median,... , forecasting missing values using machine learning models,... .

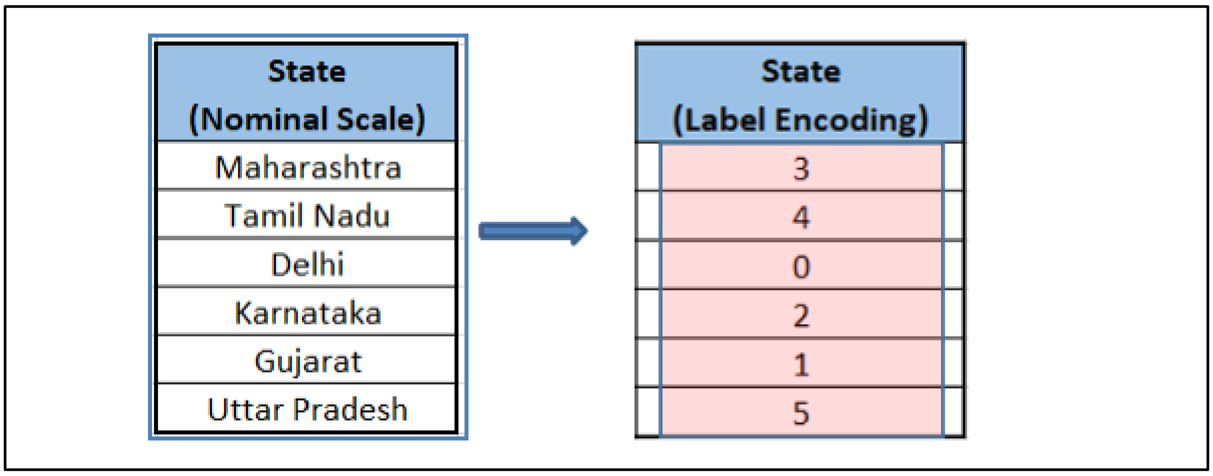


Column 0	age	years_seniority	income	parking_space	attending_party	entree	pets	emergency_contact
Tony	48	27		1	5	shrimp		Pepper
Donald	67	25	86	10	2	beef		Jane
Henry	69	21	95	6	1	chicken	62	Janet
Janet	62	21	110	3	1	beef		Henry
Nick		17		4				
Bruce	37	14	63		1	veggie		NA
Steve	83		77	7	1	chicken		n/a
Clint	27	9	118	9		shrimp	3	None
Wanda	19	7	52	2	2	shrimp		empty
Natasha	26	4	162	5	3			_
Carol		3	127	11	1	veggie	1	""""
Mandy	44	2	68	8	1	chicken		null

Figure 2. 6: Examples of missing value

Encoding

In the dataset, which can contain qualitative data types, to build a machine learning model, we need to have all the data points in the numerical data type. therefore, Label Encoder and OneHotEncoder are used to convert qualitative data types into quantitative data types.

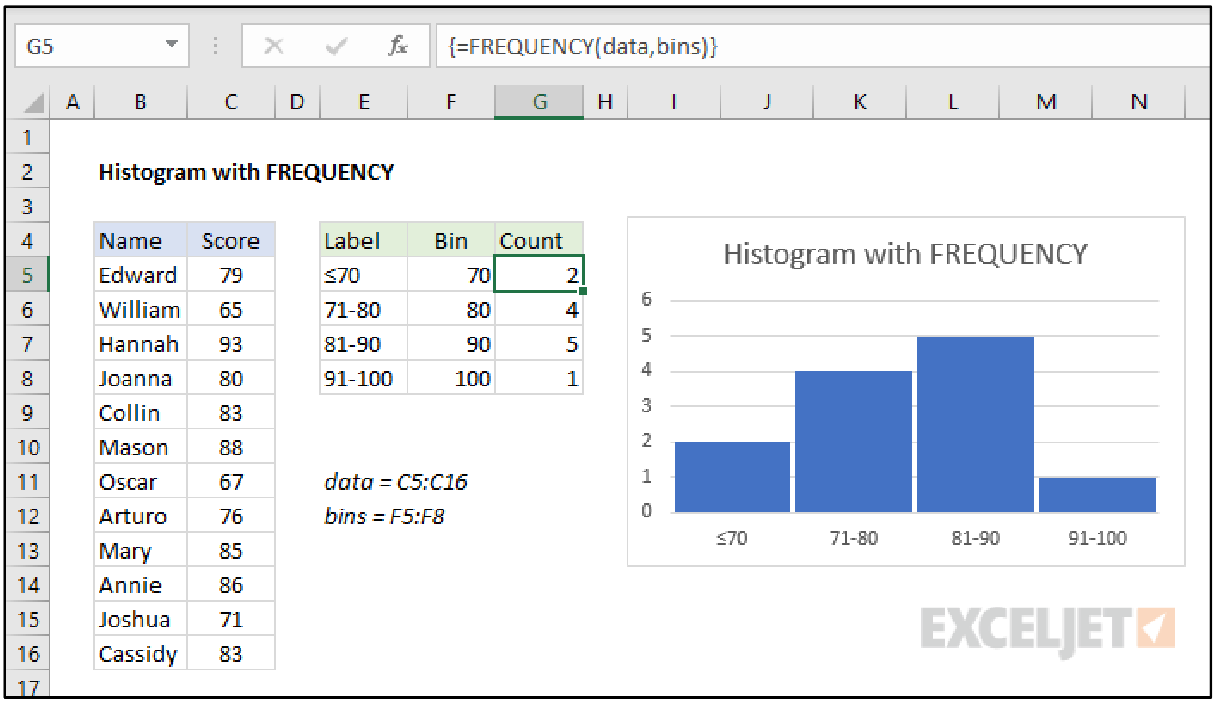


State (Nominal Scale)	State (Label Encoding)
Maharashtra	3
Tamil Nadu	4
Delhi	0
Karnataka	2
Gujarat	1
Uttar Pradesh	5

Figure 2. 7: Examples of encoding

Binning

Binning is the collection of data values into the ranges corresponding to each bin. This technique is used to transform continuous variables into discrete variables.



G5 {=FREQUENCY(data,bins)}

Histogram with FREQUENCY

Name	Score
Edward	79
William	65
Hannah	93
Joanna	80
Collin	83
Mason	88
Oscar	67
Arturo	76
Mary	85
Annie	86
Joshua	71
Cassidy	83

Label	Bin	Count
≤70	70	2
71-80	80	4
81-90	90	5
91-100	100	1

data = C5:C16
bins = F5:F8

Figure 2. 8: Examples of Binning

Handling imbalanced data

Imbalanced data is a common problem in machine learning models, usually with classification problems, Imbalanced data refers to data sets where the target variable has an uneven distribution of observations, i.e. in one variable there is a class label with a very high number of observations and the other label has a very low number of observations. For example, in a banking transaction, it is counted that for every 2000 transactions, there are only 30 "Zero Fraud" and 1970 "Fraudulent" transactions. Here, the "Fraud Space" class is called the majority class, and the much smaller "Cheat" class is called the minority class.

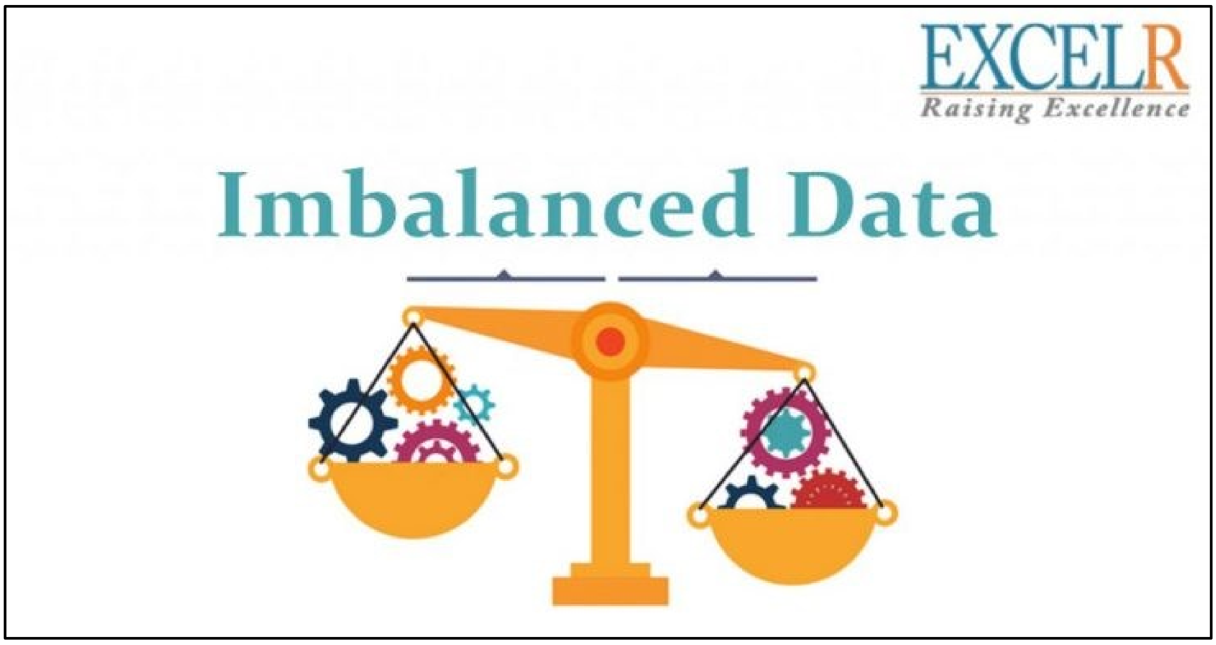


Figure 2. 9: Examples of Imbalanced Data

Handling outliers

Outliers are an observation in a given dataset that is far away from the rest of the observations. That means that the exception value is a lot greater or smaller than the rest of the values in the set. An exception may occur due to an input error or due to an unusual human transaction. Similar to missing value, we can remove or replace outliers with mean, median,...

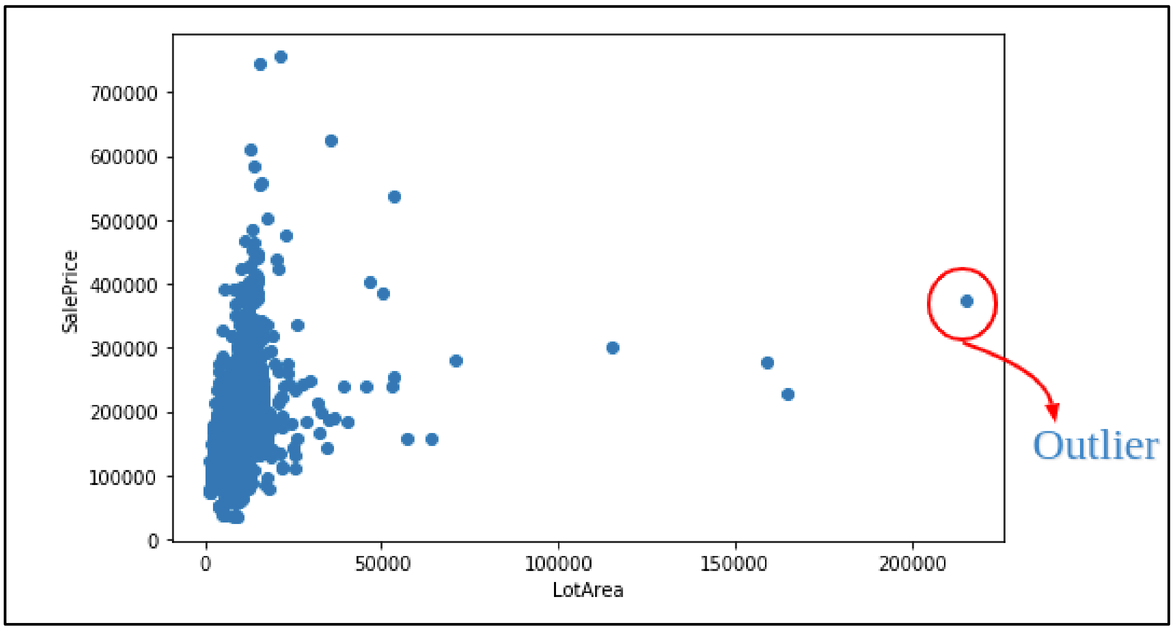
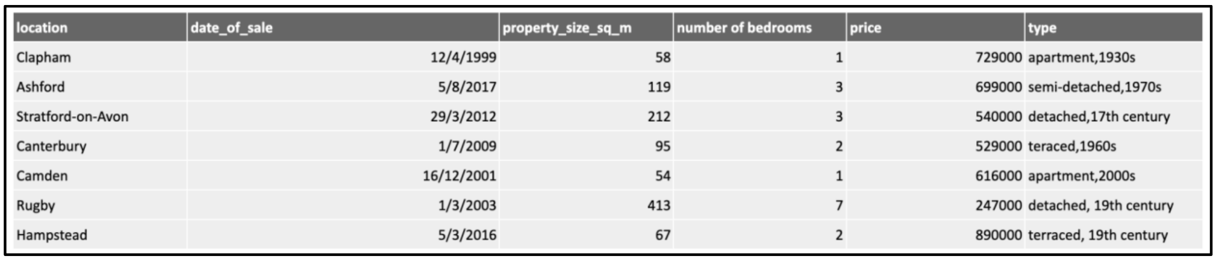


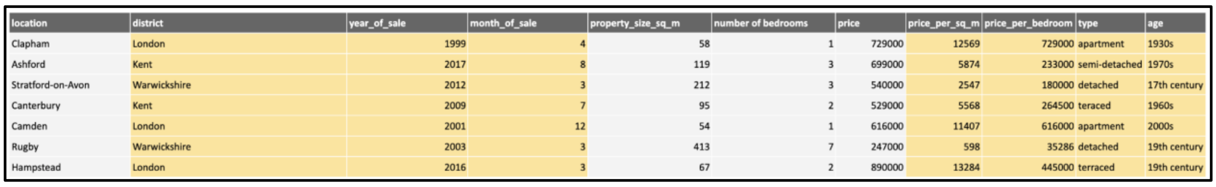
Figure 2. 10: Examples of outliers

Creating Features is the creation of new variables from existing variables. This can be done using simple math operations such as sum or multiplication and even the product of the values between 2 variables in the dataset. **Creating Features,** while derived directly from existing data, when carefully selected to relate to the target can impact the performance of the machine learning model.



location	date_of_sale	property_size_sq_m	number of bedrooms	price	type
Clapham	12/4/1999	58	1	729000	apartment,1930s
Ashford	5/8/2017	119	3	699000	semi-detached,1970s
Stratford-on-Avon	29/3/2012	212	3	540000	detached,17th century
Canterbury	1/7/2009	95	2	529000	teraced,1960s
Camden	16/12/2001	54	1	616000	apartment,2000s
Rugby	1/3/2003	413	7	247000	detached, 19th century
Hampstead	5/3/2016	67	2	890000	terraced, 19th century

Figure 2. 11: Example of creating features 1



location	district	year_of_sale	month_of_sale	property_size_sq_m	number of bedrooms	price	price_per_sq_m	price_per_bedroom	type	age
Clapham	London	1999	4	58	1	729000	12569	729000	apartment	1930s
Ashford	Kent	2017	8	119	3	699000	5874	233000	semi-detached	1970s
Stratford-on-Avon	Warwickshire	2012	3	212	3	540000	2547	180000	detached	17th century
Canterbury	Kent	2009	7	95	2	529000	5568	264500	teraced	1960s
Camden	London	2001	12	54	1	616000	11407	616000	apartment	2000s
Rugby	Warwickshire	2003	3	413	7	247000	598	35286	detached	19th century
Hampstead	London	2016	3	67	2	890000	13284	445000	terraced	19th century

Figure 2. 12: Example of creating features 2

Feature scaling

The actual dataset has many variables with a wide range of values, such as let's look at the home price prediction dataset. It will have many variables such as price, number of rooms, square meter area of the house, etc. It can be guessed that the number of bedrooms will vary from 1 to 5, but the area of the house will range from 500-2000. This is a huge difference in the range of both variables.

To minimize the difference in values for all variables or convert them all into a common value, we use feature scaling to do this. Some methods for feature scaling can be listed such as: Absolute Maximum Scaling, Min Max Scaling, Normalization, Standardization, Robust Scaling.

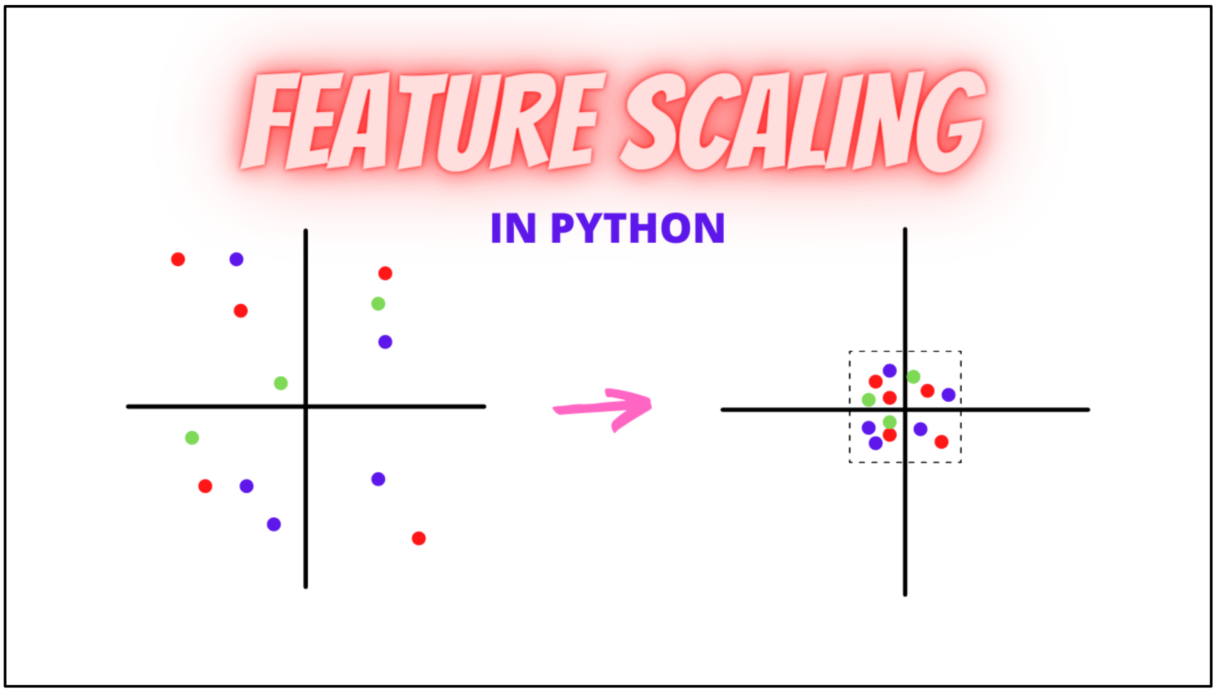


Figure 2. 13: Feature scaling examples

Feature Selection

The input variables that provide machine learning models are called features. Each column in the dataset forms a feature. To train an optimal model, it is necessary to ensure that only essential features are used. If we have too many features, the model can capture unimportant patterns and learn from the noise. The method of selecting important parameters of our data is called **Feature Selection**.

The two Feature Selection methods chosen by the team to implement Deep Learning models are L1 regularization (Lasso) and Backward Feature Selection.

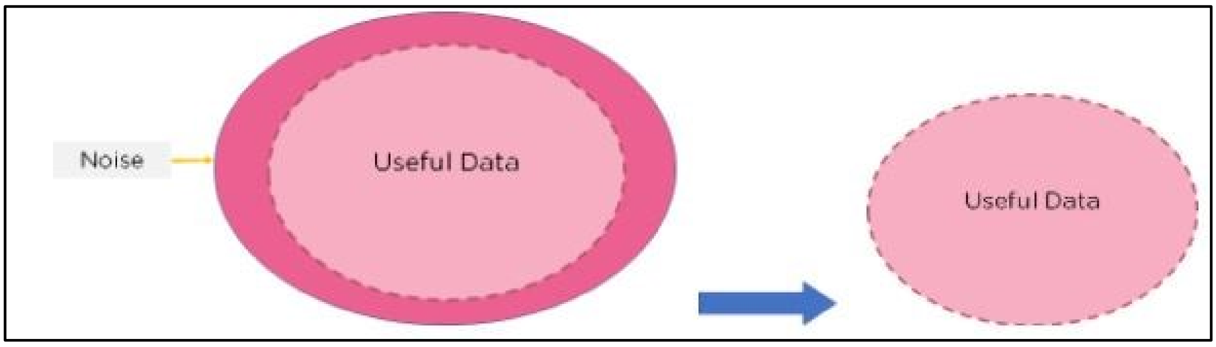


Figure 2. 14: Examples of feature selection

2.2.3 Related research

LASSO: COVID-19 Future Forecasting Using Supervised Machine Learning Models

This study demonstrates the ability of ML models to predict the upcoming number of patients affected by COVID-19 which is now considered a potential threat to humanity. In particular, four standard predictive models, such as linear regression (LR), smallest absolute shrinkage and selection operator (LASSO), assisted vector machine (SVM), and exponential smoothing (ES) were used in this study to forecast risk factors for the COVID-19 epidemic. Three types of predictions are made by each model, such as the number of new cases, the number of deaths, and the number of recoveries over a period of up to 10 days. The results generated by the study demonstrate that this is a promising mechanism to use these methods for the current scenario of the COVID-19 pandemic. The results prove that ES performs best among all the models that follow LR and LASSO perform well in forecasting new confirmed cases, mortality rates as well as recovery rates, while SVM performs poorly in all available predictive scenarios
datasets.

LASSO is a regression model that belongs to linear regression techniques that use shrinkage. Shrinkage in this context refers to narrowing the extreme values of the sample data towards the central values. Thus the shrinkage process makes the LASSO better and more stable, and reduces errors. LASSO is considered a more suitable model for multi-line scenarios. Since the implementation model L1 is regularized and the penalty is added in this case by the magnitude of the coefficients. So, LASSO makes regression simpler in terms of the number of features it is using.

It uses a canonical method to automatically sanction additional features. That is, features that can't get enough regression results can be set to a very small value that is likely to be zero. A regular multivariate regression uses all the features available to it and will assign each person a regression coefficient. In contrast, LASSO regression tries to add them one by one, and if the new feature doesn't improve properly enough to remove the penalty period by including that feature then it can't be added which means zero. Thus the rights legalized by applying a penalty period to the additional feature is that it can automatically make the choice for us. Therefore, the models are performed sparsely with few coefficients in this case of regularization because the process removes the coefficients when their values are zero.

Backward Feature Selection; A Hybrid Model to Forecast Stock Trend Using Support Vector Machine and Neural Networks

Machine learning methods struggle to handle large numbers of input features, is posing an interesting challenge for researchers. Data preprocessing is essential in order to use machine learning methods effectively. Selection of relevant features has become an essential task that accelerates data mining algorithms, improves prediction accuracy, and increases intelligibility. In this study, a new approach is proposed by combining both Assistive Vector Machines (SVMs) and Artificial Neural Networks (ANNs) in predicting stock trends. The SVM technique was introduced to remove irrelevant and redundant variables and then the neural network-based classification technique was used to forecast the stock trend with a bearish feature set. The importance of selecting the correct input features using the SVM before classification is reflected using the accuracy evaluation measures, F-gauge, ROC curve, and AUC.

A feature selection process can be used to remove terms in the training dataset that are statistically uncorrelated with class labels, thus improving both efficiency and accuracy. It implies not only reduced cardinality, which means arbitrarily cutting the number of attributes when building the model, but also the choice of attributes, implying that the model actively selects or removes attributes based on their usefulness to the analysis. The subsets selected each time are compared and analyzed according to the functional formula evaluation. If the subset selected at step m+1 is better than the subset selected at step m+1, then the subset selected at step m+1 can be selected as the optimal subset. The problem of feature selection has been modeled as a mixture.

0-1 integer program. SVM-RFE is an SVM-based feature selection algorithm created by . Also reduce the classification calculation time by reducing the feature set, it can improve the accuracy of the classification ratio. In recent years, many scholars have improved the efficiency of classification in medical diagnosis by taking advantage of this method.

2.2.4 Feature selection method

2.2.4.1 How L1 regularization (Lasso) works

$$\sum_{i=1}^{n}(y_i - \sum_{j} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

Figure 2. 15: Formula L1

The Lasso (Least Absolute Shrinkage and Selection Operator) formula used in the optimization problem reduces the number of input variables in a linear regression model.

The Lasso formula is a variation of the Ridge Regression method. In the Ridge Regression method, we look for the weighted set of the linear regression model so that the sum of the squares of the weighted coefficients does not exceed a predetermined value (usually 1). Meanwhile, Lasso looks for the weighting set of the linear regression model so that the sum of the absolute values of the weighted coefficients does not exceed a predetermined value, where:

- Y is the vector of output values
- X is the matrix of input values
- w is the vector of the weighted coefficients of the linear regression model
- λ is an adjustment parameter, which is used to control how narrow the weighting factors are. As λ increases, the weighting factors are reduced, resulting in a simpler pattern.

2.2.4.1.1 Advantages

Lasso has several advantages over other linear regression techniques:

- Feature Selection: Lasso can automatically select relevant features and remove irrelevant ones, making it useful in height settings where the number of features is much larger than the number of observations.
- Interpretability: Because Lasso creates sparse models, the outcome factors can easily be understood as the impact of each feature on the target variable.
- Robustness: Lasso is less sensitive to exceptions than other linear regression techniques, making it more robust for data that may contain noise or errors.
- Computational efficiency: Lasso can be solved efficiently using convex optimization algorithms, making it computationally feasible even for large data sets.

2.2.4.1.2 Cons

- Bias: Lasso tends to skew the estimates of the coefficients to 0, which can lead to a lack of matches if the normalization parameter is too large.
- Instability: Lasso can generate unstable estimates of coefficients, especially when input features are highly correlated.
- Normalization parameter selection: The selection of the normalization parameter λ is important for the performance of Lasso, and there is no simple rule for choosing the optimal value.

- In practice, Lasso is often used in conjunction with other regularization techniques, such as Ridge regression or Elastic Net, to balance the advantages and limitations of each method.

2.2.4.1.3 Experiments

Here are some of the applications of Lasso:

- Lasso is used in the characteristic selection problem, in which a set of characteristics is selected to form the most accurate prediction model. Lasso helps eliminate unimportant features, minimizes the number of features that need to be considered in a problem, increases accuracy, and minimizes calculation time.
- Lasso is used in the analysis of genetic data to select the most important genes associated with a particular disease. The selection of important genes can help diagnose the disease and develop treatments.
- Lasso is also used in the analysis of medical imaging data, where a large number of features are used to identify diseases and predict treatment outcomes. Lasso helps to find the most important features, increasing accuracy and minimizing the number of features to consider.
- Lasso is also used in data classification problems with many characteristics. By eliminating non-critical features, Lasso helps to increase the accuracy of the classification model and minimize the calculation time.

This code snippet is an example of how to perform a feature selection using the Lasso regression method in the scikit-learn library.

```python
import numpy as np
from sklearn.linear_model import Lasso
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.datasets import load_diabetes
```

The numpy module and other modules from scikit-learn are imported.

The load_diabetes() function from sklearn.datasets is used to load the diabetes dataset.

```
•   X = df_L1.drop('close', axis=1)
•   y = df_L1['close']
•   X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33,
    random_state=42)
•   pipeline = Pipeline([
•         ('scaler', StandardScaler()),
•         ('model', Lasso())])
```

The X and y variables are defined as the characteristic matrix and the target variable, respectively. X is obtained by removing the 'close' column from df_L1 while y is set as the 'close' column from df_L1.

The train_test_split() function is used to divide the dataset into training and test sets.

Pipelines are used to sequentially apply a list of transformations and an estimation model (Lasso regression model) to the data. The pipeline contains two steps: StandardScaler() will automatically normalize the data for mean = 0 and variance = 1, and Lasso() will apply Lasso regression to select the feature.

GridSearchCV is used to perform a search on the parameter values specified for the pipeline. In this case, it looks for the best value of the alpha parameter of the Lasso model between 0.1 and 3 using the cross-validation method with 5 folds.

```
•   selected_features = X.columns[coef != 0]
•   unselected_features = X.columns[coef == 0]
•   print("Biến được chọn: ", selected_features)
•   print("Biến loại bỏ: ", unselected_features)
```

After finding the best parameters, the coefficients returned from the Lasso model are used to determine the selected and non-selected characteristics. Selected features are those with a coefficient other than zero, while unselected features have a coefficient of zero.

Finally, the new dataset df_L1_new created by taking only the features that have been selected.

2.2.4.2 Backward Feature Selection Operation Mode

Backward Feature Selection (BFS) is a method used to select important variables or features from an initial dataset. This method starts with all the variables included in the model, and then removes the non-critical variables one after the other to re-evaluate the model's performance. This process continues until there are no more variables that can be removed without affecting the performance of the model.

The detailed steps of Backward feature selection include:

- Initialization: Prepare the input data, including the feature set and label set.
- Choose a model: Choose a machine learning model to train the prediction, such as Linear Regression, Logistic Regression, or Decision Tree, etc.
- Split data: Split data into training and test sets, usually using cross-validation or proportional splits.
- Create a list of initial features: Create a list of all the initial features you want to select.
- Backward feature selection: Perform a loop to select the best feature to remove. In each iteration, train the model on the training set with only the current features in the feature list, and evaluate the model's performance on the test set.
- Feature removal: Removes the worst-performing feature, based on the model's performance evaluation criteria, such as prediction error, accuracy, or F1-score, etc.
- Stop Condition Test: Check if feature removal improves the model's performance measurement. If not, stop and print out the list of the last selected features.
- Repeat the process: If feature removal improves the model's performance, continue the process from steps 5 through 7. If feature removal no longer improves the model's performance, stop iterating and output a model performance evaluation

2.2.4.2.1 Advantages

Simple and efficient: Backward Feature Selection starts with all features and removes each one sequentially to build the model. This method is simple and straightforward, does not require a lot of computational resources, and can be applied to a variety of machine learning models.

Reduced data dimensionality: Backward Feature Selection reduces the number of data features, while reducing the possibility of overfitting and increasing the generalization of the model. By removing unnecessary features, the model becomes simpler and easier to interpret.

Improve model performance: Backward Feature Selection helps identify the features that are most important to the prediction and remove those that are unnecessary or have a negative impact on the model's performance. As a result, models built after applying Backward Feature Selection can have higher accuracy and better predictability on new data.

2.2.4.2.2 Cons

Greed: Backward Feature Selection removes features sequentially based on the model's performance measurements, which can lead to suboptimal decision-making. There may be cases where an unnecessary feature is removed earlier, resulting in a model that is no better than retaining this feature in the feature set.

Depends on the order of removal: The results of the Backward Feature Selection may vary depending on the order in which the features are removed. A non-optimal rejection order can result in the final model not achieving the best performance.

No guarantee of finding the best feature set: Backward Feature Selection does not guarantee that the last feature set is the best feature set for the model. There may exist feature subsets that are not checked during removal, resulting in the best feature set being skipped.

Can be time-consuming: The Backward Feature Selection method requires training and evaluating the model multiple times with different subsets, which in turn can be time-consuming to execute, especially when working with large or complex datasets

2.2.4.2.3 Experiment

The steps to perform Backward feature selection in the above source code are as follows:

Data Preparation

Remove the target column (the column to be predicted) from the df_BFS dataframe to create the input dataset (X) and the output dataset (y) for the linear regression model.

```
# Remove the target column from the df_BFS dataframe
X = df_BFS.drop('close', axis=1)
y = df_BFS['close'] # Lưu cột target vào biến y
```

Select model

Initialize a GRU model to be used during training and evaluation of the model's goodness.

```
# Initialization of the GRU model
model = Sequential()
model.add(GRU(64, input_shape=(X_train.shape[1], 1))) # Input is the
characteristic number, length of the data string (1D)
```

```
model.add(Dense(1)) # Output is 1 predicted value

# Compile the model
model.compile(optimizer='adam', loss='mse')

# Convert data to 3D series to include in the GRU model
X_train_3d = np.reshape(X_train.values, (X_train.shape[0],
X_train.shape[1], 1))
X_test_3d = np.reshape(X_test.values, (X_test.shape[0], X_test.shape[1],
1))
```

Split data

Divide the data into a training set (X_train, y_train) and a test set (X_test, y_test) using the train_test_split() function with a test data ratio of 20%.

```
# Divide the data into training sets and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

- Define the function to evaluate the goodness of the model: Define the rmse() function to calculate the Root Mean Squared Error (RMSE), a measure that is often used to evaluate the goodness of a model.

```
# RMSE (Root Mean Squared Error) function to evaluate the goodness
of the model
def rmse(y_true, y_pred):
    return mean_squared_error(y_true, y_pred, squared=False)
```

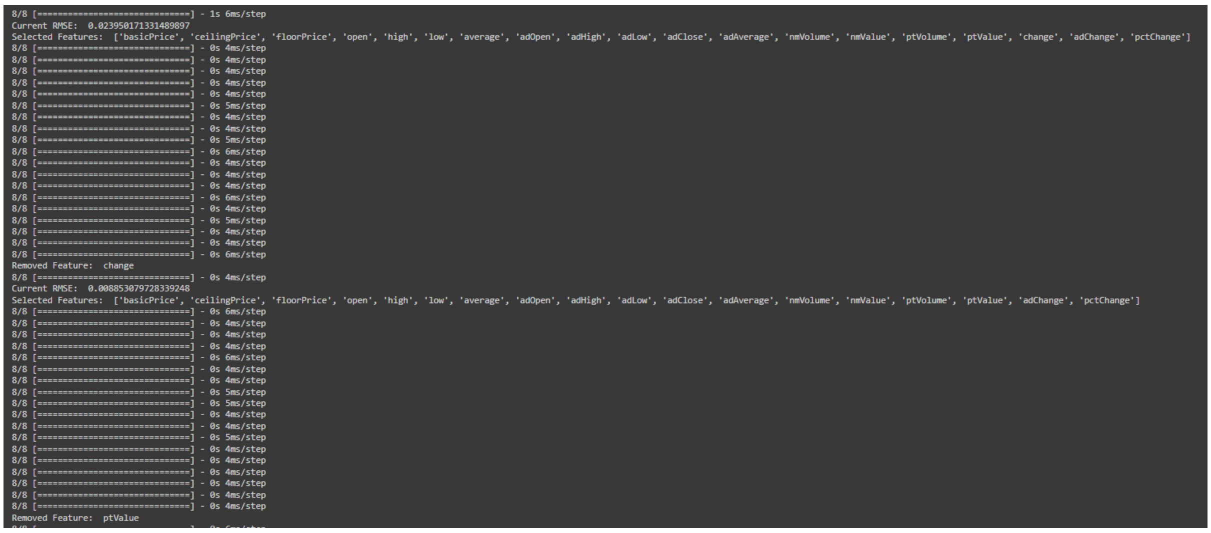
- Initialize a list of initial features: Save the list of initial features in the selected_features variable.

```
# Initialize a list of initial features
selected_features = list(X.columns)
```

- Perform Backward feature selection: In each loop, train the model on the training set with the features in the selected_features variables, predictions on the test set, and the model's RMSE calculation. Then, remove each feature from the selected_features list one by one, retrain the model, and calculate the RMSE of the new model with the test set. If the RMSE of the new model is better than the RMSE of the current model, the feature under consideration will be removed from the selected_features list.

```
# Loop thực hiện Backward feature selection
while len(selected_features) > 0:
    # Train the model on the training episode
```

```python
model.fit(X_train_3d, y_train, epochs=10, batch_size=32, verbose=0)
    # Prediction on the test set
y_pred = model.predict(X_test_3d)

    # Calculate RMSE on the test set
current_rmse = rmse(y_test, y_pred)

    # Print out the RMSE results of the current model
    print("Current RMSE: ", current_rmse)
    print("Selected Features: ", selected_features)

    # Save the original characteristic quantity
num_features = len(selected_features)

    # Delete a feature and evaluate the model's goodness with the training and test sets after deleting the feature
best_rmse = current_rmse
best_feature = None

    # Loop through each feature in the list of selected features
    for feature in selected_features:
        # Create a list of new features by removing the current one
new_features = selected_features.copy()
new_features.remove(feature)

        # Re-select training and test data with only new features
X_train_new = X_train[new_features]
X_test_new = X_test[new_features]

        # Convert data to 3D series
X_train_new_3d = np.reshape(X_train_new.values, (X_train_new.shape[0], X_train_new.shape[1], 1))
X_test_new_3d = np.reshape(X_test_new.values, (X_test_new.shape[0], X_test_new.shape[1], 1))

        # Train the model on the new training episode
model.fit(X_train_new_3d, y_train, epochs=10, batch_size=32, verbose=0)

        # Predictions on the new test set
y_pred_new = model.predict(X_test_new_3d)

        # Calculate RMSE on the new test set
current_rmse_new = rmse(y_test, y_pred_new)

        # Compare the RMSE of the new model with the RMSE of the current model
        if current_rmse_new < best_rmse:
best_rmse = current_rmse_new
best_feature = feature

    # If the RMSE of the new model is better, then remove the feature under consideration
    if best_feature is not None:
selected_features.remove(best_feature)
        print("Removed Feature: ", best_feature)
    else:
        print("No more features to remove")
        break
```



```
8/8 [==============================] - 1s 6ms/step
Current RMSE:  0.023950171331489897
Selected Features:  ['basicPrice', 'ceilingPrice', 'floorPrice', 'open', 'high', 'low', 'average', 'adOpen', 'adHigh', 'adLow', 'adClose', 'adAverage', 'nmVolume', 'nmValue', 'ptVolume', 'ptValue', 'change', 'adChange', 'pctChange']
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 5ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 5ms/step
8/8 [==============================] - 0s 6ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 6ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 5ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 6ms/step
Removed Feature:  change
8/8 [==============================] - 0s 4ms/step
Current RMSE:  0.008853079728339248
Selected Features:  ['basicPrice', 'ceilingPrice', 'floorPrice', 'open', 'high', 'low', 'average', 'adOpen', 'adHigh', 'adLow', 'adClose', 'adAverage', 'nmVolume', 'nmValue', 'ptVolume', 'ptValue', 'adChange', 'pctChange']
8/8 [==============================] - 0s 6ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 6ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 5ms/step
8/8 [==============================] - 0s 5ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 5ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
8/8 [==============================] - 0s 4ms/step
Removed Feature:  ptValue
```

- Stop Loop: If no features are removed in a loop, or if all features have been removed, the Backward feature selection process ends.

```
# If the RMSE of the new model is better, then remove the feature under consideration
    if best_feature is not None:
selected_features.remove(best_feature)
        print("Removed Feature: ", best_feature)
    else:
        print("No more features to remove")
        break
```

- Print out the final selected features: Finally, after the end of the Backward feature selection process, the final list of selected features will be printed:

```
# Print out a list of the last selected features
print("Selected Features after BFS: ", selected_features)

# Print out removed features
removed_features = list(set(X.columns) - set(selected_features))
print("Removed Features: ", removed_features)

Selected Features after BFS:  ['basicPrice', 'ceilingPrice', 'average', 'pctChange']
Removed Features:  ['open', 'nmValue', 'floorPrice', 'ptValue', 'ptVolume', 'adClose', 'change', 'adChange', 'adLow', 'adHigh', 'low', 'nmVolume', 'adAverage', 'adOpen', 'high']
```

date	basicPrice	ceilingPrice	average	pctChange
2021-12-01	0.849656	0.849799	0.796009	0.472958
2021-12-03	0.851375	0.851406	0.785169	0.385970

2021-11-10	0.834192	0.833735	0.767184	0.503431
2021-11-18	0.841924	0.841767	0.772358	0.392254
2021-11-26	0.920103	0.919679	0.883222	0.327433
...	...	...	...	...
2017-11-17	0.384021	0.383936	0.120473	0.417023
2017-10-31	0.359966	0.359839	0.083764	0.380153
2017-12-13	0.384021	0.383936	0.118133	0.354375
2017-10-26	0.350086	0.350201	0.070953	0.438803
2017-11-14	0.372852	0.372691	0.108524	0.533658

Table 2. 2: Data after BFS

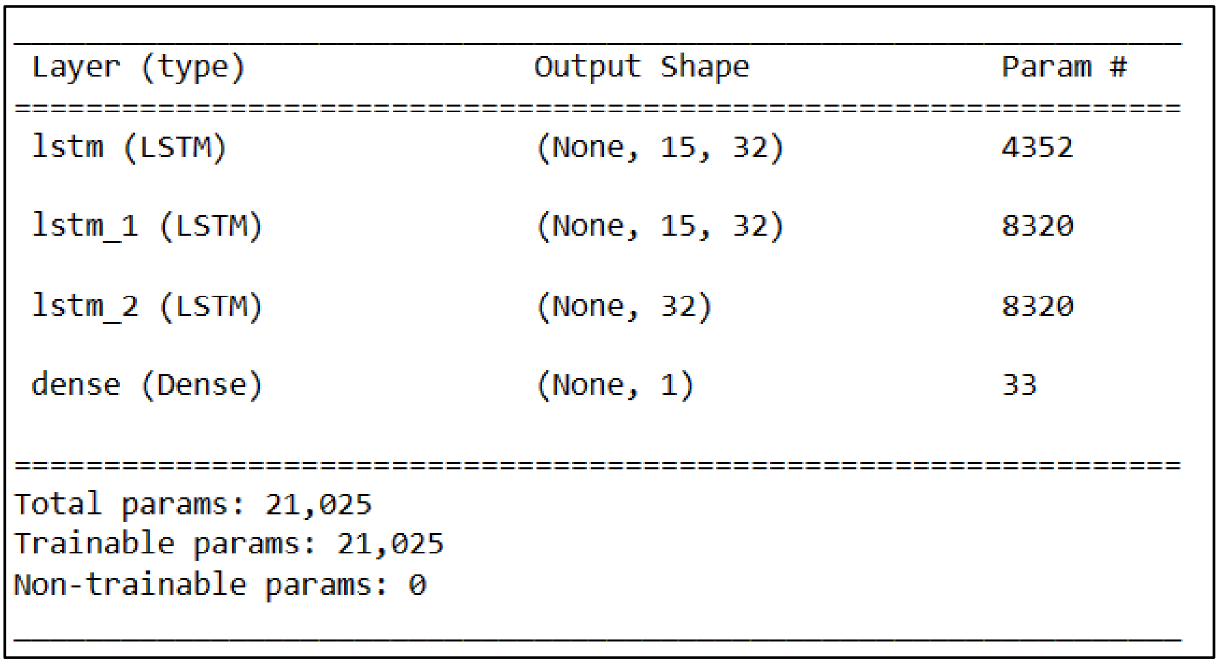
Note that the Backward feature selection process can take a lot of time and computational resources, due to the need to train and evaluate the model multiple times with subsets of the feature set. However, it can help find an optimal feature set, which improves model performance and reduces model complexity, and helps to better explain and understand the role of features in the predictive model.

2.2.4.3 Conclusion

In general, both feature selection methods, L1 regularization (Lasso) and Backward Feature Selection, produce good RMSE evaluation indexes, but the BFS method selects a smaller number of input variables (3 compared to 4 of the L1 method) and a better RMSE index, so with the current data set, we should apply the BFS method. With larger datasets, including more variables, BFS may use too many resources so it will not be optimal for machine learning models, so for datasets with many variables, we should use the L1 method.

2.3 LSTM algorithm experiment

2.3.1 Building a model



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 15, 32)            4352

 lstm_1 (LSTM)               (None, 15, 32)            8320

 lstm_2 (LSTM)               (None, 32)                8320

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 21,025
Trainable params: 21,025
Non-trainable params: 0

_________________________________________________________________
```

Figure 2. 16: Layered structure of the LSTM model

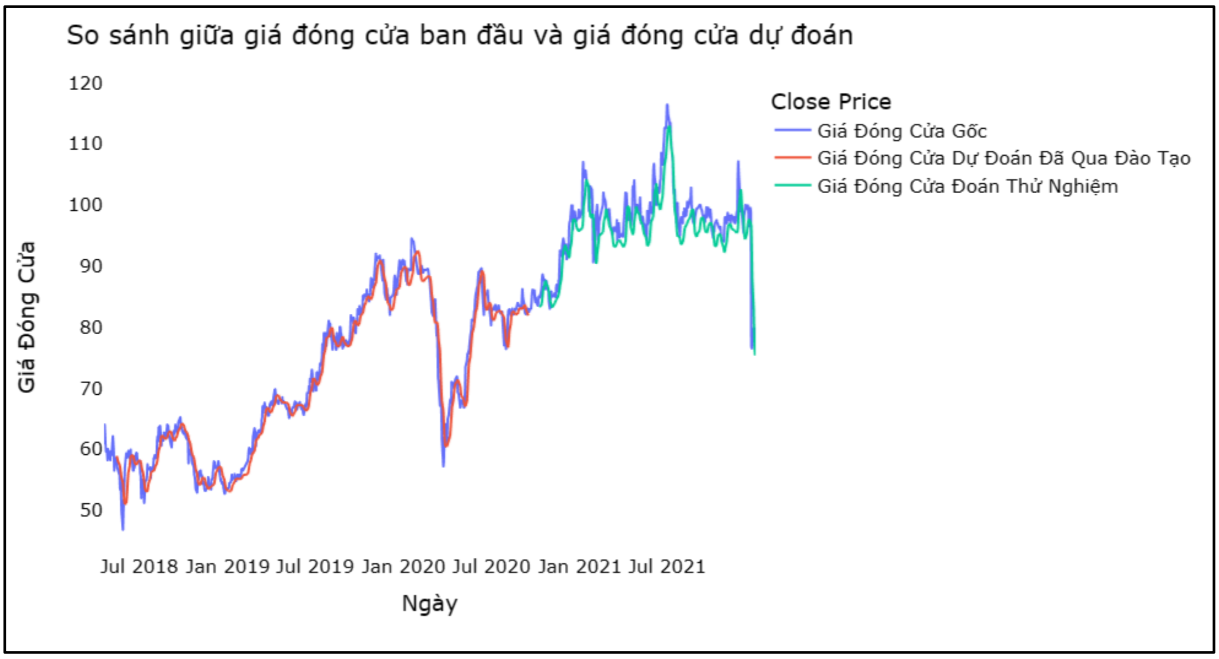


Figure 2. 17: The original closing price and the closing price predict the LSTM pattern

2.3.2 Scoring the LSTM model

		Evaluation Parameters

		Parameter	**RMSE**	**MSE**	**THERE ARE**	R^2
Data Files	Train	loss='mean_squared_error',optimizer='adam', epochs=10,batch_size=5,verbose=1	2.163	4.679	1.606	0.968
	Trial		3.310	10.958	2.383	0.731

Table 2. 3: Evaluation of LSTM model performance parameters

2.3.4 Prediction of LSTM model

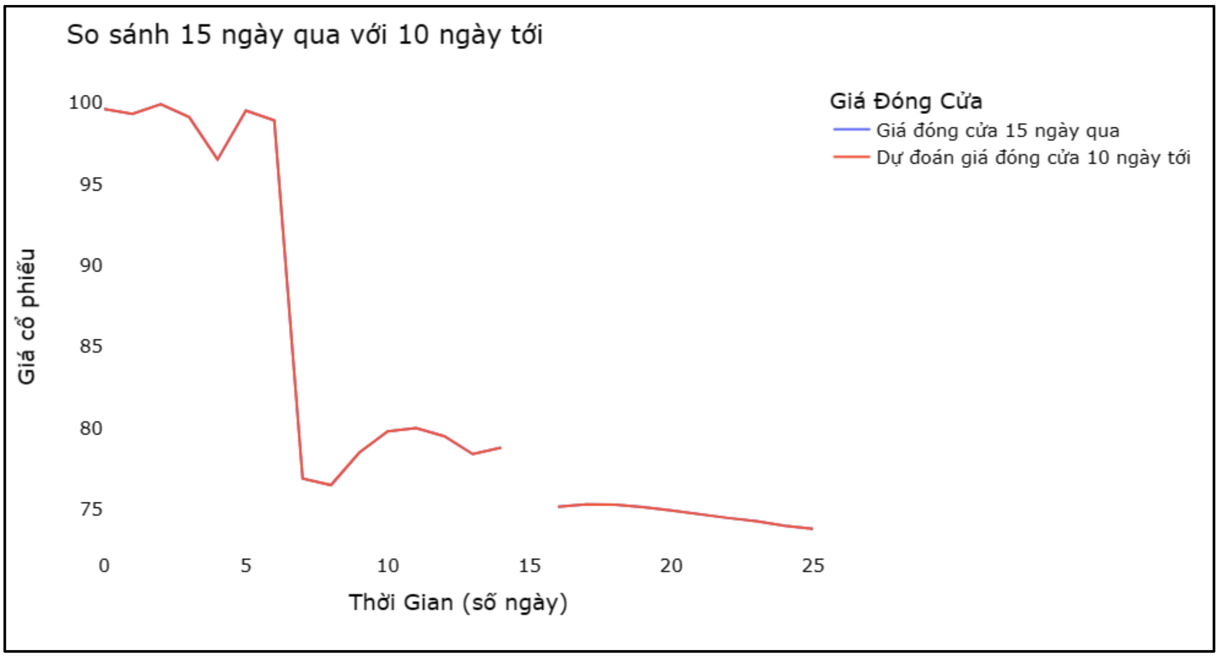
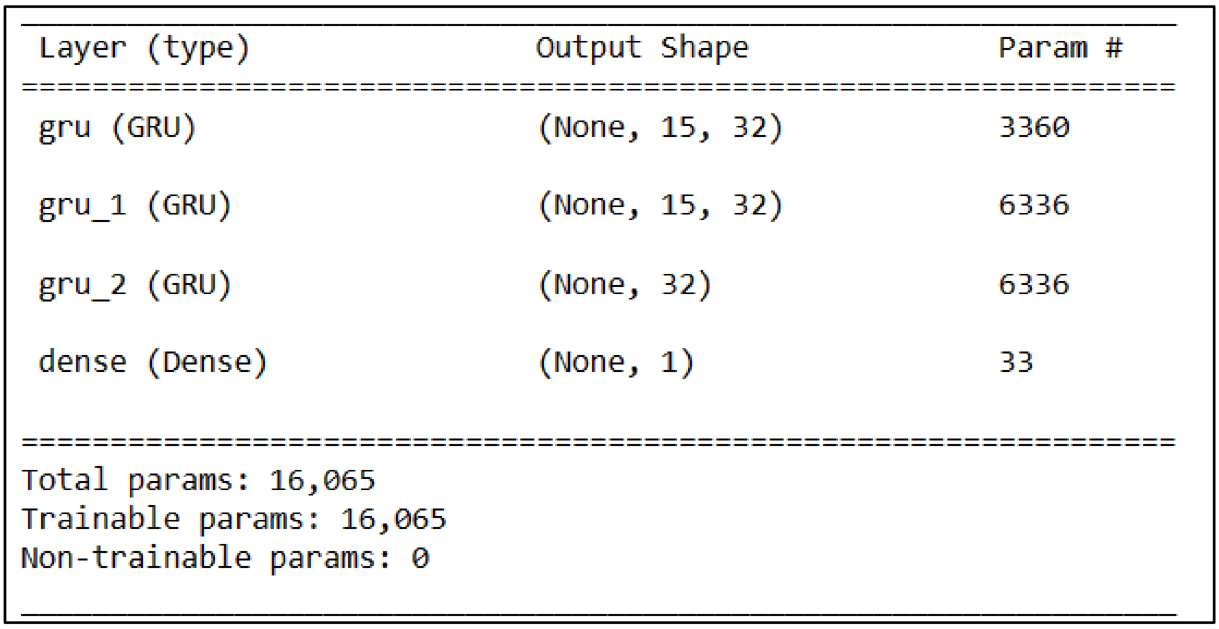


Figure 2. 18: Prediction of the upcoming 10-day closing price of the LSTM model

2.4 GRU algorithm experiments

2.4.1 Building the GRU model



```
________________________________________________________________
 Layer (type)                Output Shape              Param #
================================================================
 gru (GRU)                   (None, 15, 32)            3360

 gru_1 (GRU)                 (None, 15, 32)            6336

 gru_2 (GRU)                 (None, 32)                6336

 dense (Dense)               (None, 1)                 33

================================================================
Total params: 16,065
Trainable params: 16,065
Non-trainable params: 0
________________________________________________________________
```

Figure 2. 19: Layer Structure of the GRU Model

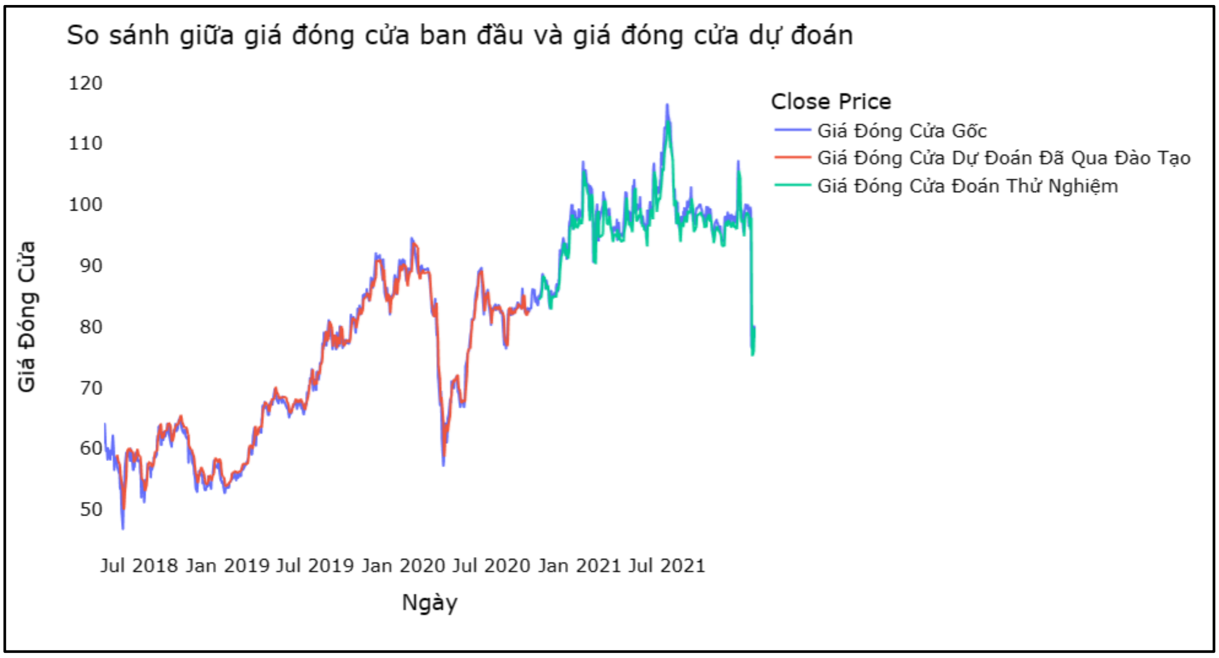


Figure 2. 20: Original closing price and GRU pattern predicted closing price

2.4.2 GRU Model Scoring

	Evaluation Parameters

		Parameter	**RMSE**	**MSE**	**MAE**	R^2
Data Files	Train	loss='mean_squared_error',optimizer='adam', epochs=10,batch_size=5,verbose=1	1.415	2.002	1.003	0.986
	Trial		2.288	5.238	1.490	0.871

Table 2. 4: Evaluate the performance parameters of the GRU model

2.4.3 GRU Model Prediction

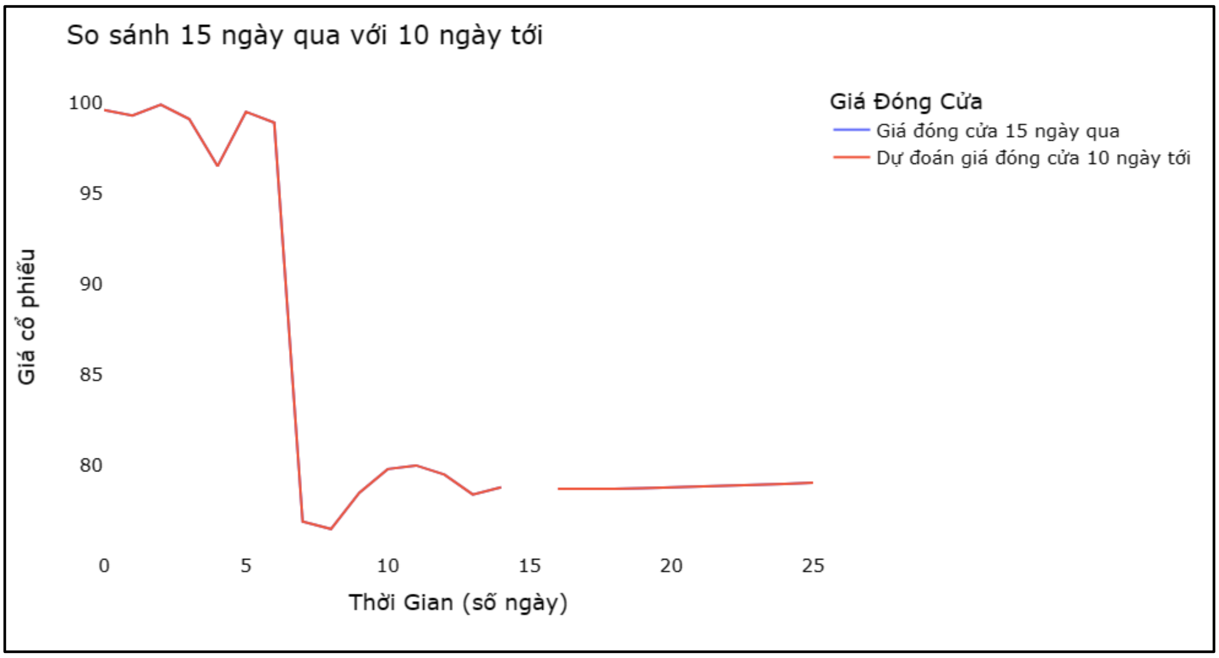
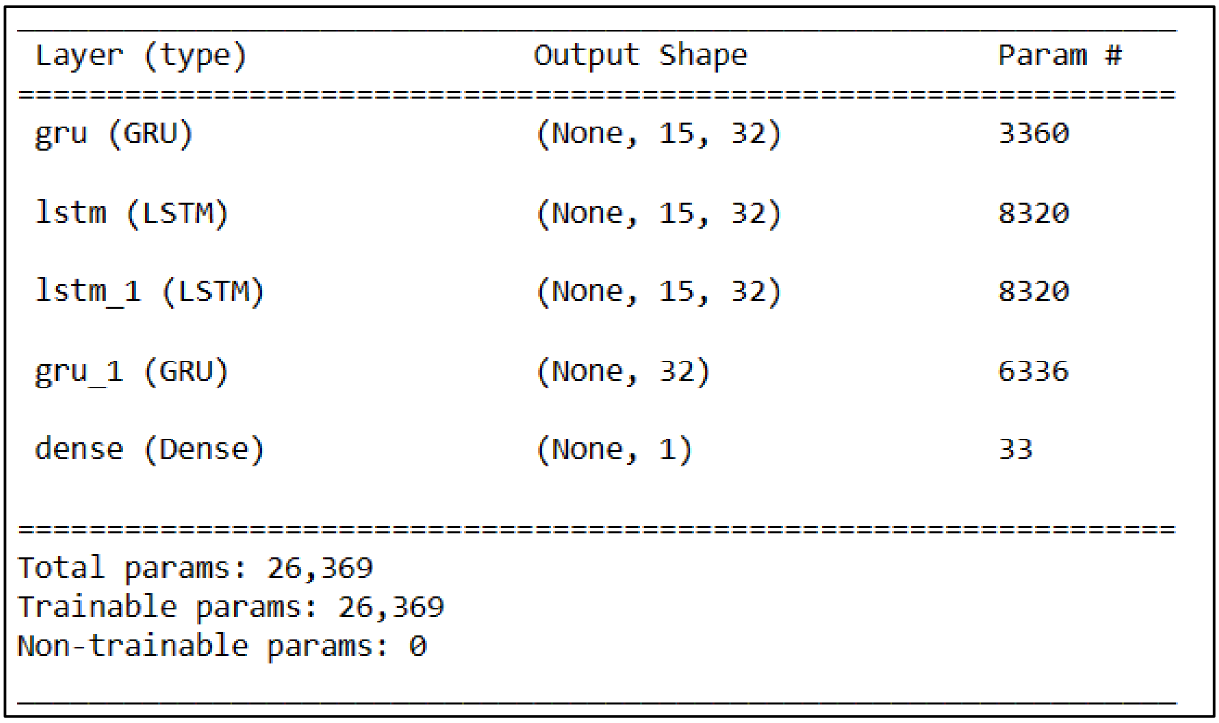


Figure 2. 21: Prediction of the next 10-day closing price of the GRU model

2.5 LSTM+GRU Experiment

2.5.1 Building the LSTM+GRU model



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 gru (GRU)                   (None, 15, 32)            3360

 lstm (LSTM)                 (None, 15, 32)            8320

 lstm_1 (LSTM)               (None, 15, 32)            8320

 gru_1 (GRU)                 (None, 32)                6336

 dense (Dense)               (None, 1)                 33

=================================================================
Total params: 26,369
Trainable params: 26,369
Non-trainable params: 0
_________________________________________________________________
```

Figure 2. 22: Layer Structure of LSTM+GRU Model

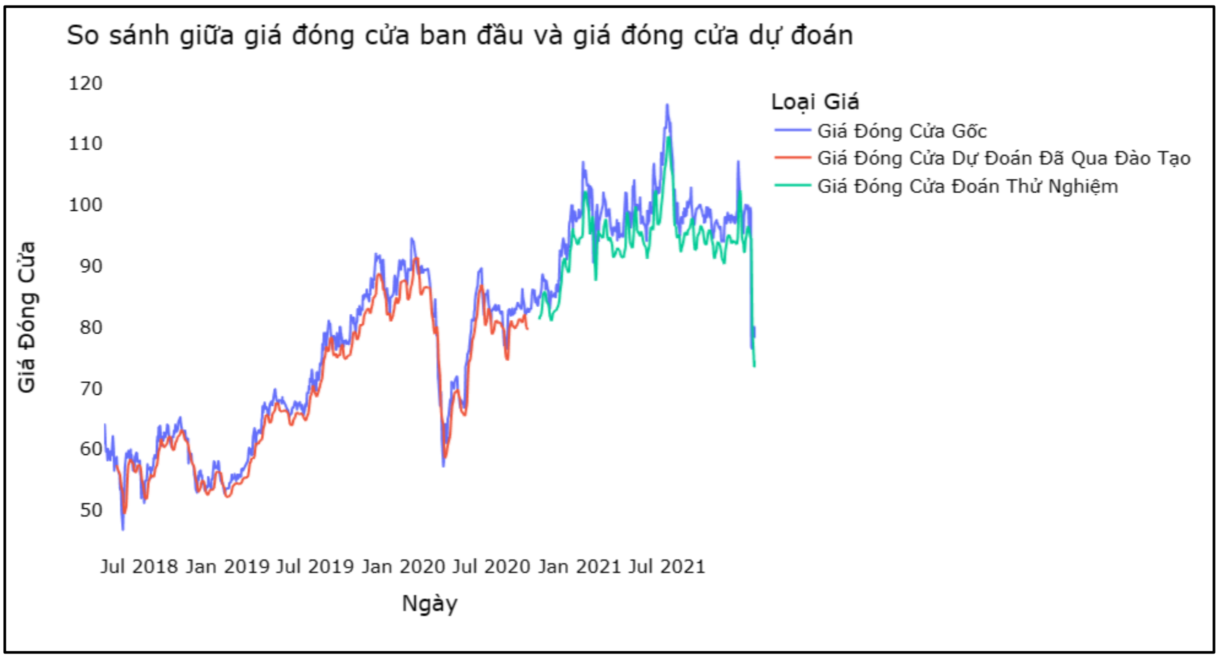


Figure 2. 23: The original closing price and the closing price predict the LSTM + GRU pattern

2.5.2 LSTM+GRU Model Scoring

		Evaluation Parameters				
		Parameter	RMSE	MSE	MAE	R^2
Data Files	Train	loss='mean_squared_error',optimizer='adam', epochs=10,batch_size=5,verbose=1	1.968	3.873	1.464	0.974
	Trial		2.659	7.074	1.752	0.826

Table 2. 5: Evaluation of LSTM+GRU model performance parameters

2.5.3 LSTM+GRU Model Prediction

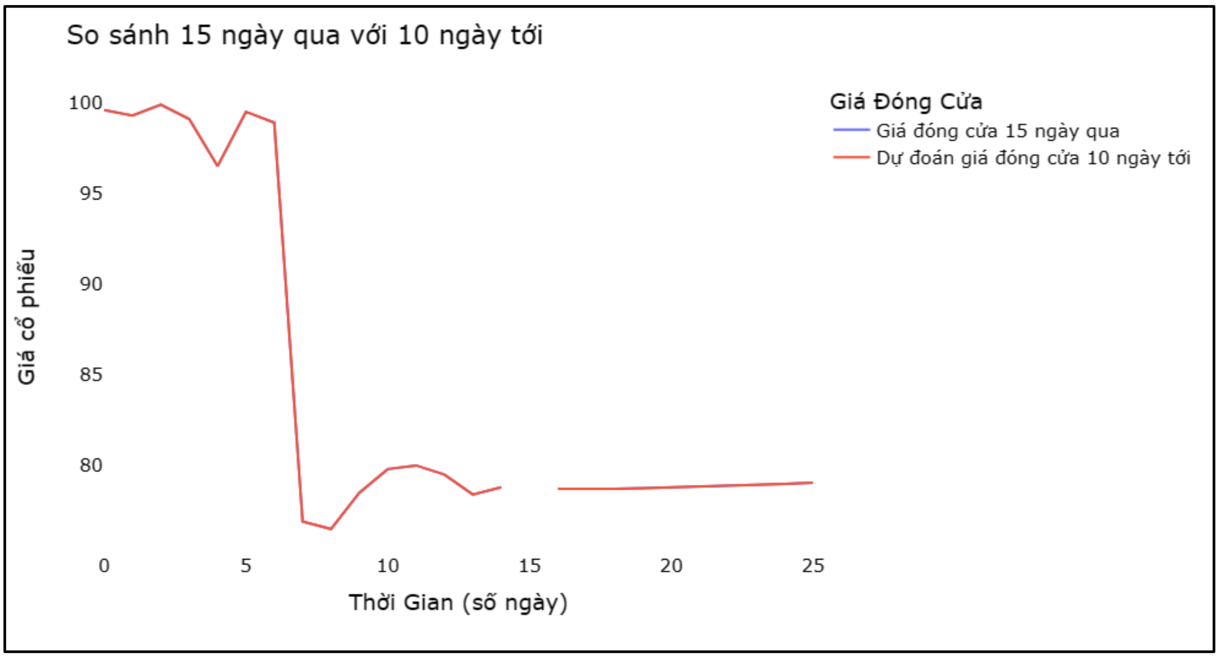


Figure 2. 24: Prediction of the upcoming 10-day closing price of the LSTM+GRU model

Chapter III: Evaluating the Performance of Algorithms

3.1 Performance Evaluation of Deep Learning Algorithms

Algorithm	Parameter	RMSE		MSE		THERE ARE		R^2	
		Training Files	Test Files	Training Files	Test Files	Training Files	Test Files	Trainin g Files	Test Files
LSTM	loss='mean_squared_error',optimizer='adam', epochs=10,batch_size=5,verbose=1	2.163	3.310	4.679	10.958	1.606	2.383	0.968	0.731
CRANE		1.415	2.288	2.002	5.238	1.003	1.490	0.986	0.871
LSTM+GRU		1.968	2.659	3.873	7.074	1.464	1.752	0.974	0.826

Table 3. 1: Evaluate the performance of Deep Learning algorithms

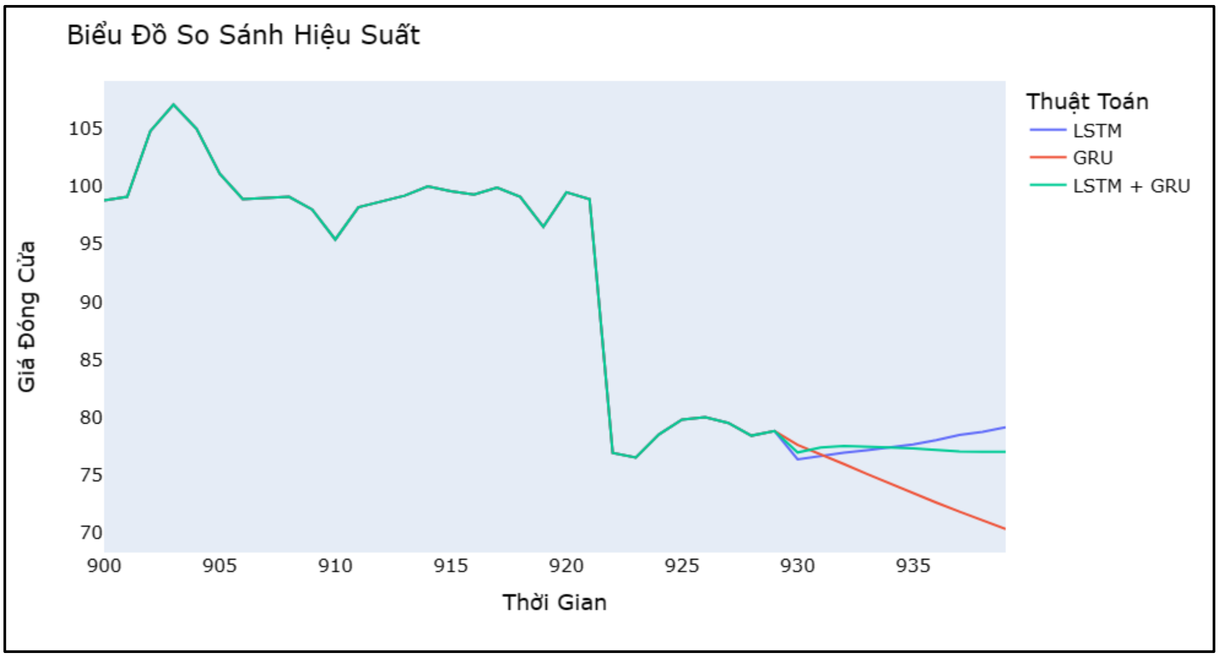


Figure 3. 1: Performance Comparison Chart

Chapter IV: Discussion of Best Algorithms and Future Research Directions

4.1 Conclusion on the Best Algorithm to Use for Prediction

Of the three algorithms, GRU shows the best results with lower RMSE, MSE, and MAE indicators than other algorithms. In addition, the GRU's R2 score is also the highest, indicating the ability to explain most of the variation in the data. This shows that GRUs have better predictive capabilities than LSTM and LSTM+GRUs.

However, when evaluating the algorithm, it is recommended to consider not only technical metrics such as RMSE, MSE, MAE and R2, but also other factors such as computational complexity, training time, feasibility and specific requirements of the problem.

4.2 Future development orientation

In the future, the team may research and apply other advanced deep learning models such as attention-shifting neural networks (Transformers) to improve predictive performance

Combine different prediction methods: Instead of using only a single prediction model, it is possible to consider combining different prediction methods such as the ARIMA (AutoRegressive Integrated Moving Average) model or traditional statistical models to take advantage of each method.