TIME SERIES ANALYSIS

by Santhi K _

Submission date: 13-Jan-2023 11:20AM (UTC+0530)
Submission ID: 1992156819
File name: time_seires.pdf (324.13K)
Word count: 2378
Character count: 12505

TIME SERIES ANALYSIS

Arjun Mitra[a], Nehal Kanwar[b] and Richa Rupera[c]
Guided By: Dr. Santhi. K
Department of MCA – SITE (School of Information Technology and Engineering) Vellore Institute of Technology Vellore (632014), Tamil Nadu, India
E-mail: arjun.mitra2022@vitstudent.ac.in[a], nehal.kanwar2022@vitstudent.ac.in[b]
richa.rupera2022@vitstudent.ac.in[c]

ABSTRACT

The World is currently running on prediction, be it the weather, any natural calamities or the stock market. In every industry, prediction plays a major part. The more accurate the prediction the more successful the industry. The key to an accurate prediction is ***time series analysis.*** Any time series is complex, ever changing and notoriously difficult to predict. Factors like economic indicators, political events, natural disasters, etc. can affect time series data like share prices, currency values, oil prices, tourism market etc. Time series forecasting is generally preferred i.e. the process of predicting future movements in any given period of any data. The main objective of our paper is to merge suitable forecasting methods, which may give accurate short term, medium term or long-term results. Further, we will compare the merged method with the independent methods.

Therefore, we will try to establish why the merged method may be better than the independent methods.

Key words: ***time series, forecasting, prediction, trends, caret.***

Detailed design and proposed work

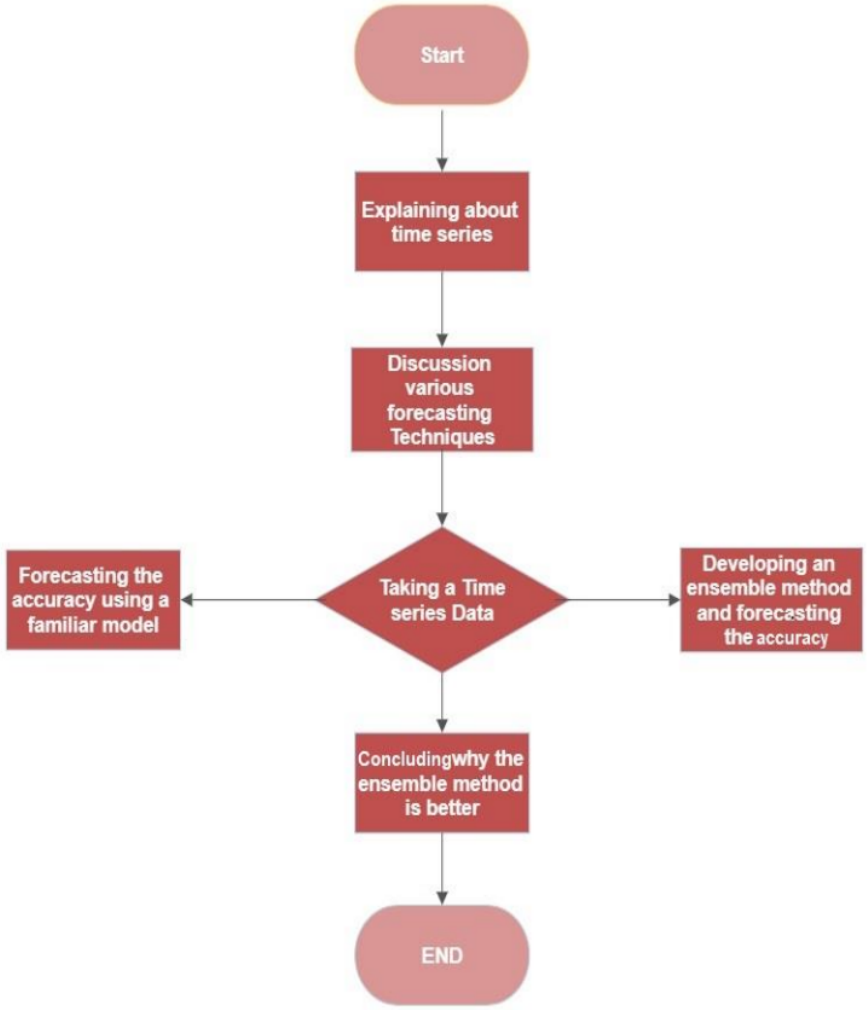

Fig. 1 - The above flowchart represents the architectural design of our work.

Introduction

Time Series

Observations are obtained from any set of data through frequent measurement in various time intervals.

For an instance, measuring the amount of rainfall each month of the year would comprise a specific time series. This is because data is well defined and accounted for at equal time intervals.

The Features of time series

Trends- The time series data has a visible pattern over time known as a trend, which can be upwards or downwards depending on the increasing or decreasing values over the period.

Seasonality- A series is affected by seasonal influences when the values of the data are remarkably alike within a certain season or period within a given time period. This phenomenon is known as seasonality. Seasonality always has a set, recognizable time frame.

Non-Seasonality- It is said when the data does not have many similarities and has different trends in different times.

Steps- We refer to any abrupt changes in the values occurring for a disproportionately long period of time or permanently as a step in a time series of observations.

Pulses- Time series observations that experience any abrupt changes in the values for brief periods of time or temporarily are referred to as pulses in the time series.

Steps to analyze Time Series Data

1) Collecting raw data and ordering it.
2) Preparing Visualization with respect to different time intervals.
3) Observing the stationary of the particular data series, we have made.
4) Developing charts and graphs to understand and determine its nature.
5) The final step is the model building of the data. Some of the common models are Naive model, Moving Average model, Linear Regression Model, etc.

Prediction - An act of saying what will or might happen in the future.

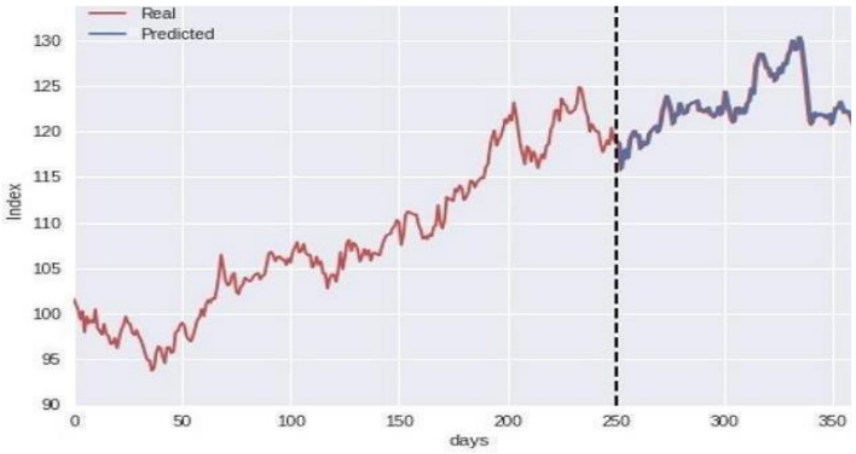

Figure 2 - A sample graph representing real and predicted data.

Forecasting

Forecasting is an important method for making decisions. Forecasting helps to anticipate trends in important business indicators, such as sales expectations or revenue. Forecasting is no ordinary asset but it requires to the point skill set and accurate data. Forecasting is a procedure of predicting by using previous and present data as major input to get the pattern of visionary trends. People use forecasting for many various motives, such as net sales, gross profit and net profit.

Time series forecasting

Making statistical predictions based on historical as well as current data. It involves producing data predictive models by historically studying data and implementing these procedures to make observations and build upcoming strategies. Time series forecasting is the procedure of analyzing time series data by using the available statistical modelling methods. It is not always an accurate prediction because we deal with data consisting of frequently changing variables along with components over which we have no control and knowledge. “Forecasting” and “prediction” has a notable difference. Forecasting might allure to data at a certain point of time in future, on the other hand prediction allure to futuristic data. Developing a model to gain an understanding of the data along with its underlying causes is the motive.

Statistical techniques for time series forecasting

- Naive()- In Naïve forecasting, the upcoming period's forecast is based on the previous period's sales. Initially predictions or adjusting the factors is not done here. Forecasted values using a naïve approach are equal to the final observed values. It can accurately predict situations, while others can be problematic because it considers only the previous period to forecast the next period. Thus, historical data is the main requirement for naïve forecasting and factors such as seasonality is not taken into account. Salespeople will use naïve forecasting to make targets and objectives as a way to make sure they are always either improving their contribution to the company. [5]

- SNaive () - Similar to the naïve forecasting method, there exists Seasonal naïve forecasting but its anticipation is based on the preceding values from the same season itself. The exponential smoothing method employs decreasing weights for forecasts but ignores seasonal components. [5]

- Treebag() - Bagging algorithm is used to increase accuracy of the model while dealing with regression and classification problems. Classification with a bagging (treebag) method in R. The fundamental idea behind bagging algorithms is to build multiple models from distinct subsets of train data before creating a final aggregated and more precise model.

- Regression () - Regression analysis is one of the most popular and straightforward techniques used when a correlation between the predicted value and other variables is desired. It is assumed that the data to be predicted are split into linear data and standard base data depending on the variables influencing the data. [3]

- Linear and non-linear models () – It represents the relationship between two variables with a steady rate of change through equations is called a linear model. A nonlinear model describes an experimental dataset's nonlinear relationships. These models, where the model is expressed as a nonlinear equation, are typically assumed parametric. [1]

- Ses()- Simple Exponential Smoothening is generally applied on the data with no specific trend or seasonal pattern. We know that in any of the exponential smoothing methods we weigh the recent values or observations more heavily rather than the old values or observations. The weight of each and every parameter is always determined by a smoothing parameter. When alpha is closer to 0 then it is considered as slow learning since the algorithm is giving more weight to the historical data. If the value of alpha is closer to 1 then it is referred to as fast learning since the algorithm is giving the recent observations or data more weight. The value of alpha lies between 0 and 1. In practice, if alpha is between 0.1 and 0.2 then SES will perform quite well. Hence, we can say that the recent changes in the values will be leaving a great impact on the forecasted values. [2]

- MA ()- Moving Average (MA) is obtained from some data's average for a specific timeframe t. In normal mean, its value gets changed with the changing data but in this type of mean it also changes with the time interval. We get the mean for some period t and then we remove some previous data. Again, we get new mean and this loop continues. This has a great application in the share market. [6]

- Hw()- when dealing with data of both seasonal patterns and trends, Holt-Winter's Seasonal method is used. This method can be implemented either by using Additive structure or by using the Multiplicative structure, depends on data to data. The Additive structure or model is used when the seasonal pattern of data has the same magnitude or is consistent throughout, while the Multiplicative structure or model is used if the magnitude of the seasonal pattern of the data increases over time. It uses three smoothing parameters, - alpha, beta, and gamma. [2]

- Rpart() - Building classification and regression trees uses Rpart, a machine learning library with many robust features in R. Recursive partitioning is implemented by this library, which is very easy to use and flexible.

- GBM()- The full form is 'Gradient Boosting Machine ', merges the predictions from various decision trees to create ultimate results, In a gradient boosting machine, the decision tree represents the total number of weak learners.

Dataset

'Vehicles' data set is obtained through Hadley Wickham's GitHub. [7] The data contains details of different components of various vehicles.

Experimental Procedure

We used 'ensembling' in the software named 'R Studio' and we did so by using 'caret' packages. An ensemble briefly means stacking models together. We can model a data set with one model and it will have an understanding of the data, but if we use different models especially diverse models that will approach the data differently. As a result, one will have a wider understanding of the dataset and especially for a complex dataset, it will increase the accuracy theoretically and stability over predictions over the long run. The packages we used are : '{caret}' ,'{RCurl}' and '{pROC}'.

We used the caret library since it offers a unified language with functions and commands that generalized, so that we can talk to variety of models using the same constructs. We then used the 'getModelnfo()' function of the caret library which displays all the models that are supported.

```
> library(caret)
> names(getModelInfo())
  [1] "ada"                  "AdaBag"               "AdaBoost.M1"          "adaboost"             "amdai"                "ANFIS"                "avNNet"
  [8] "awnb"                 "awtan"                "bag"                  "bagEarth"             "bagEarthGCV"          "bagFDA"               "bagFDAGCV"
 [15] "bam"                  "bartMachine"          "bayesglm"             "binda"                "blackboost"           "blasso"               "blassoAveraged"
 [22] "bridge"               "brnn"                 "BstLm"                "bstSm"                "bstTree"              "C5.0"                 "C5.0Cost"
 [29] "C5.0Rules"            "C5.0Tree"             "cforest"              "chaid"                "CSimca"               "ctree"                "ctree2"
 [36] "cubist"               "dda"                  "deepboost"            "DENFIS"               "dnn"                  "dwdLinear"            "dwdPoly"
 [43] "dwdRadial"            "earth"                "elm"                  "enet"                 "evtree"               "extraTrees"           "fda"
 [50] "FH.GBML"              "FIR.DM"               "foba"                 "FRBCS.CHI"            "FRBCS.W"              "FS.HGD"               "gam"
 [57] "gamboost"             "gamLoess"             "gamSpline"            "gaussprLinear"        "gaussprPoly"          "gaussprRadial"        "gbm_h2o"
 [64] "gbm"                  "gcvEarth"             "GFS.FR.MOGUL"         "GFS.LT.RS"            "GFS.THRIFT"           "glm.nb"               "glm"
 [71] "glmboost"             "glmnet_h2o"           "glmnet"               "glmStepAIC"           "gpls"                 "hda"                  "hdda"
 [78] "hdrda"                "HYFIS"                "icr"                  "J48"                  "JRip"                 "kernelpls"            "kknn"
 [85] "knn"                  "krlsPoly"             "krlsRadial"           "lars"                 "lars2"                "lasso"                "lda"
 [92] "lda2"                 "leapBackward"         "leapForward"          "leapSeq"              "Linda"                "lm"                   "lmStepAIC"
 [99] "LMT"                  "loclda"               "logicBag"             "LogitBoost"           "logreg"               "lssvmLinear"          "lssvmPoly"
[106] "lssvmRadial"          "lvq"                  "M5"                   "M5Rules"              "manb"                 "mda"                  "Mlda"
[113] "mlp"                  "mlpKerasDecay"        "mlpKerasDecayCost"    "mlpKerasDropout"      "mlpKerasDropoutCost"  "mlpML"                "mlpSGD"
[120] "mlpweightDecay"       "mlpweightDecayML"     "monmlp"               "msaenet"              "multinom"             "mxnet"                "mxnetAdam"
[127] "naive_bayes"          "nb"                   "nbDiscrete"           "nbSearch"             "neuralnet"            "nnet"                 "nnls"
[134] "nodeHarvest"          "null"                 "OneR"                 "ordinalNet"           "ordinalRF"            "ORFlog"               "ORFpls"
[141] "ORFridge"             "ORFsvm"               "ownn"                 "pam"                  "parRF"                "PART"                 "partDSA"
[148] "pcaNNet"              "pcr"                  "pda"                  "pda2"                 "penalized"            "PenalizedLDA"         "plr"
[155] "pls"                  "plsRglm"              "polr"                 "ppr"                  "pre"                  "PRIM"                 "protoclass"
[162] "qda"                  "QdaCov"               "qrf"                  "qrnn"                 "randomGLM"            "ranger"               "rbf"
[169] "rbfDDA"               "Rborist"              "rda"                  "regLogistic"          "relaxo"               "rf"                   "rFerns"
[176] "RFlda"                "rfRules"              "ridge"                "rlda"                 "rlm"                  "rmda"                 "rocc"
[183] "rotationForest"       "rotationForestCp"     "rpart"                "rpart1SE"             "rpart2"               "rpartCost"            "rpartScore"
[190] "rqlasso"              "rqnc"                 "RRF"                  "RRFglobal"            "rrlda"                "RSimca"               "rvmLinear"
[197] "rvmPoly"              "rvmRadial"            "SBC"                  "sda"                  "sdwd"                 "simpls"               "SLAVE"
[204] "slda"                 "smda"                 "snn"                  "sparseLDA"            "spikeslab"            "spls"                 "stepLDA"
[211] "stepQDA"              "superpc"              "svmBoundrangeString"  "svmExpoString"        "svmLinear"            "svmLinear2"           "svmLinear3"
[218] "svmLinearWeights"     "svmLinearWeights2"    "svmPoly"              "svmRadial"            "svmRadialCost"        "svmRadialSigma"       "svmRadialWeights"
[225] "svmSpectrumString"    "tan"                  "tanSearch"            "treebag"              "vbmpRadial"           "vglmAdjCat"           "vglmContRatio"
[232] "vglmCumulative"       "widekernelpls"        "WM"                   "wsrf"                 "xgbDART"              "xgbLinear"            "xgbTree"
[239] "xyf"
> library(RCurl)
```

Fig 3 - Introducing the 'Rcurl' package and 'caret' package with all the supported models.

The vehicle dataset basically talks about energy consumption of different vehicles. We took the first 30 columns from vehicle dataset and a variable cylinder where 6 cylinders is considered to be 1 and any other number of cylinders is considered to be 0. This acted as the target variable or response variable. All the other columns were used to predict the number of cylinders. In order to make the ensemble model work, we divided the data equally into three parts which are named 'ensembleData', 'blenderData' and 'testingData'.

```
> vehicles[is.na(vehicles)] <- 0
> vehicles$cylinders <- ifelse(vehicles$cylinders == 6, 1,0)
> prop.table(table(vehicles$cylinders))

        0         1
0.6505732 0.3494268
> # shuffle and split the data into three parts
> set.seed(1234)
> vehicles <- vehicles[sample(nrow(vehicles)),]
> split <- floor(nrow(vehicles)/3)
> ensembleData <- vehicles[0:split,]
> blenderData <- vehicles[(split+1):(split*2),]
> testingData <- vehicles[(split*2+1):nrow(vehicles),]
> # set label name and predictors
> labelName <- 'cylinders'
> predictors <- names(ensembleData)[names(ensembleData) != labelName]
> library(caret)
> # create a caret control object to control the number of cross-validations performed
> myControl <- trainControl(method='cv', number=3, returnResamp='none')
> # quick benchmark model
> test_model <- train(blenderData[,predictors], blenderData[,labelName], method='gbm', trControl=myControl)
```

Fig 4 - Representation of the division of data into three parts.

Later, we used a label name – 'cylinders' and a predictor variable to hold the variables into it. Further on, we started with calling our first caret function 'trainControl()'. The 'trainControl()' works with the classic train function which is present in the model functions in R. It instructs the train function to find the best parameters for the model. After this step, we trained all the ensemble models for which we used the dataset which we created initially. In order to execute this step, we used 'ensembleData' after which we trained the 'gbm' model, the 'rpart' model and the 'treebag' model. This was the ensemble stage where we selected three models and trained them using our dataset. This gave us the stability to predict the other two datasets and so we used these three models that we trained to predict blender dataset as well as the testing dataset. After successfully training the three models 'gdm', 'rpart' and 'treebag' to recognize a vehicle with six cylinders, we used each one of our models on the two datasets. Finally, we harvested the probabilities, adding them back to the original

datasets, thereby adding new columns to both the datasets. Then, we used our last used model, 'gbm', to train the enhanced blended data. Following this step, we predicted the area under the curve for the enhanced model.

Results and Discussions

As a result, we got the area under curve value as 0.9959. Then, we repeated the same for the original set of data and predicted the area under the curve.

```
> # See final prediction and AUC of blended ensemble
> preds <- predict(object=final_blender_model, testingData[,predictors])
> auc <- roc(testingData[,labelName], preds)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(auc$auc)
Area under the curve: 0.9959
```

Fig 5 - Prediction of Area under the Curve for blended ensemble.

We found the predicted value to be 0.9924. The model on its own 'gbm' with the original data gives a 0.9924 while the model enhanced by adding probabilities from diverse models gives us a slightly higher area under the curve. We managed to increase the value of the area under the curve with the use of three models. We got the benefit of an additional extra model as well as an extra way of analyzing the data.

```
> preds <- predict(object=test_model, testingData[,predictors])
> library(pROC)
> auc <- roc(testingData[,labelName], preds)
Setting levels: control = 0, case = 1
Setting direction: controls < cases
> print(auc$auc)
Area under the curve: 0.9924
```

Fig 6 - Predicting Area under the Curve for initial blended data.

Conclusion

This paper represents an empirical study on the basics of time series, forecasting and time series forecasting techniques. We have tried to establish why using multiple methods together can be much more helpful as well as accurate for forecasting. Upon altering the initial blender dataset and testing dataset by adding additional features to the original datasets, we successfully built the new or enhanced dataset which consisted of three new additional columns of probabilities. Further, we predicted the area under the curve for both the datasets. The value for the enhanced blender data was predicted as 0.9959. Similarly, the value for area under the curve for the original blender dataset was predicted as 0.9924. Thereby we can conclude that by using ensemble we increased the expected value by 0.0035.

References

1) Stock, J. H., & Watson, M. W. (1998). A comparison of linear and nonlinear univariate models for forecasting macroeconomic time series.

2) Mahalakshmi, G., Sridevi, S., & Rajaram, S. (2016, January).

A survey on forecasting of time series data. In 2016 International Conference on Computing Technologies and Intelligent Data Engineering (ICCTIDE'16) (pp. 1-8). IEEE.

3) Athiyarath, S., Paul, M., & Krishnaswamy, S. (2020). A comparative study and analysis of time series forecasting techniques. SN Computer Science, 1(3), 1-7.

4) Semenoglou, A. A., Spiliotis, E., Makridakis, S., & Assimakopoulos, V. (2021). Investigating the accuracy of cross-learning time series forecasting methods. International Journal of Forecasting, 37(3), 1072-1084.

5) Syu, Y., Kuo, J. Y., & Fanjiang, Y. Y. (2017). Time series forecasting for dynamic quality of web services: an empirical study. Journal of Systems and Software, 134, 279-303.

6) Deb, C., Zhang, F., Yang, J., Lee, S. E., & Shah, K. W. (2017). A review on time series forecasting techniques for building energy consumption. Renewable and Sustainable Energy Reviews, 74, 902-924.

7) Hadley Wickham (2008). Details of different components of various vehicles. https://github.com/hadley

TIME SERIES ANALYSIS

ORIGINALITY REPORT

1%	1%	1%	%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Othman Istaiteh, Tala Owais, Nailah Al-Madi, Saleh Abu-Soud. "Machine Learning Approaches for COVID-19 Forecasting", 2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA), 2020 Publication	1%
2	www2.mdpi.com Internet Source	1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		