Gunduz, Hakan

Article

An efficient stock market prediction model using hybrid feature reduction method based on variational autoencoders and recursive feature elimination

Financial Innovation

Provided in Cooperation with:
Springer Nature

Suggested Citation: Gunduz, Hakan (2021) : An efficient stock market prediction model using hybrid feature reduction method based on variational autoencoders and recursive feature elimination, Financial Innovation, ISSN 2199-4730, Springer, Heidelberg, Vol. 7, Iss. 1, pp. 1-24, https://doi.org/10.1186/s40854-021-00243-3

Gunduz *Financ Innov* (2021) 7:28
https://doi.org/10.1186/s40854-021-00243-3

Financial Innovation

RESEARCH Open Access

An efficient stock market prediction model using hybrid feature reduction method based on variational autoencoders and recursive feature elimination

Hakan Gunduz*

*Correspondence:
hgunduz@bandirma.edu.tr
Software Engineering Department, Bandirma Onyedi Eylul University, 10200 Balikesir, Turkey

Abstract

In this study, the hourly directions of eight banking stocks in Borsa Istanbul were predicted using linear-based, deep-learning (LSTM) and ensemble learning (LightGBM) models. These models were trained with four different feature sets and their performances were evaluated in terms of accuracy and F-measure metrics. While the first experiments directly used the own stock features as the model inputs, the second experiments utilized reduced stock features through Variational AutoEncoders (VAE). In the last experiments, in order to grasp the effects of the other banking stocks on individual stock performance, the features belonging to other stocks were also given as inputs to our models. While combining other stock features was done for both own (named as allstock_own) and VAE-reduced (named as allstock_VAE) stock features, the expanded dimensions of the feature sets were reduced by Recursive Feature Elimination. As the highest success rate increased up to 0.685 with allstock_own and LSTM with attention model, the combination of allstock_VAE and LSTM with the attention model obtained an accuracy rate of 0.675. Although the classification results achieved with both feature types was close, allstock_VAE achieved these results using nearly 16.67% less features compared to allstock_own. When all experimental results were examined, it was found out that the models trained with allstock_own and allstock_VAE achieved higher accuracy rates than those using individual stock features. It was also concluded that the results obtained with the VAE-reduced stock features were similar to those obtained by own stock features.

Keywords: Stock market prediction, Variational autoencoder, Recursive feature elimination, Long-short term memory, Borsa Istanbul, LightGBM

Introduction

Financial prediction, especially stock market prediction, has been one of the most attractive topics for researchers and investors over the last decade. Stock market prediction studies not only aim to forecast market prices or directions to help investors to make better investment decisions but also prevent stock market turmoil that results in notable damage to the healthy development of a capital market (Wen et al. 2019). For this

purpose, the relationship between the historical behavior of stock prices and their future movements was modeled. Current approaches in financial prediction are separated into two groups, as technical analysis and fundamental analysis. Technical analysis utilizes past price data and technical indicators for predicting future behavior of the financial time series. Although the Effective Market Hypothesis suggests that all information reflects on stock price immediately, technical analysts believe that it is possible to predict future prices by analyzing historical prices. Fundamental analysis is based on internal and external factors regarding a company. While interest rates and exchange rates are the main external factors to be considered, companies' press releases and balance sheet disclosures are the examples of internal factors used for prediction processes (Nti et al. 2019).

Over the last decade, developments in the field of artificial intelligence, specifically Machine Learning (ML), ensure opportunities for the use of computer science in the financial prediction tasks. ML models have proven to be useful in many financial activities, such as portfolio management (Yun et al. 2020), bankruptcy prediction (Kou et al. 2021), financial risk analysis (Kou et al. 2014), and stock trading (Paiva et al. 2019). Artificial Neural Networks (ANN) and Support Vector Machines (SVM) are the most common models used for financial prediction tasks (Sharma et al. 2017). These models are preferred due to the fact they can grasp nonlinear characteristics in data without prior knowledge. Statistical methods, Random Forest (RF), Linear Discriminant Analysis (LDA), Logistic Regression (LR) and Evolutionary Computation methods are the other preferred methods in financial research (Barboza et al. 2017). All aforementioned models use handcrafted features obtained from raw data as model inputs. However, the formation of handcrafted features is a process that requires heavy workload and domain expertise. Furthermore, as the size of the feature space increases, the training time of the models is extended, and the outputs produced by the models become more difficult to interpret (Gunduz et al. 2017b). Since high dimensional feature space results in poor generalization in ML models, dimensionality reduction is performed on features to eliminate the negative effects of high dimensionality and data sparsity (Zhong and Enke 2017).

While using feature selection methods to reduce the size of expanding feature space, it is difficult to find an appropriate selection method in non-linear and noisy data (Bolón-Canedo et al. 2013). In recent studies, Deep Learning (DL) models have been presented as a powerful alternative to feature selection methods. DL models can be considered as a feature extractor that form complex feature representations from raw data or simpler features in each layer at different levels of abstraction (Chen et al. 2016). Long short-term memory (LSTM), one of the popular DL models, performs particularly well in financial forecasting tasks by creating feature representations from the time series data and uses them directly in the prediction process (Fawaz et al. 2019). Unlike the traditional ANN, LSTM considers long-term dependencies and temporal effects in the time series through feedback links.

In this study, the hourly movements of 8 banking stocks in Borsa Istanbul (BIST) were predicted by using different technical indicators derived from the stock prices. While LSTM models with and without attention mechanism were used as classifiers in the prediction process, these models were trained with 4 different feature sets. While own

stock features were firstly used for the network training, Variational Autoencoder (VAE) reduced stock features were then given as inputs to the LSTM models. In the final experiments, besides the own stock features, the features of all other stocks were employed in the prediction. Since the use of all banking features had increased the dimensions of the feature space for both own and reduced feature sets, the size of the expanded space was reduced with Recursive Feature Elimination (RFE) selection. The performances of all trained LSTM models were compared with SVM and LightGBM, and their performances were evaluated with accuracy and F-measure metrics. A pictorial view of the aforementioned framework can be seen in Fig. 1.

The main contributions of this study are that first, an attention-based LSTM model was used in the prediction of Borsa Istanbul. This is the first study that has used this model to predict movement in the Turkish market. Although attention-based LSTM models have been used in many previous studies performed on the developed (Liu and Wang 2018; Li et al 2018) and emerging (Hollis et al. 2018; Chen and Ge 2019) financial markets, attention-based LSTM has not yet been used in the Turkish stock market. Second, the use of Variational Autoencoder (VAE), which allows easier handling of

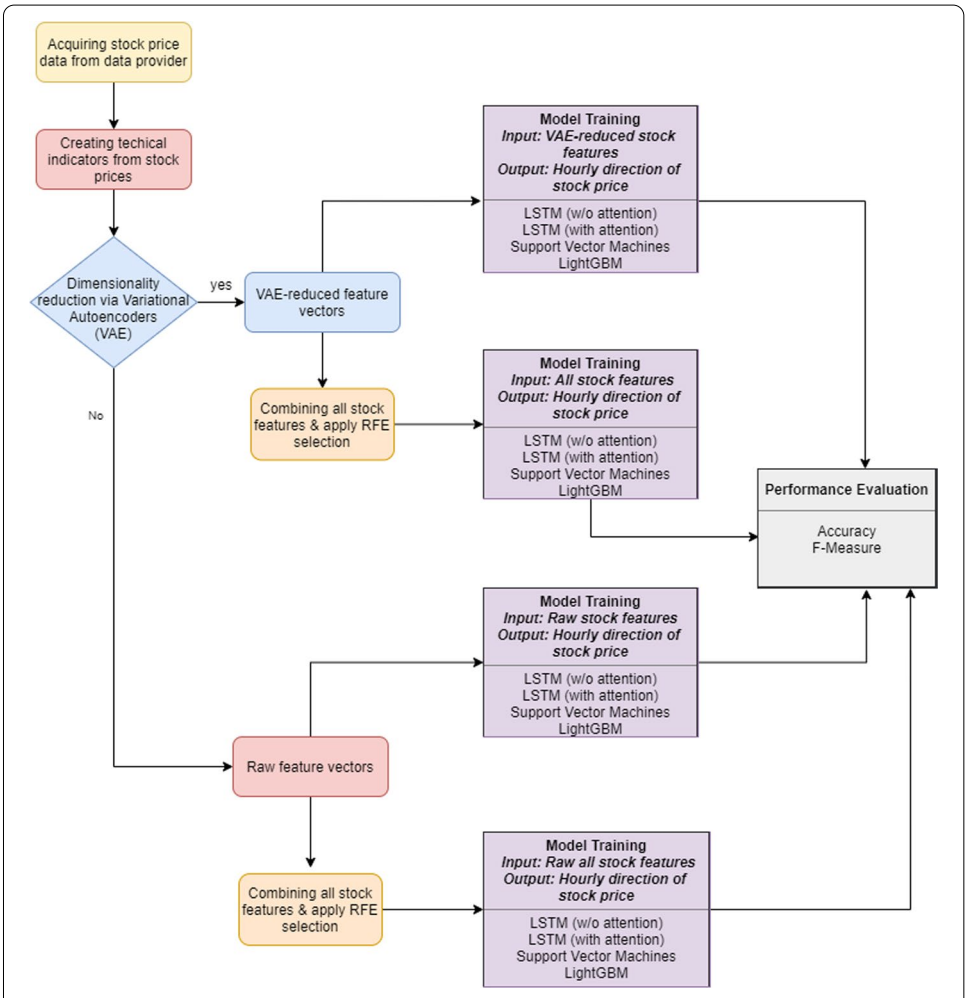

Fig. 1 A graphical view of proposed framework

the problem of the latent space irregularity (e.g. close points in latent space can produce nonadjacent points in decoded data) in time series data. Although models, such as Autoencoders (AE) (Gu et al. 2019) and Stacked-Autoencoders (SAE) (Bao et al. 2017; Gündüz 2020) have caused irregular latent space problems, they have been used in several stock market studies; VAE architecture has not yet been used for the prediction of the stock markets. Lastly, this study uses different evaluation metrics to assess model performances. This study comparatively analyzes the performances of its models on four different feature sets using not only accuracy but also Macro-Averaged (MA) F-measure. With the help of MA F-measure, the performance of the models on class level can be evaluated even in cases of imbalanced class distribution.

The remainder of this paper is organized as follows: in the next section, a brief summary is given about related work. In Sect. 3, the details of our data are explained. Section 4 provides information on dimensionality reduction, classification models and evaluation metrics used. Section 5 gives details of the experimental results, and Sect. 6 concludes the paper.

Related works

In this section, brief information is given about stock market studies used ML and DL models. Additionally, Borsa Istanbul prediction studies published in the last few years are covered.

Stock market prediction with machine learning

Machine learning models have been frequently used for making accurate predictions in financial studies. These models use various information sources to obtain financially relevant features. Among these, structured data such as past stock prices and technical indicators are at the forefront (Cavalcante et al. 2016). Financial articles, press releases, and annual reports are other sources that are commonly used in forecasting market activities (Kumar and Ravi 2016). These sources are unstructured and needed to be preprocessed before being given to ML models as inputs.

A number of studies have used different ML models to mimic the behaviors of financial markets. SVM is a leading model in financial prediction tasks due to its ability to handle the non-linear and dynamic nature of markets. For example, Lin et al. (2013) proposed a framework that predicted trends in the stock prices. Their proposed framework consisted of feature selection and classification modules that were built on the SVM. At first, SVM correlation was used to find informative features among all other features. After dimensionality reduction, a Linear SVM model was trained to classify the stock directions. Their results showed that the feature selection boosted up classification accuracy citelin2013svm. Henrique et al. (2018) used Support Vector Regression (SVR) to predict stock prices for several companies in three different markets using intraday and interday frequencies. Their study revealed that SVR had higher predictive power than the Random Walk model, especially in cases of online learning procedure. Li (2019) predicted the daily movement direction of the S&P 500 ($\hat{G}$SPC) using historical prices and the SVM classifier. The authors devised a feature selection method named Prediction Accuracy Based Hill Climbing Feature Selection Algorithm (AHCFS) and compared its performance with the Sequential Feature Selection (SFS) algorithm, and although

prediction without feature selection was determined as a baseline for both methods, AHCFS outperformed both the SFS and baseline methods in terms of accuracy.

ANN is a good alternative to SVM in modeling non-linear and noisy time series data. In a previous study (Qiu and Song 2016), the daily movement direction of the Japanese stock market was predicted with an optimized ANN model. The optimized model was a hybrid model that combined ANN with Genetic Algorithm (GA). With the help of GA, the weights and bias values were adjusted during ANN training. The proposed hybrid model achieved a satisfactory result and outperformed the standard ANN model with an accuracy rate of 86.39%. In a study conducted by Zhong and Enke (2019), 60 macro and micro economic features which belonged to a 10-year period were used to predict the daily return of the S%P 500 Index. Their prediction pipeline included dimensionality reduction and classification steps. While Principal Component Analysis (PCA), Kernel PCA, and Fast Robust PCA were used as dimensionality reduction techniques, ANN was selected as a ML model. PCA and ANN setup had the best accuracy rate among all experimental setups with a rate of 57%. Naik and Mohan (2019) designed a ML pipeline including a Boruta feature selection and ANN to predict the stock prices of the Indian National Stock Exchange. Thirty-three different technical indicators were fed to the system as the model inputs, and the model performances were evaluated with Mean absolute error (MAE) and Root mean squared error (RMSE). The results showed that the ANN model had decreased the error rate by 12% according to the baseline model.

Apart from SVM and ANN, ensemble learning has also been recently used in many stock market studies. In a study conducted by Patel et al. (2015), a model was proposed to predict the direction of the Indian Stock Market using historical stock prices and technical indicators. They selected ANN, SVM, RF, and Naive Bayes as classifiers and compared the classification performances in terms of accuracy. RF performed better than the other three models in the prediction process. Ballings et al. (2015) compared single classifiers with ensemble models in prediction accuracy of stock market direction. While RF, Adaboost, and kernel factory were chosen as ensemble models, ANN, LR, SVM, and K-nearest neighbor were determined as the single classifiers. The results showed that the ensemble models had better classification performance than the single models. Mehta et al. (2019) devised an ensemble approach for the stock price prediction. They chose diverse types of learners, such as LSTM, SVR and Multiple Regression, for their ensemble model, and compared their performances to those of the base learners. The results indicated that compared to the base learners, ensemble learning approach boosted the prediction accuracy while reducing model variance. In Basak et al. (2019), they employed the Extreme Gradient Boosting (XGBoost) model to predict the trend of the stock market index. They found out that XGBoost could successfully predict long-term trends and had surpassed the predictive performance of the conventional ML models.

Stock market prediction with deep learning

As mentioned in the previous section, although traditional ANN had high success in solving classification problems, it had difficulty with complex time correlation in the time series. LSTM was proposed to model the long-term dependencies in the neural networks and to solve the problem of the vanishing gradients in the traditional

Recurrent Neural Network models. Many studies were conducted to prove that LSTM could achieve better results in time series prediction. For example, Xingjian et al (2015) used the convolution-enhanced LSTM network for weather forecasting and achieved higher success than the other existing prediction models. Ma et al. (2015) captured the nonlinear traffic dynamics for the short-term traffic forecasting with the LSTM network.

There are also many stock market studies using the LSTM network in the literature. Chen et al. (2015) used LSTM to predict the Chinese market and estimated the 3-day earnings of the stocks with different LSTM steps. Compared to random prediction, LSTM was more successful in predicting the stock returns. Fischer and Krauss (2018) created a deep convolutional LSTM model to analyze the effects of the events of different times on stock prices. Fischer analyzed LSTM's performance in stock movement direction prediction and confirmed that LSTM had higher classification success than RF, ANN, and LR classifiers. However, Gunduz et al. (2018) estimated the financial aspects of the stocks in Borsa Istanbul using financial news and LSTM networks. In this study, news texts were converted into feature vectors with word representations and given as inputs to the LSTM networks. The performances of trained LSTM networks excelled in random and naive comparison models. Li et al. (2017) proposed an LSTM-based stock market forecasting model by combining investor sentiments and market factors to improve prediction performance. This study used the Naive Bayes model to analyze the non-rational component of the stock prices, investors' sentiments. Experiments on the CSI300 index showed that the proposed model provided 6% better performance than the other benchmark models with an accuracy of 87.86%. The study also helped investors analyze their sentiments and stock behaviors in detail. Kim and Kim (2019) proposed a hybrid model based on LSTM and Convolutional Neural Network (CNN) for the prediction of the S%P 500 index. In this study, visual features were obtained from the stock chart images with pre-trained CNN, while numerical features were created from historical stock price records with the LSTM network. Features extracted through the CNN and LSTM models were firstly used in the model training individually, after which the training was carried out by feature fusion. Compared to the individual models, feature fusion resulted in lower prediction errors.

Dataset

Hourly price data of eight banking stocks listed in the BIST 30 Index were used in this study. Price data included hourly open, close, and high and low prices. The data consists of 6705 instances collected between the years of 2011 and 2015. The first 3 years of the data were specified as training set, and the rest as test set. After the splitting process, the features used in the study were decided. Hourly raw open, close, high, and low prices of the stocks and logarithmic scale of the prices were the first added features in our dataset. Technical indicators computed from raw prices constitute the other features used in the prediction process. Technical indicators give information about the movement directions of the stocks and the continuity of the price trend in the future (Gunduz et al. 2017b). These indicators use the current point and the specified time interval as parameters. The explanations of used technical indicators are shown in Table 1.

In order to complete the computation of the technical indicators, parameters of such indicators (periods) needed to be determined. Considering that a trading day consists of

Table 1 Used technical indicators (TI) (x and y denote hourly time periods)

TI	Explanations
ROC(x)	Rate of change
MA(x)	Moving average
EMA(x)	Exponential moving average
MOM(x)	Momentum
MACD(x,y)	Moving average convergence divergence
WILLR(x)	Larry Williamś %
RSI(x)	Relative Strength Index
MEDPRICE(x)	Median price
MIDPRICE(x)	Mean price
HH(x)	Highest high
LL(x)	Lowest low

8 h, it was decided that the periods to compute the technical indicators could be 1, 2, 4, 8, 16, 32 and 64, respectively. Thus, the values of each indicator in 7 different time periods were computed, and a total of 86 features were created for 11 technical indicators. When these features were added to raw and logarithmic scale prices, a 94 features were created per hour for each stock. DL models that use gradient descent as an optimizer need input data to be scaled due to the fact the difference in range of features can cause different step sizes for each feature. For this purpose, each feature in our dataset was applied to a minimum-maximum normalization to transform the feature values into a common scale.

Since the hourly movement direction of the stock prices was predicted in the study, class labels indicating the directions were created for each trading hour. Class labels were computed as follows:

$$r(t) = \begin{cases} 1, & \text{if } c(t) \geq c(t-1) \\ -1, & \text{else} \end{cases} \quad (1)$$

In the Eq. 1, $c(t)$ and $c(t-1)$ denote the close prices of hour t and $t-1$ respectively. $r(t)$ refers to the class label assigned for hour t. Class labels determined for each trading hour were aligned with feature vectors.

Methods

This section presents the details of dimensionality reduction methods, classification models, and performance evaluation metrics used in proposed prediction framework.

Dimensionality reduction methods

Dimensionality reduction (DR) can be regarded as a preprocessing step to reduce the complexity of ML models. DR does not only improve the computational efficiency of such models but also their predictive performances (Khalid et al. 2014; Kou et al. 2020). DR can be grouped into two categories: feature selection and feature extraction. Selecting a subset of features from original feature space is defined as a feature selection, while projecting features onto a different feature space to create a low subspace is known as a feature extraction.

Obtaining high accuracy in finance studies is dependent on the use of relevant features in ML models (Gunduz et al. 2017a). However, it is difficult to find informative features for representing the latent properties of the time series data. Recently, Autoencoders, in particular, Variational AutoEncoder (VAE), can be applied to the time series data to learn robust deep feature representations (code) directly while reducing the dimensions of the feature space. The ability to create the representations with a generative approach is the main reason that we use VAE in our study.

Besides VAE, Recursive Feature Elimination selection is used as a helper method to assess the performance of the feature combination. RFE is a feature selection method that employs a wrapper approach to select a subset of features through the whole feature set.

Variational autoEncoder (VAE)

Autoencoder is a neural network that copies the values in the input layer to the output layer. In other words, the data provided as input to the neural network in this study are reconstructed in the output layer. This is an unsupervised learning model, where explicit labels are not specified when training the network (Baldi 2012).

Variational AutoEncoder (VAE) is an unsupervised and generative autoencoder model that forces the distribution of the vectors in the hidden space to a normal distribution. VAE converts the vector x in the input layer into 2 parameters in the hidden space: mean and standard deviation (sd). VAE produces new samples through learnt mean and sd vectors (Gunduz 2021). Although mean and sd values are deterministic, samples generated from these values are random (probabilistic). The randomness of the generated samples prevents the computation of the partial derivatives of mean and sd vectors with a back-propagation method. In order to eliminate this problem, the re-parametrization trick (parameter modification) and random noise (ϵ = epsilon, a random number generated from a normal distribution whose mean is 0 variance is 1) are utilized. With the help of these operations, it is possible to compute the partial derivatives in terms of mean and sd (Kingma et al. 2019).

VAE consists of two separate steps, encoder and decoder. The encoder step creates a h code vector from the input vector x in the hidden space, whereas the decoder converts this h code vector to the r output with the decoder network. This is called a reconstruction because input (x) and output (r) are identical to each other. This process is the same as that of standard AutoEncoder (AE). The key difference between AE's and VAE's is the type of the loss function used in the network training. AE's loss function is a standard mean squared error (MSE), while VAE's loss function consists of MSE + Kullback–Leibler (KL) Divergence terms. KL-Divergence is a metric for the difference between two normal distributions. Let us assume that VAE has 15 nodes in the hidden space; VAE will produce mean and sd vectors for a 15-dimensional hidden space in the first epoch. The difference between the hidden space (z) connected to the 15-dimensional mean and sd vectors and the 15-dimensional normal distribution is evaluated with KL-Divergence. KL-Divergence also acts as a regularization metric that prevents overfitting and ensures that important features are kept in the hidden space (Walker et al. 2016). Thus, close points in the latent space can produce nonadjacent point decoded data.

A lower KL-Divergence value shows that the distribution of the hidden space is closer to normal distribution. This indicates that regardless of the x input given, x will always have similar values in the hidden space. Because of this, MSE will increase too much and the total loss of VAE will also tend to increase. This case is similar to the bias-variance trade-off in ML.

Recursive feature elimination (RFE)

Recursive Feature Elimination (RFE) is known as a wrapper feature selection and employs ML models when computing the relevance scores of the features. RFE firstly trains a model with an entire feature set and computes a relevance score for each feature. In the next step, the feature with the least relevance score is neglected and the model is re-trained to compute new feature relevance scores. This process is continued until the desired number of features remain in the feature set. Therefore, the desired subset size is a parameter that needs to be set before the model initialization. Another parameter to be determined is the ML model employed in finding the relevance scores of the features in each RFE iteration. SVM is a popular model due to its high accuracy and good generalization ability. RFE commonly uses SVM model with a linear kernel to assign a weight value (feature relevance score) to each feature. In such cases, the feature is neglected in the next iteration since the lowest weighting feature will have the least effect on the classification process. RFE spends more time neglecting features one by one in case of a high dimensional feature space. In order to handle the running time issue, RFE ignores more than one feature in each iteration (Yan and Zhang 2015).

Classification models

In this study, different types of ML models, such as Support Vector Machines, LightGBM, and Long-Short Term Memory are employed to classify the directions of the stock movements. The details of the models are discussed in the subsections below.

Support vector machines (SVM)

Support Vector Machines (SVM) are an ML model employed in both classification and regression tasks. In binary classification problems, if the data are linearly separable, this separation can be done with an infinite number of decision boundaries named hyperplanes. The main goal of SVM is to find a linear function with the largest margin to both class instances. SVM also has the capability of classifying nonlinear data successfully through the "kernel trick." In order to ensure linear separability in the nonlinear data, the "kernel trick" method projects n-dimensional samples onto a new m-dimensional space (m > n) using basis functions and instances in the new feature space that are separated into two classes using hyperplanes. The parameters in SVM vary depending on the type of kernel function used. C is a common parameter that regulates the complexity of the trained model. Lower C values produce underfitted models that may have more misclassified samples, while higher C values increase the variance of the model and cause overfitting (Guenther and Schonlau 2016).

LightGBM

Boosting is an ensemble approach that combines a predefined number of base learners to produce a single strong learner. Boosting forms a learner group by training each model according to the same dataset, but adjusting the weights of the instances according to the errors of the final prediction. The main principle in boosting is to force models to focus on instances that are difficult to predict. The boosting method has been successfully applied to many problems due to their successful performance rate (Altman and Krzywinski 2017).

LightGBM is a fast, distributed, high performance ensemble model based on decision trees. It is a variant of gradient boosting that consists of many weak decision trees. Unlike a bagging approach, LightGBM combines models additively and sequentially. Boosting models use two strategies, level-oriented and leaf-oriented, while they train each decision tree and split the data. The level-oriented approach preserves the balance of the tree in the growing phase, whereas leaf-oriented approach continues to split the biggest loss decreasing leaf. LightGBM has a leaf-oriented tree structure that chooses not only losses in a particular branch but also splits based on its contribution to the entire loss. Often, it chooses the trees with fewer error rates rather than other growing models of level-oriented learning (Ke et al. 2017).

Training time of a decision tree is proportional to the number of possible node splits. Small changes in splitting often do not make a big difference in model performance. LightGBM, which is also a histogram-based method, takes advantage of this case by grouping the features into a series of bins and splitting them into the bins instead of the features. This property can reduce the computational complexity and result in reductions on model training time.

Long-short term memory

Long-Short Term Memory (LSTM) is a special variant of Recurrent Neural Networks (RNN) that has the capacity to model the long-term dependencies in a time series. Rather than having a single layer like simple neural networks, LSTM uses four layers that interact in a specific way to preserve the information for long periods. The internal structure of the LSTM is shown in Fig. 2.

The key feature of LSTM is the cell state. LSTM is capable of adding or subtracting information to the cell state ($\mathbf{C_t} - 1$) with structures called "gates." The gates are an optional way of providing information, and they are made up of a *sigmoid* layer and a dot-product. Sigmoid layer outputs the numbers from zero to one and describes how much each component is allowed. **0** means **"don't let anything pass"**; **1** means **"allow everything"**. LSTM has three of these gates to maintain and control the cell state.

The first step of LSTM is to decide which information is to be removed from the cell state. This decision is made up of a *sigmoid* layer called as **"forget gate** ($\mathbf{f_t}$)". The next step is to decide which new information will be stored in the cell state. **"Input gate** ($\mathbf{i_t}$)" layer decides which values are to be updated with a *tanh* layer. This layer creates a new candidate state vector ($\mathbf{C_{tn}}$). In the next step, ($\mathbf{C_{tn}}$) and ($\mathbf{C_t} - 1$) are combined to update the state vector. Thus, the old cell state is ($\mathbf{C_t} - 1$) replaced with a new cell state ($\mathbf{C_t}$). In the final step, LSTM's output is specified, which is relative to the last

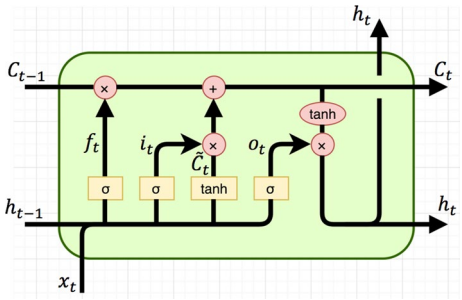

Fig. 2 LSTM network internal structure (Olah 2015)

cell state but is also a filtered version of it. An **"output gate ($\mathbf{O_t}$)"** is a layer that determines which portions of the cell state can be transferred as an output. In order to generate the output vector ($\mathbf{h_t}$), the cell state is passed through the *tanh* activation function and multiplied by $\mathbf{O_t}$ (Gunduz et al. 2018).

LSTM generates an output vector ($\mathbf{h_t}$) for each time step in the time series data to link the output vector of the current time step to the previous time steps. The most common way to use LSTM is to take the output vector of the last time step ($\mathbf{h_t}$) in the sequence as a representation of the entire sequence. This approach can result in the loss of information due to an entire sequence being reduced to a low dimensional vector. In these cases, the output vectors of all steps can be used instead of the vector of the last time step. Thus, the prediction operation depends on the aggregation of the output vectors of the input sequence, and LSTM assigns the weights to these vectors to create a fixed length vector. These weights specify which time steps are important in the classification process. This approach is called as **attention mechanism**. With this mechanism, one or more dense layers are added to the outputs of the LSTM layer, and a weight is assigned to each time step. The determination of assigned weights occurs during the training of the network (Wang ert al. 2016).

Neural networks with a large number of parameters can model functions with a high degree of complexity. However, a huge number of parameters may cause the network to not fitwell with new data. This problem, known as overfitting, is a major issue in deep neural networks with millions of parameters such as LSTM. Several techniques have been applied to overcome this problem, such as restricting the parameters and modifying cost function. Unlike other techniques, dropout is a method in a configuration that works by modifying the network itself (Srivastava et al. 2014). Dropout works randomly and temporarily by ignoring the neurons in the hidden layer during the training, based on the predefined p probability value. During the training of the network, the inputs are transmitted through the modified layers with $n * p$ active neurons and the back propagation is performed on the same neurons. During the testing phase, the inputs are fed to the unmodified layer and the output layer is scaled with p value.

With dropout, network training is done on a set of different networks, and the final output is generated by averaging all their outputs. This method is a powerful way to reduce overfitting like in the ensemble learning approach. Since a neuron cannot rely on the presence of other neurons, it also has to learn characteristics that do not depend on the presence of other neurons. Thus, the network learns the robust

properties and reduces noise sensitivity. Dropout does not restrict the network parameters and can be used with such as L2.

Performance evaluation

Evaluation metrics are used to measure the predictive performance of ML models. Although accuracy is the most preferred metric in performance evaluation, it does not solely provide sufficient information to decide whether a model is good enough. Accuracy can also cause misleading results in cases of imbalanced data, which is a concept used to define the datasets where the distribution between classes is not close. Assessment metrics such as F-measure can calculate how well a classifier can distinguish between different classes even in the case of class imbalance (Gunduz et al. 2017a).

Accuracy and F-measure are both computed based on a confusion matrix, the clear and simple way to present the predictive results of the classifier. Confusion matrix (CM) is a table commonly used to describe the performance of classification models with a set of test instances where ground truths are known. In a binary classification case, CM refers to the number of correctly and incorrectly classified instances per class. The elements of the confusion matrix are expressed in Table 2.

In Table 2, *tp*, *fp*, *fn*, and *tn* denote the numbers of true positive (tp), false positive (fp), false negative (fn) and true negative (tn) instances, respectively.

Accuracy is an overall measure of the predictive performance and is defined as a ratio of accurate prediction counts to the total number of instances. However, in cases where the difference between *fp* and *fn* values is high, other parameters need to be considered to evaluate the performance of the model. Precision is a metric that computes the ratio of accurately predicted positive instances to predicted total positive instances (Eq. 2). Recall is another metric, which is used to reveal the ratio of number of correctly classified positive samples (*tp*) to total number of actual positive samples (Eq. 3). Low precision rates also indicate many false positives in model performance, while low recall rates show us that the classification result contains many instances of false negatives (Song et al. 2018).

F-measure is defined as a harmonic mean of precision and recall. Therefore, evaluating classification performance with F-measure also considers both false positives and false negatives. Since F-measure can directly assess the discriminative power of the classifier, it is more useful to look at the F-measure, especially when there is an imbalanced class distribution.

Based on counts in confusion matrix, F-measure is computed as follows:

$$\text{precision} = \frac{tp}{tp + fp} \tag{2}$$

Table 2 Confusion matrix for two-class classification

Actual/predicted as	Positive	Negative
Positive	*tp*	*fn*
Negative	*fp*	*tn*

$$\text{recall} = \frac{tp}{tp + fn} \tag{3}$$

$$\text{F-measure} = \frac{2 \times precision \times recall}{precision + recall} \tag{4}$$

In case of a class imbalance problem, model performance is assessed for each class using F-measure, and overall evaluation is performed by computing the mean of the class-level F-measure (a.k.a Macro-Averaged (MA)) rates (Pillai et al. 2017).

Experimental results

Two different benchmark methods, random and naive, were used to compare the performances of this study's ML models. In the random method, the prior probability of each class in the dataset was found, and class labels were assigned to the samples randomly by considering the probabilities. The naive approach assigned the class label of the previous time step to the current time step. The classification results obtained by these methods are shown in Table 3 on a stock basis. The results showed that the naive method was better than the random method in terms of accuracy and MA F-measure.

After results obtained by benchmark methods, the first experiments were conducted with the own stock features (raw stock prices, log-scaled price, and technical indicators) using SVM, LightGBM, and two LSTM classifiers. While feature vectors were given to SVM and LightGBM models in a 1-Dimensional (1D) form, vectors were transformed into 2-Dimensional (2D) tensors for the LSTM models, which were formed by combining the stock features of past 8 h (each trading day consisted of 8 h). Thus, LSTM models used the feature vectors of the last 8 h in the hourly prediction. LSTM with attention models were also utilized from an attention mechanism to weight the output vectors of all time steps. A pictorial view of LSTM basic and LSTM with attention models are shown in Figs. 3 and 4, respectively.

LSTM basic model included one input, one LSTM, one batch-normalization, one activation, and one dense layer. The input layer transmitted 2D instances of size 8 by 94 to the LSTM layer, which consisted of 60 cells. The outputs of the LSTM layer were then passed through the batch-normalization and activation ("ELU") layers and were then

Table 3 Classification results using random and naive methods

Stocks	Random		Naive	
	Acc	F-Mea	Acc	F-Mea
ALBRK	0.501	0.493	0.511	0.501
AKBNK	0.493	0.481	0.509	0.493
HALKB	0.510	0.505	0.523	0.511
ISCTR	0.487	0.478	0.502	0.487
SKBNK	0.501	0.487	0.486	0.464
TSKB	0.503	0.492	0.534	0.513
VAKBN	0.499	0.468	0.513	0.483
YKBNK	0.493	0.476	0.503	0.491
Average	0.498	0.485	**0.510**	**0.492**

Bold numbers indicate the best performance

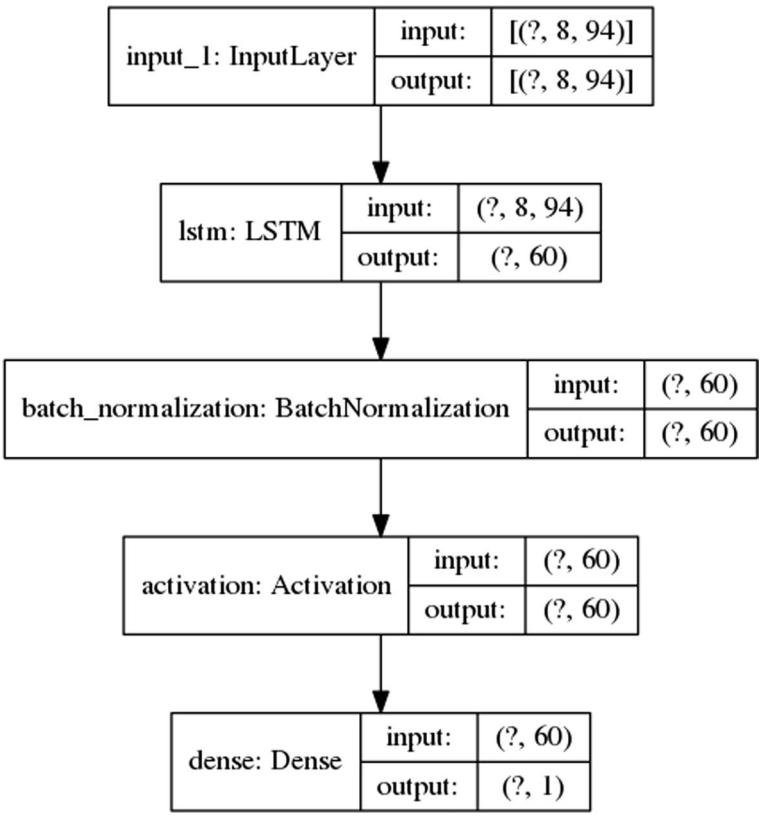

Fig. 3 Graphical view of LSTM basic model

finally transferred to the dense layer. The difference between the attention and basic models is that the former had an attention block after the activation layer. In the attention block, the contributions of all time step vectors from the LSTM layer were found in the attention_scores layer, and these contributions were converted into the weights in the attention_weight layer. The generated attention_weight vector was then dot-produced with the LSTM time step vectors to create a context vector. After obtaining this vector from the attention block, its values were compared with those of the vector obtained from the last time step of LSTM to find maximum vector values. The classification process was completed by finding maximum values through dropout and dense layers.

Since model parameters directly affected classification performance, hyper-parameters of used models were specified by a grid search with five-fold cross-validation on the training data. LSTM models were trained using KERAS (Chollet 2018) package with the specified hyper-parameters shown in Table 4. Although the number of epochs was determined to be 250, early-stopping was applied if there was no decrease in the validation error during 20 iterations. In order to avoid overfitting, dropout layers were also used in the fully connected and recurrent layers. For the evaluation results of the LSTM models, each model was executed 11 times and the accuracy rates of these models were sorted in ascending order. After sorting the execution results, the 6th model (the median of the 11 executed models) was chosen as a key model for reporting model performances.

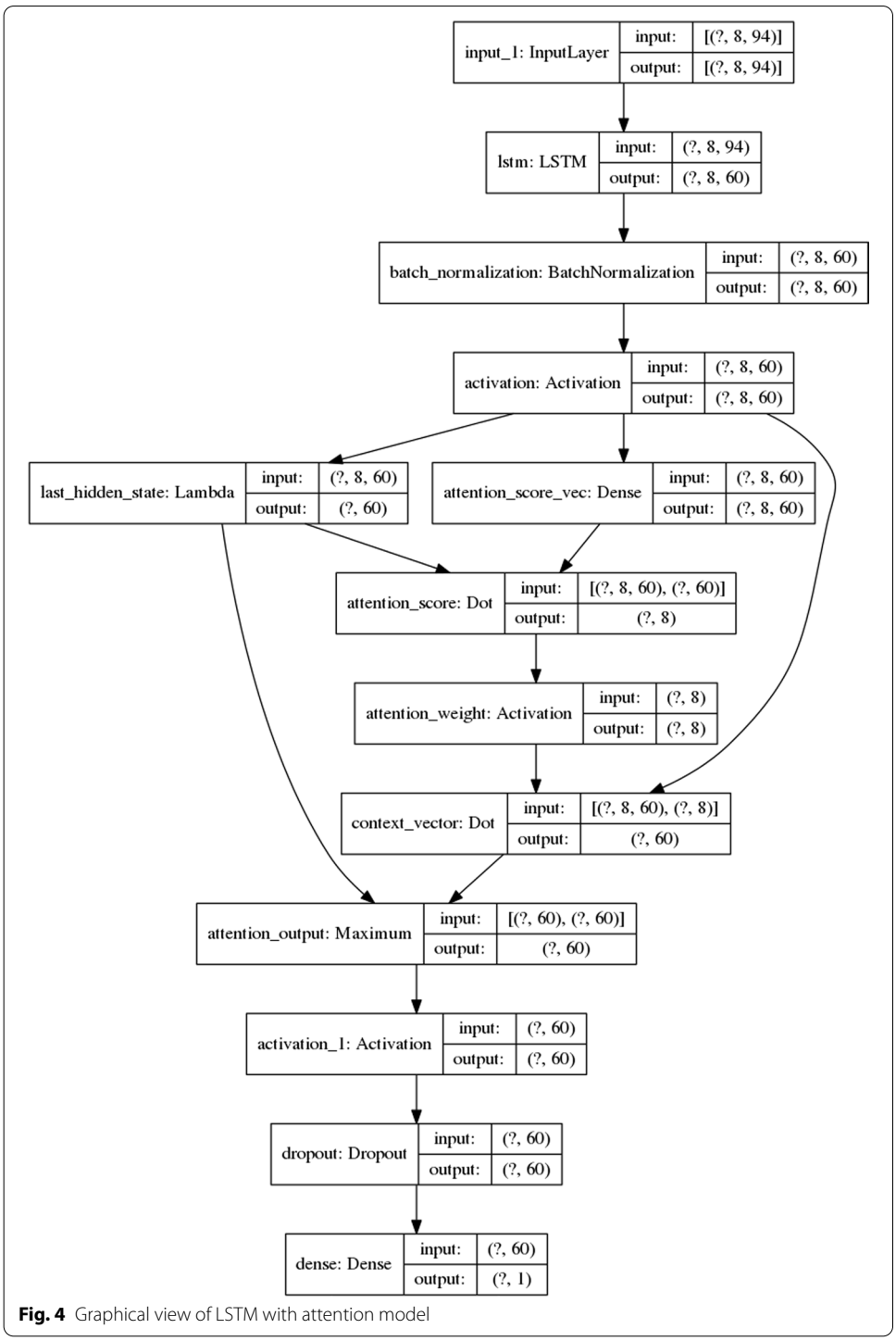

Fig. 4 Graphical view of LSTM with attention model

LSTM considers previous n-time step instances in the prediction of the current time step. Unlike LSTM models, SVM and LightGBM predict the current time step using only previous time step instance. Our study used the time series cross-validation (cv) procedure in the training of SVM and LightGBM in order to fairly compare the performance of both models to those of LSTM. In this cv procedure, the test set was first divided into

Table 4 Parameters of LSTM network

Parameters	Value
Number of memory cells	40
Dropout rate	0.3
Optimizer	Adam
Activation function	ELU
Number of epochs	250
Batch size	32

the predefined number of folds. In this case, the test data were separated into the folds that contained 8 instances each (due to using time step parameter in LSTM as 8 h, the number of instances per fold was defined as 8). Time series cv began with the training set in the first iteration. After completing the model training, the first predictions were done for the instances in the first test fold. In the next iteration, the instances in the first test fold were added to the existing training set and the predictions for the second fold instances were made. This process was additively continued until the final test fold was predicted. This way, SVM and LightGBM model performances were compared with the LSTM models' fairly by including previous instances gradually in the model training. Like in the LSTM models, the parameters of SVM and LightGBM were defined by a grid search process over search space listed in Tables 5 and 6.

Our first models were trained with the own stock features and their performances were assessed by both individual and overall levels. While individual level performances were computed for each stock in terms of accuracy and MA F-measure, overall level performances were calculated by computing the mean of individual stock accuracy and F-measure scores. Classification performances of trained models are listed in Table 7.

The results showed that the LSTM with attention model was superior to the LSTM basic model, SVM, and LightGBM, with an overall accuracy of 0.658, whereas the LSTM basic, SVM, and LightGBM models had accuracy rates of 0.629, 0.620, and 0.631, respectively. The same results could be seen in overall MA F-measure rates. The LSTM with attention model (0.598) had a 1.4% higher performance than the SVM (0.585) classifier in terms of mean MA F-measure. LSTM with attention was also superior in the results obtained on the stock basis. LSTM with attention had higher accuracy rates in seven of

Table 5 Parameter space of SVM

Parameters	Value
Kernel type	{rbf, poly}
Regularization (C)	{1, 2, 4, 8}

Table 6 Parameter space of LightGBM

Parameters	Value
number of learners	{100, 200, 300, 400, 500}
learning rate	{0.1, 0.01}
L2-regularizer	{0.001, 0.0001}
max_depth	{9, 11, 13}

Table 7 Classification results using own stock features

Stocks	LSTM (basic)		LSTM (with attention)		SVM		LightGBM	
	Acc	F-Mea	Acc	F-Mea	Acc	F-Mea	Acc	F-Mea
ALBRK	0.704	0.642	**0.737**	0.668	0.691	0.661	0.711	**0.669**
AKBNK	**0.625**	0.537	0.621	**0.554**	0.585	0.551	0.598	0.539
HALKB	0.569	0.542	**0.629**	**0.595**	0.568	0.557	0.578	0.553
ISCTR	0.629	0.573	**0.668**	**0.574**	0.646	0.564	0.622	0.548
SKBNK	0.674	0.602	**0.744**	**0.653**	0.693	0.649	0.713	0.629
TSKB	0.666	0.593	**0.688**	**0.609**	0.611	0.586	0.681	0.562
VAKBN	0.552	0.545	0.567	0.549	**0.575**	**0.557**	0.562	0.543
YKBNK	**0.615**	0.556	0.609	**0.585**	0.587	0.556	0.585	0.558
Average	0.629	0.574	**0.658**	**0.598**	0.620	0.585	0.631	0.575

Bold numbers indicate the best performance

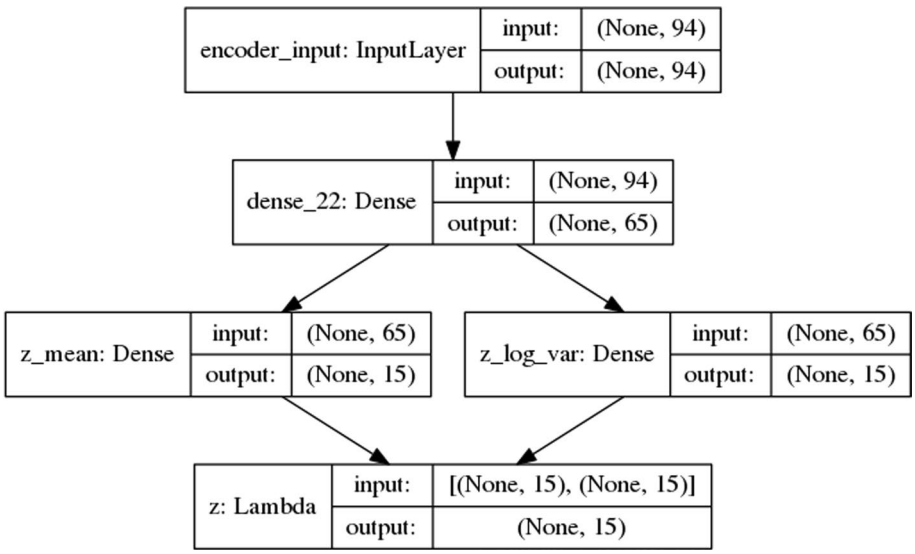

Fig. 5 Encoder component of the VAE

eight stocks compared to the other models. The stocks with the highest accuracies were ALBRK and SKBNK, with a rate of approximately 0.74.

In the second experiment, reduced stock features were given as inputs to the models. VAE was used to reduce the size of the feature vectors while extracting deep and latent properties from the entire feature. In order to decide the size of the reduced feature vectors, reconstruction errors were searched for each stock in terms of MSE. 10, 15, and 30 were selected as a search space for size of reduction, and the obtained results showed that vectors with sizes reduced to 10, 15, and 30 had average MSE rates of 14.21, 10.47, and 9.58, respectively. Considering the reduced dimensions and reconstruction errors, 15 was decided as a reduction parameter. KERAS framework was used to implement our VAE model. The graphical representations of the created VAE are listed in Figs. 5 and 6.

As seen in the illustrations, the size of the feature vectors was reduced from 94 to 15 with the help of the encoding component of the VAE model. As in the first experiments, reduced stock features were provided as inputs to the LSTM basic, LSTM with

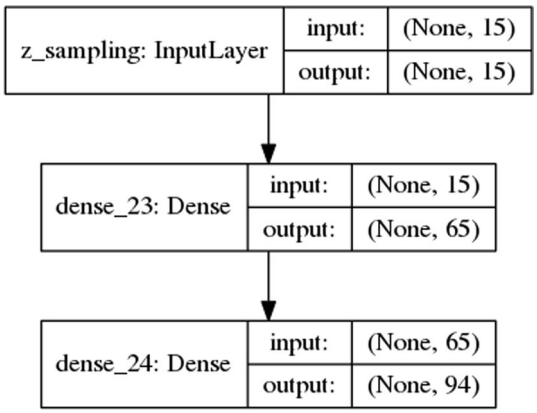

Fig. 6 Decoder component of the VAE

Table 8 Classification results using VAE-reduced stock features

Stocks	LSTM (basic)		LSTM (with attention)		SVM		LightGBM	
	Acc	F-Mea	Acc	F-Mea	Acc	F-Mea	Acc	F-Mea
ALBRK	0.656	0.530	**0.733**	**0.629**	0.712	0.596	0.712	0.601
AKBNK	**0.652**	**0.592**	0.652	0.557	0.645	0.556	0.603	0.520
HALKB	0.569	0.542	0.613	0.546	**0.615**	**0.549**	0.588	0.525
ISCTR	0.621	0.535	0.663	0.549	**0.680**	**0.556**	0.654	0.534
SKBNK	0.685	**0.610**	**0.737**	0.595	0.726	0.564	0.722	0.576
TSKB	0.650	0.530	**0.694**	**0.544**	0.685	0.526	0.645	0.504
VAKBN	0.563	0.553	0.563	0.546	**0.576**	**0.548**	0.566	0.530
YKBNK	**0.583**	**0.541**	0.537	0.529	0.565	0.536	0.575	0.527
Average	0.622	0.554	0.649	**0.562**	**0.651**	0.554	0.633	0.540

Bold numbers indicate the best performance

attention, SVM and LightGBM models. Results of those with reduced features are shown in Table 8.

Classification results showed that the highest accuracy rates were achieved by SVM and the LSTM with attention model. Both models had an accuracy rate of around 0.65, followed by the LightGBM and LSTM basic models with accuracy rates of 0.63 and 0.62. When analyzed in terms of F-measure, the LSTM with attention model had an MA F-measure rate of 0.562, followed by the LSTM basic, SVM and LightGBM models with the rates of 0.554, 0.554, and 0.540, respectively. The LSTM with attention model also led in success rate at the individual level with 4 stocks compared to the LSTM basic and SVM models, which had 2 high performer stocks. ALBRK and SKBNK stocks were again at the forefront with an accuracy of about 0.74.

In order to show the effects of the dimensionality reduction in easing overfitting, we also noted the training accuracies for the own and VAE-reduced stock features. The training accuracy rates for all models are listed in Table 9. The results showed that the biggest change in training accuracy had been made in the SVM model. Since LSTM models use dropout and L2 regularization to prevent overfitting and LightGBM is an

Table 9 Training accuracy rates of own and VAE-reduced stock features

Stocks	LSTM (basic)		LSTM (with attention)		SVM		LightGBM	
	Own	VAE	Own	VAE	Own	VAE	Own	VAE
ALBRK	0.729	0.691	0.773	0.756	0.863	0.823	0.783	0.761
AKBNK	0.626	0.589	0.685	0.686	0.820	0.784	0.720	0.691
HALKB	0.653	0.637	0.665	0.652	0.841	0.780	0.721	0.696
ISCTR	0.619	0.638	0.696	0.679	0.825	0.773	0.726	0.708
SKBNK	0.723	0.723	0.780	0.742	0.862	0.804	0.771	0.741
TSKB	0.640	0.681	0.721	0.723	0.835	0.799	0.767	0.733
VAKBN	0.637	0.618	0.682	0.659	0.839	0.795	0.717	0.685
YKBNK	0.669	0.623	0.677	0.662	0.839	0.781	0.709	0.671
Average	0.662	0.651	0,709	0.694	0.840	0.792	0.739	0.710

Table 10 Classification results using allstock_own features

Stocks	LSTM (basic)		LSTM (with attention)		SVM		LightGBM	
	Acc	F-Mea	Acc	F-Mea	Acc	F-Mea	Acc	F-Mea
ALBRK	0.723	0.632	**0.750**	**0.670**	0.738	0.658	0.740	0.673
AKBNK	0.646	0.554	**0.674**	**0.537**	0.637	0.547	0.623	0.556
HALKB	0.624	0.568	**0.662**	**0.543**	0.637	0.588	0.627	0.593
ISCTR	0.664	0.543	**0.686**	**0.550**	0.680	0.573	0.666	0.572
SKBNK	0.737	0.634	**0.743**	**0.641**	0.741	0.639	0.744	0.652
TSKB	0.689	0.536	**0.694**	**0.611**	0.695	0.603	0.687	0.605
VAKBN	0.594	**0.560**	**0.620**	**0.590**	0.558	0.547	0.562	0.544
YKBNK	0.613	**0.555**	**0.652**	0.531	0.632	0.587	0.608	0.584
Average	0.661	0.573	**0.685**	**0.574**	0.669	0.594	0.657	0.597

Bold numbers indicate the best performance

ensemble learner that reduces model variance, changes in the training accuracy of these models remained limited compared to SVM.

In the last experiments, besides individual stock features, the features belonging to other stocks were also given as inputs to our models. In order to grasp the effects of other banking stocks on individual stock performance, other stock features were merged. Combining the features was done for both own and VAE-reduced stock features. The combination resulted in a 658-dimensional vector for the own stock and a 105-dimensional for the VAE-reduced features. The size of other stocks features was reduced from 658 to 94 using RFE selection for the own stock features. Selected features were combined with the own stock features, and created 188-dimensional feature vectors called allstock_own features. Results of the allstock_own features are shown in Table 10.

In the experiments using allstock_own features, the highest accuracy rate was achieved by the LSTM with attention model, and its overall success was 0.685 in terms of accuracy, followed by the SVM, LSTM basic, and LightGBM models, with 0.669, 0.661, and 0.657 accuracy rates, respectively. Compared to the own stock

features, the use of own and other stock features increased success rate by about 4% in both the LSTM basic and with attention models, and by 2.4% in the LightGBM model. The main contribution of own and other stock features was also seen in overall MA F-measure rates and the overall F-measure of all four models increased by 1.2 to 4.7 %. When stock performance was analyzed at the individual level, LSTM with attention again reached the highest accuracy rates in SKBNK and ALBRK stocks.

Similar to the own stock features, the same procedures were done for VAE-reduced features. The size of other stock features was reduced from 105 to 15 using RFE selection. Selected features were combined with VAE-reduced stock features to create 30-dimensional feature vectors called as allstock_VAE. The classification results obtained with the created feature set are shown in Table 11.

allstock_VAE results showed that the LSTM with attention model was superior to the other models in terms of average accuracy. It had an accuracy of 0.675, followed by the SVM, LightGBM, and LSTM basic models with rates of 0.662, 0.657, and 0.649, respectively. allstock_VAE features resulted in an accuracy improvement of approximately 2.5% in both LSTM and LightGBM models compared to the results obtained with only VAE-reduced features. When the results of allstock_VAE features were compared with allstock_own, it could be seen that the differences between model accuracies were around 1%, and an approximately 3% decrease in the F-measure rates of the SVM and LightGBM models could be seen.

Since allstock_VAE achieved sufficient results with few features, we selected the models trained with these features as best performers and applied a statistical significance test to compare model performances. We employed McNemar's test, a non-parametric statistical test for paired model comparisons, to compare the predictions of the model pairs. McNemar's test uses a contingency table that holds the counts of instances in which two models disagree or agree in the same way. McNemar's test rejects the null hypothesis if the computed p value is below than a defined significance threshold (alpha $= 0.05$), which means the performances of the models are different. If p value is higher than the defined significance level, McNemar's test fails to reject the null hypothesis, which indicates that the compared models have a similar proportion of errors (the two model's performances are equal). The results of McNemar's test with an alpha of 0.05 on allstock_VAE models are presented in Table 12.

Table 11 Classification results using allstock_VAE features

Stocks	LSTM (basic)		LSTM (with attention)		SVM		LightGBM	
	Acc	F-Mea	Acc	F-Mea	Acc	F-Mea	Acc	F-Mea
ALBRK	0.737	**0.656**	**0.741**	0.653	0.741	0.623	0.735	0.617
AKBNK	0.626	0.558	**0.667**	**0.567**	0.643	0.541	0.635	0.537
HALKB	0.579	0.563	**0.660**	**0.562**	0.619	0.555	0.613	0.555
ISCTR	**0.678**	0.554	0.657	**0.575**	0.660	0.567	0.651	0.557
SKBNK	0.749	0.633	**0.750**	**0.624**	0.743	0.580	0.746	0.581
TSKB	0.664	0.576	**0.705**	**0.580**	0.680	0.538	0.677	0.537
VAKBN	0.569	0.549	**0.603**	**0.555**	0.589	0.545	0.580	0.536
YKBNK	0.577	0.554	**0.628**	0.564	0.623	**0.575**	0.618	0.573
Average	0.647	0.580	**0.676**	**0.585**	0.662	0.566	0.657	0.562

Bold numbers indicate the best performance

Table 12 Results of McNemar's test on allstock_VAE trained models

Stocks	LSTM (with attention) versus LSTM basic		LSTM (with attention) versus light GBM		LSTM (with attention) versus SVM	
	Score	p value	Score	p value	Score	p value
ALBRK	62	1	80	0.698	72	0.683
AKBNK	92	**4.77E−06**	102	**0**	174	**0**
HALKB	175	**4.03E−09**	164	0.066	104	0.286
ISCTR	18	0.064	66	**0.004**	106	**0**
SKBNK	41	0.594	71	0.742	77	**0.001**
TSKB	108	**0.002**	119	0.702	129	0.365
VAKBN	131	**1.16E−06**	159	**0**	190	**0**
YKBNK	150	**2.80E−10**	184	**0.001**	71	0.204

Bold numbers indicate the best performance

The results also revealed that the LSTM with attention model performed statistically better within a significance value of 0.05 than the LSTM basic model in 5 stocks. LSTM with attention also had a significant performance difference according to SVM and LightGBM in 4 stocks. Additionally, while ALBRK was the only stock in which all models made errors in similar proportions, AKBNK and VAKBN were two stocks for which the results of the tests were significant between all model pairs, which rejects the null hypothesis.

Conclusions

In this study, we predicted the hourly movement directions of eight banking stocks in Borsa Istanbul using stock prices and technical indicators as features. We selected linear-based (SVM), deep-learning (LSTM) and ensemble learning (LightGBM) models in the prediction process and assessed the model performances in terms of accuracy and F-measure metrics.

We performed our experiments based on different types of feature sets (own stock features, VAE-reduced stock features, allstock_own and allstock_VAE features). In the first experiments, the models were trained with own features and their classification performances in accuracy and F-measure were evaluated. Among the trained models, LSTM with attention excelled compared to LSTM basic, SVM, and LightGBM both in terms of average and individual stock performances. LSTM with attention predicted the movement direction of the stocks with an average of 0.658 accuracy and 0.598 F-measure rates. In order to extract informative and hidden feature representations from stock features, an effective dimensionality reduction architecture, VAE, was used in the second experiment. The size of the stock feature was reduced from 94 to 15 through the VAE. In models trained with reduced stock features, high accuracy rates were achieved in the LSTM with attention and SVM models. Compared to the results obtained without dimensionality reduction, the average classification performances of the reduced features were satisfactory in terms of accuracy, and although the accuracy rates are the equal or higher than those of the models without reduction are, there was a 3% decrease in F-measure rates.

In the last experiments, besides the individual stock features, the features of the other banking stocks were also used. While the size of increased feature space was reduced by RFE selection, models trained with low dimensional features (allstock_own and allstock_VAE) achieved higher accuracy rates than those using individual stock features. As the highest success rate increased up to 0.685 with allstock_own and LSTM with attention model, the combination of allstock_VAE and LSTM with attention model resulted in an accuracy rate of 0.675. The classification results achieved with both feature types were close, but allstock_VAE achieved these results using nearly 16.67% fewer features compared to allstock_own. Additionally, the results of McNemar's test were significant for LSTM with attention in at least in four stocks according to the LSTM basic, SVM, and LightGBM models.

When all experimental results were evaluated, it was found that models trained with VAE-reduced features had similar accuracy rates to those trained without dimensionality reduction. Thus, the conclusion could be made that using all stock features in a prediction boosted the classification performance for all stocks in terms of accuracy. Furthermore, all models had higher classification performance than naive and random benchmark models.

It was also difficult to compare the results of this study with other Borsa Istanbul studies due to the difference in datasets, prediction horizon, and experimental methods used for predicting. For example, the study conducted by Gunduz and Cataltepe (2015) predicted the daily movement direction of the BIST 100 index with Turkish financial news texts, and Term Frequency-Inverse Document Frequency (TF-IDF) was used as a document representative to generate the feature vectors. The classification process was performed with the Naive Bayes algorithm resulting in an accuracy of 0.75. In another study conducted by Gunduz et al. (2017b), a novel Convolutional Neural Network (CNN) architecture was proposed for the prediction of hourly directions of 100 stocks in Borsa Istanbul. The proposed architecture achieved an average F-measure rate of 0.563 for 100 stocks. This study differs from the aforementioned studies (Gunduz and Cataltepe 2015; Gunduz et al. 2017b) in that it gives low-dimensional features extracted by VAE from technical indicators as inputs to several ML models and utilizes other stock features besides their own features. Because of these operations, the number of used features decreased from 94 to 30, while average classifier performance increased up to 0.59 in terms of MA F-measure. In another recent Borsa Istanbul study, deep ensemble models were developed in order to predict the daily direction of the BIST 100 index (Kilimci 2020). Twitter media was used as a news source in the estimation process, and tweets were transformed into feature vectors with different document representation methods such as Word2vec, Glove and TF-IDF. Deep learning architectures such as CNN, RNN and LSTM were used as single learners and ensemble strategies. The success of the proposed method was 0.78 in terms of accuracy with the deep ensembles. The predictions at hourly scales and the use of F-measure in addition to accuracy in performance evaluation make the present study superior.

In the future, the stock networks and graph embedding methods are planned to be used to mine the temporal dependencies between the stocks. We believe that this could allow modelling of the causal dependencies between the stocks and trading hours.

Abbreviations
LSTM: Long-short term memory; SVM: Support vector machines; ML: Machine learning; ANN: Artificial neural networks; RF: Random forest; LDA: Linear discriminant analysis; LR: Logistic regression; DL: Deep learning; BIST: Borsa Istanbul; RFE: Recursive feature elimination; VAE: Variational autoencoder; CNN: Convolutional neural network; MA: Macro-averaged; AE: Autoencoders; SAE: Stacked-autoencoders; SVM: Support vector regression; CM: Confusion matrix; MSE: Mean squared error; RMSE: Root mean squared error; MAE: Mean absolute error; KL: Kullback–Leibler; std: Standard deviation; GA: Genetic algorithm; cv: Cross-validation; DR: Dimensionality reduction; TF-IDF: Term frequency-inverse document frequency.

Acknowledgements
Not applicable.

Authors' contributions
This is a single authored article. HG contributed to model development and coding, dataset preparation and model evaluation. Additionally, the author read and approved the final manuscript.

Funding
This research received no specific Grant from any funding agency in the public, commercial, or not-for-profit sectors.

Data availability statement
The datasets used and analyzed in the current study are allowed by the corresponding authors on reasonable request.

Received: 8 July 2020 Accepted: 12 April 2021
Published online: 21 April 2021

References
Altman N, Krzywinski M (2017) Points of significance: ensemble methods: bagging and random forests
Baldi P (2012) Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML workshop on unsupervised and transfer learning, pp 37–49
Ballings M, Van den Poel D, Hespeels N, Gryp R (2015) Evaluating multiple classifiers for stock price direction prediction. Expert Syst Appl 42(20):7046–7056
Bao W, Yue J, Rao Y (2017) A deep learning framework for financial time series using stacked autoencoders and long-short term memory. PloS one 12(7)
Barboza F, Kimura H, Altman E (2017) Machine learning models and bankruptcy prediction. Expert Syst Appl 83:405–417
Basak S, Kar S, Saha S, Khaidem L, Dey SR (2019) Predicting the direction of stock market prices using tree-based classifiers. North Am J Econ Finance 47:552–567
Bolón-Canedo V, Sánchez-Maroño N, Alonso-Betanzos A (2013) A review of feature selection methods on synthetic data. Knowl Inf Syst 34(3):483–519
Cavalcante RC, Brasileiro RC, Souza VL, Nobrega JP, Oliveira AL (2016) Computational intelligence and financial markets: a survey and future directions. Expert Syst Appl 55:194–211
Chen K, Zhou Y, Dai F (2015) A lstm-based method for stock returns prediction: a case study of china stock market. In: 2015 IEEE international conference on big data (big data). IEEE, pp 2823–2824
Chen S, Ge L (2019) Exploring the attention mechanism in LSTM-based Hong Kong stock price movement prediction. Quant Finance 19(9):1507–1515
Chen Y, Jiang H, Li C, Jia X, Ghamisi P (2016) Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. IEEE Trans Geosci Remote Sens 54(10):6232–6251
Chollet F (2018) Deep Learning mit Python und Keras: Das Praxis-Handbuch vom Entwickler der Keras-Bibliothek. MITP-Verlags GmbH & Co, KG
Fawaz HI, Forestier G, Weber J, Idoumghar L, Muller PA (2019) Deep learning for time series classification: a review. Data Min Knowl Disc 33(4):917–963
Fischer T, Krauss C (2018) Deep learning with long short-term memory networks for financial market predictions. Eur J Oper Res 270(2):654–669
Gu S, Kelly BT, Xiu D (2019) Autoencoder asset pricing models
Guenther N, Schonlau M (2016) Support vector machines. Stata J 16(4):917–937
Gündüz H (2020) Stock market prediction with stacked autoencoder based feature reduction. In: 28th signal processing and communications applications conference. IEEE
Gunduz H (2021) An efficient dimensionality reduction method using filter-based feature selection and variational autoencoders on parkinson's disease classification. Biomed Signal Process Control 66(102):452
Gunduz H, Cataltepe Z (2015) Borsa Istanbul (BIST) daily prediction using financial news and balanced feature selection. Expert Syst Appl 42(22):9001–9011
Gunduz H, Cataltepe Z, Yaslan Y (2017a) Stock daily return prediction using expanded features and feature selection. Turk J Electr Eng Comput Sci 25(6):4829–4840
Gunduz H, Yaslan Y, Cataltepe Z (2017b) Intraday prediction of Borsa Istanbul using convolutional neural networks and feature correlations. Knowl Based Syst 137:138–148
Gunduz H, Yaslan Y, Cataltepe Z (2018) Stock market prediction with deep learning using financial news. In: 2018 26th signal processing and communications applications conference (SIU). IEEE, pp 1–4
Henrique BM, Sobreiro VA, Kimura H (2018) Stock price prediction using support vector regression on daily and up to the minute prices. J Finance Data Sci 4(3):183–201
Hollis T, Viscardi A, Yi SE (2018) A comparison of LSTMs and attention mechanisms for forecasting financial time series. arXiv preprint arXiv:181207699

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017) Lightgbm: a highly efficient gradient boosting decision tree. In: Advances in neural information processing systems, pp 3146–3154
Khalid S, Khalil T, Nasreen S (2014) A survey of feature selection and feature extraction techniques in machine learning. In: 2014 science and information conference. IEEE, pp 372–378
Kilimci ZH (2020) Financial sentiment analysis with deep ensemble models (DEMS) for stock market prediction. J Fac Eng Arch Gazi Univ 35(2):635–650
Kim T, Kim HY (2019) Forecasting stock prices with a feature fusion LSTM-CNN model using different representations of the same data. PloS one 14(2)
Kingma DP, Welling M et al (2019) An introduction to variational autoencoders. Found Trends® Mach Learn 12(4):307–392
Kou G, Peng Y, Wang G (2014) Evaluation of clustering algorithms for financial risk analysis using MCDM methods. Inf Sci 275:1–12
Kou G, Yang P, Peng Y, Xiao F, Chen Y, Alsaadi FE (2020) Evaluation of feature selection methods for text classification with small datasets using multiple criteria decision-making methods. Appl Soft Comput 86(105):836
Kou G, Xu Y, Peng Y, Shen F, Chen Y, Chang K, Kou S (2021) Bankruptcy prediction for SMES using transactional data and two-stage multiobjective feature selection. Decis Support Syst 140(113):429
Kumar BS, Ravi V (2016) A survey of the applications of text mining in financial domain. Knowl Based Syst 114:128–147
Li H, Shen Y, Zhu Y (2018) Stock price prediction using attention-based multi-input LSTM. In: Asian conference on machine learning, pp 454–469
Li J, Bu H, Wu J (2017) Sentiment-aware stock market prediction: a deep learning method. In: 2017 international conference on service systems and service management. IEEE, pp 1–6
Li Q (2019) Application of improved feature selection algorithm in SVM based market trend prediction model. Master Thesis
Liu G, Wang X (2018) A numerical-based attention method for stock market prediction with dual information. IEEE Access 7:7357–7367
Ma X, Tao Z, Wang Y, Yu H, Wang Y (2015) Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. Transp Res Part C Emerg Technol 54:187–197
Mehta S, Rana P, Singh S, Sharma A, Agarwal P (2019) Ensemble learning approach for enhanced stock prediction. In: 2019 twelfth international conference on contemporary computing (IC3). IEEE, pp 1–5
Naik N, Mohan BR (2019) Optimal feature selection of technical indicator and stock prediction using machine learning technique. In: International conference on emerging technologies in computer engineering. Springer, pp 261–268
Nti IK, Adekoya AF, Weyori BA (2019) A systematic review of fundamental and technical analysis of stock market predictions. Artif Intell Rev, pp 1–51
Olah C (2015) Understanding LSTM networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed 19 May 2020
Paiva FD, Cardoso RTN, Hanaoka GP, Duarte WM (2019) Decision-making for financial trading: a fusion approach of machine learning and portfolio selection. Expert Syst Appl 115:635–655
Patel J, Shah S, Thakkar P, Kotecha K (2015) Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. Expert Syst Appl 42(1):259–268
Pillai I, Fumera G, Roli F (2017) Designing multi-label classifiers that maximize f measures: state of the art. Pattern Recogn 61:394–404
Qiu M, Song Y (2016) Predicting the direction of stock market index movement using an optimized artificial neural network model. PloS one 11(5)
Sharma A, Bhuriya D, Singh U (2017) Survey of stock market prediction using machine learning approach. In: 2017 international conference of electronics, communication and aerospace technology (ICECA), vol 2. IEEE, pp 506–509
Song Q, Guo Y, Shepperd M (2018) A comprehensive investigation of the role of imbalanced learning for software defect prediction. IEEE Trans Softw Eng 45(12):1253–1269
Srivastava N, Hinton GE, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
Walker J, Doersch C, Gupta A, Hebert M (2016) An uncertain future: forecasting from static images using variational autoencoders. In: European conference on computer vision. Springer, pp 835–851
Wang Y, Huang M, Zhu X, Zhao L (2016) Attention-based LSTM for aspect-level sentiment classification. In: Proceedings of the 2016 conference on empirical methods in natural language processing, pp 606–615
Wen F, Xu L, Ouyang G, Kou G (2019) Retail investor attention and stock price crash risk: evidence from china. Int Rev Financial Anal 65(101):376
Xingjian S, Chen Z, Wang H, Yeung DY, Wong WK, Woo WC (2015) Convolutional LSTM network: a machine learning approach for precipitation nowcasting. In: Advances in neural information processing systems, pp 802–810
Yan K, Zhang D (2015) Feature selection and analysis on correlated gas sensor data with recursive feature elimination. Sens Actuators B Chem 212:353–363
Yun H, Lee M, Kang YS, Seok J (2020) Portfolio management via two-stage deep learning with a joint cost. Expert Syst Appl 143(113):041
Zhong X, Enke D (2017) Forecasting daily stock market return using dimensionality reduction. Expert Syst Appl 67:126–139
Zhong X, Enke D (2019) Predicting the daily return direction of the stock market using hybrid machine learning algorithms. Financial Innov 5(1):4

Publisher's Note