Supervised Machine Learning Algorithms for Stock Market Prediction

V.ChandraSekhar[1], V.Satish Kumar[2]
[1]*M.tech, Dept. of CSE, Acharya Nagarjuna University, India*
[2]*Asst.Profeesor, Dept. of CSE, Acharya Nagarjuna University, India*

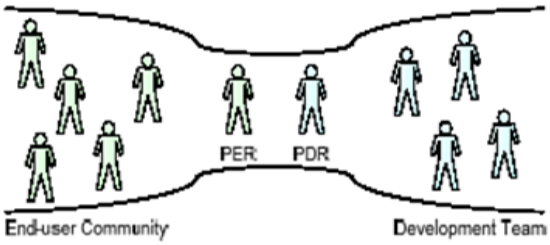
***Abstract*- In this paper, the comparative study of the supervised machine learning algorithms using the time window of size 1 to 90 has been proposed. The algorithms have been compared based upon the parameters: Size of the dataset and Number of technical indicators used. Accuracy and F measure values have been computed for each algorithm. Long term model has been used to compute the accuracy and F-measure. The proposed architecture for the implemented work mainly consist of four steps: feature extraction from the given dataset, supervised classification of the training dataset, supervised classification of the test dataset, and result evaluation.**

Index terms- Machine learning, Classifier, random forest.

I. INTRODUCTION

Too many software development efforts go awry when development team and customer personnel get caught up in the possibilities of automation. Instead of focusing on high priority features, the team can become mired in a sea of nice to have features that are not essential to solve the problem, but in themselves are highly attractive. This is the root cause of large percentage of failed and or abandoned development efforts and is the primary reason the development team utilizes the iterative model.

The PER and PDR are the brain trust for the development effort. The PER has the skills and domain knowledge necessary to understand the issues associated with the business processes to the supported by the application and has a close working relationship with the other members of the end-user community. The PDR has the same advantages regarding the application development process and the other members of the development team together, they act as the concentration points for knowledge about the application to be developed.



The objective of this approach is to create the close relationship that is characteristic of a software project with one developer and one end-user in essence, this approach the "pair programming" concept from Agile methodologies and extends it to the end-user community. While it is difficult to create close relationships between the diverse members of an end-user community and a software development team, it is much simpler to create a close relationship between the lead representatives for each group.

When multiple end-users are placed into relationship with multiple members of a development team, communication between the two groups degrades as the number of participants grows.

In this model, members of end-user community may communicate with members of the development team as needed, but it is the responsibility of all participants to keep the PER and PDR apprised of the communications for example, this allows the PER and PDR to resolve conflicts that arise when two different end-users communicate different requirements for the same application feature to different members of the development team.

II. RELATED WORK

When Object orientation is used in analysis as well as design, the boundary between OOA and OOD is blurred. This is particularly true in methods that combine analysis and design. One reason for this

blurring is the similarity of basic constructs (i.e.,objects and classes) that are used in OOA and OOD. Through there is no agreement about what parts of the object-oriented development process belongs to analysis and what parts to design, there is some general agreement about the domains of the two activities.

The fundamental difference between OOA and OOD is that the former models the problem domain, leading to an understanding and specification of the problem, while the latter models the solution to the problem. That is, analysis deals with the problem domain, while design deals with the solution domain. However, in OOAD subsumed in the solution domain representation. That is, the solution domain representation, created by OOD, generally contains much of the representation created by OOA. The separating line is matter of perception, and different people have different views on it. The lack of clear separation between analysis and design can also be considered one of the strong points of the object-oriented approach the transition from analysis to design is "seamless". This is also the main reason OOAD methods-where analysis and designs are both performed.

The main difference between OOA and OOD, due to the different domains of modeling, is in the type of objects that come out of the analysis and design process.

III.RESEARCH DATA

Software engineering has slowly become part of our everyday life. From washing machines to compact disc player, through cash machines and phones, most of our daily activities use software, and as time goes by, the more complex and costly this software becomes.

The demand for sophisticated software greatly increases the constraints imposed on development teams. Software engineers are facing a world of growing complexity due to the nature of applications, the distributed and heterogeneous environments, the size of programs, the organization of software development teams, and the end-users ergonomic expectations.

To surmount these difficulties, software engineers will have to learn not only how to do their job, but also how to explain their work to others, and how to understand when others work is explained to them. For these reasons, they have (and will always have) an increasing need for methods.

IV. PROPOSED METHODOLOGY

The unification of object-oriented modeling methods became possible as experience allowed evaluation of the various concepts proposed by existing methods. Based on the fact that differences between the various methods were becoming smaller, and that the method wars did not move object-oriented technology forward any longer, Jim Rumbaugh and Grady Booch decided at the end of 1994 to unify their work within a single method: the Unified Method. A year later they were joined by Ivar Jacobson, the father of use cases, a very efficient technique for the determination of requirements.

Booch, Rumbaugh and Jacobson adopted four goals:

- To represent complete systems (instead of only the software portion) using object oriented concepts.
- To establish an explicit coupling between concepts and executable code.
- To take into account the scaling factors that are inherent to complex and critical systems.
- To creating a modeling language usable by both humans and machines.

4.1 FIRST NORMAL FORM:

A relation is said to be in first normal form if the values in the relation are atomic for every attribute in the relation. By this we mean simply that no attribute value can be a set of values or, as it is sometimes expressed, a repeating group.

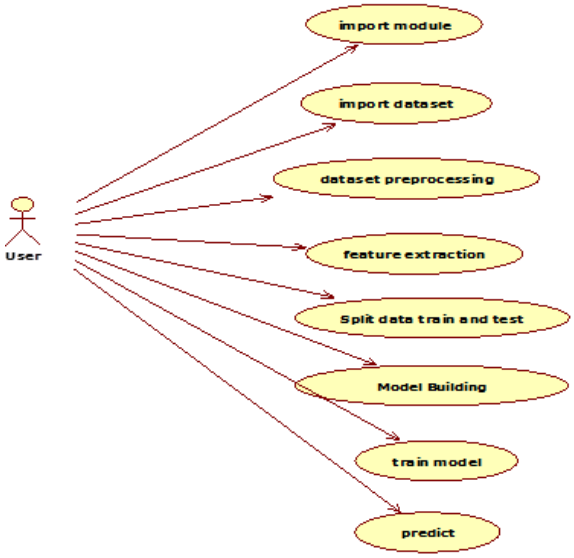
4.2 SECOND NORMAL FORM:

A relation is said to be in second Normal form is it is in first normal form and it should satisfy any one of the following rules.

1. Primary key is a not a composite primary key
2. No non key attributes are present
3. Every non key attribute is fully functionally dependent on full set of primary key.

Transitive Dependency: If two non key attributes depend on each other as well as on the primary key then they are said to be transitively dependent.

The above normalization principles were applied to decompose the data in multiple tables thereby making the data to be maintained in a consistent state.



4.4 E – R DIAGRAMS

- The relation upon the system is structure through a conceptual ER-Diagram, which not only specifics the existential entities but also the standard relations through which the system exists and the cardinalities that are necessary for the system state to continue.
- The entity Relationship Diagram (ERD) depicts the relationship between the data objects. The ERD is the notation that is used to conduct the date modeling activity the attributes of each data object noted is the ERD can be described resign a data object descriptions.

4.5 DATA FLOW DIAGRAMS

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

V. EXPERIMENTAL RESULTS

Input

These are the input predictions taken by the stock market.

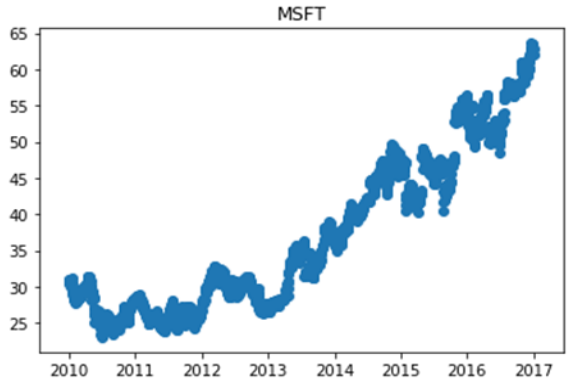
date	open	low	high	close	volume
2010-01-04	30.620001	30.590000	31.100000	30.950001	38409100.0
2010-01-05	30.850000	30.639999	31.100000	30.959999	49749600.0
2010-01-06	30.879999	30.520000	31.080000	30.770000	58182400.0
2010-01-07	30.629999	30.190001	30.700001	30.450001	50559700.0
2010-01-08	30.280001	30.240000	30.879999	30.660000	51197400.0
2010-01-11	30.709999	30.120001	30.760000	30.270000	68754700.0
2010-01-12	30.150000	29.910000	30.400000	30.070000	65912100.0
2010-01-13	30.260000	30.010000	30.520000	30.350000	51863500.0
2010-01-14	30.309999	30.260000	31.100000	30.959999	63228100.0
2010-01-15	31.080000	30.709999	31.240000	30.860001	79913200.0
2010-01-19	30.750000	30.680000	31.240000	31.100000	46575700.0
2010-01-20	30.809999	30.309999	30.940001	30.590000	54849500.0
2010-01-21	30.610001	30.000000	30.719999	30.010000	73086700.0
2010-01-22	30.000000	28.840000	30.200001	28.959999	102004600.0
2010-01-25	29.240000	29.100000	29.660000	29.320000	63373000.0
2010-01-26	29.200001	29.090000	29.850000	29.500000	66639900.0
2010-01-27	29.350000	29.020000	29.820000	29.670000	63949500.0
2010-01-28	29.840000	28.889999	29.870001	29.160000	117513700.0
2010-01-29	29.900000	27.660000	29.920000	28.180000	193888500.0
2010-02-01	28.389999	27.920000	28.480000	28.410000	85931100.0
2010-02-02	28.370001	28.139999	28.500000	28.459999	54413700.0
2010-02-03	28.260000	28.120001	28.790001	28.629999	61397900.0
2010-02-04	28.379999	27.809999	28.500000	27.840000	77850000.0
2010-02-05	28.000000	27.570000	28.280001	28.020000	80960100.0
2010-02-08	28.010000	27.570000	28.080000	27.719999	52820600.0
2010-02-09	27.969999	27.750000	28.340000	28.010000	59195800.0
2010-02-10	28.030001	27.840000	28.240000	27.990000	48591300.0
2010-02-11	27.930000	27.700001	28.400000	28.120001	65993700.0
2010-02-12	27.809999	27.580000	28.059999	27.930000	81117200.0
2010-02-16	28.129999	28.020000	28.370001	28.350000	51935600.0
...	...	...	...	...	...

Output

5.1 MSFT

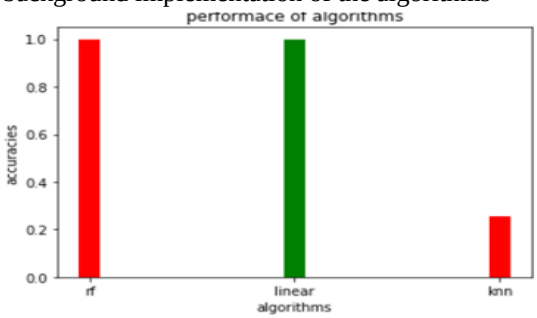
It is used to specify, visualize, modify, construct and document the artifacts of an object-oriented software

intensive system under development. UML offers a standard way to visualize a system's architectural blueprints, including elements.



5.2 Algorithm Perfomance

It is the final output for the three algorithms supported at front end anthe remaining two are helps at background implementation of the algorithms



VI.CONCLUSION

The Supervised machine learning algorithms SVM, Random Forest, KNN, Naive Bayes Algorithm, and Softmax Algorithm have been applied for the stock price prediction. The results reveal that for large dataset, Random Forest Algorithm outperforms all the other algorithms in terms of accuracy and when the size of the dataset is reduced to almost half of the original, thenNaïve Bayes Algorithm shows the best results in terms of accuracy. Also, reduction in the number of technical indicators reduces the accuracy of each algorithm in predicting the stock market trends.

REFERENCES

[1] G. Zhang, B.E. Patuwo, M.Y. Hu, Forecasting with arti-cial neural networks: the state of the, Int. J. Forecasting 14 (1998) 35–62.

[2] K. Kim, Financial time series forecasting using support vector machines. Neurocomputing, 55, pp. 307–319, 2003.

[3] T. Manojlović* and I. Štajduhar*, Predicting Stock Market Trends Using Random Forest: A Sample of the Zagreb Stock Exchange, IEEE International Convention, pp. 1189- 1193, 2015.

[4] Yuqing Dai, Yuning Zhang, Machine Learning in Stock Price Trend Forecasting, 2013.

[5] Steven B. Achelis, “Technical Analysis from A to Z”, 2nd ed., McGraw-Hill Education, 2000.

[6] Koosha Golmohammadi, Osmar R. Zaiane and David Díaz, Detecting Stock Market Manipulation using Supervised Learning Algorithms, IEEE International Conference on Data Science and Advanced Analytics, pp. 435-441, 2014.