https://doi.org/10.5281/zenodo.13682826 *Ife Journal of Technology, Vol. 29(1), 2024*

Requirements Engineering Process of Recommender System for the Nigerian Stock Market using Machine Learning Approach

Ishaya P. Gambo
Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria

ABSTRACT

Stock prices are influenced by numerous complex and unpredictable factors, posing significant challenges for novice traders in identifying and utilizing pertinent information for optimal stock selection. This study proposes an innovative approach that integrates requirements engineering (RE) principles with machine learning (ML) techniques to predict stock prices in the Nigerian market based on historical data. The data, encompassing ten years of stock prices and volumes, as well as user preferences and budgets, was utilized to train and test an Artificial Neural Network (ANN) model. The model's predictions were then compared with those from Support Vector Regression (SVR), Linear Regression (LR), K-Nearest Neighbor Regression (KNNR), and Neural Network Regression (NNR). The implementation of the recommender system was carried out using the Python programming language, enabling the detection of patterns in stock price movements and providing stock suggestions based on these patterns to aid traders in making informed decisions. The NNR algorithm demonstrated superior performance, achieving an 85% confidence level. These results suggest that the proposed system can generate reliable predictions to guide stock trading decisions. This research offers a significant contribution to the field by demonstrating the effective synergy between RE and ML in enhancing stock trading strategies. The findings have important implications for both novice and experienced traders, providing a robust tool for navigating the complexities of stock market investments.

KEYWORDS

Artificial neural network
Design specifications
Nigeria stock market
Neural network regression
Linear regression
Support vector regression
Requirements engineering

1. INTRODUCTION

Requirements engineering (RE) is a critical factor in the development of systems, particularly recommender systems (RS) (Gambo *et al.*, 2021a). The RE process for RS is focused on identifying the data that drives recommendations, which are fundamentally the users' requirements (Portugal *et al.*, 2015). This process involves engaging users to deliver high-quality software-intensive systems capable of providing accurate recommendations. RS enables users to discover new and relevant content by monitoring, aggregating data, and displaying tailored suggestions (Resnick and Varian, 1997). RS acts as a software agent that learns its users' interests and needs (Wei *et al.*, 2005).

Despite the advancements, developing RS remains challenging (Portugal *et al.*, 2015). The RE process can help define, articulate, and analyze user requirements for recommendations. For instance, RS for stock market time series requires a comprehensive RE process to simplify complexities and capture users' preferences accurately, thereby predicting trends to avoid financial losses. Machine Learning (ML) techniques are often employed to facilitate these predictions. RE can collaborate with ML to determine the best functional and non-functional requirements for RS.

ML has seen widespread application in various automation problems, enabling significant achievements and advances in applied computing. These include predictive analysis modeling of RS, image and pattern recognition, natural language processing, and outperforming humans in complex games (Khomh *et al.*, 2018).

Recent studies have analyzed stock price behaviors, predictions, and index variations using ML algorithms (Singh & Khushi, 2021; Nabipour *et al.*, 2020; Khan *et al.*, 2019; Gurav & Sidnal, 2018; Cheng *et al.*, 2016). These ML techniques have been instrumental in improving decision-making for stock investors and brokers. Therefore, this paper aims to establish a synergy between RE and ML to provide optimal recommendations for trading in the stock market.

Stock market trading involves leveraging extensive data to support clients. These data sets are often vast, making it challenging to uncover necessary insights for decision-making. Stock trading requires significant experience and the ability to identify patterns and trends in stock price movements (Nair *et al.*, 2015). The RE process can address these challenges by identifying patterns, users' preferences, and expectations, thus providing a suitable platform for applying ML techniques.

Numerous factors can influence stock prices in nonlinear patterns (Caginalp & Desantis, 2011; Jabbarzadeh *et al.*, 2016). This complexity can overwhelm inexperienced traders, who may struggle to identify the crucial information needed for making informed decisions. Consequently, new traders may find it difficult to make profitable decisions without the experience or skills of seasoned traders.

Trading involves buying and selling goods or services, where the selling party is compensated for the value of the goods or services. In financial markets, trading involves buying and selling securities such as stocks, bonds, and commodities (Williams, 2010). The stock market facilitates the buying, selling, and issuing of shares belonging to publicly owned companies, with stock exchanges providing services that include inventory management of exchanged stocks and offering exceptional services to investors and brokers (Jaiswal, 2018). Nigeria hosts five stock exchanges: The Nigerian Stock Exchange, FMDQ over-the-counter (OTC), National Association of Securities Dealers over-the-counter (NASD OTC), Nigerian Commodities Exchange (NCE), and African Commodities Exchange (ACE).

The internet has made transactions faster and more convenient, eliminating the need for intermediaries (Özkan *et al.*, 2010). However, one of the biggest challenges of online trading is not just obtaining adequate information but making the right decision amidst overwhelming data. Users often interact with RS, which suggest content based on previous

Corresponding Author: I. P. Gambo (*ipgambo@oauife.edu.ng*)
The editor responsible for coordinating the review of this article and approving its publication was *O.B. Akinwale*.

activities, thus preventing information overload (Ricci *et al.*, 2010).

This paper focuses on the Nigerian stock market (NSM) as a case study. It gathers NSM characteristics through the RE process, learns factors affecting stock prices, identifies patterns in price movements, and recommends stocks based on user preferences using ML techniques. The system uses product knowledge—either hardcoded by experts or mined from market events and user inputs—to make stock purchase recommendations.

The paper centers on user stock preferences, historical stock price data, and user budgets. It aims to study and examine seasoned stock traders' methods and design an RS to uncover interest patterns in stock price movements. The RS can then generate stock recommendations, supplementing traders' decision-making processes.

The remainder of the paper is structured as follows: Section 2 reviews relevant and earlier work and its limitations. Section 3 details the data collection, training, and RE procedures, along with the techniques used to achieve the research goals. Section 4 outlines the method for model evaluation. Section 5 describes the system evaluation. Section 6 presents the results and discussion, and Section 7 offers conclusions and suggestions for further research.

2. LITERATURE REVIEW

Recommender systems (RS) have their foundations in cognitive science, approximation theory, information retrieval, forecasting theory, management science, and marketing (Portugal *et al.*, 2015; Resnick and Varian, 1997). RS techniques can be broadly categorized into personalized and non-personalized recommendations. Personalized recommendations, which include collaborative filtering, factorization models, and artificial neural networks, tailor suggestions based on individual user preferences (Uchyigit and Ma, 2008; Khatwani and Chandak, 2016). For instance, collaborative filtering employs k-nearest algorithms to build user-centric recommendations by leveraging user preferences (Gong, 2010; Verstrepen and Goethals, 2014; Sun *et al.*, 2016). Non-personalized recommendations, conversely, aggregate general opinions and utilize basic product association recommenders without considering individual user preferences (Poriya *et al.*, 2014; Pereira and Varma, 2016; Falk, 2019).

Another notable approach in RS is content-based filtering, which makes recommendations based on metadata from a user's history and other available items (Pazzani and Billsus, 2007; Lops *et al.*, 2011; Bagher *et al.*, 2017). Knowledge-based recommendations formalize domain-specific knowledge and apply various constraints and item ontologies to make suggestions (Burke, 2000; Bouraga *et al.*, 2014; Tarus *et al.*, 2018; Ameen, 2019). Case-based reasoning focuses on using defined cases to provide explainable recommendations (Musto and Semeraro, 2015; Musto *et al.*, 2015; Caro-Martinez *et al.*, 2019). Hybrid methods combine different decision support methods to enhance precision in recommendations (Burke, 2007; Zhang *et al.*, 2015; Çano and Morisio, 2017; Sharma *et al.*, 2016; Verma, 2018), while complementary methods introduce fuzzy concepts for stock recommendations (Yager, 2003; Yujun *et al.*, 2016).

RS are widely utilized in e-commerce to predict user preferences and recommend products, as seen in platforms like Amazon, Jumia, Konga, Best Buy, Spotify, and Netflix (Itmazi and Gea, 2006; Rao, 2008; Lu *et al.*, 2015). According to Li *et al.* (2012) and Lu *et al.* (2015), RS applications span various domains, including content development (e.g., social media, e-learning, email filtering), entertainment (e.g., movie and music recommendations by Netflix, YouTube, Spotify), services (e.g., travel, expert consulting, rental home recommendations), and finance (e.g., stock, borrowing money, insurance policies, real estate purchases). This study specifically focuses on the application of RS in stock market recommendations.

Previous research in RS for stock markets has employed various methodologies. Paranjape-Voditel and Deshpande (2013) used association rule mining (ARM) to aid stock portfolio recommendations by identifying correlations and hidden relationships between stocks. Nair and Mohandas (2015) developed an intelligent RS using a decision tree-support vector machine and a genetic algorithm to identify patterns in stock price movements and recommend optimal trading strategies. Additionally, Nair *et al.* (2015) proposed mining temporal association rules from stock price data using a genetic algorithm-optimized Symbolic Aggregate approXimation (SAX)-Apriori method to provide insights into the capital market's success rates.

While these studies highlight the potential of RS in stock market applications, gaps remain in effectively integrating user constraints such as budget and leveraging collaborative filtering techniques. This paper aims to address these gaps by applying collaborative filtering techniques to stock recommendations, considering the vast amount of available data and user-provided budget constraints. By doing so, this study contributes to the advancement of RS methodologies in the context of stock trading, providing a robust tool for both novice and experienced traders to make informed decisions.

3. METHODOLOGY

The paper adopted quantitative and case study research approaches. The Nigerian stock market (NSM) was used as the case study. The quantitative approach involves measuring and predicting data (Takhteyev & Hilts, 2010; Gousios *et al.*, 2014). The quantitative approach's goal centers on the extraction of data on the prices in the stock market from the Knoema Corporation. Thus, the developed system's model begins with collecting data on past stock prices. For prediction, SVR, LR, KNNR, and NNR were used. These models were compared to ascertain the one that performed better. Figure 1 shows the context diagram for this paper.

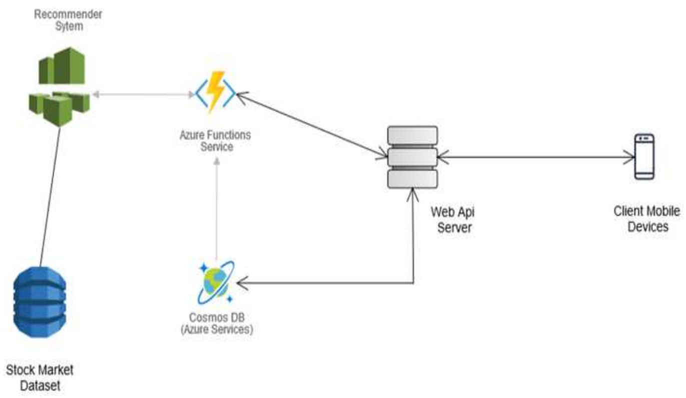


Figure 1. Context diagram for the developed RS

In this paper, initial data was inputted into the RS to analyze these data and map out what factors to look out for to recommend credible suggestions successfully. Additionally, the RS requires certain information from the user if, perchance, the user has stocks they have a preference for and the budget constraint that should be considered while recommending what stocks to buy and in what quantity.

The Recommender System (RS) was built on the .NET framework, a robust platform that enables scalability for both web and mobile applications. This framework supports object-oriented programming, allowing developers to treat all components as objects, which aids in abstracting real-world complexities. The RS utilizes collaborative filtering, content-based, and hybrid methods to build user profiles and generate recommendations. It incorporates trader preference mining (Vu, 2014), stock data association mining (Ting *et al.*, 2006), and rule-based algorithms (Isinkaye *et al.*, 2015) to compute recommendations.

3.1. Data Collection and Cleaning

Data was extracted from the Knoema Corporation historical data repository on the Nigerian stock market (https://knoema.com/atlas/Nigeria/datasets). The historical data[1] contains the dates and other relevant information that provides a tremendous amount of data set with greater efficiency. These datasets are available through multiple mediums such as JavaScript object notation (JSON) and text files Excel, Comma Separated Values (csv), or pdf.

Moreover, the data set collected was preprocessed before any meaningful activity was done. The data was cleaned for missing values and inconsistencies in the data values, smoothing the noisy data. The preprocessed data were supplied to the machine learning algorithms mentioned in section 3.1 to use as a training set for the RS. The system's recommendations are then delivered to the users via some application programming interface (API) endpoints. These endpoints are also used to send more data from the user to the RS to improve each user's recommendations. As such, the system is a personalized RS.

With the personalized system, each user indicates what stocks they are interested in based on these preferences. The recommendation is made on a combination of stocks to buy that would give the user the highest yield on his investment. The system was implemented in Python to exploit the vast amount of readily available machine learning algorithms written in Python. The intermediary API between the RS and third-party client applications (Web and Mobile) was implemented using Microsoft's ASP/Net Core framework. The system was then tested for accuracy, and the conference level was compared.

3.2. Training Process

The training process needs current data on the intended results. The cleaned-up data set was used to train the neural network prediction model. To ensure that a minimal network performance function was created, the network of neurons and their biases were reused during training. Since hidden units lack a training output value that can be applied, backpropagation was utilized as one of the primary training methods to achieve this. They are therefore forced to rely on mistakes from the prior layer. In this regard, comparisons were made using the goal values of the output layer (Sadeghian & Tahayori, 2015). Also, the Multilayer Perception (MLP) was used to describe the feedforward network. In this context, the MLP has three layers: an input layer, a hidden layer, and the output layer. Figure 2 depicts the flowchart of the MLP training process (Zainal-Mokhtar & Mohamad-Saleh, 2012). The derived Artificial Neural Network (ANN) is shown in Figure 3.

The model's training was done in 3 cycles to derive the ANN in Figure 3. The training data set was partitioned randomly into a 70:30 ratio for the first cycle. The remaining 30% was utilized to validate by determining whether the network is generalizing and halting training before overfitting, while the remaining 70% was used to train the data set. The data set was separated into two parts for the remaining two cycles, 75:25 and 80:20, respectively. This was done to ensure that the model was generalizing the data and not merely fitting itself with the given data set.

3.3. Requirements Engineering Process

The focus of requirements specification was on the user's stock preferences and the user's stocks to enable the recommender to provide a personalized experience for the user. The requirements specifications are used for the software design and coding process (Gambo *et al.*, 2020; Gambo *et al.*, 2014; Gambo *et al.*, 2021b). Table 1 shows the various requirements specification and their categories.

For the design specification, and architecture of the RS was described, as shown in Figure 4. The system architecture in this context shows the processes and identifies how the data set was classified for training purposes. Besides, the Unified Modelling Language (UML) was used in designing the system. UML gives the diagrammatical views of the system. It serves as a blueprint that guides the development of the system from a software development perspective. Next, the system was further represented using the use case, activity, sequence, and class diagrams, as shown in Figure 5, 6, 7, and 8.

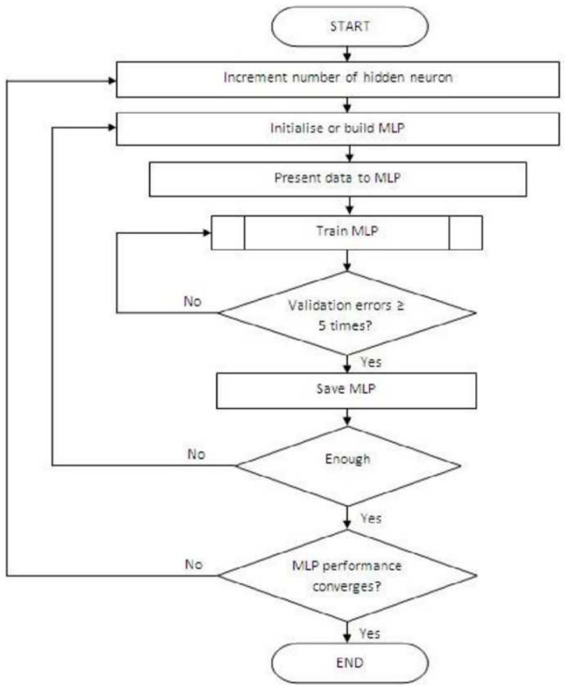


Figure 2. Flowchart of the MLP training process (Zainal-Mokhtar & Mohamad-Saleh, 2012)

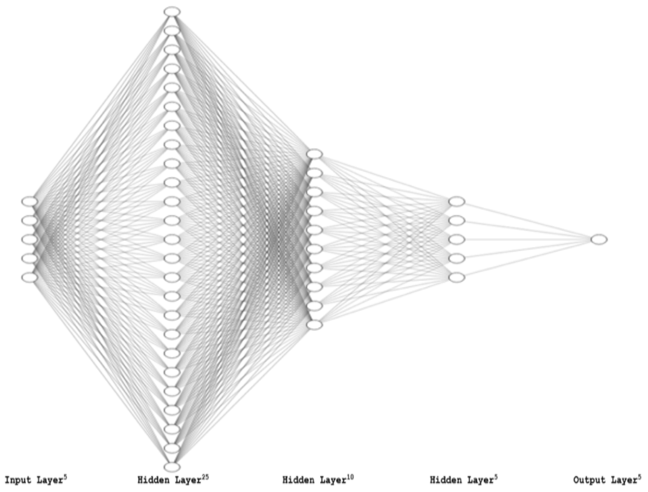


Figure 3. Derived Artificial Neural Network

The use case diagram in Figure 5 depicts the system from a user's point of view. It consists of the users (i.e., actors') use cases which are the actions that the users can carry out. The unadorned line shows the interaction between actors and use

[1] http://doi.org/10.5281/zenodo.4718403

cases. The use case aids the analysis of the system by showing users the system and the different actions they can perform. In particular, the RS and the user are the two actors in the system. On the one hand, the user interacts with the system by inputting preferences, making a recommendation request, accepting or rejecting the recommendation, and updating profile information. On the other hand, the RS provides the recommendation, gets users' preferences, and fetches the data set.

Table 1. Description of requirements specification and categories

SN	Requirement description	Category
1	There must be no loss of user data	Functional Requirements
2	Output must be just the topics recommended	
3	The system must display information to the user via the graphical user interface.	
4	The user data should be kept sensitive with no security breaches.	Non-functional
5	The user can navigate to different sections of the client application.	
6	Users should get real-time updates on the changes in prices.	
7	The user may or may not have prior knowledge of buying stocks.	User Requirements
8	Users need to specify their budget for each recommendation they require.	
9	Users need to specify stocks they have a preference for.	
10	The RS should provide real-time updates on the prices of stocks.	System Requirements
11	The system should generate suggestions based on the stock preferences of the user.	

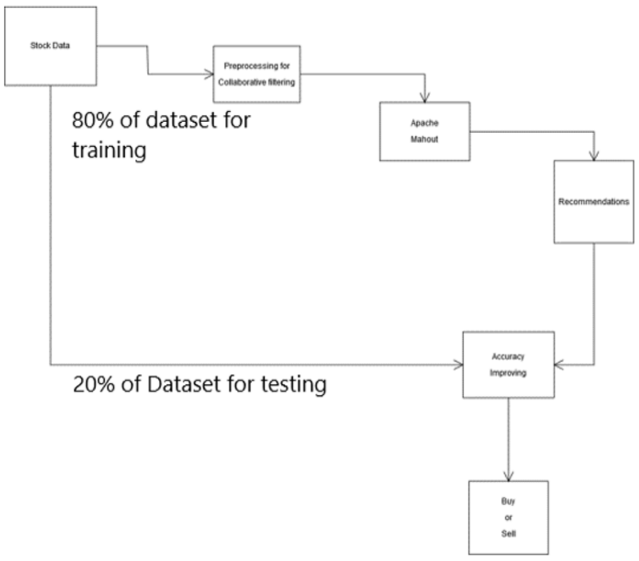

Stock Data
Preprocessing for Collaborative filtering
Apache Mahout
Recommendations
80% of dataset for training
20% of Dataset for testing
Accuracy Improving
Buy or Sell

Figure 4. System architecture for RS

The activity diagram in Figure 6 was used to model the various Dynamics of the RS. It describes the flow of control between the different activities in the RS. Also, the sequence diagram in Figure 7 shows the complete integrated system and the ordering process. Additionally, the class diagram in Figure 8 established the relationship between the objects in the RS.

4. STRATEGY FOR MODEL EVALUATION

After the design phase, the implementation of the system started, in which the design was implemented as a prototype tool. The Model View Controller (MVC) concepts and Model View View-Controller (MVVM) were employed in the implementation process. Python programming language was used for the RS. The source code for implementation is described here[2].

4.1. Processing Historical Stock Market Prices Data

Historical prices of the NSM were collected via the official SDK (Software Development Kit) of Knoema Corporation (https://knoema.com/atlas/Nigeria/datasets). The data had multiple rows for each date for indexes Open, High, Low, Close, Volume, and Change percentage. These indexes with the same dates were merged as a row of data, and the data set was then saved as a CSV file.

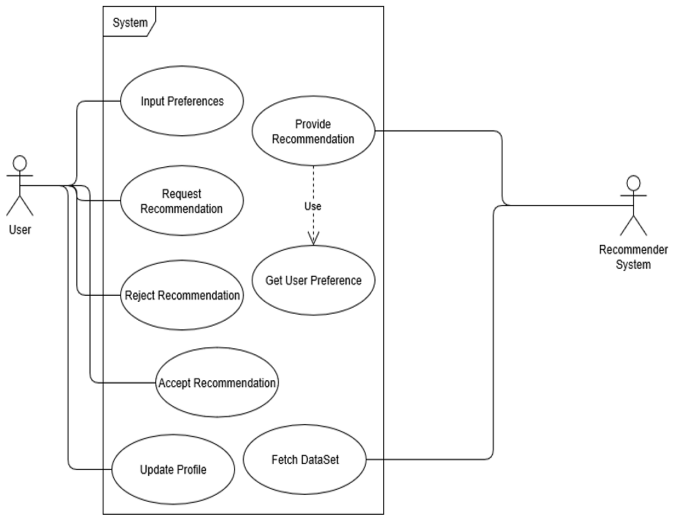


Figure 5. Use case diagram for the RS

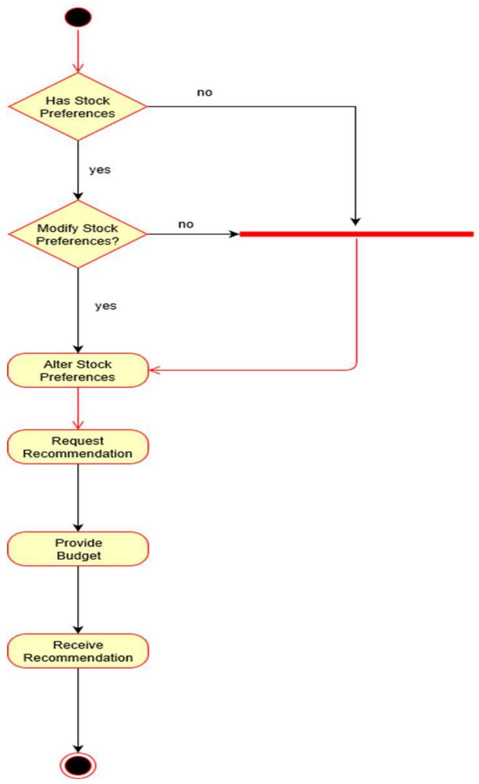


Figure 6. Activity diagram for the RS

[2] http://doi.org/10.5281/zenodo.4718395

4.2. Model training and Validation

Multiple models were trained with SVR using kernel option Radial Basis Function (RBF), SVR with kernel option Linear, SVR with kernel option Polynomial, LR, and KNNR. The data was divided into 3 groups: (i) 80% for training and 20% for testing, (ii) 75% for training and 25% for testing, and (iii) 70% for training and 30% for testing. The whole data sets obtained were for one thousand, nine hundred, and eighteen (1918) days of the NSM stock market prices. The data set for the implementation date is from January 1, 2012, to October 29, 2019.

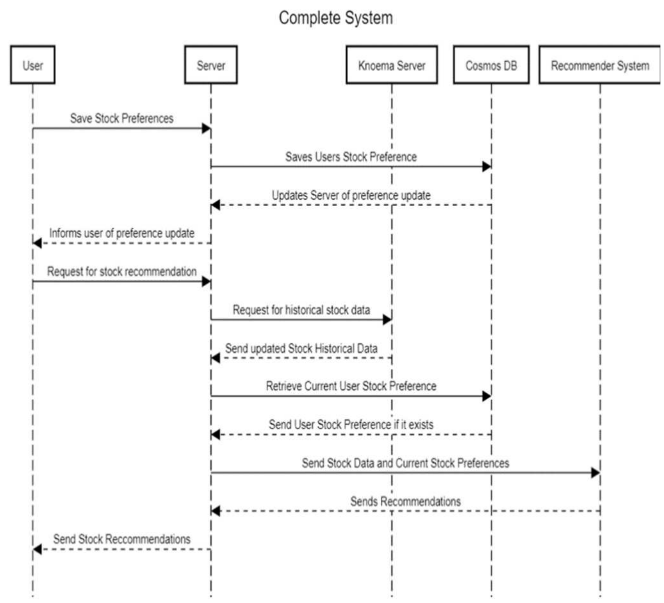

Complete System
User
Server
Knoema Server
Cosmos DB
Recommender System
Save Stock Preferences
Saves Users Stock Preference
Updates Server of preference update
Informs user of preference update
Request for stock recommendation
Request for historical stock data
Send updated Stock Historical Data
Retrieve Current User Stock Preference
Send User Stock Preference if it exists
Send Stock Data and Current Stock Preferences
Sends Recommendations
Send Stock Reccommendations

Figure 7. Sequence diagram for the complete integrated system

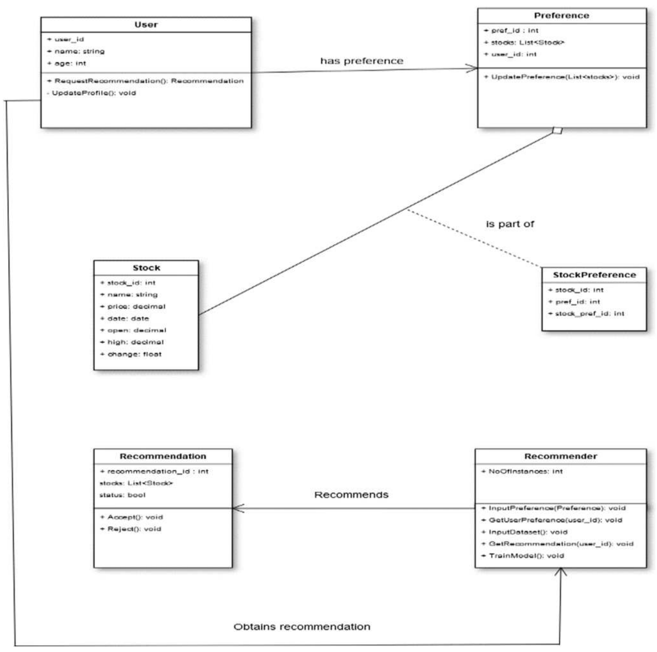


Figure 8. Class diagram for the RS

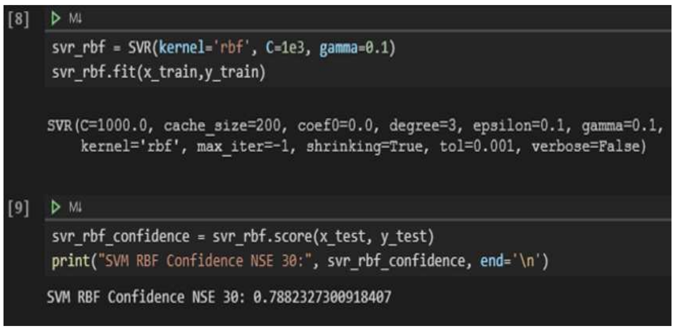
4.3. Procedures for Model Evaluation

Date (in days), Open (stock price at opening of trading), High (stock price at highest trading volume), Low (stock price at lowest trading volume), Close (stock price at closing of trading), and Adjusted Close make up the data used (the price of the stock at the closing of the trading with adjusted dividends). The initial data set is a set of files containing data about a particular price index on the stock market. Each of these files (in CSV format) was read and combined to provide a single dataset used in the developed system. It was then noted that the new data set contained a few null values. This situation was tackled by filling these cells with the median values of the mean of the other indexes in its row. The pandas' framework in Python allows for a description of the dataset, which gives a statistic on the whole dataset.

Furthermore, a forecast value of 30 days is set in a variable 'forecast_out' to specify the number of days the prediction will occur. A new column, 'prediction,' was added to the data frame to handle the storage of the predictions made by the model. The model was configured to predict the opening price for the specified forecast days. To achieve this, the values from the 'Opening' price column are copied into the predictions column except for its last number of forecast_out value values specified (30 in this case). This means that the 'prediction' column has the same values as the 'Opening' column bar the last 30 cells in the 'prediction' column, which are left empty, so the predicted values will be stored and compared with their equivalent from the 'Opening' column.

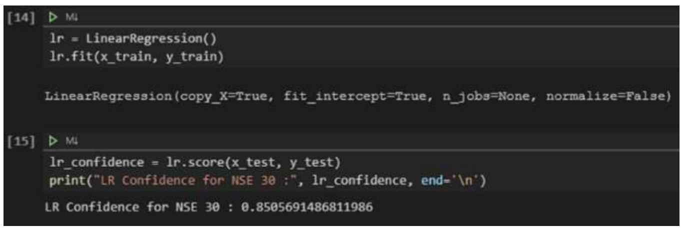
The 'prediction' column is then reshaped into a 1D array using Numpy's reshape function and then passed into a variable' y'. A new data frame was derived by dropping the 'prediction' column from the initial data frame. This new data frame is then converted to an array using Numpy's array function. This function is then assigned to a variable y'. These new variables were then passed as parameters for Sklearns' model_selection class' "train_test_split" method that is used for splitting the dataset into "test" and "training" data for training a model as well as testing the trained model. The method accepts an additional parameter, 'test-size,' which specifies the ratio of the dataset to be used for testing while using the rest are stored for training. To achieve a satisfactory result, this was done with 3 different ratios; the first was 0.3 (meaning 30% of the dataset was used for testing while the other 70% was used in training the model). The subsequent training had values of 0.25 and 0.2.

The confidence values for the SVR model, LR model, KNNR model, and NNR model are shown in Figure 9, 10, 11, and 12, respectively.



```
[8]  ▷ M↓
     svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
     svr_rbf.fit(x_train,y_train)

    SVR(C=1000.0, cache_size=200, coef0=0.0, degree=3, epsilon=0.1, gamma=0.1,
        kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)

[9]  ▷ M↓
     svr_rbf_confidence = svr_rbf.score(x_test, y_test)
     print("SVM RBF Confidence NSE 30:", svr_rbf_confidence, end='\n')

    SVM RBF Confidence NSE 30: 0.7882327300918407
```

Figure 9. Support vector regression model confidence value

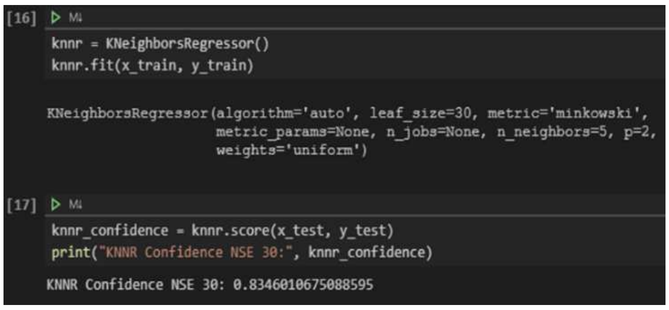


```
[14]  ▷ M↓
      lr = LinearRegression()
      lr.fit(x_train, y_train)

     LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)

[15]  ▷ M↓
      lr_confidence = lr.score(x_test, y_test)
      print("LR Confidence for NSE 30 :", lr_confidence, end='\n')

     LR Confidence for NSE 30 : 0.8505691486811986
```

Figure 10. Linear regression model confidence value

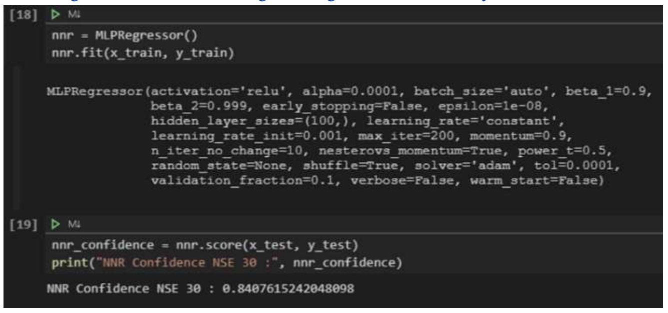
5. SYSTEM EVALUATION

The neural network input later was supplied with five (5) inputs: the indexes from the stock data, the Opening Price, the Highest Price, the Lowest Price, the Adjusted Close price, and the Volume of the stock bought. The expected output was an opening price prediction forecast for the next 30 days. The result of this experiment was compared with the results of models of LR, SVR, and KNNR. The results are shown in Tables 2, 3, and 4.



```
[16]
knnr = KNeighborsRegressor()
knnr.fit(x_train, y_train)

KNeighborsRegressor(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                    weights='uniform')

[17]
knnr_confidence = knnr.score(x_test, y_test)
print("KNNR Confidence NSE 30:", knnr_confidence)

KNNR Confidence NSE 30: 0.8346010675088595
```

Figure 11. K-Nearest neighbor regression model confidence value



```
[18]
nnr = MLPRegressor()
nnr.fit(x_train, y_train)

MLPRegressor(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
             beta_2=0.999, early_stopping=False, epsilon=1e-08,
             hidden_layer_sizes=(100,), learning_rate='constant',
             learning_rate_init=0.001, max_iter=200, momentum=0.9,
             n_iter_no_change=10, nesterovs_momentum=True, power_t=0.5,
             random_state=None, shuffle=True, solver='adam', tol=0.0001,
             validation_fraction=0.1, verbose=False, warm_start=False)

[19]
nnr_confidence = nnr.score(x_test, y_test)
print("NNR Confidence NSE 30 :", nnr_confidence)

NNR Confidence NSE 30 : 0.8407615242048098
```

Figure 12. Neural network regression model confidence value

Table 2. Testing with 20% dataset

SN	Model	Confidence level(s)	Value in Percentage
1	Support Vector Regression with kernel option RBF	0.7882	78.82%
2	Linear Regression	0.8505	85.05%
3	K-Nearest Neighbor Regression	0.8346	83.46%
4	Neural Network Regression	0.8407	84.07%

Table 3. Testing with 25% dataset

SN	Model	Confidence level(s)	Value in Percentage
1	Support Vector Regression with kernel option RBF	0.7664	76.64%
2	Linear Regression	0.8275	82.75%
3	K-Nearest Neighbor Regression	0.8098	80.98%
4	Neural Network Regression	0.8138	81.38%

Table 4. Testing with 30% dataset

SN	Model	Confidence level(s)	Value in Percentage
1	Support Vector Regression with kernel option RBF	0.7661	76.61%
2	Linear Regression	0.8460	84.60%
3	K-Nearest Neighbor Regression	0.8310	83.10%
4	Neural Network Regression	0.8413	84.13%

6. RESULTS AND DISCUSSION

The visual aids for each model were done using Matplot lib to plot the regression graph, as seen in Figure 13. Other graphs plotted are the Time series of the original dataset in Figure 14. The graphical comparisons between the forecast of the prediction models and the original dataset are shown in Figure 15, 16, 17, and 18, respectively.

The results depicted in Figure 13 only showed the predictions for a forecast of 7 days. Carrying out the test for more extended forecasts indicates that predictions cannot be trusted for longer periods. This can be attributed to the factors that affect stock prices. They are more likely to change more often over a more extended period. Therefore, the model fails because it can no longer work as it does not consider the other factors that affect the prices. This implies that the current system can only be trusted to make accurate predictions for a short period.

Figure 13 indicates that in terms of the shape of the predictions made by the various models, only the graph predictions of the LR and the Neural Network Prediction are similar to the actual data graph. This suggests that the Linear and Neural Network Regressors can understand the provided data and, therefore, project values that are similar to the expected values.

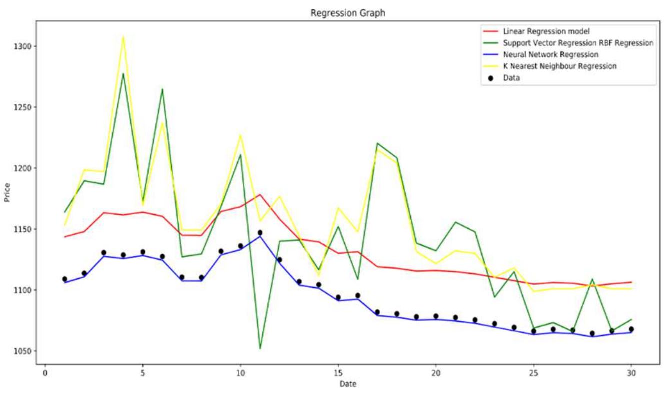


Figure 13. Regression graph of the prediction models comparing the forecasts and actual data set

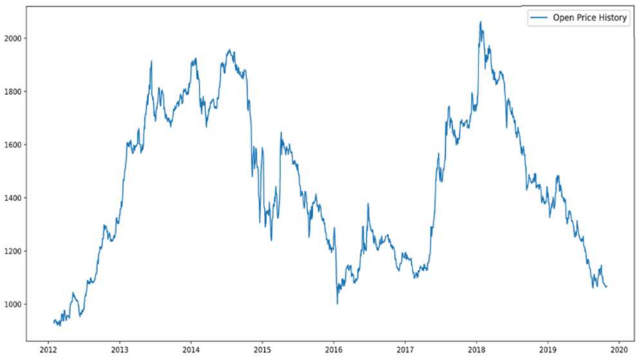


Figure 14. Time series data plot graph for the data set

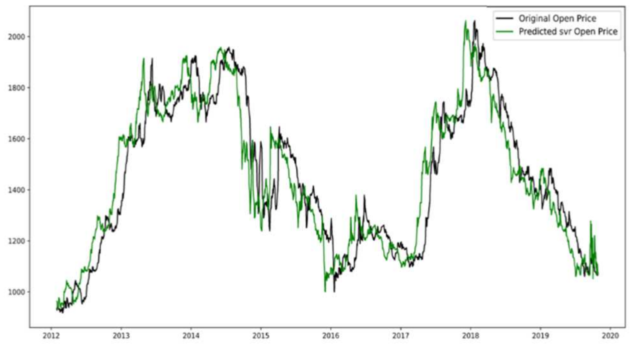


Figure 15. Time series data plot graph comparing data set and SVR Prediction

Regressors can deduce the modeling was correctly done by analyzing the graph and comparing the Linear (Figure 16) and Neural Network (Figure 18). However, considering the data trend, only the Neural Network Regressor (NNR) prediction values are similar to the actual values. This means the NNR values are more accurate than that of the LR. Thus, LR tries to fit the data using a Linear equation y = mx + c; where c is a constant. Moreover, since the stock market is not a linear phenomenon affected by several factors, it is nearly impossible to get an accurate value for this constant. Table 5 shows the results of the Neural Network prediction model as compared with other models.

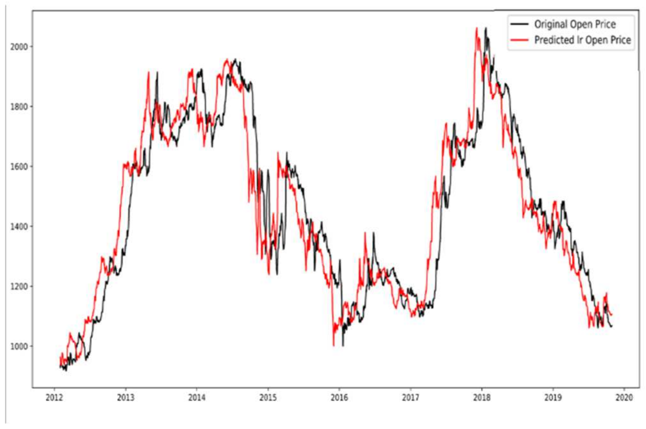


Figure 16. Time series data plot graph comparing data set and LR Prediction

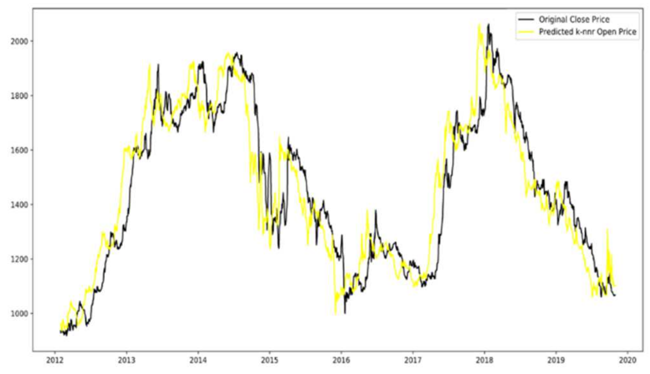


Figure 17. Time series data plot graph comparing data set and KNNR prediction

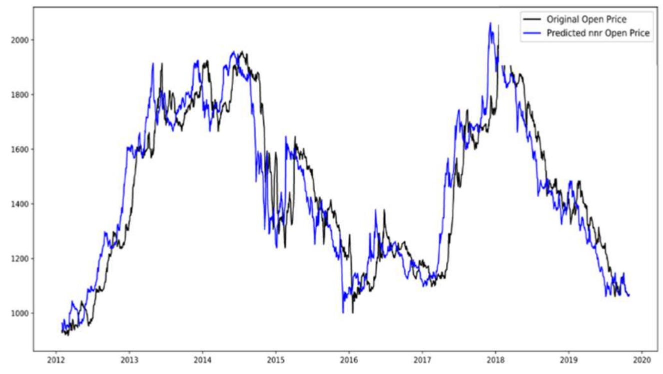


Figure 18. Time series data plot graph comparing data set and NNR prediction

Table 5. Comparison of prediction models for evaluation

Model/Dataset division ratio	Neural Network Regression Model	Linear Regression	Support Vector Regression	K-Nearest Neighbor Regression Model
80:20	0.8407	0.8505	0.7882	0.8346
75:25	0.8138	0.8275	0.7664	0.8098
70:30	0.8413	0.8460	0.7661	0.8310

The Neural Network outperformed other regression models when the results of this model were compared to those of other models. There are three (3) algorithms available when using Neural Networks in deep learning; Scaled Conjugate Gradient (SCG), Levenberg-Marquardt (LM), and Bayesian Regularization (BR) (Kayri, 2016; Selvamuthu *et al.*, 2019). The algorithm used in this project is the BR (Burden & Winkler, 2008) because it gives the least mean squared error when compared to the other two algorithms (LM and SCG) on an overall dataset. The BR uses both the training and validation parts; hence, the data is split into two (2) parts for training and testing. The performance is validated through regression plots, as shown in Figure 13.

To calculate the price at the end of the day, the neural network was given the Opening price, Highest price, Lowest price, Adjusted close price, and the volume of the stock bought (where data is available) during the day. These indexes represent the five (5) inputs at the input layer of the neural network, and other parameters used are the 25, 10, and 5 neutrons in three (3) hidden layers, respectively, learning rate (alpha) of 0.001, Momentum constant of 0.9 and Max Epochs of 200. Therefore, the model can be described as ANNM.5.25.10.5.1; 5 inputs, 25 hidden neurons, 10 hidden neurons, 5 hidden neurons, and 1 output. The ANNR model is a multilayer perceptron (MLP) using a feedforward network, trained using backpropagation.

To develop the prediction model, the implementation method should incorporate processes like data collecting, preprocessing, and model evaluation. The network's performance was about 85% in five inputs, 3 hidden layers with neurons of 25, 10, and 5. It can be seen that although the Linear Regressor has a better performance than the Neural network, it does not fit closely with the data set; hence there is a more significant error in its forecast values.

Therefore, this NNR model may predict future stock price behavior and movement, allowing individual and institutional investors, financial analysts, and consumers of financial news to act accordingly in their trading to increase profits and minimize losses.

7. CONCLUSION AND FUTURE WORK

This paper focuses on applying the Requirements Engineering (RE) process to design a Recommender System (RS) for the Nigerian Stock Market (NSM) and using a machine learning approach for predictive analysis. The dataset comprises historical stock prices in Nigeria. The designed RS aims to assist stock market traders, fund managers, and individual investors in making informed decisions by predicting stock prices. The implementation tools include Python, Visual Studio Code, and Jupyter Notebook, along with libraries such as NumPy, Scikit-Learn, Pandas, and Matplotlib.

The system was trained with several regression algorithms and achieved the best performance when using the NNR algorithm, with an 85% confidence value. This result shows that the system can confidently predict prices that can be considered when making decisions in stock trading.

Further, the paper contributes to knowledge by providing the design specifications from the RE perspective of a stock market RS in the Nigerian context that: (i) helps users select stocks that they might prefer, as well as other stocks, and (iii) serves as a decision support system for users who may or may not have prior knowledge of stocks.

In future work, additional features can be used as inputs with the price indexes for price prediction. These features may include analyzing news surrounding the entities and companies that the stocks belong to and natural event occurrences.

REFERENCES

Adomavicius, G., & Tuzhilin, A. (2002). An architecture of e-butler: A consumer-centric online personalization system. *International Journal of Computational Intelligence and Applications*, *2*(03), 313-327.

Ameen, A (2019). Knowledge based Recommendation System in Semantic Web - A Survey, *Int. J. Comput. Appl.,* 182(43), 20–25, 2019, doi: 10.5120/ijca2019918538

Bagher, R. C., Hassanpour, H., & Mashayekhi, H. (2017). User trends modeling for a content-based recommender system. *Expert Systems with Applications*, *87*, 209-219.

Bouraga, S., Jureta, I., Faulkner, S., & Herssens, C. (2014). Knowledge-based recommendation systems: a survey. *International Journal of Intelligent Information Technologies (IJIIT)*, *10*(2), 1-19.

Burden, F., & Winkler, D. (2008). Bayesian regularization of neural networks. In *Artificial neural networks*, 23-42. Humana Press.

Burke, R. (2000). Knowledge-based recommender systems, *Encyclopedia of library and information systems*, 69 (Supplement 32), 175-186.

Burke, R. (2007). Hybrid web recommender systems, In *The adaptive web*, 377-408. Springer, Berlin, Heidelberg.

Caginalp, G., & Desantis, M. (2011). Stock price dynamics: nonlinear trend, volume, volatility, resistance and money supply. *Quantitative Finance*, *11*(6), 849-861.

Çano, E., & Morisio, M. (2017). Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, *21*(6), 1487-1524.

Caro-Martinez, M., Recio-Garcia, J. A., & Jimenez-Diaz, G. (2019). An algorithm independent case-based explanation approach for recommender systems using interaction graphs. In *International Conference on Case-Based Reasoning,* 17-32. Springer, Cham.

Falk, K. (2019): *Practical recommender systems*. Manning Publications, 2019.

Gambo, I.P. and Taveter, K., (2021a). A Pragmatic View on Resolving Conflicts in Goal-oriented Requirements Engineering for Socio-technical Systems. In *ICSOFT* (pp. 333-341).

Gambo, I. and Taveter, K., (2021b). Stakeholder-Centric Clustering Methods for Conflict Resolution in the Requirements Engineering Process. In International Conference on Evaluation of Novel Approaches to Software Engineering (pp. 183-210). Springer, Cham.

Gambo, I. P., Odukoya, H. O., Oke, A. A., & Adagunodo, E. R. (2020). Analysis and Classification of Requirements Specification for Web Application Development: A Case Study Approach. Journal of Computer Science and Its Application, 27(1), 144-160, 2020. https://dx.doi.org/10.4314/jcsia.v27i1.12.

Gambo, I., Soriyan, A., & Ikono, R. (2014). Framework for enhancing requirements engineering processes: a conceptual view of health information system. *International Journal of Computer Applications*, 93(2), 19-26.

Gong, S. (2010): A collaborative filtering recommendation algorithm based on user clustering and item clustering, *JSW*, *5*(7), 745-752.

Gousios, G., Pinzger, M., & Deursen, A. V. (2014). An exploratory study of the pull-based software development model. In *36th International Conference Proceedings on Software Engineering (ICSE'14), Hyderabad, India, May 31 to June 7, 2014,* 345–355.

https://knoema.com/atlas/Nigeria/datasets

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. *Egyptian Informatics Journal*, *16*(3), 261-273.

Itmazi, J., & Gea, M. (2006). The recommendation systems: Types, domains and the ability usage in learning management system. In *Proceedings of the International Arab Conference on Information Technology (ACIT'2006), Yarmouk University, Jordan*.

Jabbarzadeh, A., Shavvalpour, S., Khanjarpanah, H., & Dourvash, D. (2016). A multiple-criteria approach for forecasting stock price direction: nonlinear probability models with application in S&P 500 Index. *International Journal of Applied Engineering Research*, *11*(6), 3870-3878.

Jaiswal, A. (2018). *What is the difference between a Stock Market and a Stock Exchange?* Retrieved from Quora: https://www.quora.com/What-is-the-difference-between-a-Stock-Market-and-a-Stock-Exchange.

Kayri, M. (2016). Predictive abilities of Bayesian regularization and Levenberg–Marquardt algorithms in artificial neural networks: a comparative empirical study on social data. *Mathematical and Computational Applications*, *21*(2), 20(2016). https://doi.org/10.3390/mca21020020

Khatwani, S. and Chandak, M. B. (2016). Building personalized and non-personalized recommendation systems, In *2016 IEEE International Conference on Automatic Control and Dynamic Optimization Techniques (ICACDOT)*, 623-628.

Li, L. H., Hsu, R. and Lee, F. M. (2012). Review of Recommender Systems and Their Applications, *T&S Journal Publications*, 1-38.

Lops, P., De Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In *Recommender systems handbook* (pp. 73-105). Springer, Boston, MA.

Lops, P., Jannach, D., Musto, C., Bogers, T., & Koolen, M. (2019). Trends in content-based recommendation. *User Modeling and User-Adapted Interaction*, *29*(2), 239-249.

Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments: a survey. *Decision Support Systems*, *74*, 12-32.

Musto, C., & Semeraro, G. (2015). Case-based Recommender Systems for Personalized Finance Advisory. In *FINREC,* 35-36.

Musto, C., Semeraro, G., Lops, P., De Gemmis, M., & Lekkas, G. (2015). Personalized finance advisory through case-based recommender systems and diversification strategies. *Decision Support Systems*, *77*, 100-111.

Nair, B. B., & Mohandas, V. P. (2015). An intelligent recommender system for stock trading. *Intelligent Decision Technologies*, *9*(3), 243-269.

Nair, B. B., Mohandas, V. P., Nayanar, N., Teja, E. S. R., Vigneshwari, S., & Teja, K. V. N. S. (2015). A stock trading recommender system based on temporal association rule mining. *SAGE Open*, *5*(2), 2158244015579941.

Paranjape-Voditel, P., & Deshpande, U. (2013). A stock market portfolio recommender system based on association rule mining. *Applied Soft Computing*, *13*(2), 1055-1063.

Pazzani, M. J. and Billsus, D. (2007). Content-based recommendation systems", In *The adaptive web*, 325-341. Springer, Berlin, Heidelberg.

Pereira, N., & Varma, S. (2016). Survey on content based recommendation system. *Int. J. Comput. Sci. Inf. Technol*, *7*(1), 281-284.

Poriya, A., Bhagat, T., Patel, N., & Sharma, R. (2014). Non-personalized recommender systems and user-based collaborative recommender systems. *Int. J. Appl. Inf. Syst*, *6*(9), 22-27.

Rao, K. N. (2008). Application domain and functional classification of recommender systems--a survey. *DESIDOC Journal of Library & Information Technology*, *28*(3), 17-35.

Ricci, F. Rokach, L. and Shapira, B. (2015). Recommender systems: introduction and challenges. In *Recommender systems handbook,* 1-34. Springer, Boston, MA.

Sadeghian, A., & Tahayori, H. (Eds.). (2015). *Frontiers of Higher Order Fuzzy Sets*. Springer New York.

Selvamuthu, D., Kumar, V., & Mishra, A. (2019). Indian stock market prediction using artificial neural networks on tick data. *Financial Innovation*, *5*(1), 16(2019).

Sharma, S., Sharma, A., Sharma, Y., & Bhatia, M. (2016, April). Recommender system using hybrid approach. In *2016 International Conference on Computing, Communication and Automation (ICCCA)* (pp. 219-223). IEEE.

Sun, L., Michael, E. I., Wang, S., & Li, Y. (2016, December). A Time-Sensitive Collaborative Filtering Model in Recommendation Systems. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* (pp. 340-344). IEEE.

Zainal-Mokhtar, K., & Mohamad-Saleh, J. (2012). A Generic Intelligent Oil-Gas Flow Classifier Based On ECT Sensor Data. *International Journal of Innovative Computing, Information and Control ICIC*, *8*, 953-965.

Zhang, H. R., Min, F., He, X., & Xu, Y. Y. (2015). A hybrid recommender system based on user-recommender interaction. *Mathematical Problems in Engineering*.

Takhteyev, Y. and Hilts, A. (2010). Investigating the geography of open source software through GitHub. *Manuscript submitted for publication,* 2010.

Tarus, J. K., Niu, Z., & Mustafa, G. (2018). Knowledge-based recommendation: a review of ontology-based recommender systems for e-learning. *Artificial intelligence review*, *50*(1), 21-48.

Ting, J., Fu, T. C., & Chung, F. L. (2006). Mining of stock data: intra-and inter-stock pattern associative classification. *Threshold*, *5*(100), 5-99.

Uchyigit, G. and Ma, M. Y. (Eds.) (2008). *Personalization techniques and recommender systems* (Vol. 70). World Scientific.

Wei, Y. Z., Moreau, L., & Jennings, N. R. (2005). Learning users' interests by quality classification in market-based recommender systems. *IEEE Transactions on Knowledge and Data Engineering*, *17*(12), 1678-1688.

Verma, L. (2018). Design and Development of Decision Support Based Hybrid Recommender System, PhD thesis Department of Physics and

Computer Science Faculty of Science Dayalbagh Educational Institute Dayalbagh, Agra.

Verstrepen, K. and Goethals, B. (2014): Unifying nearest neighbors collaborative filtering, In *Proceedings of the 8th ACM Conference on Recommender Systems*, 177-184.

Williams, R. T. (2010). *An introduction to trading in the financial markets: Global markets, risk, compliance, and regulation.* Academic Press.

Vu, H. Q. (2014). *Preference mining techniques for customer behavior analysis.* PhD thesis dissertation at Deakin University.

Özkan, S., Bindusara, G., & Hackney, R. (2010). Facilitating the adoption of e-payment systems: theoretical constructs and empirical analysis. *Journal of enterprise information management,* 23(3), 305-325.

Yager, R. R. (2003). Fuzzy logic methods in recommender systems, *Fuzzy Sets and Systems, 136*(2), 133-149.

Yujun, Y., Jianping, L., & Yimei, Y. (2016). An efficient stock recommendation model based on big order net inflow. *Mathematical Problems in Engineering,* 1-15(*2016*). https://doi.org/10.1155/2016/5725