Time Series Modelling and Forecasting with Applications in R

Prof. Sudeep Bapat

Shailesh J. Mehta School of Management

Indian Institute of Technology Bombay

Department of Applied Statistics/Finance

Week 12

Lecture 57: Linear Regression for Time Series and Beyond

Hello all, welcome to this course on Time Series Modeling and Forecasting using R. So, now we are in the last week, and as discussed briefly in the last session, the broad idea we are covering this week is the integration of machine learning into time series models. The entire last session was more from an introductory perspective, where I gave you a flavor of different ML techniques that are deployed and what models could be integrated into a practical time series setting. Again, if you remember, we started by discussing regression or classification models, decision trees, random forests, gradient boosting, and support vector machines. The other idea we discussed was neural networks, right?

So, RNN, CNN, LSTM, etc. Then we talked about a few other advanced ML techniques or deep learning techniques. Again, if you remember, some examples pertained particularly to time series, like the Facebook Prophet model and all of that. So, I think now is the time to explore in a bit more detail at least the well-known ones. Let us say linear regression, decision trees, random forests, etc.

Along with some examples, so you know what exactly the integration of some ML models with a time series setting could be, okay? But just before that, I will talk very briefly about the advantages of classical ML models and, at the same time, the cons or limitations of classical ML models, okay? So, let us start with the advantages. So, the first advantage is speed, of course. Fast training and prediction are suitable for low-resource environments, okay?

So, all the ML models have the capability of training different ML models very fast, which is again helpful for, let us say, forecasting or prediction, etc., and suitable for low-resource environments. So, if you do not require very high computing facilities or capabilities, and then on the resource front, it is on the lower end, then this is one

advantage of the classical ML models, right. Now, again, I will pause here and say that classical ML models are some of the basic ML models. So, regression, classification, decision trees, random forest, etc. So, all these models do not require a very high-resource environment, unlike neural networks or deep learning techniques.

The second one is interpretability. So, models like decision trees and linear regression are straightforward to explain. So, one can actually explain to any layman what is going on inside the model. Third is flexibility, of course. So, they can handle diverse tasks like classification, regression, clustering, etc.

So, when it comes to classical ML models, these are some of the advantages and why one should apply or use some of the classical ML models. So, for speed, interpretability, flexibility, etc. But on the other hand, of course, what are the challenges or limitations of all the classical ML models? The first one is feature engineering dependency. So, performance hinges on the quality of features.

So feature engineering is a very very important idea where feature engineering is a task where one chooses the appropriate features. So again as discussed in the last session also that what are my features. So features are nothing but all the variables or all the inputs that are feeded inside the ML model. So of course one should choose the appropriate features very very carefully. And then this step is called as feature engineering.

And then in general, feature engineering dependency is always there. So performance hinges on quality of all the features. Then second is data stationarity. So, many models assume that the data is stationary requiring pre-processing steps like differencing etcetera right. So, again all these classical ML models require the idea about stationarity.

So, if the model is not stationary then one has to difference it number of times one has to transform it a number of times etcetera. So, many models assume that the data is stationary requiring pre-processing steps like differencing etcetera right. And third limitation is scalability. Some models like KNN, etc. struggle with very large datasets.

So, by the way, KNN is K-nearest neighbors, right? Some models like KNN struggle if the datasets go beyond a certain limit or for very large datasets. So, at the same time, these are some challenges when it comes to classical ML models. So, again, the takeaway point from both these slides is that one should always find the suitable model and try to balance the advantages and the limitations of that classical ML model, okay? So, now the next thing we will discuss is linear regression for time series and beyond.

So, we will start with linear regression. Again, try to understand how a regression model could be suited to bring in the time series data or practical time series data and beyond, meaning some of the other advanced ML models, and then we will go a bit deeper into each one of them. Alright, so we will start with a bit of an introduction about the linear regression setting in general. So, the linear regression model is the relationship between a dependent variable, let us say yt in this case, and one or more independent variables xt by assuming linearity as follows. So, I think this slide is just a very basic introduction to linear regression.

A usual sort of, or rather a standard, linear regression model. So, let us say you have yt on one hand, which is equal to beta naught plus beta 1 xt minus 1 plus beta 2 xt minus 2, and so on and so forth, plus a random error which is et. So, on the left-hand side, you have the dependent variable, which is yt; on the other hand, you have all the independent inputs, let us say xt minus 1, xt minus 2, etc., right, as the independent variables, which are xt's, okay? And et is a usual random error component. Note that the independent variables are time lags of x itself. And again, one can actually change this to y generally, so that each one of xt minus 1, xt minus 2 actually becomes a lagged variable of y itself, generally, right?

$$y_t = \beta_0 + \beta_1 x_{t-1} + \beta_2 x_{t-2} + \cdots + e_t,$$

So, yt minus 1, yt minus 2. So, broadly speaking what is going on is that you are trying to predict the current state which is yt using all its historical past or all its lagged values right. and a regression sort of a setting. So, y t minus 1, y t minus 2, y t minus 3, etcetera plus e t. And of course, all these coefficients, so beta naught, beta 1, beta 2 have to be estimated suitably, ok. So, just a bit of an introduction as to how can you integrate the idea of linear regression into a time series or a temporal sort of a setting.

Now, what are the applications? So, let us say trend estimation. So, one can use linear regression for very very easily to estimate the trend. Now, of course, one very important point is since the name itself contains linear. So, you cannot actually handle some of the complex patterns or some of the non-linear patterns.

So, trend estimation. So, if you have a linear trend sort of let us say the time series is behaving like that for example. Now, again seasonality be it there or may not be there say does not does not matter, but let us say if you have a trend which looks something like

that one might as well try to fit a linear line which sort of superimposes on the practical setting here and then if you have a linear line then why not explain using a linear regression basically. Then the second application could be capturing some seasonal effects via dummy variables or engineered features. So one can actually capture the seasonality in the data set as well by bringing the idea about some other dummy variables or some other engineered inputs or engineered features, etc.

And third very, very important application is that all the linear regression models in general form the baseline models for further forecasting. So, even as discussed in the last session that linear regression stands at the very basic level and then all the other advanced ML models follow by the way, ok. So, one always has to start with a linear regression model try to explain if not linearity then one has to sort of move to a non-linear setting and then try to let us say use some decision trees ideas or gradient boosting, XG boost, etc., But one very important application of a linear regression sort of a setting and integration to time series is that it follows the or forms the baseline models for forecasting usually. So, I think this one slide would give you some idea about what do you mean by this term called as feature engineering.

So, again feature engineering means that how do you choose the features carefully or how do you pick the features carefully. Or how do you sort of engineer all the inputs or engineer all the features? Okay. So, this entire exercise, this entire idea is called as feature engineering, which is a very, very important part of any ML exercise or any ML technique in general. So, the first one is called as lagged features.

So, what do you mean by that? So, lagged features means its own lagged values as seen even before. So, the first point could be create lagged versions of the target variable itself. So, if the target variable or the dependent variable is yt, then why not have all the features which are nothing but the lagged versions of the target variable itself. Let us say yt minus 1, yt minus 2, etc.

One small example in this regard could be, let us say, if you want to forecast the temperature, right? So, if y t is a temperature, then one can actually involve features like its own lag values, let us say y t minus 1, which simply denotes yesterday's temperature, right. So, what you are forming or what you are having as inputs for the ML model are nothing but the prior or historical values of the target variable itself. So, if y t is the current temperature, then one can actually use yesterday's temperature, day before yesterday's temperature, etc. as inputs in the ML model.

Now, the second kind of inputs could be called rolling statistics. So, rolling statistics means one very widely applicable example is moving averages, right. So, again, the idea of moving average—I will give you a small example. Let us say if you have y t and then y t minus 1, y t minus 2, y t minus 3, and then let us say y t minus 4, ok. So, what one can do is one can form moving averages of 3

in a category or 3 in a bunch. So, let us say y t minus 4, y t minus 3, y t minus 2. Take the average of these 3 and present some value. Now, let us say I will give a 1. So, a 1 is the first average, and then you roll down the window, then you move the window, and then the next 3 would be let us say y t minus 1, y t minus 2, and y t minus 3, right, and then you ignore the last one.

So, in every bunch, you pick 3 and take the average of that. So, this one would be a 2, and lastly, I can have y t, y t minus 1, y t minus 2, and this could be my a 3. So, what you are doing is you are sort of taking the averages, and you are also moving the window, rolling the window, or rather shifting the window. So, all such techniques are called rolling statistics because what you are having is different statistics, which are nothing but averages here, on a rolling basis. So, one can add some moving averages or rolling standard deviations as predictors.

So, if not averages, let us say if you want to take standard deviations. So, I can do that, right? I mean, let us say again, if you have three in a bunch. So, rather than finding the averages, I can find out variances, I can find out any other statistics I want, say standard deviations, and again try to roll down the window. So, all these individual statistics, which are collected on a rolling basis, are called rolling statistics. So, why not add some of the rolling statistics as features or as the inputs, isn't it?

So, I can add A1, A2, A3 as the inputs in the ML model or as the features in the ML model. So, the first example we saw was rolling or other lagged features, where the lagged versions of the target variable itself are fed as inputs. On the other hand, I can take any statistic I want, let us say averages or standard deviations, on a rolling basis and then input those as features. So, one can add some moving averages or rolling standard deviations as predictors. One example could be, let us say, use the average of the past 7 days to predict, let us say, the rainfall of the next day or the temperature of the next day, etcetera.

So, this is a classical example of let us say the last 7 days moving average basically. And thirdly, the features could be of seasonality indicator. So, let us say encode day of the

week, month or holiday information as some dummy variables. So, one example I can give you is let us say holiday information. So, whether or not it is a holiday, let us say 1 or 0.

So, I can code the variable as let us say 1 or 0. So, 1 means yesterday was a holiday, let us say. And 0 means yesterday was not holiday. And then you use all these indicators 1, 0 as features. And hence they're called as seasonality indicators.

So all these coded variables using let us say 01 or some other technique are indicator variables and these could be served as features for the ML model. So, let us say day of the week or day of the month or which month is it in the year or holiday information. So, I can actually code all these individually using some 01 sort of a technique and then feed all these predictors as inputs in the ML model. So, the idea broadly speaking the idea is that how do you pick or how do you curate the correct features or curate the correct predictors right or curate the correct inputs into the ML model is important.

And this entire exercise is called feature engineering in ML literature. Now, the advantages. So, the advantages of feature engineering, or let us say in general, how do you curate the features? It is interpretability. So, coefficients provide clear insights into the influence of predictors, right? And second could be efficiency.

So, it is computationally inexpensive for large datasets. By the way, these are advantages for linear regression, okay. So, once you have chosen linear regression, once you have curated the features, done the feature engineering exercise, what are the advantages of linear regression? First is interpretability, that all the estimated coefficients provide clearer insights into the influence of predictors. So, one can actually see how each and every predictor or how each and every variable influences the overall output or influences the overall model.

And second is efficiency. So, it is computationally inexpensive even for large datasets, right? I mean, the entire model is explainable. So, it's quite not a black-box kind of model per se, right? So, it is computationally inexpensive; you don't require any high resources, right?

Even if the dataset goes outside a certain limit, you can still fit a linear regression very, very cheaply. But again, at the same time, what are the limitations of linear regression? Of course, the first very important limitation is assumed linearity, which may not hold in complex time series examples. So, if you have a really complex sort of setting, then

again, of course, as discussed even in the last session or a short while back, that linearity might go for a toss, and then you have to go with some of the advanced ML models or deep learning techniques, etc. Second, any of the linear regression exercises or models are really sensitive to multicollinearity between features, for example, overlapping lags.

So, the idea about multicollinearity is what? So, multicollinearity means that, let us say, if any of the predictors—so let us say x1 and x2 are two predictors—and these are highly correlated. Then we have a situation which is called multicollinearity, right? So, you have y on one hand, which is the response variable. And let us say any of the two predictors are highly correlated, then that can pose a problem.

Because, let us say, I will give you an example. Let us say y could be the age of a person. And let us say x1 could be the weight of a person. And x2 could be, let us say, the height of a person. Okay, and the exercises depending on or based on the weight of the person and the height of the person, how do you sort of predict the age of the person, right?

But in general, weight and height might show some high correlation, right? I mean, let us say if weight is more or height is more, then weight could be more, something like that. So, one can actually assume some high correlation between the weight and height features, and then this problem is called multicollinearity, okay? So, linear regression is quite sensitive to multicollinearity between features, and one has to take care of that. And thirdly, as discussed in the first point here, it cannot model any non-linear relationships effectively.

So, of course, linear regression cannot be used to model any non-linearity in the time series structure. So, even though the linear regression model is easy to understand, easy to deploy, and so on, it comes with its own limitations. A small example: let us say forecasting stock prices. So, what is the objective? So, the objective is to predict the next day's closing price of a stock based on some historical data.

So, let us say if you want to predict the next day's closing price of a particular stock based on some past or historical data, I can actually use a linear regression setting. Now, what could be the features that go into the model? So, some of the features would be, let us say, previous closing prices. So, as discussed a couple of slides back, one can actually use some lagged variables or lagged values of the variable itself. So, y t minus 1, y t minus 2, y t minus 3, etc.

So, some of the previous closing prices would be used as features in that linear regression sort of a setting or I can use some rolling averages which are called as rolling statistics. Let us say 5 day or 10 day moving averages. Or even otherwise, I can have some other features, let us say volume. So, for forecasting stock price of a company, I can actually have the volume of the stock as an important predictor also. So, volume traded or other indicators like let us say RSI or relative strength index or MACD, which is called as moving average convergence divergence, etc.,

So, all these are suggested features. So, previous closing prices, rolling averages, volumes or some indicator based values from let us say RSI, MACD, etc. And then lastly implementation. So, how do you implement the linear regression model? So, create the lag variables and rolling statistics and keep them ready.

Train a linear regression model using these features to predict the next closing price. So the objective is very simple here. The objective is to predict the next day's closing price of a stock, let us say Reliance or Adani, whatever, based on some past values or historical data. How do you eventually implement the model? So again, once you formulate or rather engineer the features or once you have the features ready, then create the linear regression model and then try to estimate all the unknown parameters in the model and then try to forecast the next day's closing price.

Now, the next sort of terminology or rather methodology in ML is support vector machines, or in short, SVM. So, what exactly is the idea in SVM? So, SVMs aim to find a hyperplane. So, again, this is a very typical sort of terminology pertaining to SVM. So, SVMs aim to find a hyperplane.

that best separates data in feature space for classification or fits the data within a margin of tolerance for regression, known as support vector regression. So, again, SVM could be used in both ideas. So, regression problems and classification problems, okay. But the main idea in SVM is that you have to find out a hyperplane. So, a hyperplane is nothing but, let us say, a plane which separates the data into two different categories, right?

So, let us say if you have a classification problem—yes versus no, red versus blue—then it sort of does something like this. Let us say it creates a hyperplane. On one side of the hyperplane, you might have all the red dots. On the other side of the hyperplane, you might have all the blue dots, right? So, let us say these are blues here, okay?

And how effectively do you sort of find the hyperplane that divides all the red circles with the blue pointers is important. And this is called a support vector machine. So, the hyperplane itself is called a support vector. So, the support vector is nothing but a vector or a hyperplane that separates all the data points into categories. So, if you have a classification problem, then finding a hyperplane that distinguishes between the red circles and the blue pointers is important.

Or on the other hand, if you have a regression problem, regression problem means what? That is not a classification problem, where you have some numerical inputs and a numerical response variable. Then in that case, it is called a support vector regression or in short SVR. A bit more about let us say support vector regression is optimizes the margin of tolerance. So, there is a thing called as margin of tolerance which is epsilon around the hyperplane.

So, how do you optimize the margin of tolerance around the hyperplane? And secondly penalizes points that fall outside the margin minimizing the error. So, penalizes points which fall outside the margin minimizing the error. So, this is called as a support vector regression or SVR. Secondly, what could be the applications of SVM?

So, the first one could be time series regression for forecasting, of course. I would say that SVM is an extension of the linear regression model, where again the regression sort of an application or regression sort of an idea could come into the picture here. So, time series regression for forecasting. On the other hand, unlike linear regression, SVMs can also be used for classification tasks. Let's say anomaly detection, regime change prediction—whether the regime has changed or not changed, something like that.

So, classification means putting different outputs into categories, right? So, anomaly detection—whether it's an anomaly or not an anomaly. So, something like a yes versus no kind of problem, right? Or regime change prediction—again, yes versus no. Has the shift happened or has the shift not happened?

Has the regime changed or not changed, etc., okay? So, SVMs can go a step ahead of linear regression models, both in terms of capturing regression sort of problems as well as classification tasks. Then, the third set of models are called ensemble learning models. And then, SVMs rely on kernel functions to model non-linear relationships. And then, these are called ensemble learning.

And ensemble learning means you have some kernel functions. So what exactly are some of the well-known kernel functions? First one could be linear kernel for linear relationships. Second could be polynomial kernel, which captures polynomial relationships. Our third one could be radial basis function kernel or RBF kernel, which models highly nonlinear relationships, etc.

Third one is feature engineering. So again, feature engineering is exactly as similar to what we discussed a short while back that even in SVM, I can engineer the features, right? So, again engineering the features means picking the features carefully and then the first kind of features could be lagged features as we saw earlier. So, create features like y t minus 1, y t minus 2, y t minus 3, etc., which are lagged features. Then seasonality features, encode periodic patterns, right?

Seasonality features means encode some periodic patterns or encode the seasonality. So, again, let us say if you want to categorize whether it is a holiday yesterday or not a holiday, I can use coding variables like 0, 1, etc. So, 1 would stand for holiday yesterday and 0 would stand for not a holiday yesterday. So, I can actually code all the variables suitably and then these are called as seasonal features and then involve in the ML model. Third one is normalization.

So, normalization means scale the features to ensure that the SVM works efficiently. Example, let us say min-max scaling. So, min-max scaling is one sort of a scaling which is applied to all the features to ensure that the features are scaled down. It is called a normalization process. So, normalization even means let us say standardization.

Standardization means that let us say subtract the means or divide by the standard deviation something like that. So, how do you ensure that all the features are normalized or scaled down such that the SVM works bit more efficiently? Now, what exactly are the advantages and limitations? Let us say advantages are handles non-linear relationship via kernels and the robust to outliers when using appropriate margin parameters. On the other hand, limitations.

So, computationally intensive for large data sets due to quadratic complexity. And on the other hand, requires careful tuning of hyper parameters. So, how do you choose the kernels or how do you estimate the parameters, right? So, what are the hyper parameters? Epsilon, kernel type, regularization parameters called as C, etc.

So, all these have to be handled really carefully when it comes to, let us say, the limitations of SVM in general. Now, the next one could be random forest. So, random forests are nothing but ensemble models that build multiple decision trees on random subsets of data and aggregate their predictions. Let us say, usually via averaging for regression or majority voting for classification. So, random forests are nothing but ensemble models that combine several different decision trees.

So, let us say, individually if you have multiple decision trees. And if you want to combine all these decision trees, then the category of ML model that you get is called a random forest model. Similarly, here I can have the exact same feature engineering as before. So, lag variables, rolling features, seasonality coding or seasonality features, etc. So, here I can add dummy variables for day of the week, month of the year, holidays, etc.

Again, advantages: handles non-linear relationships well and interactions naturally, robust to overfitting when many trees are used, does not require extensive pre-processing or normalization. On the other hand, what are the limitations or struggles with extrapolation? Computationally expensive for large data sets with many trees. So, if you want to have an ensemble of a really large number of trees. Then that can get out of hand.

So, it is computationally expensive for large datasets or integrating or combining many trees, and it can become less interpretable with increasing complexity. So, again, these are some advantages and limitations of Random Forest in general. Applications—what are the applications? So, time series forecasting using lag variables or engineered features.

Feature importance analysis to identify key drivers in the data. Anomaly detection by modeling residuals, etc. And then, hyperparameters. So, as discussed a short while back, what exactly do you mean by hyperparameters? So, let us say, the number of trees.

So, how do you decide on the number of trees? So, the more the number of trees, the better the robustness, but it increases the training time. Maximum depth. What do you mean by maximum depth? So, control the depth of each tree to prevent overfitting.

So, the depth of each tree means how many nodes or how many layers each tree has. And thirdly, of course, the number of features. So, determine the subset of features used for splits. So, one has to actually control all these hyperparameters carefully and then create the random forest in general. And lastly, I have a very small table called a comparative analysis of all the models discussed today.

So, linear regression, SVM, and then random forest. In terms of complexity, interpretability, non-linearity, feature engineering, computational cost, and handling large datasets. So, probably I will not read each and every cell here, but just to give an example, let's say linear regression is low in complexity. High in interpretability, but at the same time, low in computational cost, and it can actually handle some large datasets excellently. On the other hand, let us say SVM.

So, what are the features of SVM pertaining to all these categories? Let us say medium to high complexity, medium interpretability, but at the same time, it cannot handle large datasets. So, it says poor here. And lastly, random forest. So, medium complexity, low to medium interpretability, excellent non-linearity.

So, again, you have to balance around, right. So, let us say linear regression is really poor at handling non-linearity. At the same time, SVM or random forest can handle non-linearity quite well, right. And then, let us say feature engineering is very critical for linear regression, very important in SVM, but not so important in random forests. And computational cost is low for regression, high for SVM, and medium for random forest, etc.

Now, again, in the next session, we will try to delve deeper into some of the other models. And then, in the session to follow, we have a complete session on neural networks. And then, there we will see some examples of, let us say, some particular models like NMAR, etc. Thank you.