**DISCOVERY AND PREDICTION OF STOCK INDEX PATTERN USING TPA-LSTM**

**ABSTRACT**

Stock index prediction is one of the most important subjects in financial time series forecasting. Investors can invest passively through a stock index or compare active investment performance with stock index. Therefore, developing a more realistic model to predict stock index is of great importance for investors and professional analysts. However, stock index characteristics, including “noisy” and “non-stationary”, make prediction face challenges. “Noisy” implies that there is insufficient information for investors to observe past behaviors of stock index. ‘Nonstationary’ means that stock index may change dramatically in different periods. These characteristics lead to poor stock index prediction results as predicted by traditional econometric models such as linear model, Auto-Regressive Integrated Moving Average (ARIMA), and Vector Auto Regression (VAR). The aforementioned methods belong to short-term predictions in time series, which are seriously affected by “noisy” and “non-stationary”. However, if stock index prediction only focuses on forecasting the trend over a certain period, the effects of “noisy” and “non-stationary” on the prediction results will be eliminated. One of methods of stock index trend prediction over a certain period is to decompose stock index over long-period into many stock index fragments over short-period, and then stock index fragments over short-period are classified into some pattern in pattern sets including “W” shape, “head and shoulders”, etc. The core of the process above is to find some inherent patterns in the existing stock index sequences and make appropriate prediction, which is referred to as “pattern discovery” and “pattern prediction”. According to the repeated patterns which were discovered in previous work, we can predict repeated patterns of stock index over a certain period in the future, and then make appropriate actions to take gains and avoid losses more effectively. Therefore, this paper focuses on discovering and forecasting stock index patterns.

In this project, we attempt to discover and predict stock index patterns through analysis of multivariate time series. Our motivation is based on the notion that financial planning guided by pattern discovery and prediction of stock index prices maybe more realistic and effective than traditional approaches, such as Autoregressive Integrated Moving Average (ARIMA) model. A three-stage architecture constructed by combining Temporal Pattern Attention and Long- Short-Term Memory (TPA-LSTM) is applied for pattern discovery and prediction of stock index.

**CHAPTER 1**

**INTRODUCTION**

Stock index prediction is one of the most important subjects in financial time series forecasting. Investors can invest passively through a stock index or compare active investment performance with stock index. Therefore, developing a more realistic model to predict stock index is of great importance for investors and professional analysts. However, stock index characteristics, including ‘‘noisy’’ and ‘‘non-stationary’’, make prediction face challenges. ‘‘Noisy’’ implies that there is insufficient information for investors to observe past behaviors of stock index. ‘Nonstationary’ means that stock index may change dramatically in different periods. These characteristics lead to poor stock index prediction results as predicted by traditional econometric models such as linear model, Auto-Regressive Integrated Moving Average (ARIMA), and Vector Auto Regression (VAR) [1]. The aforementioned methods belong to short-term predictions in time series, which are seriously affected by ‘‘noisy’’ and ‘‘non-stationary’’. However, if stock index prediction only focuses on forecasting the trend over a certain period, the effects of ‘‘noisy’’ and ‘‘non-stationary’’ on the prediction results will be eliminated. One of methods of stock index trend prediction over a certain period is to decompose stock index over long-period into many stock index fragments over short-period, and then stock index fragments over short period are classified into some pattern in pattern sets including ‘‘W’’ shape, ‘‘head and shoulders’’, etc. The core of the process above is to find some inherent patterns in the existing stock index sequences and make appropriate prediction, which is referred to as ‘‘pattern discovery’’ and ‘‘pattern prediction’’. According to the repeated patterns which were discovered in previous work, we can predict repeated patterns of stock index over a certain period in the future, and then make appropriate actions to take gains and avoid losses more effectively. Therefore, this paper focuses on discovering and forecasting stock index patterns.

In this study, we attempt to discover and predict stock index pattern through a three-stage architecture that consists of Toeplitz Inverse Covariance-Based Clustering (TICC), Temporal Pattern Attention and Long- Short Term Memory (TPA-LSTM) and Multivariate LSTM-FCNs (MLSTM-FCN, MALSTM-FCN), which are developed by Hallac et al. [2], Shih et al. [3] and Karim et al. [4], respectively. Taking Hangseng Composite Stock Index (HSCI) and 11 industry stock indices in HSCI as an example, this paper investigates the feasibility of proposed three-stage architecture in financial time series. In the first stage, this paper applies TICC algorithm to cluster prices of industrial indices in HSCI, including consumer good manufacturing, consumer service, energy, finance, industry, information technology, integrated industry, raw material, real estate, utilities. Then, this paper maps the clustering results of industry indices to HSCI and discover repeated patterns of HSCI. In the second stage, TPA-LSTM is used to predict industry indices. In the third stage, this paper applies Multivariate LSTM-FCNs to classify industry indices and predict the pattern of HSCI in the future. Based on the idea that industry indicators are predominant factors in explaining stock market co-movements [5], the proposed three-stage architecture with TICC, TPA-LSTM and Multivariate LSTM-FCNs might be more effective in discovery and prediction of stock index patterns. Moreover, we could conduct early warnings of stock index and make corresponding measures more efficiently.

**CHAPTER 2**

**LITERATURE SURVEY**

Ouyang et al. [6] attempted to discover and predict stock index patterns through analysis of multivariate time series. This motivation is based on the notion that financial planning guided by pattern discovery and prediction of stock index prices maybe more realistic and effective than traditional approaches, such as Autoregressive Integrated Moving Average (ARIMA) model. A three-stage architecture constructed by combining Toeplitz Inverse Covariance-Based Clustering (TICC), Temporal Pattern Attention and Long-Short-Term Memory (TPA-LSTM) and Multivariate LSTM-FCNs (MLSTM-FCN and MALSTM-FCN) is applied for pattern discovery and prediction of stock index. In the first stage, this paper used TICC to discover repeated patterns of stock index. Then, in the second stage, TPA-LSTM that considered weak periodic patterns and long short-term information is used to predict multivariate stock indices. Finally, in the third stage, MALSTM-FCN is applied to predict stock index price pattern. The Hangseng Stock Index and eleven industrial sub-indices are used in the experiment.

Wei et al. [7] applied a different approach, Temporal Pattern Attention and Long Short-Term Memory (TPA-LSTM), to predict stock indexes’ prices in different industries included in the Hangseng Composite Index. TPA-LSTM method is a new prediction model that enables the prediction of multivariate time series simultaneously with a top concern of weak periodic patterns and a mixture of linear and nonlinear structures. Further, the TPA-LSTM method comprises four components, the Temporal Pattern Attention component, the RNN component, and the Autoregressive component. The experiment results indicate that by combining the strengths of convolutional network, recurrent network, temporal attention component, and autoregressive component, the TPA-LSTM method significantly improves state-of-the-art results in multivariate time series forecasting on the dataset of industry stock index prices. With the empirical results, this paper shows that the applied TPA-LSTM method is a satisfactory alternative for multivariate time series forecasting in stock indices.

Sayavong et al. [8] proposed a stock price prediction model based on convolution neural network, which has obvious self-adaptability and self-learning ability. Combining the characteristics of CNN (Convolution Neural Network) and Thai stock market, the data set is trained and tested after pretreatment. On this basis, three stocks (BBL, CAPLL&PTT) listed on the Thai Stock Exchange are tested and compared with the actual stock price. The results showed that the model based on CNN can effectively identify the changing trend of stock price and predict it which can provide valuable reference for stock price forecast. The prediction accuracy is high, and it is worth further promotion in the financial field.

Wen et al. [9] introduced a new method to simplify noisy-filled financial temporal series via sequence reconstruction by leveraging motifs (frequent patterns), and then utilized a convolutional neural network to capture spatial structure of time series. The experimental results showed the efficiency of this proposed method in feature learning and outperformance with 4%–7% accuracy improvement compared with the traditional signal process methods and frequency trading patterns modeling approach with deep learning in stock trend prediction.

Thakkar et al. [10] conducted a systematic approach to present a survey for the years 2011–2020 by considering articles that have used fusion techniques for various stock market applications and broadly categorize them into information fusion, feature fusion, and model fusion. The major applications of stock market include stock price and trend prediction, risk analysis and return forecasting, index prediction, as well as portfolio management. This work also provided an infographic overview of fusion in stock market prediction and extend this survey for other finely addressed financial prediction problems. Based on the surveyed articles, this paper provided potential future directions and concluding remarks on the significance of applying fusion in stock market.

Dattatray et al. [11] presented the detailed review of 50 research papers suggesting the methodologies, like Bayesian model, Fuzzy classifier, Artificial Neural Networks (ANN), Support Vector Machine (SVM) classifier, Neural Network (NN), Machine Learning Methods and so on, based on stock market prediction. The obtained papers are classified based on different prediction and clustering techniques. The research gaps and the challenges faced by the existing techniques are listed and elaborated, which help the researchers to upgrade the future works. The works are analyzed using certain datasets, software tools, performance evaluation measures, prediction techniques utilized, and performance attained by different techniques. The commonly used technique for attaining effective stock market prediction is ANN and the fuzzy-based technique. Even though a lot of research efforts, the current stock market prediction technique still has many limits. From this survey, it can be concluded that the stock market prediction is a very complex task, and different factors should be considered for predicting the future of the market more accurately and efficiently.

Chung et al. [12] focused on the optimization of feature extraction part of CNN, because this is the most important part of the computational procedure of CNN. This study proposed a method to systematically optimize the parameters for the CNN model by using genetic algorithm (GA). To verify the effectiveness of this model, this work compared the prediction result with standard artificial neural networks (ANNs) and CNN models. The experimental results showed that the GA-CNN outperforms the comparative models and demonstrated the effectiveness of the hybrid approach of GA and CNN.

Zhong et al. [13] presented a comprehensive big data analytics process to predict the daily return direction of the SPDR S&P 500 ETF (ticker symbol: SPY) based on 60 financial and economic features. DNNs and traditional artificial neural networks (ANNs) are then deployed over the entire pre-processed but untransformed dataset, along with two datasets transformed via principal component analysis (PCA), to predict the daily direction of future stock market index returns. While controlling for overfitting, a pattern for the classification accuracy of the DNNs is detected and demonstrated as the number of the hidden layers increases gradually from 12 to 1000. Moreover, a set of hypothesis testing procedures are implemented on the classification, and the simulation results showed that the DNNs using two PCA-represented datasets give significantly higher classification accuracy than those using the entire untransformed dataset, as well as several other hybrid machine learning algorithms. In addition, the trading strategies guided by the DNN classification process based on PCA-represented data perform slightly better than the others tested, including in a comparison against two standard benchmarks.

Jiang et al. [14] pertained to Random Forest (RF), extremely randomized trees (ERT), extreme gradient boosting (XGBoost) and light gradient boosting machine (LightGBM), and recurrent neural networks (RNN), bidirectional RNN, RNN with long short-term memory (LSTM) and gated recurrent unit (GRU) layer, which pertain to the deep learning algorithms, are stacked as base classifiers in the first layer. Cross-validation method is then implemented to iteratively generate the input for the second level classifier to prevent overfitting. In the second layer, logistic regression, as well as its regularized version, are employed as meta-classifiers to identify the unique learning pattern of the base classifiers. Empirical results over three major U.S. stock indices indicate that our improved Stacking method outperforms state-of-the-art ensemble learning algorithms and deep learning models, achieving a higher level of accuracy, F-score and AUC value. Besides, another contribution in this research paper is the design of a Lasso (least absolute shrinkage and selection operator) based meta-classifier that is capable of automatically weighting/selecting the optimal base learners for the forecasting task. These findings provide an integrated Stacking framework in the financial area.

Kumar et al. [15] focused is on application of computational intelligent approaches such as artificial neural network, fuzzy logic, genetic algorithms, and other evolutionary techniques for stock market forecasting. This paper presented an up-to-date survey of existing literature on stock market forecasting based on computational intelligent methods. In this article, the selected papers are organized and discussed according to six main point of view: (1) the stock market analyzed and the related dataset, (2) the type of input variables investigated, (3) the pre-processing techniques used, (4) the feature selection techniques to choose effective variables, (5) the forecasting models to deal with the stock price forecasting problem and (6) performance metrics utilized to evaluate the models. The major contribution of this work is to provide the researcher and financial analyst a systematic approach for development of intelligent methodology to forecast stock market. This paper also presented the outlines of proposed work with the aim to enhance the performance of existing techniques.

**CHAPTER 3**

**EXISTING SYSTEM**

**3.1 Random Forest Algorithm**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Diagram

Description automatically generated

Fig. 3.1: Random Forest algorithm.

**Random Forest algorithm**

**Step 1:** In Random Forest n number of random records are taken from the data set having k number of records.

**Step 2:** Individual decision trees are constructed for each sample.

**Step 3:** Each decision tree will generate an output.

**Step 4:** Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

**Important Features of Random Forest**

* **Diversity**- Not all attributes/variables/features are considered while making an individual tree, each tree is different.
* **Immune** **to** **the** **curse** **of** **dimensionality**- Since each tree does not consider all the features, the feature space is reduced.
* **Parallelization**-Each tree is created independently out of different data and attributes. This means that we can make full use of the CPU to build random forests.
* **Train-Test** **split**- In a random forest we don’t have to segregate the data for train and test as there will always be 30% of the data which is not seen by the decision tree.
* **Stability**- Stability arises because the result is based on majority voting/ averaging.

**Assumptions for Random Forest**

Since the random forest combines multiple trees to predict the class of the dataset, it is possible that some decision trees may predict the correct output, while others may not. But together, all the trees predict the correct output. Therefore, below are two assumptions for a better Random Forest classifier:

* There should be some actual values in the feature variable of the dataset so that the classifier can predict accurate results rather than a guessed result.
* The predictions from each tree must have very low correlations.

Below are some points that explain why we should use the Random Forest algorithm

* It takes less training time as compared to other algorithms.
* It predicts output with high accuracy, even for the large dataset it runs efficiently.
* It can also maintain accuracy when a large proportion of data is missing.

**Types of Ensembles**

Before understanding the working of the random forest, we must look into the ensemble technique. Ensemble simply means combining multiple models. Thus, a collection of models is used to make predictions rather than an individual model. Ensemble uses two types of methods:

**Bagging**– It creates a different training subset from sample training data with replacement & the final output is based on majority voting. For example, Random Forest. Bagging, also known as Bootstrap Aggregation, is the ensemble technique used by random forest. Bagging chooses a random sample from the data set. Hence each model is generated from the samples (Bootstrap Samples) provided by the Original Data with replacement known as row sampling. This step of row sampling with replacement is called bootstrap. Now each model is trained independently which generates results. The final output is based on majority voting after combining the results of all models. This step which involves combining all the results and generating output based on majority voting is known as aggregation.

**Diagram

Description automatically generated**

Fig. 3.2: RF Classifier analysis.

**Boosting**– It combines weak learners into strong learners by creating sequential models such that the final model has the highest accuracy. For example, ADA BOOST, XG BOOST.

Diagram

Description automatically generated

Fig. 3.3: Boosting RF classifier.

**Advantages of Random Forest**

* It can be used in classification and regression problems.
* It solves the problem of overfitting as output is based on majority voting or averaging.
* It performs well even if the data contains null/missing values.
* Each decision tree created is independent of the other thus it shows the property of parallelization.
* It is highly stable as the average answers given by a large number of trees are taken.
* It maintains diversity as all the attributes are not considered while making each decision tree though it is not true in all cases.
* It is immune to the curse of dimensionality. Since each tree does not consider all the attributes, feature space is reduced.

**Applications of Random Forest**

There are mainly four sectors where Random Forest mostly used:

* Banking: Banking sector mostly uses this algorithm for the identification of loan risk.
* Medicine: With the help of this algorithm, disease trends and risks of the disease scan be identified.
* Land Use: We can identify the areas of similar land use by this algorithm.
* Marketing: Marketing trends can be identified using this algorithm.

**3.2 Naïve Bayes**

***What is the Naive Bayes algorithm?***

Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

Naive Bayes classifier is a collection of many algorithms where all the algorithms share one common principle, and that is each feature being classified is not related to any other feature. The presence or absence of a feature does not affect the presence or absence of the other feature.

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. ... Naïve Bayes Classifier is one of the simple and most effective Classification algorithms which helps in building the fast machine learning models that can make quick predictions.

***How Naive Bayes works?***

Naive Bayes is a powerful algorithm that is used for text data analysis and with problems with multiple classes. To understand Naive Bayes theorem’s working, it is important to understand the Bayes theorem concept first as it is based on the latter.

Bayes theorem, formulated by Thomas Bayes, calculates the probability of an event occurring based on the prior knowledge of conditions related to an event. It is based on the following formula:

P(A|B) = P(A) \* P(B|A)/P(B)

Where we are calculating the probability of class A when predictor B is already provided.

P(B) = prior probability of B

P(A) = prior probability of class A

P(B|A) = occurrence of predictor B given class A probability

**Drawbacks of Naïve Bayes**

The Naive Bayes algorithm has the following disadvantages:

* The prediction accuracy of this algorithm is lower than the other probability algorithms.
* It is not suitable for regression. Naive Bayes algorithm is only used for textual data classification and cannot be used to predict numeric values.

**3.3 Support Vector Machine Algorithm**

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane:



Fig. 3.4: Analysis of SVM.

**Example:** SVM can be understood with the example that we have used in the KNN classifier. Suppose we see a strange cat that also has some features of dogs, so if we want a model that can accurately identify whether it is a cat or dog, so such a model can be created by using the SVM algorithm. We will first train our model with lots of images of cats and dogs so that it can learn about different features of cats and dogs, and then we test it with this strange creature. So as support vector creates a decision boundary between these two data (cat and dog) and choose extreme cases (support vectors), it will see the extreme case of cat and dog. On the basis of the support vectors, it will classify it as a cat. Consider the below diagram:



Fig. 3.5: Basic classification using SVM.

**Types of SVM:** SVM can be of two types:

**Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.

**Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier

**SVM working**

**Linear SVM:** The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue. Consider the below image:



Fig. 3.6: Linear SVM.

So as it is 2-d space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. Consider the below image:



Fig. 3.7: Test-Vector in SVM.

Hence, the SVM algorithm helps to find the best line or decision boundary; this best boundary or region is called as a hyperplane. SVM algorithm finds the closest point of the lines from both the classes. These points are called support vectors. The distance between the vectors and the hyperplane is called as margin. And the goal of SVM is to maximize this margin. The hyperplane with maximum margin is called the optimal hyperplane.



Fig. 3.8: Classification in SVM.

**Non-Linear SVM:** If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. Consider the below image:



Fig. 3.9: Non-Linear SVM.

So, to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third-dimension z. It can be calculated as:

z=x2 +y2

By adding the third dimension, the sample space will become as below image:

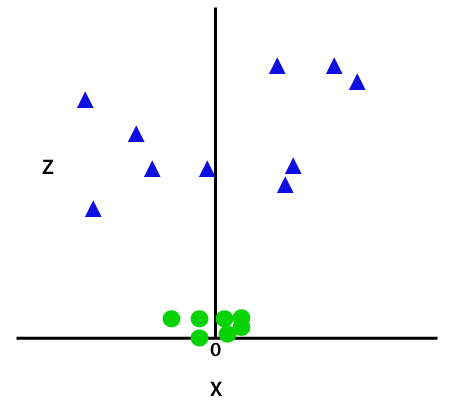


Fig. 3.10: Non-Linear SVM data seperation.

So now, SVM will divide the datasets into classes in the following way. Consider the below image:



Fig. 3.11: Non-Linear SVM best hyperplane.

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=1, then it will become as:

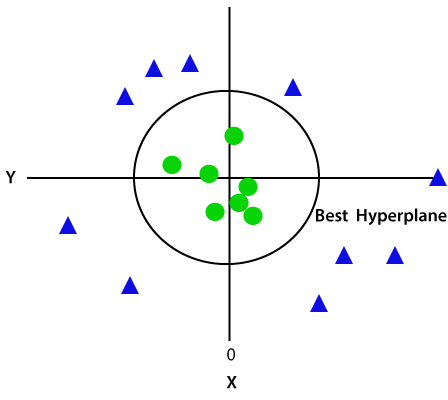


Fig. 3.12: Non-Linear SVM with ROC.

Hence, we get a circumference of radius 1 in case of non-linear data.

**Disadvantages of support vector machine:**

* Support vector machine algorithm is not acceptable for large data sets.
* It does not execute very well when the data set has more sound i.e. target classes are overlapping.
* In cases where the number of properties for each data point outstrips the number of training data specimens, the support vector machine will underperform.
* As the support vector classifier works by placing data points, above and below the classifying hyperplane there is no probabilistic clarification for the classification.

**CHAPTER 4**

**PROPOSED SYSTEM**

The block diagram of the proposed system is shown in Fig. 4.1

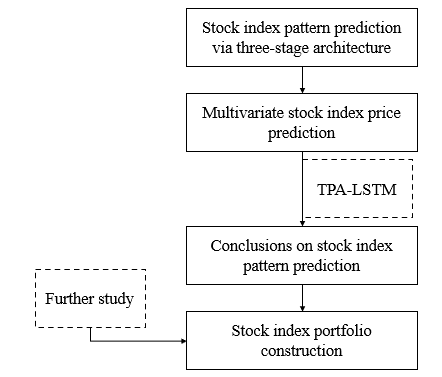
****

Fig. 4.1: Block diagram of proposed system.

**4.1 Dataset description**

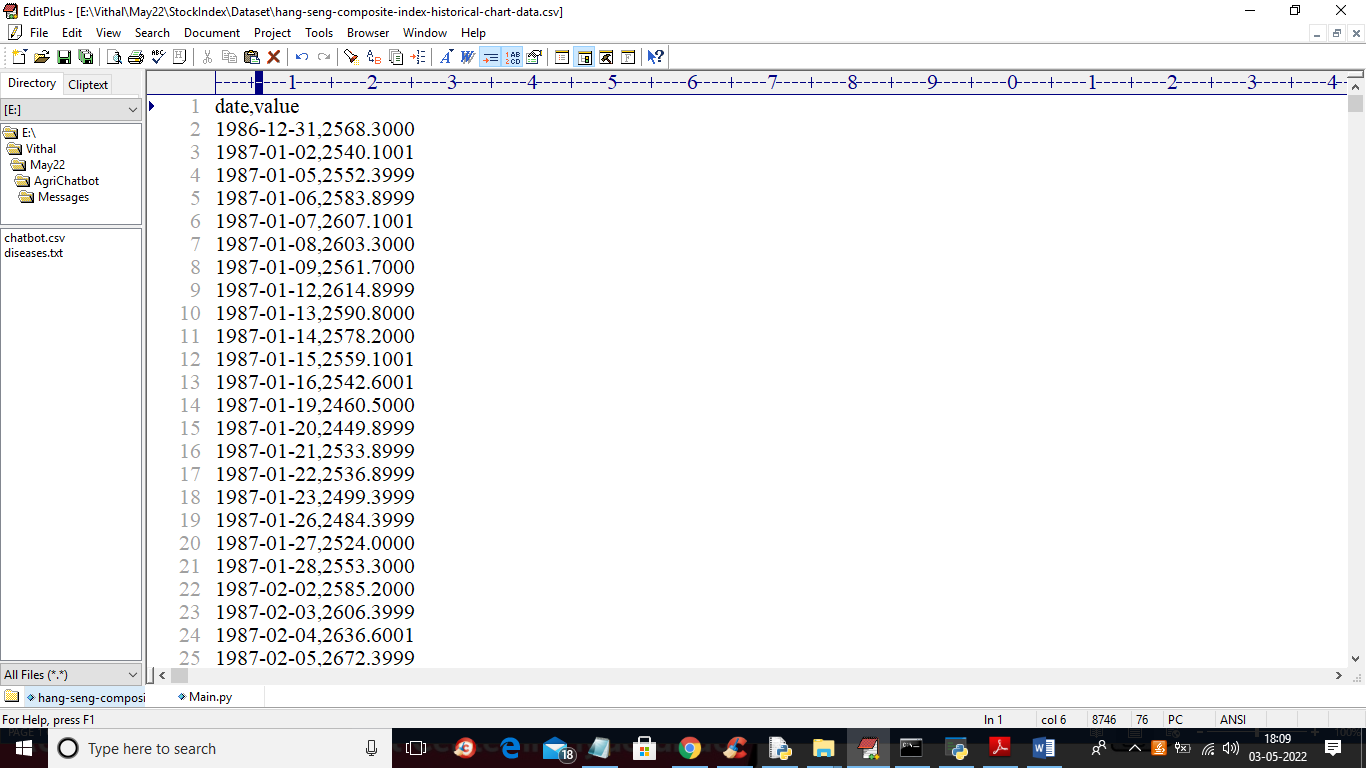
Discovery and Prediction of Stock Index Pattern via Three-Stage Architecture of TICC, TPA-LSTM and Multivariate LSTM-FCNs

In this paper to predict stock index author is using multivariate time series (data which contains time information) data and then extracting repeated values from that data by using Toeplitz Inverse Covariance-Based Clustering (TICC) as this algorithm put similar data in same cluster. Patterns will be extracted by using Temporal Pattern Attention and Long-Short-Term Memory (TPA-LSTM) and then extracted pattern will get trained with Multivariate LSTM-FCNs (fully connected network) to predict stock index.

All existing algorithms such as ARIMA work on time series data but not extract any patterns so its prediction accuracy will be low and Relative Absolute Error will be high.

Author compares propose algorithm TPA-LSTM with various existing algorithms such as SVM, Random forest and Naïve Bayes and evaluate their performance in terms of accuracy and RAE.

To implement this project author has used Hang-Sang dataset and I am also using same dataset and below screen showing dataset details.



In above dataset we have date and stock value and by using this time series and stock data we will train algorithms and then predict stock index and then find difference between original stock index and predicted index as RAE error.

**4.2 Pre-processing**

Data pre-processing is a process of preparing the raw data and making it suitable for a machine learning model. It is the first and crucial step while creating a machine learning model.

When creating a project, it is not always a case that we come across the clean and formatted data. And while doing any operation with data, it is mandatory to clean it and put in a formatted way. So, for this, we use data pre-processing task.

***Why do we need Data Pre-processing?***

A real-world data generally contains noises, missing values, and maybe in an unusable format which cannot be directly used for machine learning models. Data pre-processing is required tasks for cleaning the data and making it suitable for a machine learning model which also increases the accuracy and efficiency of a machine learning model.

* Getting the dataset
* Importing libraries
* Importing datasets
* Finding Missing Data
* Encoding Categorical Data
* Splitting dataset into training and test set
* Feature scaling

**4.2.1 Splitting the Dataset into the Training set and Test set**

In machine learning data pre-processing, we divide our dataset into a training set and test set. This is one of the crucial steps of data pre-processing as by doing this, we can enhance the performance of our machine learning model.

Supposeif we have given training to our machine learning model by a dataset and we test it by a completely different dataset. Then, it will create difficulties for our model to understand the correlations between the models.

If we train our model very well and its training accuracy is also very high, but we provide a new dataset to it, then it will decrease the performance. So we always try to make a machine learning model which performs well with the training set and also with the test dataset. Here, we can define these datasets as:

A picture containing shape

Description automatically generated

Fig. 4.2: Splitting the dataset.

**Training** **Set**: A subset of dataset to train the machine learning model, and we already know the output.

**Test** **set**: A subset of dataset to test the machine learning model, and by using the test set, model predicts the output.

**4.3 Temporal Pattern Attention and Long Short-Term Memory**

Compared to other forecasting methods, TPA-LSTM is the first method to predict n dimensional time series with a mixture of weak long- and short-term patterns. Based on TPA-LSTM method, we could fully consider the mixed structure of weak long- and short-term repeated patterns inherent in financial time series and predict multivariate stock indices more accurately. In this section, we describe the details of TPA-LSTM algorithm applied in this paper.

TPA-LSTM consists of a non-linear part and a linear part. The non-linear part is a temporal attention mechanism which includes a recurrent layer, convolutional layer, and a temporal pattern attention layer, while the linear part uses an autoregressive model (AR) to forecast the result.

**1) Recurrent Layer**

The first layer of TPA-LSTM is a long short-term memory network (LSTM). Given the input matrix wherein 𝑥, this recurrent layer aims at capturing long-term information. The outputs of recurrent layer are the hidden states at each time stamp. The hidden sates of recurrent layer’s units at time can be formulated as

Which is defined by the following equations,

Where ,  , , ∈ , , , , ∈ , ⨀ is the element-wise product, and 𝜎 is the sigmoid function.

**2) Convolutional Layer**

Given previous LSTM hidden states 𝐻 = { and initial input matrix , this section extracts short-term signal patterns and interdependencies among eleven variables. The output in this section can be expressed as

Where represents the convolutional value of the -th row vector and the -th filter, denotes the filters we have, is the maximum length this paper is paying attention which is set to be 30.

**3) Temporal Pattern Attention Layer**

Traditional attention mechanism selects relevant information relative to current time step, which may lead to a failure to ignore noisy variables and detect temporal useful patterns in multivariate time series forecasting. To alleviate this problem, TPA-LSTM develops a new temporal pattern attention mechanism which could select useful variables and capture temporal information for forecasting. Given previous convolutional value , recurrent value 𝐻, and initial input matrix 𝑋, the output of this temporal pattern attention layer is a non-linear projection part, which is computed as

where 𝑣𝑡 = 𝐻𝑖 𝐶𝛼𝑡 is the weighted context of hidden states of the convolutional matrix, 𝛼𝑡 is the attention weights which can be expressed as

**4) Autoregressive Layer**

Due to non-linear property of the proposed attention mechanism, TPA-LSTM method decomposes the prediction into a non-linear part and a linear part. The prediction of the non-linear part is captured by a recurrent layer, a convolutional layer, and a temporal pattern attention layer. In contrast, the prediction of the linear part is solved by the Autoregressive (AR) model in this section. Given the initial input *X*, we can get the forecasting result of the non-linear part through AR Layer, which is formulated as follows,

Then the forecasting result of TPA-LSTM can be expressed as follows,

**CHAPTER 5**

**UML DIAGRAMS**

Class Diagram: Class diagram is a static diagram. It represents the static view of an application.

Note:

TPA-LSTM: Temporal Pattern Attention and Long-Short-Term Memory

RAE: Relative Absolute Error

A screenshot of a computer

Description automatically generated with low confidence

Sequence Diagram: Sequence diagram is an interaction diagram that details how operations are carried out.

A picture containing text, diagram, parallel, line

Description automatically generated

Activity diagram: Activity diagram is another important diagram in UML to describe the dynamic aspects of the system.

A picture containing text, screenshot, number, font

Description automatically generated

Deployment diagram: The deployment diagram visualizes the physical hardware on which the software will be deployed.

A screenshot of a computer

Description automatically generated with low confidence

Use case diagram: The purpose of use case diagram is to capture the dynamic aspect of a system.

A picture containing text, diagram, line, font

Description automatically generated

Component diagram: Component diagram describes the organization and wiring of the physical components in a system.

A picture containing text, diagram, line, parallel

Description automatically generated

**CHAPTER 6**

**MACHINE LEARNING**

**What is Machine Learning**

Before we look at the details of various machine learning methods, let's start by looking at what machine learning is, and what it isn't. Machine learning is often categorized as a subfield of artificial intelligence, but I find that categorization can often be misleading at first brush. The study of machine learning certainly arose from research in this context, but in the data science application of machine learning methods, it's more helpful to think of machine learning as a means of building models of data.

Fundamentally, machine learning involves building mathematical models to help understand data. "Learning" enters the fray when we give these models tunable parameters that can be adapted to observed data; in this way the program can be "learning" from the data. Once these models have been fit to previously seen data, they can be used to predict and understand aspects of newly observed data. I'll leave to the reader the more philosophical digression regarding the extent to which this type of mathematical, model-based "learning" is like the "learning" exhibited by the human brain. Understanding the problem setting in machine learning is essential to using these tools effectively, and so we will start with some broad categorizations of the types of approaches we'll discuss here.

**Categories of Machine Leaning**

At the most fundamental level, machine learning can be categorized into two main types: supervised learning and unsupervised learning.

Supervised learning involves somehow modeling the relationship between measured features of data and some label associated with the data; once this model is determined, it can be used to apply labels to new, unknown data. This is further subdivided into classification tasks and regression tasks: in classification, the labels are discrete categories, while in regression, the labels are continuous quantities. We will see examples of both types of supervised learning in the following section.

Unsupervised learning involves modeling the features of a dataset without reference to any label and is often described as "letting the dataset speak for itself." These models include tasks such as clustering and dimensionality reduction. Clustering algorithms identify distinct groups of data, while dimensionality reduction algorithms search for more succinct representations of the data. We will see examples of both types of unsupervised learning in the following section.

**Need for Machine Learning**

Human beings, at this moment, are the most intelligent and advanced species on earth because they can think, evaluate, and solve complex problems. On the other side, AI is still in its initial stage and have not surpassed human intelligence in many aspects. Then the question is that what is the need to make machine learn? The most suitable reason for doing this is, “to make decisions, based on data, with efficiency and scale”.

Lately, organizations are investing heavily in newer technologies like Artificial Intelligence, Machine Learning and Deep Learning to get the key information from data to perform several real-world tasks and solve problems. We can call it data-driven decisions taken by machines, particularly to automate the process. These data-driven decisions can be used, instead of using programming logic, in the problems that cannot be programmed inherently. The fact is that we can’t do without human intelligence, but other aspect is that we all need to solve real-world problems with efficiency at a huge scale. That is why the need for machine learning arises.

**Challenges in Machines Learning**

While Machine Learning is rapidly evolving, making significant strides with cybersecurity and autonomous cars, this segment of AI as whole still has a long way to go. The reason behind is that ML has not been able to overcome number of challenges. The challenges that ML is facing currently are −

1. Quality of data − Having good-quality data for ML algorithms is one of the biggest challenges. Use of low-quality data leads to the problems related to data preprocessing and feature extraction.
2. Time-Consuming task − Another challenge faced by ML models is the consumption of time especially for data acquisition, feature extraction and retrieval.
3. Lack of specialist persons − As ML technology is still in its infancy stage, availability of expert resources is a tough job.
4. No clear objective for formulating business problems − Having no clear objective and well-defined goal for business problems is another key challenge for ML because this technology is not that mature yet.
5. Issue of overfitting & underfitting − If the model is overfitting or underfitting, it cannot be represented well for the problem.
6. Curse of dimensionality − Another challenge ML model faces is too many features of data points. This can be a real hindrance.
7. Difficulty in deployment − Complexity of the ML model makes it quite difficult to be deployed in real life.

**Applications of Machines Learning**

Machine Learning is the most rapidly growing technology and according to researchers we are in the golden year of AI and ML. It is used to solve many real-world complex problems which cannot be solved with traditional approach. Following are some real-world applications of ML.

* Emotion analysis
* Sentiment analysis
* Error detection and prevention
* Weather forecasting and prediction
* Stock market analysis and forecasting
* Speech synthesis
* Speech recognition
* Customer segmentation
* Object recognition
* Fraud detection
* Fraud prevention
* Recommendation of products to customer in online shopping

**How to Start Learning Machine Learning?**

Arthur Samuel coined the term “Machine Learning” in 1959 and defined it as a “Field of study that gives computers the capability to learn without being explicitly programmed”.

And that was the beginning of Machine Learning! In modern times, Machine Learning is one of the most popular (if not the most!) career choices. According to Indeed, Machine Learning Engineer Is the Best Job of 2019 with a 344% growth and an average base salary of $146,085 per year.

But there is still a lot of doubt about what exactly is Machine Learning and how to start learning it? So, this article deals with the Basics of Machine Learning and the path you can follow to eventually become a full-fledged Machine Learning Engineer. Now let’s get started!!!

**How to start learning ML?**

This is a rough roadmap you can follow on your way to becoming an insanely talented Machine Learning Engineer. Of course, you can always modify the steps according to your needs to reach your desired end-goal!

**Step 1 – Understand the Prerequisites**

In case you are a genius, you could start ML directly but normally, there are some prerequisites that you need to know which include Linear Algebra, Multivariate Calculus, Statistics, and Python. And if you don’t know these, never fear! You don’t need a Ph.D. degree in these topics to get started but you do need a basic understanding.

**(a) Learn Linear Algebra and Multivariate Calculus**

Both Linear Algebra and Multivariate Calculus are important in Machine Learning. However, the extent to which you need them depends on your role as a data scientist. If you are more focused on application heavy machine learning, then you will not be that heavily focused on maths as there are many common libraries available. But if you want to focus on R&D in Machine Learning, then mastery of Linear Algebra and Multivariate Calculus is very important as you will have to implement many ML algorithms from scratch.

**(b) Learn Statistics**

Data plays a huge role in Machine Learning. In fact, around 80% of your time as an ML expert will be spent collecting and cleaning data. And statistics is a field that handles the collection, analysis, and presentation of data. So, it is no surprise that you need to learn it!!!  
Some of the key concepts in statistics that are important are Statistical Significance, Probability Distributions, Hypothesis Testing, Regression, etc. Also, Bayesian Thinking is also a very important part of ML which deals with various concepts like Conditional Probability, Priors, and Posteriors, Maximum Likelihood, etc.

**(c) Learn Python**

Some people prefer to skip Linear Algebra, Multivariate Calculus and Statistics and learn them as they go along with trial and error. But the one thing that you absolutely cannot skip is Python! While there are other languages you can use for Machine Learning like R, Scala, etc. Python is currently the most popular language for ML. In fact, there are many Python libraries that are specifically useful for Artificial Intelligence and Machine Learning such as Keras, TensorFlow, Scikit-learn, etc.

So, if you want to learn ML, it’s best if you learn Python! You can do that using various online resources and courses such as Fork Python available Free on GeeksforGeeks.

**Step 2 – Learn Various ML Concepts**

Now that you are done with the prerequisites, you can move on to learning ML (Which is the fun part!!!) It’s best to start with the basics and then move on to the more complicated stuff. Some of the basic concepts in ML are:

**(a) Terminologies of Machine Learning**

* Model – A model is a specific representation learned from data by applying some machine learning algorithm. A model is also called a hypothesis.
* Feature – A feature is an individual measurable property of the data. A set of numeric features can be conveniently described by a feature vector. Feature vectors are fed as input to the model. For example, in order to predict a fruit, there may be features like color, smell, taste, etc.
* Target (Label) – A target variable or label is the value to be predicted by our model. For the fruit example discussed in the feature section, the label with each set of input would be the name of the fruit like apple, orange, banana, etc.
* Training – The idea is to give a set of inputs(features) and it’s expected outputs(labels), so after training, we will have a model (hypothesis) that will then map new data to one of the categories trained on.
* Prediction – Once our model is ready, it can be fed a set of inputs to which it will provide a predicted output(label).

**(b) Types of Machine Learning**

* Supervised Learning – This involves learning from a training dataset with labeled data using classification and regression models. This learning process continues until the required level of performance is achieved.
* Unsupervised Learning – This involves using unlabelled data and then finding the underlying structure in the data in order to learn more and more about the data itself using factor and cluster analysis models.
* Semi-supervised Learning – This involves using unlabelled data like Unsupervised Learning with a small amount of labeled data. Using labeled data vastly increases the learning accuracy and is also more cost-effective than Supervised Learning.
* Reinforcement Learning – This involves learning optimal actions through trial and error. So, the next action is decided by learning behaviors that are based on the current state and that will maximize the reward in the future.

**Advantages of Machine learning**

**1. Easily identifies trends and patterns -**

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

**2. No human intervention needed (automation)**

With ML, you don’t need to babysit your project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus softwares; they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

**3. Continuous Improvement**

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say you need to make a weather forecast model. As the amount of data you have keeps growing, your algorithms learn to make more accurate predictions faster.

**4. Handling multi-dimensional and multi-variety data**

Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

**5. Wide Applications**

You could be an e-tailer or a healthcare provider and make ML work for you. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

**Disadvantages of Machine Learning**

**1. Data Acquisition**

Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.

**2. Time and Resources**

ML needs enough time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power for you.

**3. Interpretation of Results**

Another major challenge is the ability to accurately interpret results generated by the algorithms. You must also carefully choose the algorithms for your purpose.

**4. High error-susceptibility**

Machine Learning is autonomous but highly susceptible to errors. Suppose you train an algorithm with data sets small enough to not be inclusive. You end up with biased predictions coming from a biased training set. This leads to irrelevant advertisements being displayed to customers. In the case of ML, such blunders can set off a chain of errors that can go undetected for long periods of time. And when they do get noticed, it takes quite some time to recognize the source of the issue, and even longer to correct it.

**CHAPTER 7**

**SOFTWARE ENVIRONMENT**

**What is Python?**

Below are some facts about Python.

* Python is currently the most widely used multi-purpose, high-level programming language.
* Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
* Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
* Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber… etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

* + Machine Learning
  + GUI Applications (like Kivy, Tkinter, PyQt etc. )
  + Web frameworks like Django (used by YouTube, Instagram, Dropbox)
  + Image processing (like Opencv, Pillow)
  + Web scraping (like Scrapy, BeautifulSoup, Selenium)
  + Test frameworks
  + Multimedia

**Advantages of Python**

Let’s see how Python dominates over other languages.

**1. Extensive Libraries**

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don’t have to write the complete code for that manually.

**2. Extensible**

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

**3. Embeddable**

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

**4. Improved Productivity**

The language’s simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

**5. IOT Opportunities**

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

**6. Simple and Easy**

When working with Java, you may have to create a class to print ‘Hello World’. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

**7. Readable**

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

**8. Object-Oriented**

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

**9. Free and Open-Source**

Like we said earlier, Python is freely available. But not only can you download Python for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

**10. Portable**

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn’t the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

**11. Interpreted**

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

**Advantages of Python Over Other Languages**

**1. Less Coding**

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don’t have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

**2. Affordable**

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

**3. Python is for Everyone**

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

**Disadvantages of Python**

So far, we’ve seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let’s now see the downsides of choosing Python over another language.

**1. Speed Limitations**

We have seen that Python code is executed line by line. But since Python is interpreted, it often results in slow execution. This, however, isn’t a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

**2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonnelle.

The reason it is not so famous despite the existence of Brython is that it isn’t that secure.

**3. Design Restrictions**

As you know, Python is dynamically-typed. This means that you don’t need to declare the type of variable while writing the code. It uses duck-typing. But wait, what’s that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

**4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python’s database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

**5. Simple**

No, we’re not kidding. Python’s simplicity can indeed be a problem. Take my example. I don’t do Java, I’m more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

**History of Python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde &Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners1, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

**Python Development Steps**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of list, dict, str and others. It was also object oriented and had a module system.  
Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

* Print is now a function.
* Views and iterators instead of lists
* The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
* There is only one integer type left, i.e., int. long is int as well.
* The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behaviour.
* Text Vs. Data Instead of Unicode Vs. 8-bit

**Purpose**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

**Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Modules Used in Project**

**TensorFlow**

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.‍

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

**NumPy**

NumPy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

* A powerful N-dimensional array object
* Sophisticated (broadcasting) functions
* Tools for integrating C/C++ and Fortran code
* Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary datatypes can be defined using NumPy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**Pandas**

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

**Matplotlib**

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

**Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

* Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

**Install Python Step-by-Step in Windows and Mac**

Python a versatile programming language doesn’t come pre-installed on your computer devices. Python was first released in the year 1991 and until today it is a very popular high-level programming language. Its style philosophy emphasizes code readability with its notable use of great whitespace.

The object-oriented approach and language construct provided by Python enables programmers to write both clear and logical code for projects. This software does not come pre-packaged with Windows.

**How to Install Python on Windows and Mac**

There have been several updates in the Python version over the years. The question is how to install Python? It might be confusing for the beginner who is willing to start learning Python but this tutorial will solve your query. The latest or the newest version of Python is version 3.7.4 or in other words, it is Python 3.

Note: The python version 3.7.4 cannot be used on Windows XP or earlier devices.

Before you start with the installation process of Python. First, you need to know about your System Requirements. Based on your system type i.e. operating system and based processor, you must download the python version. My system type is a Windows 64-bit operating system. So the steps below are to install python version 3.7.4 on Windows 7 device or to install Python 3. Download the Python Cheatsheet here.The steps on how to install Python on Windows 10, 8 and 7 are divided into 4 parts to help understand better.

**Download the Correct version into the system**

**Step 1:** Go to the official site to download and install python using Google Chrome or any other web browser. OR Click on the following link: https://www.python.org

A screenshot of a computer

Description automatically generated with medium confidence

Now, check for the latest and the correct version for your operating system.

**Step 2:** Click on the Download Tab.

Graphical user interface, application

Description automatically generated

**Step 3:** You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Graphical user interface, application

Description automatically generated

**Step 4:** Scroll down the page until you find the Files option.

**Step 5:** Here you see a different version of python along with the operating system.

Graphical user interface, text

Description automatically generated

* To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.
* To download Windows 64-bit python, you can select any one from the three options: Windows x86-64 embeddable zip file, Windows x86-64 executable installer or Windows x86-64 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

**Installation of Python**

**Step 1:** Go to Download and Open the downloaded python version to carry out the installation process.

Graphical user interface, text, application

Description automatically generated

**Step 2:** Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.

Graphical user interface, text, application, chat or text message

Description automatically generated

**Step 3:** Click on Install NOW After the installation is successful. Click on Close.

Graphical user interface, text, application, chat or text message

Description automatically generated

With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

Note: The installation process might take a couple of minutes.

**Verify the Python Installation**

**Step 1:** Click on Start

**Step 2:** In the Windows Run Command, type “cmd”.

Graphical user interface, application

Description automatically generated

**Step 3:** Open the Command prompt option.

**Step 4:** Let us test whether the python is correctly installed. Type python –V and press Enter.

A screenshot of a computer

Description automatically generated with medium confidence

**Step 5:** You will get the answer as 3.7.4

Note: If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

**Check how the Python IDLE works**

**Step 1:** Click on Start

**Step 2:** In the Windows Run command, type “python idle”.

Application

Description automatically generated with low confidence

**Step 3:** Click on IDLE (Python 3.7 64-bit) and launch the program

**Step 4:** To go ahead with working in IDLE you must first save the file. Click on File > Click on Save

Graphical user interface, text, application, email

Description automatically generated

**Step 5:** Name the file and save as type should be Python files. Click on SAVE. Here I have named the files as Hey World.

**Step 6:** Now for e.g., enter print (“Hey World”) and Press Enter.  
Graphical user interface, text, application, email

Description automatically generated

You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won’t work.

**CHAPTER 8**

**SYSTEM REQUIREMENTS**

**Hardware Requirements**

System : Pentium IV 2.4 GHz.

Hard Disk : 40 GB.

Floppy Drive : 1.44 Mb.

Monitor : 15 VGA Colour.

Mouse : Logitech.

Ram : 512 Mb.

**Software Requirements**

Operating System : Windows

Coding Language : Python 3.7

**CHAPTER 9**

**FUNCTIONAL REQUIREMENTS**

**Output Design**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization.
* Internal Outputs whose destination is within organization, and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**Output Definition**

**The outputs should be defined in terms of the following points:**

1. Type of the output
2. Content of the output
3. Format of the output
4. Location of the output
5. Frequency of the output
6. Volume of the output
7. Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**Input Design**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**Input Stages**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**Error Detection**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

**User Interface Systems Can Be Broadly Classified As:**

1. User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
2. Computer initiated interfaces.

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**User Initiated Interfaces**

User initiated interfaces fall into tow approximate classes:

1. Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
2. Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms oriented interface is chosen because it is the best choice.

**Computer-Initiated Interfaces**

The following computer – initiated interfaces were used:

1. The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
2. Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**Error Message Design:**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**Performance Requirements**

Performance is measured in terms of the output provided by the application.

Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system.
* The system should be accurate.
* The system should be better than the existing system.
* The existing system is completely dependent on the user to perform all the duties.

**CHAPTER 10**

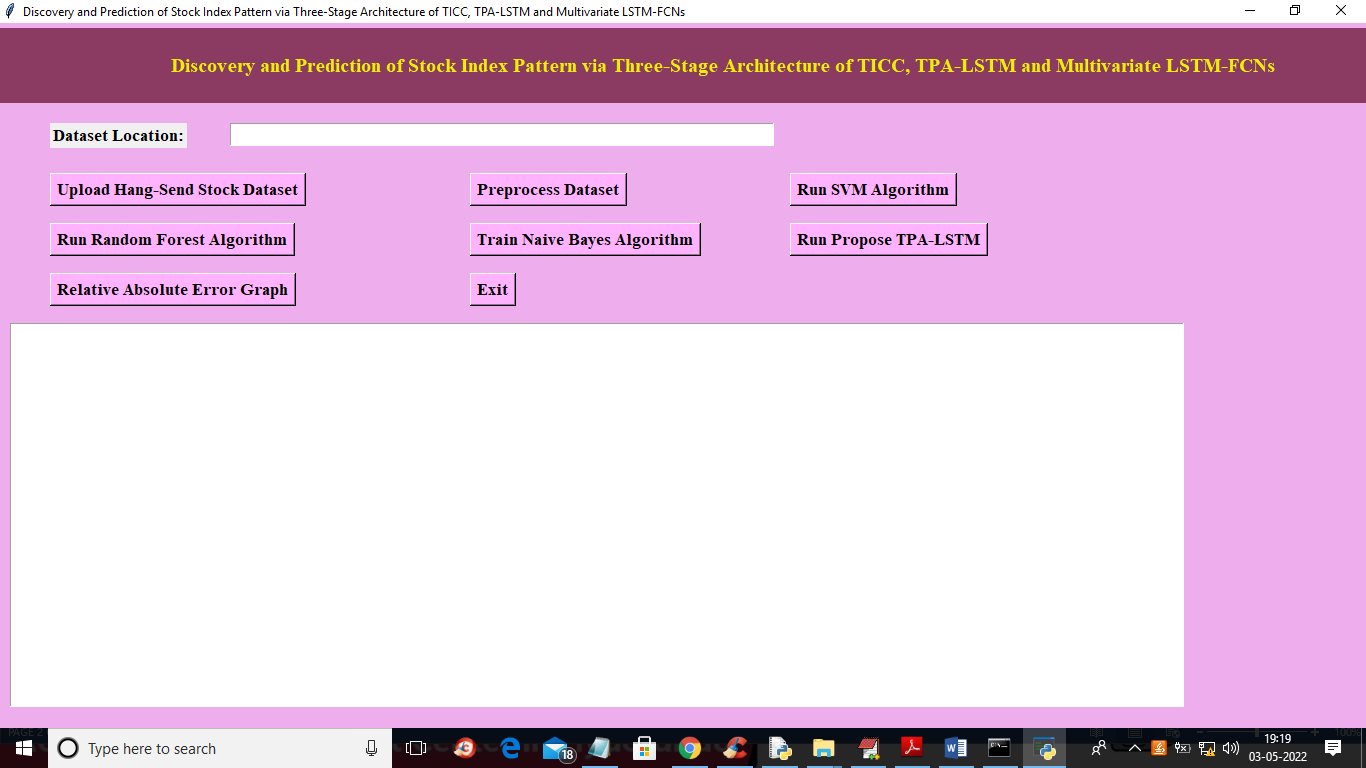
**RESULTS AND DISCUSSIONS**

To implement this project, we have designed following modules.

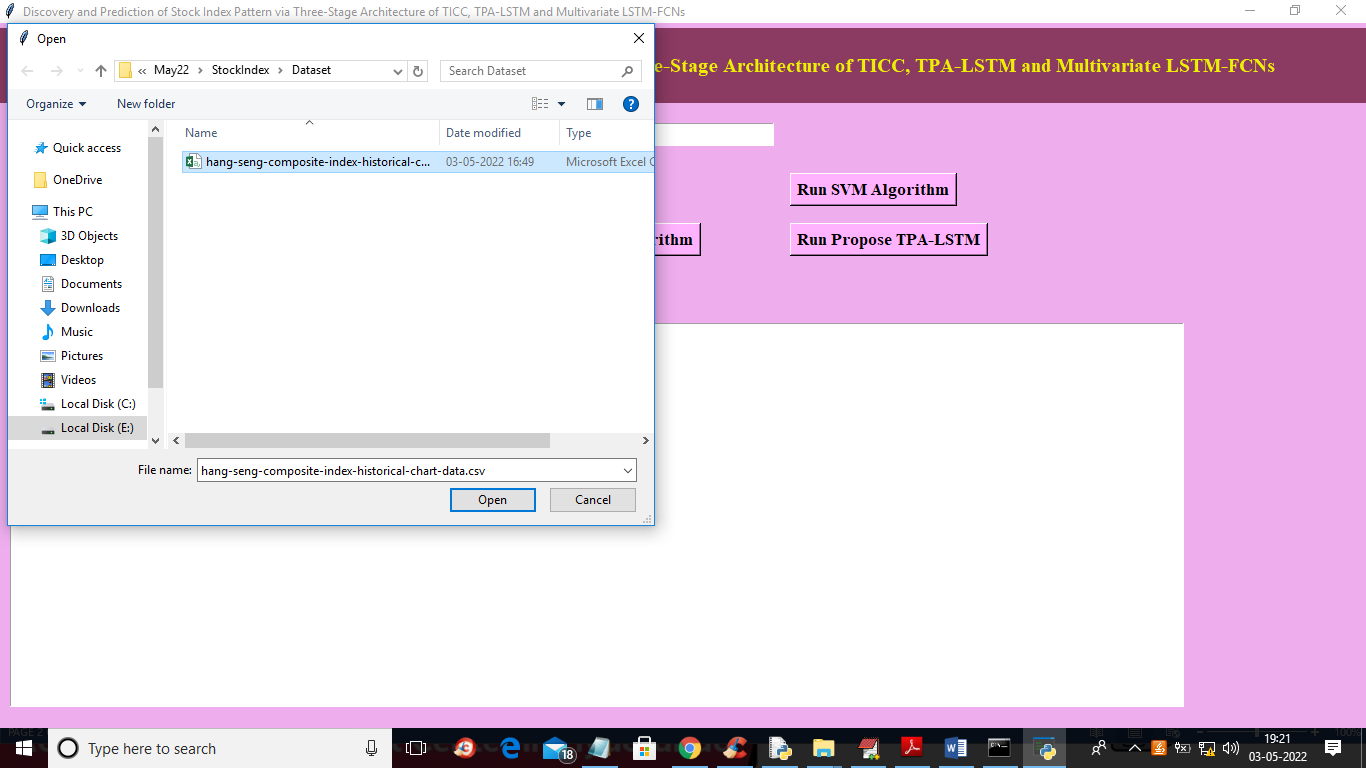
1. Upload Hang-Send Stock Dataset: using this module we will upload dataset to application.
2. Preprocess Dataset: using this module we will read all dataset values and then normalize values using MIN-MAX scaler.
3. Run SVM Algorithm: using this module we will split dataset into train and test and then train SVM on training dataset and then calculate accuracy and RAE on test data prediction.
4. Run Random Forest Algorithm: using this module we will split dataset into train and test and then train Random Forest on training dataset and then calculate accuracy and RAE on test data prediction.
5. Train Naive Bayes Algorithm: using this module we will split dataset into train and test and then train Naïve Bayes on training dataset and then calculate accuracy and RAE on test data prediction.
6. Run Propose TPA-LSTM: using this module we will split dataset into train and test and then train TPA-LSTM on training dataset and then calculate accuracy and RAE on test data prediction.
7. Relative Absolute Error Graph: using this module we will plot RAE (relative absolute error) graph between all algorithms.

**SCREENSHOTS**

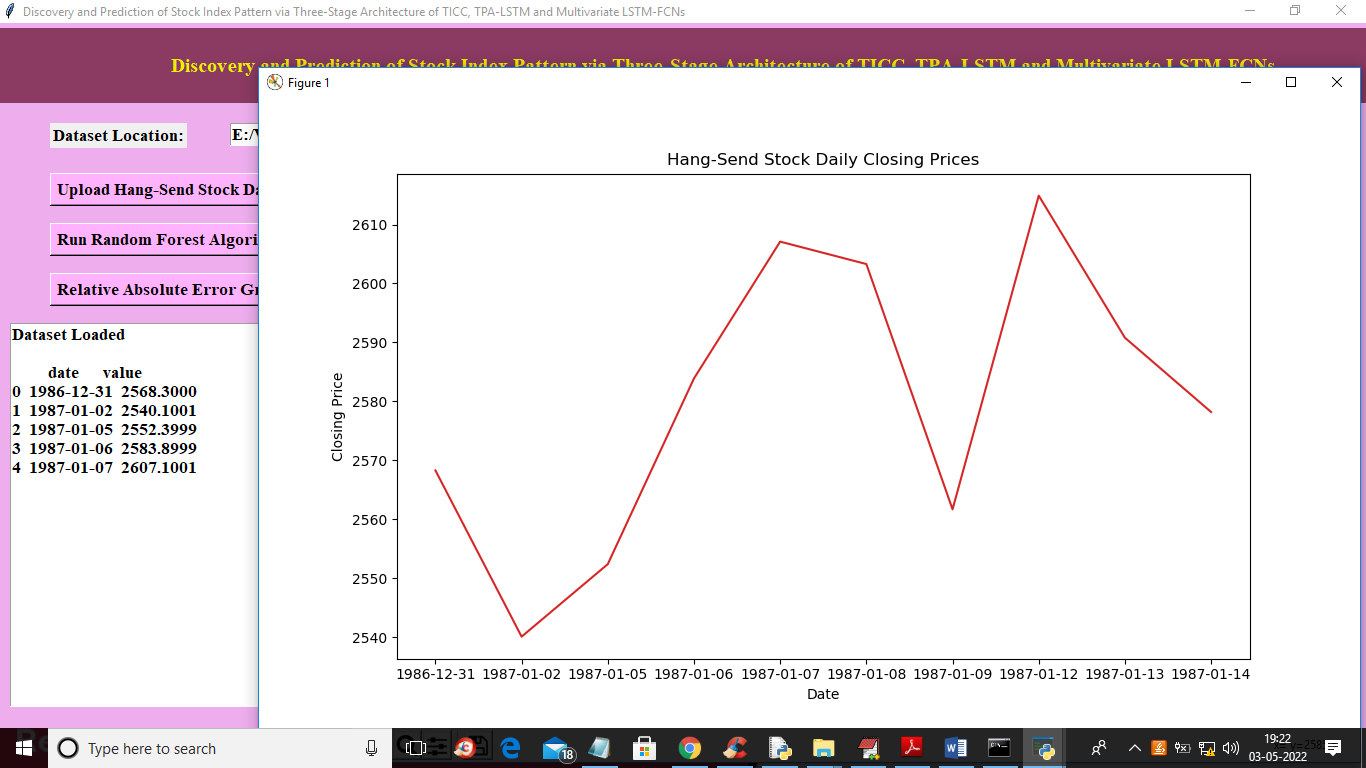
To run project double click on ‘run.bat’ file to get below screen.



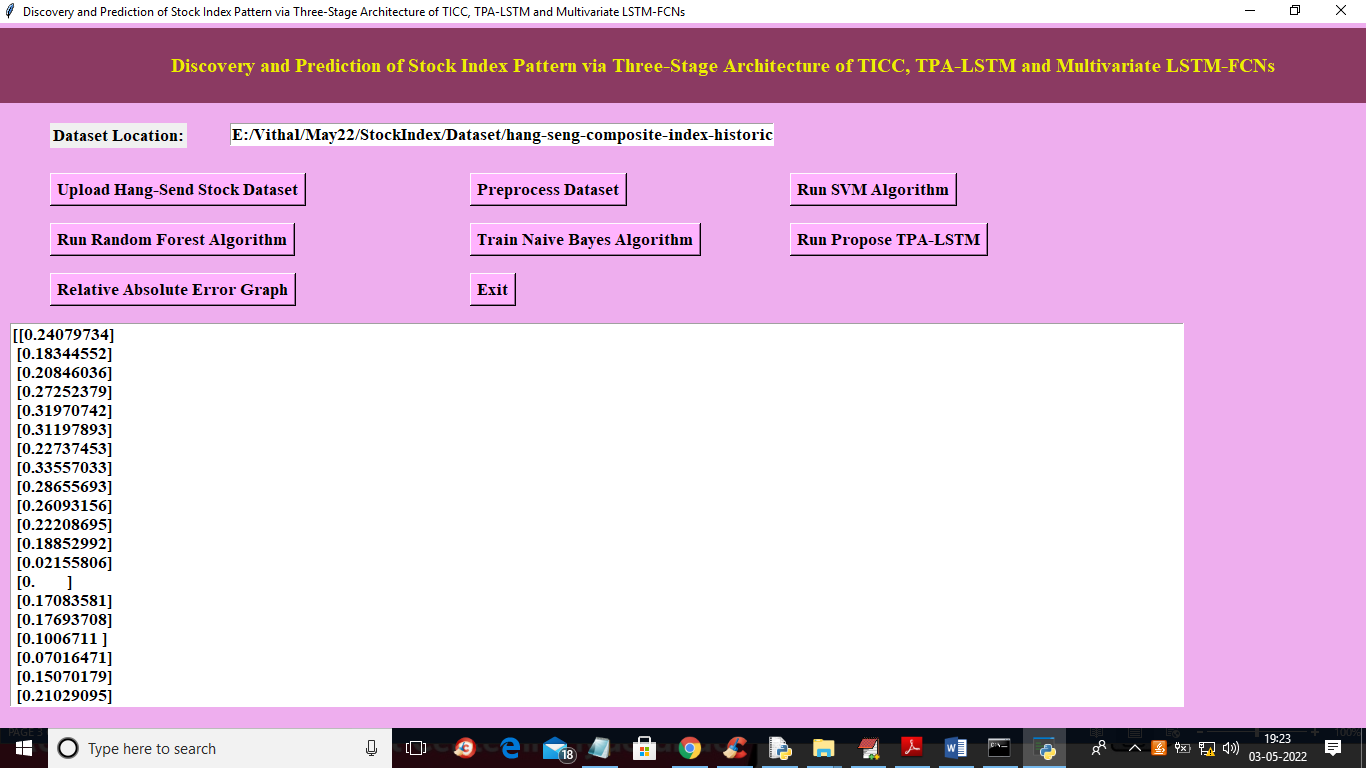
In above screen click on ‘Upload Hang Send Stock Dataset’ button to upload dataset and to get below output



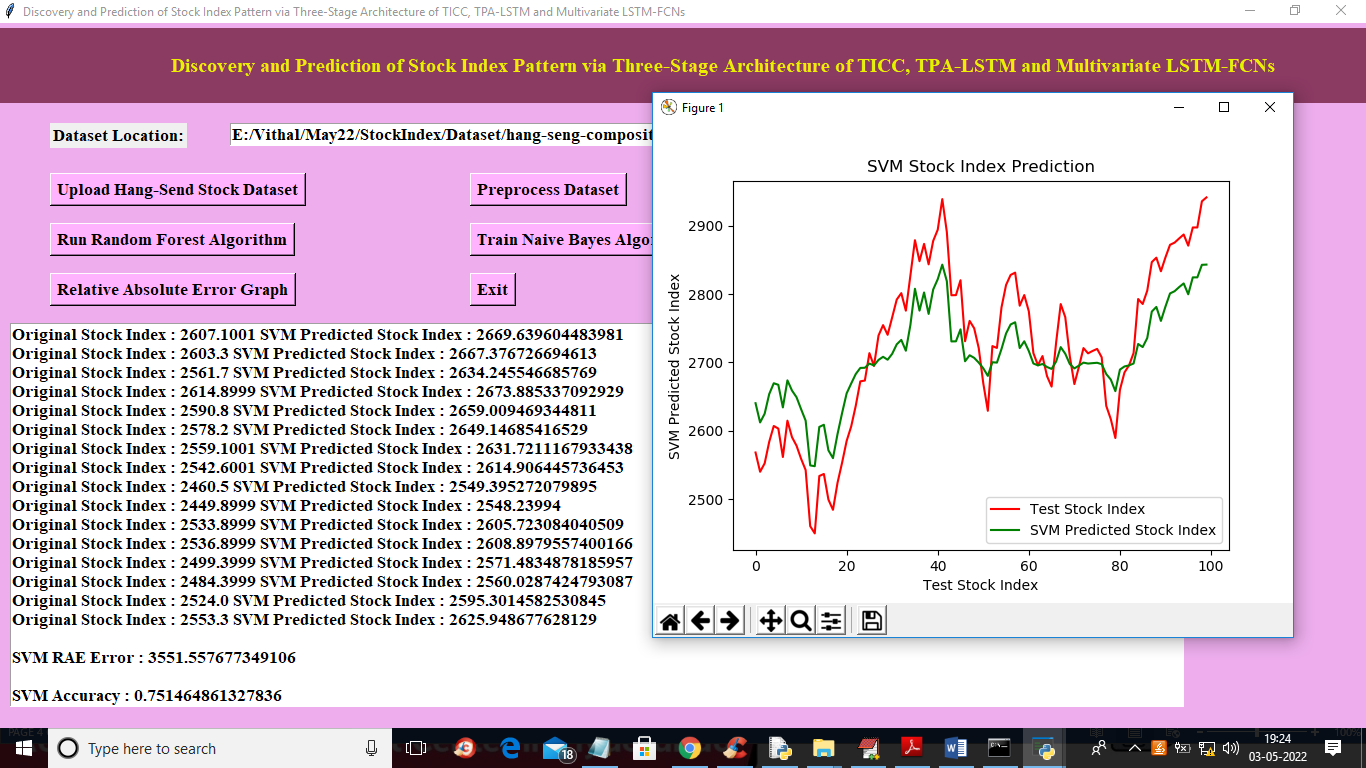
In above screen selecting and uploading ‘Hang Seng’ dataset file and then click on ‘Open’ button to load dataset and to get below output



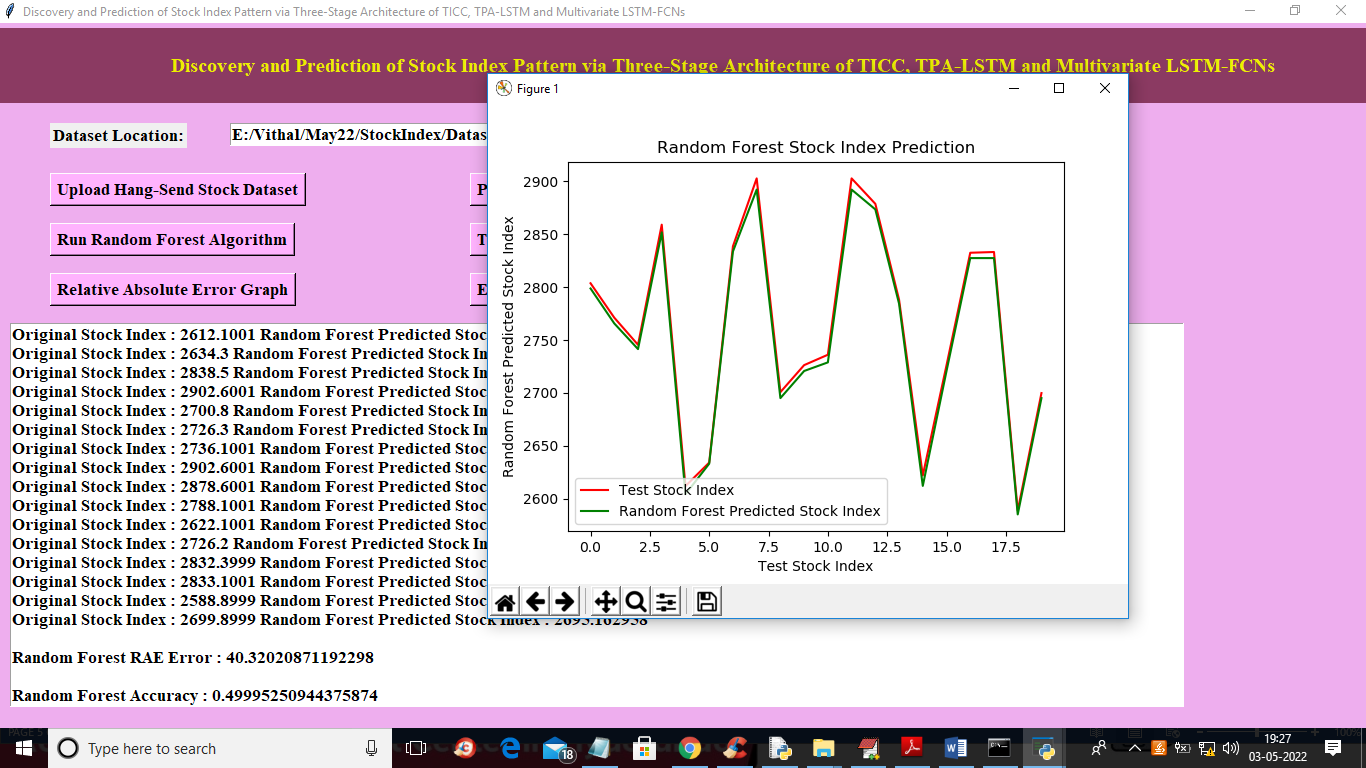
In above screen in text area we can see dataset loaded and in graph x-axis represents DATE and y-axis represent stock value on that date and now close above graph and then click on ‘Preprocess Dataset’ button to read all values and then normalize them and get below output



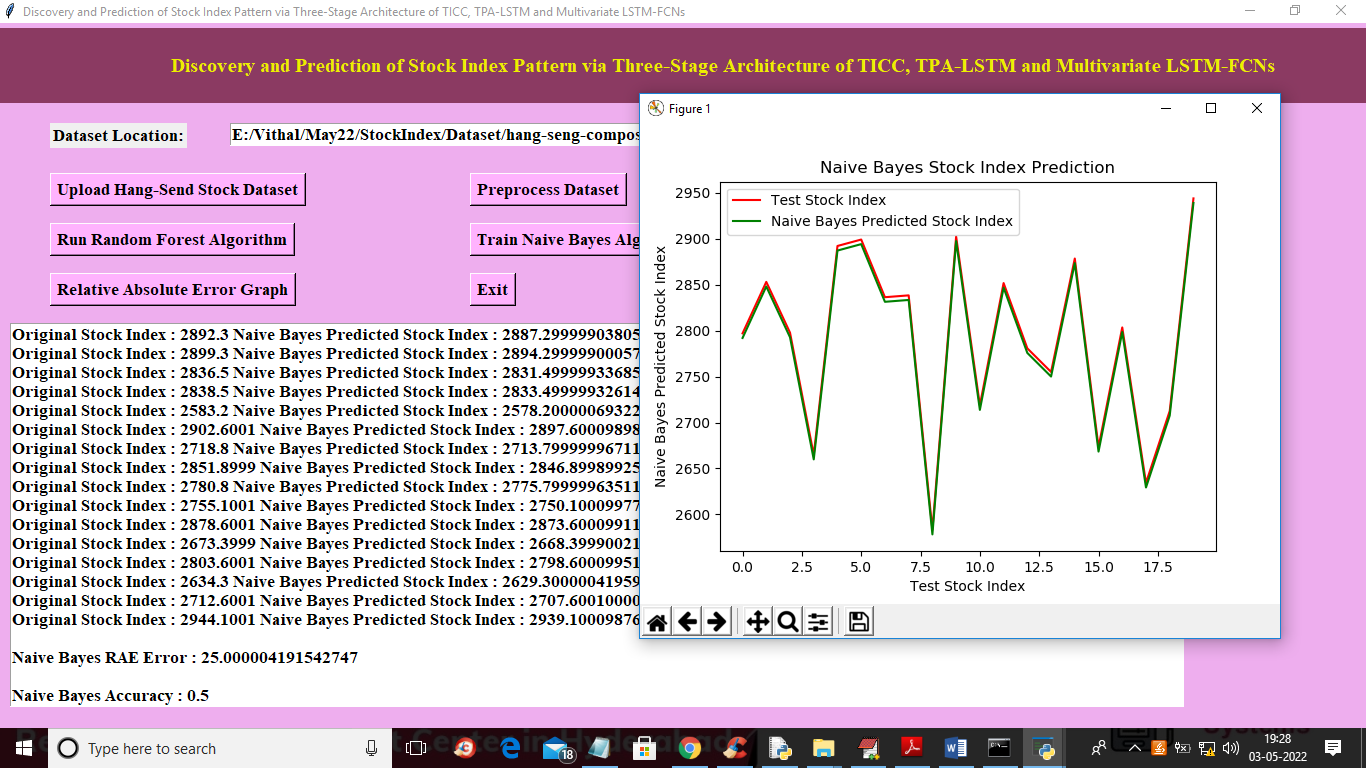
In above screen all stock values are normalize between 0 and 1 and now click on ‘Run SVM Algorithm’ button to get below output



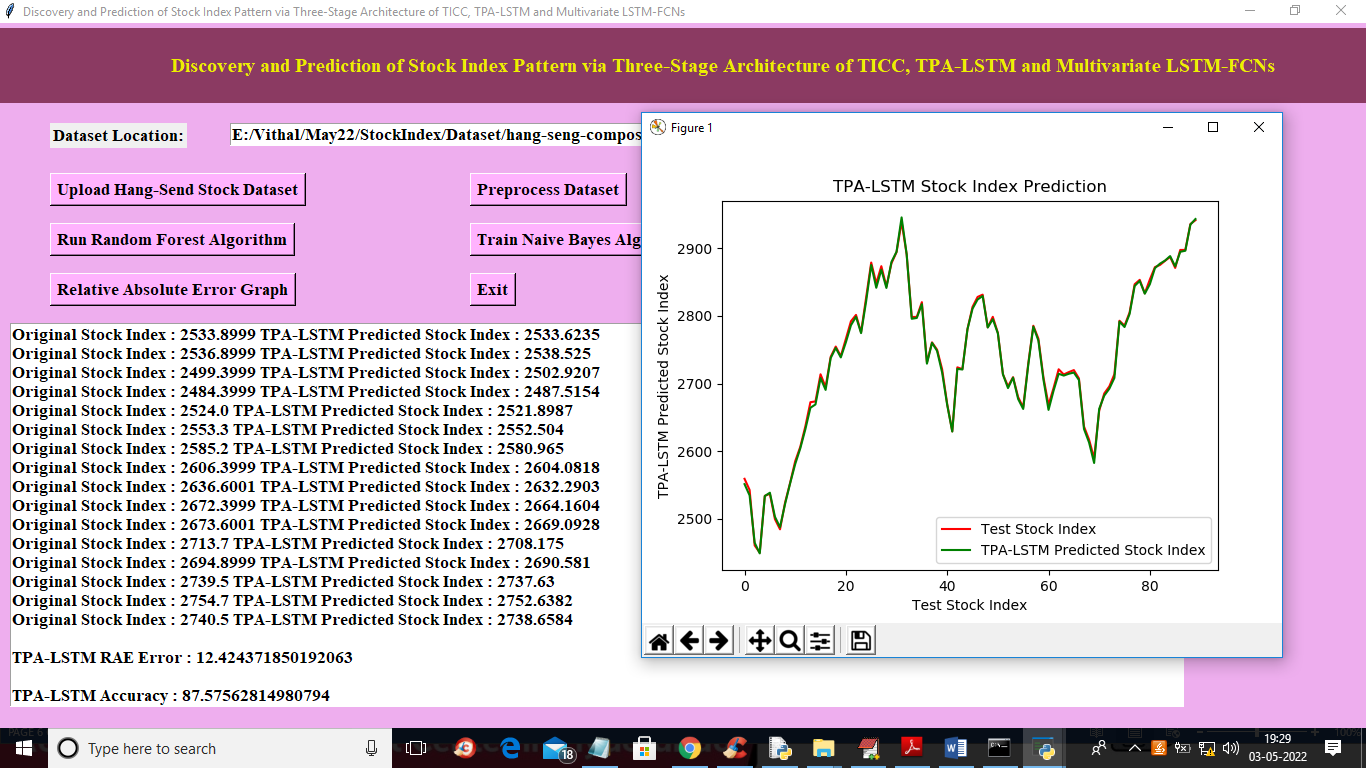
In above screen text area we can see original test value and predicted value from SVM and then calculate difference between original test value and predicted value as RAE and we got SVM RAE as 3551 and accuracy as 75% and in graph x-axis represents days and y-axis represents stock values and RED line represents original test value and green line represents predicted value and we can see there is huge gap between red and green line so prediction is not accurate and if prediction is accurate then both lines get overlap and now close above graph and then click on ‘Run Random Forest’ button to get below output



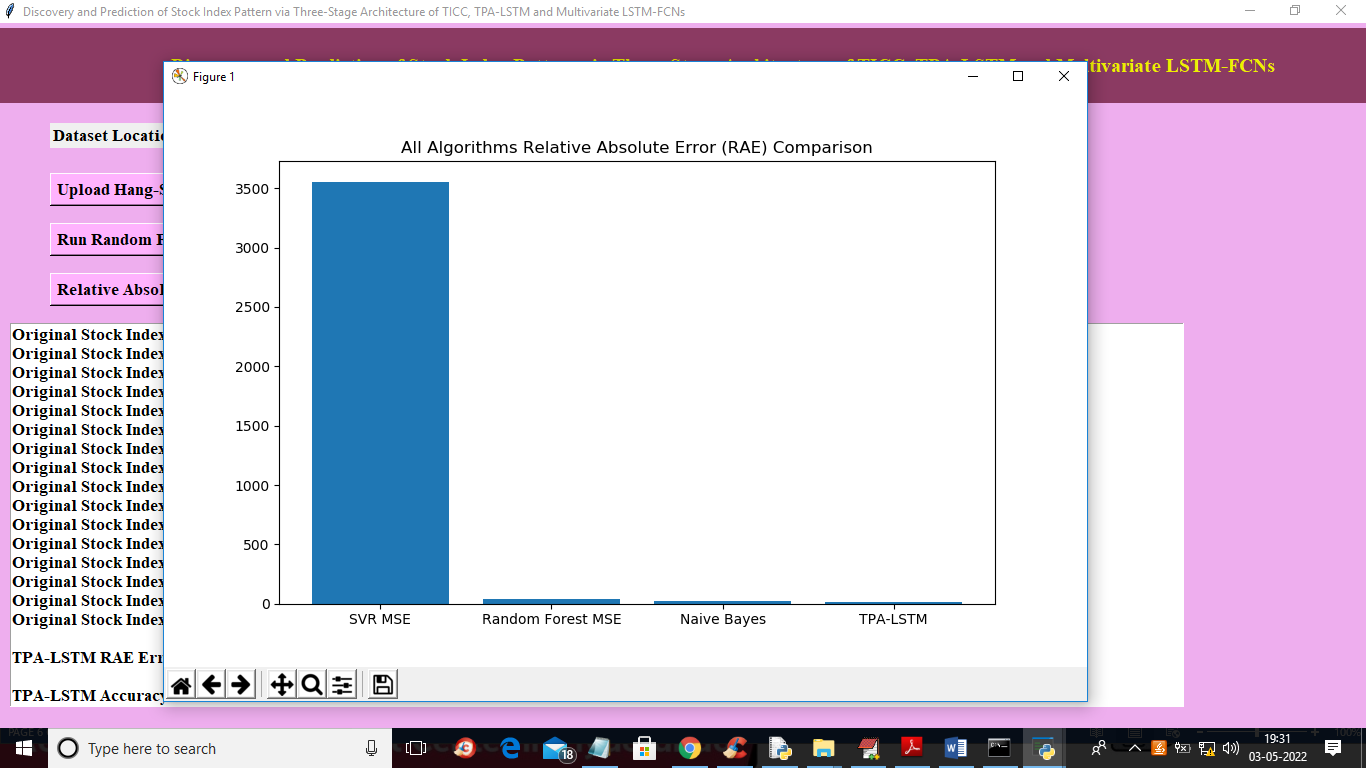
In above screen with random forest both lines are overlapping so its RAE error reduce to 40 and its prediction is little accurate and now close above graph and then click on ‘Run Naïve Bayes Algorithm’ button to get below output



In above screen with Naïve Bayes we got 25% error rate and both lines are overlapping so its prediction is also little accurate and now close above graph and then click on ‘Run Propose TPA-LSTM’ button to train propose algorithm and get below output



In above screen with propose LSTM-TPA we got RAE as 12% and accuracy as 87% and we can see both lines are overlapping without any gap so propose algorithm prediction is accurate. Now close above graph and then click on ‘Relative Absolute Error Graph’ button to get below graph



In above screen x-axis represents algorithm names and y-axis represents RAE error and, in all algorithms, propose LSTM-TPA got less error rate, so its performance is good.

**CHAPTER 11**

**CONCLUSION AND FUTURE SCOPE**

Discovery and prediction of stock index pattern are of great importance to reduce uncertainty and risks in financial markets and, more specifically, is crucial in constructing a financial portfolio. In the literature of stock index pattern discovery and prediction through neural networks, previous studies mainly focus on pattern discovery and up-down prediction of stock index with strong repeated patterns and fixed time periods. This paper makes up for the shortcomings of previous research, which forms a complete structure of stock index pattern discovery and prediction through a proposed three stage architecture of TICC, TPA-LSTM, and Multivariate LSTM-FCNs. Through proposed three-stage architecture, this paper could analyze and predict stock index prices with weak periodic and flexible patterns.

The proposed three-stage architecture contains three stages. In the first stage, we apply TICC to cluster industry stock indices in the comprehensive stock index and map cluster results to that stock index. Based on the mapping results, we could discover repeated patterns of the comprehensive stock index on the training dataset. In the second stage, we predict multivariate time series of industry stock indices simultaneously through TPA-LSTM. In the third section, we predict repeated patterns of the comprehensive stock index on the test dataset through Multivariate LSTM-FCNs. HSCI and eleven industry indices that are included in the HSCI are used in the experiment. The empirical results show that the proposed three-stage architecture, including TICC, TPA-LSTM, and Multivariate LSTM-FCNs significantly improves the state-ofthe-art results in pattern discovery and prediction of HSCI. Moreover, we propose a bullish trading rule and construct an equal proportion portfolio based on this trading rule and the prediction results of the proposed three-stage architecture. Seven comprehensive stock indices are used in the experiment. The empirical results show that, the constructed portfolio based on the bullish trading rule and the proposed three-stage architecture performs significantly better than the market-based portfolio. Therefore, the proposed three-stage architecture is a feasible and promising method to discover and predict repeated patterns of stock index in financial markets. There are two promising extensions of pattern discovery and prediction in stock index prices. One possible extension of stock index pattern prediction is to conduct proactive index tracking or construct other trading strategies with predicted patterns of stock index. The other extension is to search for more suitable and effective price prediction and pattern matching methods to improve the performance of the proposed structure.

**REFERENCES**

1. Kazem, E. Sharifi, F. K. Hussain, M. Saberi, and O. K. Hussain, ‘‘Support vector regression with chaos-based firefly algorithm for stock market price forecasting,’’ Appl. Soft Comput., vol. 13, no. 2, pp. 947–958, Feb. 2013, doi: 10.1016/j.asoc.2012.09.024.
2. D. Hallac, S. Vare, S. Boyd, and J. Leskovec, ‘‘Toeplitz inverse covariance based clustering of multivariate time series data,’’ in Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Halifax, NS, Canada, Aug. 2017, pp. 215–223.
3. S.-Y. Shih, F.-K. Sun, and H.-Y. Lee, ‘‘Temporal pattern attention for multivariate time series forecasting,’’ Mach. Learn., vol. 108, nos. 8–9, pp. 1421–1441, Sep. 2019, doi: 10.1007/s10994-019-05815-0.
4. F. Karim, S. Majumdar, H. Darabi, and S. Harford, ‘‘Multivariate LSTMFCNs for time series classification,’’ Neural Netw., vol. 116, pp. 237–245, Aug. 2019, doi: 10.1016/j.neunet.2019.04.014.
5. D. Lamponi, ‘‘Is industry classification useful to predict US Stock Price co-movements?’’ J. Wealth Manage., vol. 17, no. 1, pp. 71–77, Apr. 2014.
6. Ouyang, Hongbing & Wei, Xiaolu & Wu, Qiufeng. (2020). Discovery and Prediction of Stock Index Pattern via three-stage architecture of TICC, TPA-LSTM and Multivariate LSTM-FCNs (February 2019). IEEE Access. PP. 1-1. 10.1109/ACCESS.2020.3005994.
7. X. Wei, B. Lei, H. Ouyang, Q. Wu, "Stock Index Prices Prediction via Temporal Pattern Attention and Long-Short-Term Memory", Advances in Multimedia, vol. 2020, Article ID 8831893, 7 pages, 2020. https://doi.org/10.1155/2020/8831893
8. L. Sayavong, Z. Wu and S. Chalita, "Research on Stock Price Prediction Method Based on Convolutional Neural Network," 2019 International Conference on Virtual Reality and Intelligent Systems (ICVRIS), Jishou, China, 2019, pp. 173-176, doi: 10.1109/ICVRIS.2019.00050.
9. M. Wen, P. Li, L. Zhang, and Y. Chen, "Stock Market Trend Prediction Using High-Order Information of Time Series," in IEEE Access, vol. 7, pp. 28299-28308, 2019, doi: 10.1109/ACCESS.2019.2901842.
10. A. Thakkar, K. Chaudhari, Fusion in stock market prediction: A decade survey on the necessity, recent developments, and potential future directions, Information Fusion, Vol. 65, 2021, Pages 95-107, ISSN 1566-2535, https://doi.org/10.1016/j.inffus.2020.08.019.
11. Dattatray P. Gandhmal, K. Kumar, Systematic analysis and review of stock market prediction techniques, Computer Science Review, Volume 34, 2019, 100190, ISSN 1574-0137, https://doi.org/10.1016/j.cosrev.2019.08.001.
12. Chung, H., Shin, Ks. Genetic algorithm-optimized multi-channel convolutional neural network for stock market prediction. Neural Comput & Applic 32, 7897–7914 (2020). https://doi.org/10.1007/s00521-019-04236-3
13. Zhong, X., Enke, D. Predicting the daily return direction of the stock market using hybrid machine learning algorithms. Financ Innov 5, 24 (2019). https://doi.org/10.1186/s40854-019-0138-0
14. M. Jiang, J. Liu, L. Zhang, C. Liu, An improved Stacking framework for stock index prediction by leveraging tree-based ensemble models and deep learning algorithms, Physica A: Statistical Mechanics and its Applications, Volume 541, 2020, 122272, ISSN 0378-4371, https://doi.org/10.1016/j.physa.2019.122272.
15. Kumar, G., Jain, S. & Singh, U.P. Stock Market Forecasting Using Computational Intelligence: A Survey. Arch Computat Methods Eng 28, 1069–1101 (2021). https://doi.org/10.1007/s11831-020-09413-5.