Artificial Intelligence (AI) for Investments
Prof. Abhinava Tripathi
Department of Industrial and Management Engineering
Indian Institute of Technology, Kanpur

Lecture- 44

In this lesson, we will apply classification algorithm in security price prediction using our programming. We will revisit the ABC price case study. First, we will train linear logit and probit models using our training dataset. We will evaluate various statistical goodness of it measures and visualize the performance of these algorithms. These measures include classification and confusion matrix and ROC curve and area under the curve. We will also train a simple algorithmic model to find the best performing algorithm using a simple performance object which is the simple average of model classification, accuracy, sensitivity and specificity measures.

In this video, we will recap the ABC stock price forecasting case study and then we will see the application of classification algorithm in the context of security price prediction. Recall our discussion about ABC security price prediction case study in the context of regression algorithm in a series of videos. We noted that stock price prediction or stock return prediction is an attempt to determine the future value of a company based on the analysis of factors which impact its price movement. We noted that there are a number of factors that help in predicting stock prices.

Case Study: Stock Price Prediction

So, we are given the data for stock market price for ABC company, along with Nifty and Sensex (market indices). We are also given the data of dividend announcement and a sentiment index.

Date	Price	ABC	Sensex	Dividend Announced	Sentiment	Nifty
03-01-2007	718.15	0.079925	0.073772	0	0.048936	0.095816
04-01-2007	712.9	–0.00731	0.021562	0	–0.05504	0.009706
05-01-2007	730	0.023987	–0.02441	0	0.019135	–0.03221
06-01-2007	788.35	0.079932	0.012046	0	0.080355	0.011205
07-01-2007	851.4	0.079977	–0.0013	0	0.094038	–0.0004
10-01-2007	919.5	0.079986	0.019191	1	0.015229	0.030168
11-01-2007	880	–0.04296	–0.04025	0	–0.07217	–0.04966
12-01-2007	893.75	0.015625	0.036799	0	0.01396	0.020999
13-01-2007	875	–0.02098	–0.00845	0	0.057518	–0.01164
14-01-2007	891	0.018286	0.004858	1	0.008828	0.020714
17-01-2007	819.75	–0.07997	–0.01228	0	–0.12395	–0.00962
.....	.....	.....	.....	.....	.....	.....
.....	.....	.....	.....	.....	.....	.....

These can be macroeconomic factors like the state of the country's economy, growth rate, inflation, among others. There are also various factors that are more specific to a stock like profit margin, debt equity issues, sales of a company and so on. In this particular case, we are given the data for a stock market price of ABC company, its returns along with the Nifty

and SINSEX returns. So we have Nifty, SINSEX and Nifty returns along with the dividend announcement and sentiment. Sentiment variable is provided.

Sentiment essentially reflects investors view about a market. It can be positive or negative as well and then dividend announced. For example, when dividend is announced, here the value 1 reflects that on this particular date the dividend was announced. Now consider a portfolio manager who has built a model for a particular stock. The manager wants to predict whether in the next period the stock price returns will go up or down.

That means whether it will be positive, return will be positive or negative. The market price starts from 2007 and goes on till 2019. So we have approximately 13 years of data and we have daily returns for ABC which is basically the change in price here in this column. Along with that we have market index variables like Nifty, SINSEX, Sentiment and dividend announced as we discussed. Please note that these SINSEX and Nifty are two major stock market in eastern India.

They are benchmark stock market indices that represent the weighted average of largest Indian companies. SINSEX represents average 30 largest companies that are most actively traded and similarly Nifty represents a weighted average of 50 largest Indian companies. To summarize this video, we will perform the following tasks using R programming. First, we will create a dummy variable which is 1 when stock prices go up and 0 when stock prices go down and then we will segregate the data into training and press data sets similarly as we did in the regression model. Then we will train and build the model using simple logit probit classification algorithms using market index as the independent variable and up and down dummy variable.

So remember the up and down, the positive and integrative returns will be represented through a dummy variable like we said. So 1 when there is a positive return, 0 when there is a negative return. So this will act as a dependent variable while market returns like SINSEX or Nifty will be employed as an independent variable. Next we will evaluate in-sample performance and out-of-sample performance of these models. We will compute the marginal effects of the independent variable, visualize the performance of these models using ROC curve and then examine the classification accuracy of the model and compare it with a similar linear probability model.

In this video, we will start with our implementation of classification algorithm with ABC security price study. So we will talk about ABC security price prediction UB classification algorithm. So we will conduct the following sizes. First we will set the working directory, we will load the relevant packages and then we will read the data and then we will create the up down variable which is 1 when returns are positive and 0 when returns are negative. So first we will set the working directory here using this session tab, choose directory and click on it.

So notice a setwd command has appeared. So for reference, for future reference, I will just copy paste this command. So whenever I am working on this file in future, I know where the working directory has been set. And we will import our ABC data from text. I will import this ABC data.

So the file has been read. And we can see that a view of file has appeared. Now instead of naming it, you can change the name also I will just copy paste this command, the full command. For future reference, I will use this command. And instead of using ABC, I will rather use the name, we can change the name here.

So I will choose the name data file. Already there is one read excel package which has read through this command library read excel but we need some more important libraries. First lubridate to handle date time operations. Then we need library of moments if we need to do some statistical operation like moments and all computation of moments and so on. Then library car, we have already seen the application of library car.

Another important library that we will be using is lmtest library. So these are some of the important library that we are going to use some of them already used in the regression modeling library tidyverse for any kind of tidy operations. Library of margins for examining the marginal effect of logistic regression coefficients. For ROC, for generating ROC plot, we have ROCR library. We have carrot library for classification matrix and some various important measures.

And lastly, there is also one patchwork library, which is for some good plotting. So these are some of the libraries that we are going to use in our operations. We have read the data also now. So first we loaded the relevant libraries here. Loaded the relevant libraries then we read the ABC data.

Another very important operation is checking whether your date often it is important to check whether date variable in your data is actually in the date format or not. So sometimes the date variable is read as character. Let's see if that is the case. So it is read as that. In fact, it is read as date time.

So POSIX-CT represents not only date but time as well. But we are only dealing with the date part of it. So I would to in the interest of clarity, I will just change the class to date. This is important so that there is no confusion while we are doing some operations. So when I run this command, I will convert my date from date time to exactly date.

Now if I run again run back this class command notice in the console window you have date it is appearing as date so that time aspect is. Lastly I need to convert or rather create up down variable which is 1. So this variable should be 1 when returns are positive and 0 when returns are negative. So a very simple operation is needed to create this variable. Now we are working with todayverse.

It's quite simple. I'll use this assignment operation. Then I start working with this symbol. I start working in data. I request it to create through mutate command. I request it to create a new variable.

Its name is up down. This variable is equal to and I'll use if else operation. If else operation. If and remember ABC variable contain the returns. So if the returns are greater than 0 then I want it to be 1 and 0 if the returns are negative. Now if I run this command I run this command you would like to check our data so I'll use the head since we are in readyverse I can simply type data and notice there is a variable up down created when returns are positive ABC returns are positive.

This variable is 1 and when returns are negative as here in red it is 0. To summarize this video we started with setting the working directory then we loaded the relevant triab libraries then we loaded the data and converted its date variable which was actually date time to pure date variable and then we created a new variable called up down variable which is 1 when returns are positive and which is 0 when returns are negative. We created this variable because ultimately we are doing classification with the help of limited dependent variable where this up down is the limited dependent variable or binary response variable or discrete you can say discrete variable. In this video we will create our sample of test and training datasets.

Training and test datasets. So as a first step what we'll be doing is we'll be separating only those observations that are after 2007 so we'll first in our data we'll create those observations using this filter year of date is greater than 2006. So notice in the console now only those observations are there where year is greater than 2006 and we'll assign this to our new data so this will create a new data assign this data to our same name and now our year starts from 2007 so this is first step. Now we'll randomly select 80 percent of the observations as training dataset using these 80 percent observations will train our data and remaining 20 percent will be used to test our trained algorithm. So we'll create an index variable the application of this index variable it will be quite simple. So using the sample command we are randomly creating an index variable.

So notice with the sample command we are selecting data number of rows with the number of rows and here we'll specify that out of these rows only 80 percent that is 0.8 into only 80 percent of the rows are to be selected in this index data. So the objective of this index this will act as an index while we are extracting observations from our data variable. This is how we do it so we use train na.omit so we want to so first we'll also sometimes there are missing observations the best way to deal with them is use this na.

Omit command so all the missing observations are removed and so we save some unavoidable problems. Now we'll simply select only those observations and see how we use this index variable to extract only those rows so index will act as an index while extracting those rows all the columns notice on the right side of the comma there is nothing that means

you want to select all the variables if you wanted to select some specific variables like variables 1 to 3 then we would have used this but we want to use all the variables I am simply using index comma and now the train variable will contain all the rows randomly selected 80 percent of the rows and please remember these are randomly selected. Now in the test data again we want to use only those rows that were not used earlier so very intuitive and interesting use of this minus sign which tells us that only those rows that are excluded in that are not there in the index variable those rows are to be selected and again the same comma operation and we get that in test data. So we have our test data and train data ready. Now as a practice you just have a look at we will just have a look at these data that we created so see that they are not properly ordered train data and test data they are not properly ordered so the next thing we will do we will try to order them with time so what we will do is what is this command train and I want to arrange this with date so I will use the date command to arrange the train data and now you notice that now the date command is with the order of time order with time.

Similarly we would like to order our test data also with this I will order the test data with time. Date basically a date variable indicates the date so we will arrange this as well with date so now that we have created our test and data train data which are arranged with time. As a good check it is important to see whether the distribution of our dichotomous or binary variable which is up down variable has same proportion what do I mean by proportion let's look at so I want to know what is the proportion of ones and zeros in this up down variable in our original data so I will use this prop.table command and then I will put it in tab with table I will put in table form I have the original data which is data $up down and I will multiply it with 100 to get the number in percentage form. So notice it is around 50 50 percent approximately 50 50 in the main data set so I want to see whether this remains the same in our test data also so I will use this first I will check with the train data and let's see very close it is almost 50 50 and they should not come as surprise for us because we randomly selected our sample was very randomly selected there is no systematic bias so ideally it should the proportion of up and down variable ones and zeros should be same as the parent data and it supposed to be indeed the case although in test data there is slight deviation but it almost it is same so it's okay so broadly this not skewed so ones and zeros are not skewed in any of our samples test and train.

To summarize this video first we filtered the observations that are after 2006 we wanted more fresh observations so we started our experiment with 2007 then we removed any any observation not available observation for missing observations then we randomly selected only we randomly selected 80 percent of the observations as part of our training data set and remaining 20 percent were assigned to the test data set. Finally we also tested whether the proportions of ones and zeros is skewed in any of these data sets that is the parent data the training data and test data and we found that broadly the distributions of ones and zeros is fairly similar close to 50 50 although it is slightly different in the test data set which is 45 54 but still it is okay for our experiment. In this video we will train a linear regression model using the training data set and then compute the classification confusion matrix and performance measures like accuracy specificity and sensitivity and see the model

performance we will evaluate the model performance. So in this video we will do the linear probability modeling and performance evaluation. So first as a first step I will assign it to a linear object I will create a linear object where I will assign the train model remember that we use lm command for train model then up down is are the is the variable relevant variable dependent binary variable ones and zeros then we have sensex as market index and the data that we are using is the train data.

Now this trained algorithm is assigned to our linear object if you want to see the summary of this object you can simply type the summary command and linear and notice that this is able to explain approximately 20.35 percent of the model that just our square is approximately 20.35 so we are able to explain that percentage and also notice the coefficient which is 12.34 that means one percentage increase one percentage increase in market variable leads to an increase in probability of approximately 12 percent in return being positive and similarly we can interpret in a vice versa manner that one percent decrease in market variable leads to 12 percent probability in a negative or zero observing a zero that means negative return so 12 percent increase in a probability of negative return. Now we cannot simply model like this we need to assign or classify these results into ones and zeros because we only observe what we call probabilities we do not observe these probabilities we observe ones and zeros we always observe ones and zeros so we need to convert these probability results into ones and zeros and a very simple way to do that is through this if else we will extract the fitted values so these are our fitted values inside the linear object we have our fitted values and those fitted values let us say we start with that threshold value of 0.

4 we will use three threshold values 0.4, 0.6 and 0.8 so if it is greater than 0.4 we will assign a value of 1 and if the probability threshold is less than 0.4 then we will assign a value of 0. So the fitted results are put here in the fitted result object.

In the next step we will compute the classification or what we call confusion matrix it is very simple confusion matrix which is facilitated by the caret package that we installed earlier and here we need to provide our fitted result in the factor form so using as.factor we will create these results in the form of factor so there we have our fitted results and then we will also provide our actual data so then we have actual data as.factor actual data so we have our actual data so this is our actual data and we will put our computed results we will put our computed results in the object named as cm so the cm will carry our fitted objects now I want to compute my performance measures accuracy sensitivity and so on so I will create this performance performance let us go for this performance object and this will carry I will assign in the table format I will create a sort of triple threshold so threshold value of 0.4 and accuracy now please note for accuracy we will assign a name of accuracy but I need to extract this from the accuracy how do I do it I type it accuracy like this and then I also need to extract the sensitivity so I put a name of sensitivity I assign again I use that cm object cm dollar by class and I put a name of sensitivity and lastly I also want to extract specificity and for that again I will use this cm dollar by class and I will put the argument specificity. So here I will get my performance object and notice I can print the performance object very nicely it has for a given value 0.

4 whatever the accuracy sensitivity and specificity that are computed now I need not write the code again I want to compute it for a threshold value of 0.6 I can simply create another object performance underscore 6 and I will use a value of 0.6 I will run this and all the results are stored means later we will examine the results first we will create for 8 or 0.8 also so I will create the performance object for 0.8 this is we are doing for linear probability models lpn so we are creating the objects next we will create 0.

8 for 0.8 also very simple we will create it for 0.8 also now lastly we will combine all these objects for our future purposes we will combine this linear underscore performance object and we will type it as rbind there are three objects that we need to combine performance 4, performance 6 and performance 8 now I have done that but I will also add another command where I would like to add a class column we are doing it because in future when we are comparing it with logit probit models I will make use of this new variable class and we will see the use shortly we will understand why I am adding this but for now I will just print this object linear performance and you notice we have threshold levels accuracy levels and sensitivity levels so I also need so I my mistake I forgot one step I needed to add so I will just add this step here I simply needed to compute every time I do that I need to compute again as well I forgot that so I will add 0.6 here and similarly I will again add here 0.8 here so I just rerun the commands again so that it is done properly so my model is already there I simply need to add from this and once I run this now we have all the threshold value their accuracy and sensitivity then specificity to summarize this video we create we trained a linear probability model using the train data set and then we using these fitted results we converted them into ones and zeros with our threshold values of 0.

4 0.6 and 0.8 then we computed the confusion matrix and subsequently we computed the important measures of performance which is accuracy sensitivity specificity and then we stored these in our linear performance object for future use in this video we will train our logit and probit algorithms using the train data set and then we will compute some performance evaluation measures such as pseudo R square we will also see their marginal effects and then we will compute their performance evaluate their performance using threshold values of 0.4 0.6 0.8 by using measures of accuracy specificity and sensitivity so we will first train our logit performance object now this is quite simple we will use this we have assigned the trained object into logit logit object and we need to simply use GLM generalized linear models for GLM short for generalized linear models now I will not write the formula again I will simply use the same formula that I used in our linear object remember the formula was dependent variable was up down and independent variable was market variable now again our data set remains the same the train data set only we need to specify the family of models that we want to use and this is the binomial family of models which is logit so our trained object is assigned inside the logit variable remember the concept of pseudo R square where we wanted to create another restricted or null model sort of null model where we are not using any variable it's a constrained model so the dependent there is no variable we are using simply putting one to reflect that there is no variable only constant term it will be again trained with train data set it will its only purpose is to check the pseudo

R square remember the discussion on pseudo R square in the video topic goodness of it measures in lesson 1 so family binomial again binomial and logic so this null object restricted object where there are no variables it will help us in creating our pseudo R square measure now this pseudo R square measure is simply remember the formula 1 minus log log likelihood log I select the log likelihood so on numerator we have log likelihood of our logic object and it is divided by our log likelihood of null which is restricted model let's see what value we have so in the pseudo R square we are able to now please note this value we have 11 percent of pseudo R square but please remember on a standalone absolute basis we cannot interpret much it is more of a relative concept so we need to have a competing classification model logit or probit class if competing model which can help us compare we can compare with that but on absolute standalone basis this is not exactly that intuitive as compared to R square or distro R square measures also we have margins package we installed earlier where we can check the marginal effect of the coefficient so we had the market coefficient sensex and seems that its impact is 15.75 percent that means if market increases by 1 percent on average a positive return the probability of a positive return increases by 15.

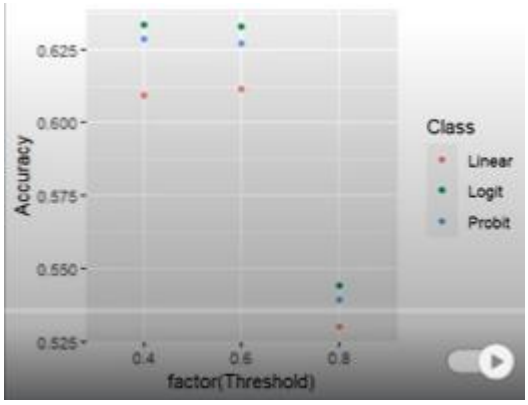
75 percent and vice versa if market decreases or market goes down by 1 percent then probability of negative return increases by 15.75 percent now please remember in order to fit our results we need not write the whole code again what we'll do is we'll simply copy paste the codes that we wrote for the linear model it's quite intuitive and simple we'll just simply copy paste these codes and see how it is done with R it's quite intuitive I'll simply copy paste these codes very minor modifications are needed so we need not write the full codes again just have a look at this so here I am using linear instead of linear I'll just simply use the logit I'll replace it with logit so that is first change so I put it as logit and the second change that I want to do is similarly in the 0.8 I'll again here change as well logit and lastly the third fitted results where we are converting the probabilities here we are basically converting the probabilities into ones and zeros like we did for the linear model now I'll just change the name of these performance objects to logit logit performance so and then lastly this logit logit performance and again as we did earlier we'll change this name from linear to logit and we'll just run these commands to assign these values so it's quite easy right we didn't have to write the entire code again we simply copy pasted and that is how it works in R so we simply copy pasted the values and we have our measures of sensitivity specifically ready for us now again now we want to compute it for probit and we want to compute the performance evaluation of probit so we'll do that so at this stage I will not do many changes I'll simply replace the logit with probit so I'll simply just copy paste the entire thing and the only thing I have to do is replace logit with probit so I'll simply replace wherever I have written logit I'll just simply replace it with probit so I'll just type probit here probit probit probit I'll just simply replace this logit object with probit object and just replace all all the places where I have written logit I can just copy paste probit everywhere where we have written logit we can write probit so everywhere we have written logit we are simply writing probit and with this we are done so we'll start with the implement this one is left so I'll just type okay so now we'll start with training our probit model it's quite simple I'll just run this command so we'll run the probit we'll train the probit model and then we'll also run this null command to get that restricted model as we did earlier we'll compute the pseudo R square pseudo R square is

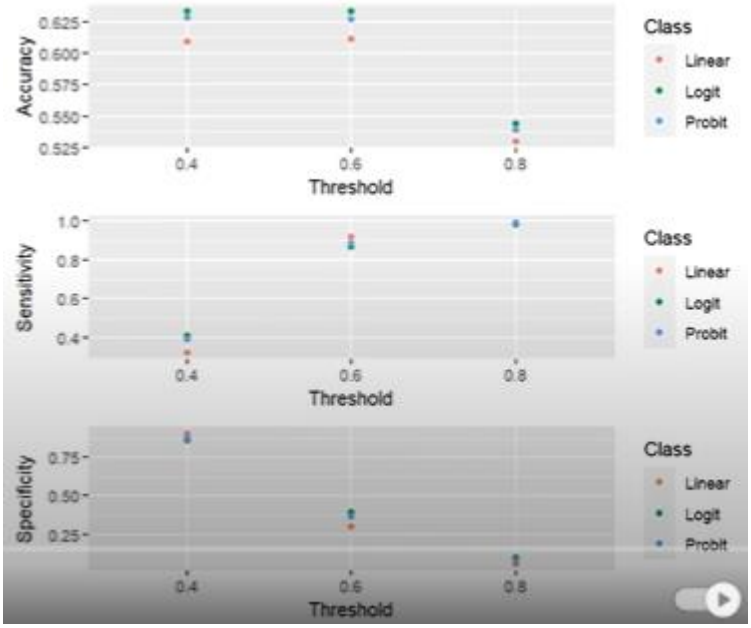
also there it's around 10.5% more importantly we look at the marginal effects of the product model which is 14.56 now the coefficient has similar value as we had for logit and the interpretation goes like this if there is a 1% increase in market then the probability of a positive return increases by 14.

56% and vice versa that is if market goes down then probability of a negative return increases by 14.56% now we'll run the remaining commands to store our fitted results and objects we as we did earlier for the logic model we will do for the probit model also as well interesting to note if I compute the correlation the correlation between the fitted logit dollar fitted values and probit dollar fitted values the correlation is very very high that means almost 99% this is intuitive as well because we were expecting this result we had the distribution of ones and zeros is very similar across the objects 50 almost 50 50 percent in such scenarios it is expected that logit and probit will give similar results and therefore we also expect that the performance in terms of various measures is going to be similar for logit and probit approaches to summarize this video we trained our logit and probit model objects through logit and probit algorithms we computed we evaluated their performance with pseudo R square measure and also we examined the marginal effects of the market coefficient subsequently we evaluated their performance on measures of accuracy and sensitivity and specificity using confusion or classification metrics at threshold values of 0.4 0.6 and 0.8 now that we have computed our performance objects in the next set of video we will compare their performance in this video we will evaluate the performance of logit probit and linear objects algorithms by computing the correlation between them and also through visualization so we will first compute the performance by computing the correlation and for that let's create the correlation object correlation is simply C bind correlation object is simply C bind we compute the data frame which will include the linear fitted object logit fitted object and also the probit fitted object so we have created our saved our fitted values inside this score underscore of and now we will compute the correlation metrics for this which is very simple in our core and the score of let's see the control in the console window let's see the printed correlations we can see the correlations are very high in fact 99.

9 which we already saw in the previous video between probit and logit fitted values for linear values also it's very high on almost 95% so this is expected generally this kind of high correlation expected when the distribution is very symmetric we have 50 50 percent ones and zeros for our training data set so such high correlation is expected but still we need to compare whether which one which one of these models linear logit and probit is performing well which we will do shortly for that we need our measures of accuracy specificity and sensitivity so we will first as a starting point we will first combine all the performance objects in our performance object we will combine all of them by creating a table the kind of data frame which is easy to process we combine all the three performance objects that is first is our linear remember in the previous video we computed the linear performance object logit performance object and we also computed the probit performance object now that we have combined we can see what is there inside this performance object so all the variables threshold accuracy sensitivity along with their class not here the class variable will be important when we will be doing the plot exercise so we will try to visualize them for

visualization visual so we will visualize them now for visualizing we will make use of ggplot command it is very interesting so first we will create the p1 plot ggplot and here we will have the performance object for ggplot we need to provide the aesthetics so what exactly we want threshold levels now threshold levels can be in the numeric form also but we will make use in the factor form that is category variable so threshold object threshold object and then we also want to plot accuracy but for accuracy we use the numerical properties accuracy so this is our accuracy then we would like to print because these are simply points so we like to make use of geom point while plotting them so geom point will ensure that we have a plot of and now here notice now in this aesthetics we are providing color to be segregated as per the class and that is why class variable that we added this class variable that we added is important because this will help us segregate the plots in terms of their different classes so this is first object in fact we can print it right now and we can see that p1 object you can see that accuracy is printed three objects notice the accuracy while we can examine them later on but just to give us a flavor you can see how they are plotted but we will compare them later on then similarly like p1 I can also plot p2 I have p2 object I can create a graph object p2 where instead of accuracy I can use the variable sensitivity I can simply just copy this sensitivity variable here so this is my sensitivity I will run state store this in p2 and the last one is the specificity p3 specificity so here I will use the specificity object so here p3 I will use the specificity object I just copied one more thing I need to adjust for the x and y axis so I need to add those xy axis lab command so for example I need to have let's say x lab in my accuracy object I will have the x axis always remains the threshold and y axis here y axis let me use next line for this so that will be better make it more clear so y axis I am using as since accuracy because this is p1 is for accuracy so I will use the accuracy here I will just copy paste this sorry I will just copy paste this and make use of in the next p2 we have sensitivity so instead of accuracy I'll replace it with sensitivity similarly for p3 I have specificity so I'll add specificity now I have all the three plots I'll again just run these commands to ensure that we have proper p1 p2 p3 now it's time to plot them and very intuitive very nice because we had one patchwork library if you remember we added which will help us plotting these objects like this p1 p2 p3 they will be very nicely plotted and notice they are they are very nicely lined in rows and now we can compare their performance so just notice the logit object which is the green window is always most of the times it is higher on the higher side while the values are very close to each other which which is which we also saw in terms of the correlation number the correlation was very high in the fitted values nonetheless the green one the logit seems to be doing better most of the times the green one you know especially on the accuracy level it is particularly high sometimes there is change in the sequence but still most of the time the values of the logit for accuracy and sensitivity is reasonably high except a few scenarios for example here the linear one is doing well in terms of sensitivity of 0.6 and so on but mostly it seems the logic model is doing well to summarize this video we computed the correlation across all the three linear logit and probit models fitted value then we plotted them for the three threshold values of 0.4, 0.6, 0.8 we noted that while the performance on all these measures are very close for all of them nonetheless logit model seems to be doing marginally better than the other fit models we were expecting this close performance this close column performance because the correlation across all the three linear logit and probit fitted values seems to be very close to each other.





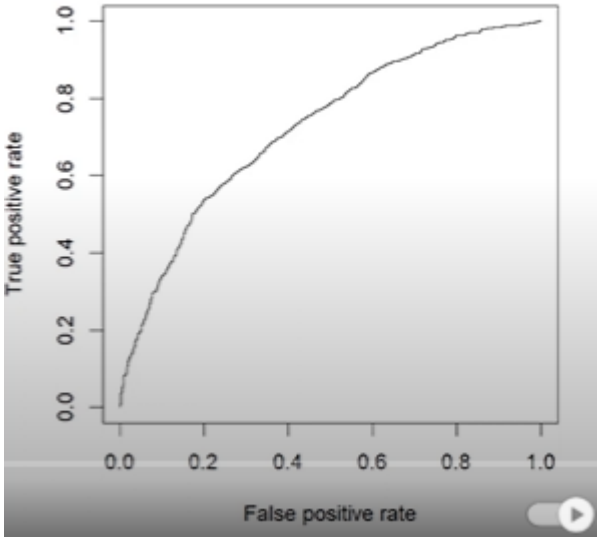
In this video we will compare the performance of linear logit and probit algorithms with the help of ROC curve recall our discussion about ROC curve in the lesson 1 video topic ROC curve characteristics ROC curve. So, first we will create a performance object with prediction object prediction and we will start with the linear algorithm so linear model that we fitted values and also we need to supply the train training data up and down value values. So, this will create a prediction object and from this prediction object we create we create our performance measurement object with the help of performance command PR and we need as a measure we need two positive rate or sensitivity on as our y-axis and we need on x-axis we need false positivity or 1 minus specificity as we have discussed in the ROC characteristic curve so we will run this command. So, now I will plot the ROC command it is a very interesting measure to compare the performance or evaluate the performance so I will plot with the help of this plot PRF so when I plot this notice this is the plot I will again clarify slightly smaller so we need to have proper PRF plot so this is our PRF plot now a good evaluation of this PRF plot can be done with the help of center line which is 45 degree line which gives 50 50 probability so I will use this 0 1 which is basically 0 to 1 line and remember this if you recall this ROC curve essentially measures different values different

coordinates of TPR true positive T rate and false positive T rate at different values of threshold different threshold values so it gives a complete view of the performance so we are lined up to and notice the line type now we can see that this model is somewhere between 50 percent line as we saw and the complete hundred percent accuracy it is somewhere in between but in order to know the exact accuracy number we need to compute the area under the curve so for that we will use our performance object PR and the measure is area under the curve so we will assign this AUC object now with the help of this AUC object we can simply access the area under the curve with this command and is around 72 percent so this is our area under the curve which visual so visualization we also we can also see the performance let us also calculate this for the logit object we will visualize also so the same thing we will do for our logit object let's do that for logit object we get the plot looks like this again very similar plot in fact it appears to be very similar to the linear object but we can compute the numbers again very same values we are getting very similar values which was expected because of their high correlation almost 95 to 99 correlation let's do this for probit also now so for probit I will do the same thing so probit object let's plot again almost identical graph we are getting so if I compute the AOC again see very similar values notice that we observed exactly identical values of area under the curve so could it be the case that we have done some mistake in computing our fitted values are these some result of some computational error let's check the summary of these fitted values so I will compute the summary linear $ these are fitted values so just compute the summary of linear model then we will do for it logit also logit logit $ fitted values and we will see the distribution the numbers are very close in fact to the extent of 4 digit they are close but they are not identical as we can see they are not identical they are very close to each other we can see for probit also so you can see there they are not identical but they are so close they are so close in terms of their distribution the minimum first quantile median mean and this was also borne out by the fact that the correlation between them is almost from 90 95 percent between linear and logit probit and between logit probit almost 99.9 percent so because of such closeness such high correlation the classification is also the ones and zeros would be very similar and that is the result that different thresholding values the area under the curve the accuracy of performance is almost identical so area under the curve ROC curve is almost identical area and for ROC curve area under the curve is very identical quickly to just summarize the video we plotted ROC curve for all the three linear logit and probit objects and we noted that area under the curve for ROC curve is almost identical this is ascribed to the fact that the correlation between all these the fitted values of all three algorithms that is linear algorithm and logit probit algorithms the correlation is very high and because of that the area under the curve the performance measure is almost identical in this video we'll develop a simple machine learning system which will help the computer learn how to select the best classification algorithm across a class of algorithms so we'll develop a simple machine learning system very often we would have heard of a cost function to be minimized in the context of machine learning algorithms in this particular case we will talk about a simple performance function which we will try to maximize the idea here is to develop a performance function a well suited performance function to the algorithm to these classification algorithms and see which one of these algorithms help us maximize the performance function so let's call this performance function as OPT for optimize and let's

define the parameters of this function so so in this function we'll try to simulate some numbers we'll use our fitted values we'll use the actual values and we'll use different threshold values so we'll simulate different fitted values for a given threshold compare them with the actual values and try to create a performance object which will be maximized across different algorithms so first we need to use these fitted value we need these fitted values so for that as we did earlier we'll use the very simple if else function the idea is to using these threshold values create ones and zeros so actual fit values are in the form of probabilities if you remember so we need to convert them into ones and zeros using our performance evaluation function once we have that recall that we computed confusion or classification matrix of and in that we will create that object again it's a very simple object so we'll just use this command cm equal to confusion confusion matrix in fact we can simply copy paste this number we have already typed this so we need not type this again we can simply use this we already have computed this so I can type this command so this command will help me extract all the three relevant objects that is accuracy specificity and sensitivity so we'll use this also because we are using the number of zero and when all as for tau the threshold we need to tell that there are two possible levels to this variable so two possible levels that are 0 comma 1 so because it may give us error from my past experience I know that it will give error if you do not specify levels because for 0 and 1 there is one extreme case where all the fitted values are either 0 so in that case we may get an error to avoid such error we simply define the levels in advance now with the help of the CM object I can obtain all the three relevant measures assuming that specificity accuracy and sensitivity are the relevant measures will define our complex combined performance object and this can be anything you can define you can give any weights or any complex function for your performance matrix of accuracy specificity and sensitivity but to keep the elaboration simple we I simply use this average simple average of all these three matrix what are these three matrix these three matrix are of overall performance are accuracy sensitivity and specificity so I'll just use these three matrix so it's one is accuracy one is accuracy I'm simply using their averages just to demonstrate the point you can think of you can imagine any complex performance function depending upon your taste choice and preference but to keep the demonstration simple I am using their simple averages just the average of three measures which is accuracy sensitivity sensitivity and specificity so I'm using the simple average to test my trained algorithms and tell my machine or the machine learning process that this function has which the algorithm whichever maximizes this function need to be considered so specificity now the idea is it throw this function I'll simulate a large number of threshold value and simulate a large number of tau or threshold values and then I'll generate number of performance measures for different algorithms and then I'll see which one of these algorithms is able to maximize or provide the maximum level of performance across competing algorithms to summarize this video we created an optimization function where the arguments that is fitted value actual value and threshold values will be passed and this will give us our performance or objective function which is simple average of accuracy sensitivity and specificity the idea is to simulate a set of threshold values and through which we will generate a number of performance measurement values for different competing algorithms in the next set of videos we'll generate these performance values for our three competing that is linear logit and probit trained models we'll generate these performance values and we'll see and compare and

contrast which of these models are able to generate better performance in the previous video we defined our performance objective function in this video we'll simulate thousand threshold values and then calculate the value of our performance objective function for each of the classification algorithms that is linear logit and probit models so we'll compute the performance objective performance function the starting point let's start with the linear performance object performance object of the linear model so how do we do it first and foremost we we have remember the function we have the fit object we'll assign the linear dollar fitted values so these are our fitted values then we also need to supply actual values which are trained dollar up down so these are actual values we'll start with the threshold value of 0 so that is our starting point and we'll create a performance linear object so this is our performance linear object where we'll assign the values here we are putting we are putting the values of tau so tau we have as a sequence from 0 to 1 so our tau values are from 0 to 1 in the sequence so minimum is 0 maximum is 1 and we need to give the length so length is 1000 so 1000 is the length of our assigned values in fact we can remove tau equal to 0 from here we already defined tau from 0 to 1 and then the performance object we initiate the performance object with the value of 0 later these values will be filled when the value is calculated as we will see shortly so we have created a performance linear object now in the next set of codes we'll run a for loop a very simple for loop where we'll simulate the values and for each value of threshold we will compute our performance object value so notice a simple for loop it starts from 0 to 1000 now the code is very simple it's a very simple code just notice so I have my performance object performance underscore linear object and in this performance for each I remember the I moves from 0 1000 in each I I will assign the value computed from my optimization function where I supply in the optimization function here I am supplying the fitted values fit that are the fitted values by the linear object then my actual values and then here I provide this threshold simulated threshold value so notice this I moves from 0 to 1000 and it is divided by 1000 so I supply 1000 values simulated values which will be used to compute the value of performance object now I run these values so we will run our for loop here and notice now that the loop has run it will simulate a set of values you can check also you can run this we can print these simulated values in the performance linear object so we can check the head of our performance linear object we can also check the tray tail of performance linear object seems they are varying from around 50% from smaller to larger value but we will compare with the actual thing so this is how you compute the performance object for the linear algorithm similarly very similar manner we will compute it for logit object for the logit model now I will not write the code again I simply just change the small line of code here to ensure that these are my logit fitted values and then here instead of performance linear I will type performance logit I will run this function and just replace the performance logit here and I will run this loop so this will assign in the performance logit function it will assign the values similarly I will create a performance probit function where I will assign the probit performance probit model so I will use this probit model here I will change the names from probit here and probit here so I will just run these codes probit I will create a probit object here also when I am simulating the threshold values I will run the for loop with probit objects to summarize this video we created three performance objects first is the linear logit and probit using our optimize function which we created for measurement of our performance we simulated thousand values of threshold and

for different values for each object we computed the performance objective function value and assign them into a performance object of linear logit and probit in the previous video we computed the values of our performance object for all the three classification algorithms that is linear logistic and probit using our simulated threshold values in this video we will try to compare the performance through visualization we will compare the performance through visualization as a starting point let us plot the values of the performance object for logit object so we will plot the logit object first its tau values threshold then the values of the performance for different threshold values we will plot as a type line color red line width of two seems to be appropriate we will not name the y-axis we will keep the y-axis as 70 x-axis is common so we can give a name of threshold and the central heading is performance so the central heading is performance let us plot this object so we can see the object that is plotted now we will superimpose other performance objects for first for the probit again same we can use the same commands only that we change the color to green to make a distinction other things remain same line width and so on now we'll just copy paste this and instead of probit we'll use linear here and color will change to blue to make a distinction let's superimpose these now it would be nice if we can add a legend also so we'll add a legend on the probably top left seems to be empty so we'll use top left corner to add the legend so that a user can see the distinction legend so we have three items logit probit let's make it capital logit probit linear then we need to add the color also fill out feature red logit is in red probit is in green and linear is in blue so with this we add the legend finally with this our plotting will be complete now notice the performance objects here for all the values in fact for most of the values the red one seems to be offering in terms of performance higher value so the red one attains higher values for most of the threshold values and red one is logit next is the probit which is green one although they are very close but still for most of the cases logit is higher linear is the lowest one they are pretty close as we said this is expected because the distribution of ones and zeros is pretty symmetric 50 50 percent but nonetheless logit performs better than second number we have probit and third we have linear lastly if you want to extract a particular value for example you would mostly be interested in the maximum performance value and what threshold it is to train our algorithm so we'll be extracting it very easily in our notice the command so I am extracting that performance value which is equal to max so the maximum value we are extracting here just taking the maximum value and we'll extract that so along with its threshold as well so the maximum performance is 0.



672 performance object value at a threshold of 0.536 so an algorithm which is trained through this manner would probably be better placed to have a threshold level of 0.536 and at that level the performance object will give a value of 0.672 to summarize this video we computed various performance object values for different classification algorithms that is linear logit and probit for our thousand simulated threshold values we plotted all these performance values and superimpose for all the three classification algorithms and we noted that for most of the values logit algorithm performs the best next probit and finally the linear although the performances are close to each other nonetheless for most of the threshold values logit seems to be the best algorithm for classification in this video we'll start with out of sample fit performance evaluation recall that in the discussion around linear regression modeling on the video topic out of sample fit performance evaluation we have already covered a certain number of measures such as error measures like RMSE, RSE and so on so we will not repeat that discussion here we can refer to that video so as a first step we will using our trained algorithm on the linear object linear performance object will create the test object first remember this was our linear object it was simply computed as linear equal to lm up down variable and it was modeled using sensex with the training data set so this was our linear object if you recall so we run this command this was our linear object now we'll use the same trained so this is our trained object linear and we'll now work out with the test data set the command is simple predict we'll use our linear object but this time the data that we'll use the new data is test data so this is our predicted object the values are contained in predicted linear object next we'll train our logit performance object logit performance object again remember we computed the logit object with the simple glm command glm we'll use the same formula formula for the linear one so we'll extract the formula from there and we use the train data set to train our algorithm family of binomial models under logit so this was our logit model train model using train data and now we'll compute the test object using test data logit and again we'll use the predict command this is our trained object which is logit and we'll use the new data which is test data again the type is response this is for logit probit type of variables sorry the response type is equal to response for binary variables so this is

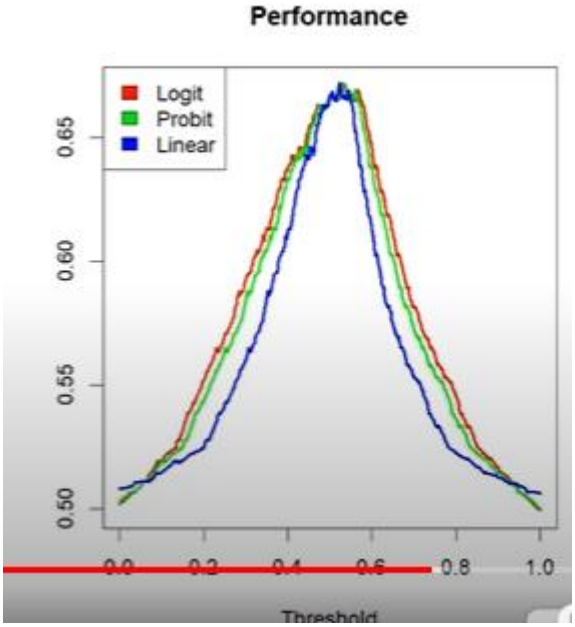
our predicted logit variable next we'll generate our probit object probit performance object again similar command probit we'll first train our algorithm in the similar manner glm in fact I'll just copy paste the previous commands and just make few spelling changes and save time so I'll just copy paste it here notice yes here now I'll change it to probit the object will be changed to probit and in the family I'll change it to probit here so I'll run this command again we change here the predicted object the trained object is probit so we'll change it to probit here also we'll change it to probit and remaining things will remain same so we'll project this is our predicted probit object as a summary aspect we can check the summary of predicted object for example we can check the summary of predict linear or if you want you can check the summary of predict probit and so on lastly we'll just combine them combine all the predicted object let's call it table cbind data frame we'll combine our predicted objects linear predict logit predict probit so we'll combine them in the consolidated table all the three objects that are there in fact you can check the correlation also if you want you can just check the correlation across all the three objects it's pretty high between probit and logit we can see very high correlation that means we can expect similar results and similar performance results with logit probit the correlation with linears is less still it is high at 91 percent but it's still much less as compared to between logit and probit in this video we computed the predicted values for all the three classification algorithm using our trained object so using the trained object we predicted the values for all the three algorithms and we noted that in the predicted values using out of sample test data which is out of sample fit the correlation between all the three algorithms predictions are pretty high in fact for logit and probit it is close to 99 percent while with linear object it is 91 to 92 percent this is ascribed to the fact that in our data once and zeros that is positives and negatives are pretty symmetric that is 50 50 percent both in test train and original data set therefore we expect the overall performance of logit and probit to be pretty similar and that of linear also to be close to logit and probit models in the previous video we predicted the values using the trained model with the test data set in this video using these predicted values for all the three algorithms we'll compute the ROC curve ROC curve and compare and visualize performance we'll also compute area under the curve to compare the performance of three algorithms please note that a detailed discussion on ROC curves have already been done in lesson one video topic ROC curve also a detailed R implementation of these ROC curves has been done with the trained data so we'll not have a detailed discussion we will keep the discussion brief and we'll focus on the R implementation part so we'll not have a detailed discussion we'll keep the discussion brief and we'll focus on the R implementation part so we'll first create prediction object as we already done similarly in there with the training data set for the first for the linear object and we'll compare it with the test data actual up down values so this is our prediction object using this we go to the performance same as we did earlier PR and we use on x-axis we have true positive period TPR on the sorry on y-axis which is also our sensitivity and on x-axis we'll have false positive period which is our 1 minus specificity so this is our performance object we can plot this performance object very well so we can plot it as we can see here along with that we can also draw the 50% area line as we did earlier so this is very simple using a blind command 0 comma 1 line width of 2 maybe line type of 2 we can give it a color of maybe red and this is the line one can also compute area under the curve very simply a UC equal to performance and we'll supply our prediction object PR

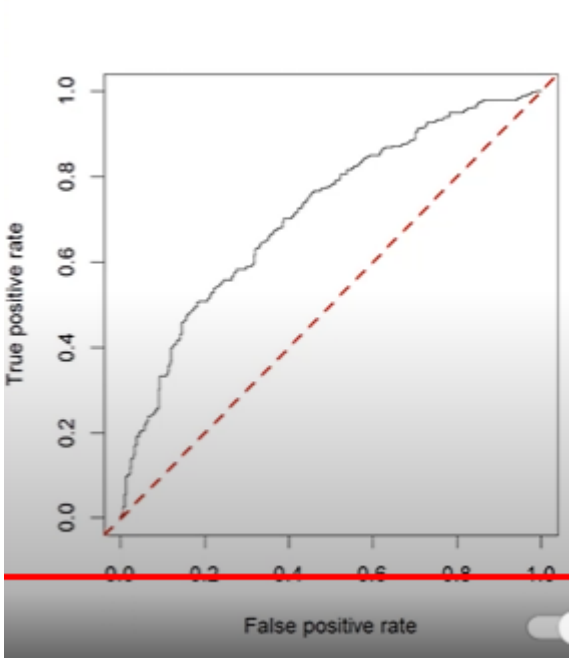
along with measure which is area under the curve so this will store in a UC I can simply extract the area under curve with this a UC y dot values which is around 71.5% now I'll not write the quote again I'll simply copy paste these and this was created for the linear object linear performance object similarly I'll create it for the logic performance object object performance object only I'll replace from linear I'll replace it to logic and I'll just compute the performance now we have the plot here this is the blind the values are expected to be pretty similar given the high correlation we are expecting high values similarly I can compute it for probit as noted earlier we may expect a very high similarity predict probit so this time we'll be using probit object and I'll compute so again pretty similar value this is this ascribed to the fact that the prediction the probabilities may be different but ones and zeros will be almost identical so we are expecting a very high similarity given the very high correlation in the fitted predicted objects we saw the correlations were in fact more than 90% so we are expecting very similar performance as we can see here around 71.5% area under the curve so as per our OC the performance is quite similar but we'll create our complex more complex performance object to see how they are comparing with each other and contrasting with each other to summarize in this video first we computed the ROC receiver operator characteristic curve and then we also computed area under the curve we find that given the high correlation across the fitted objects the area under the curve and performance as per ROC characteristic curve is very similar in the next video we'll use these values our trained algorithms and then compute our objective performance function for which we have already written the user defined function and we'll say using the simulated threshold values we'll compare the performance for all the three algorithms linear logit and probit recall that we have already set up a performance object which was simply the average of accuracy sensitivity and specificity now using our predicted values for all the three algorithms we compute various performance object values for thousand threshold values that we'll simulate using the user defined function that we have already created so we'll start with the performance object for linear model this is quite simple we have already seen that in with the trained data so we'll not repeat the discussion in gray detail we'll simply execute it so this is our fitted values predicted values using the test data and trained algorithm these are our actual values test dollar these are actual values and then our performance object let's move this name performance engineer we'll keep the same name and we'll assign a table which is nothing but a sort of data frame object tau will be a sequence remember the taus here will be sequence of zero to one and we'll keep the length thousand same as earlier and then this thousand so thousand threshold values will be created from zero to one at equal intervals and we'll initiate the performance object with the value of zero so this initiation value is zero so this will be our linear performance object now recall we'll start with the for loop where iv is the counter and zero to thousand so we'll start with zero to thousand so we include zero and one so that zero and one both are included in the threshold and now performance underscore linear in this we'll assign the value of perform we have initiated a value of zero by default so by default there is a zero value each time this for loop will run one new performance value be assigned and this opt user defined function we have already created linear dollar fitted values in fact i can simply use the fit we have already assigned to the predicted value into fit object actual values are also assigned here and then i upon thousand so this will be our performance object i have done this so this performance linear object will carry my performance object

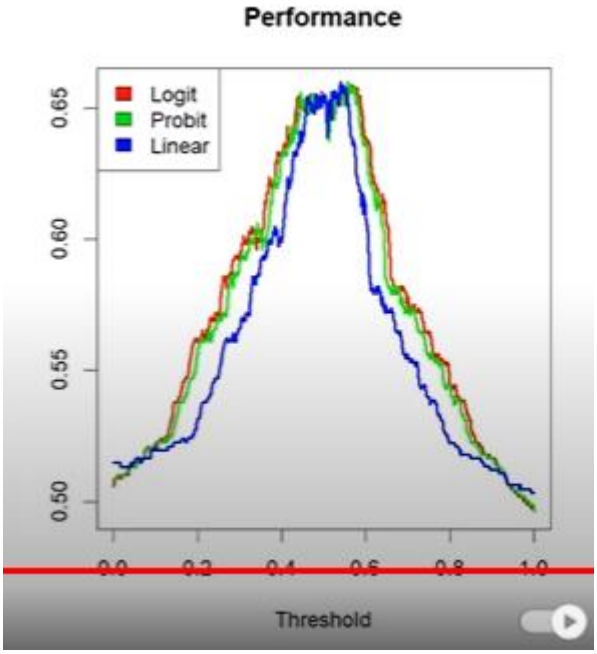
values cross simulated threshold values in a similar manner i need to create it for the logit object so i'll just copy paste this code i'll not change it much so i'll use it performance object for the logic model only thing i need to do here is i simply need to just change this to logit and rest of all it will also this one as well logit this one also needs to be changed other than the okay this one also needs to be changed so now i'll run this entire segment so this will also run in the meanwhile while the loop is running i'll also create my probit object this is for probit again i'll make the same changes probit here so now i have computed the values of my performance object for the simulated threshold values for all the three classification algorithms to summarize in this video we computed the value of our performance object using the predicted values for all the three algorithms with simulated thousand threshold value and then we computed the values of our performance object using the predicted values for all the three algorithms with simulated thousand threshold value in the next video using these performance object values we'll come compare the out of sample performance of the all the three algorithms and also visualize the performance in the previous video we have simulated thousand performance object values for our test data in this video we'll compare the performance of all the three classification algorithms using these simulated values since we have already done this exercise for training data set we'll keep the discussion brief and focus on the implementation so we'll compare out of sample performance first we'll plot in the simpler manner we'll plot the performance for the linear object now performance linear object for the performance again same commands we'll keep the type as l line color as red line width of two seems to be appropriate here y lab access y label will keep empty access y label will keep empty again x lab will keep as threshold name which is appropriate and the central heading will keep as performance so this will be a with this we'll start plot and we can see that the performance plot next we'll to this we'll add probit performance object so just to maintain consistency i'll use logit here next we'll start with the probit object again i'll i'll use the same set of commands performance probit performance other commands we'll keep same type l column red line with two we'll change the color to green we'll maintain the consistency here for consistency we'll change we'll run this as well and next we'll post linear so linear and i need to change this to logit as well here we'll take linear i'll keep the color blue so i'll again re-plot because there was some error so i'll re-plot it so this is the these are the plots let me add the legends here as we did earlier so we'll add the legend again the top left seems to be empty so i'll create the legend at top left names of legend will give the name same as earlier logit probit and linear color scheme remains the same so we'll use that again fill logit is with red probit is with green and linear is with blue so this is my legend so i'll run the legend command the legend has appeared now notice very identical to our train data set test in test data set also we note that the red one is the most outer one that means the performance value which is the average of accuracy sensitivity and specificity is highest for most of the probability region most of the threshold region it remains higher for logit in red probit in green and linear in blue so for logit it is highest probit is very close but is still lower but very close to it which is expected because of the very high correlation in the fitted values linear is the lowest linear the blue one is the lowest although relatively it's not that lower also but still it if one has to choose one will choose the logit and next probit so lastly if one wants to finally select the best trained model one can simply as we can see the logit is the most optimum model here from our predicted object so and if you want to select that particular

observation and threshold value as well as we did earlier we will simply choose this logit dollar performance and equal to maximum value so we'll select the maximum value for our performance object and this will help us extract the corresponding threshold value along with the performance object so I'll just run this and again we can see at tau of 0.

569 which is the threshold value we get a performance object value of 0.659 so this is the maximum performance value which we obtain for the logit object and logit is supposed arguably the best classification algorithm at tau value of 0.569 to summarize in this video we plotted our simulated performance object values we observed that for most of the threshold region logit model seems to perform best as compared to other two probit and linear it is closely followed by probit and then linear we also extracted the best performance threshold value for logit object which was around 0.569 and the performance object has a value of 0.659 which is simply the average of three accuracy sensitivity and specificity values for the model to summarize this lesson we modeled abc stop price up down measure that is a dummy variable which is one for up and zero for down moments using linear logit probit classification algorithms first we trained the model using the training data set and examined various measures of model performance evaluation these included parameters of accuracy sensitivity and specificity derived from confusion classification matrix we also visually examined the model performance using roc curve and computed the area under the curve next we created a simple algorithmic system by simulating thousand thresholding values and computed a comprehensive model performance object using parameters namely accuracy sensitivity and specificity with the test data to examine the out of sample prediction efficiency of the model overall the logit model seems to have provided better fit and model efficiency followed by probit and linear models we also computed the marginal effects of the independent variables on the dependent variable for logit probit class of models and discuss the interpretation of the same thing.




True positive rate
0.0
0.2
0.4
0.6
0.8
1.0
False positive rate


Performance
Logit
Probit
Linear
0.50
0.55
0.60
0.65
Threshold