Linear Models

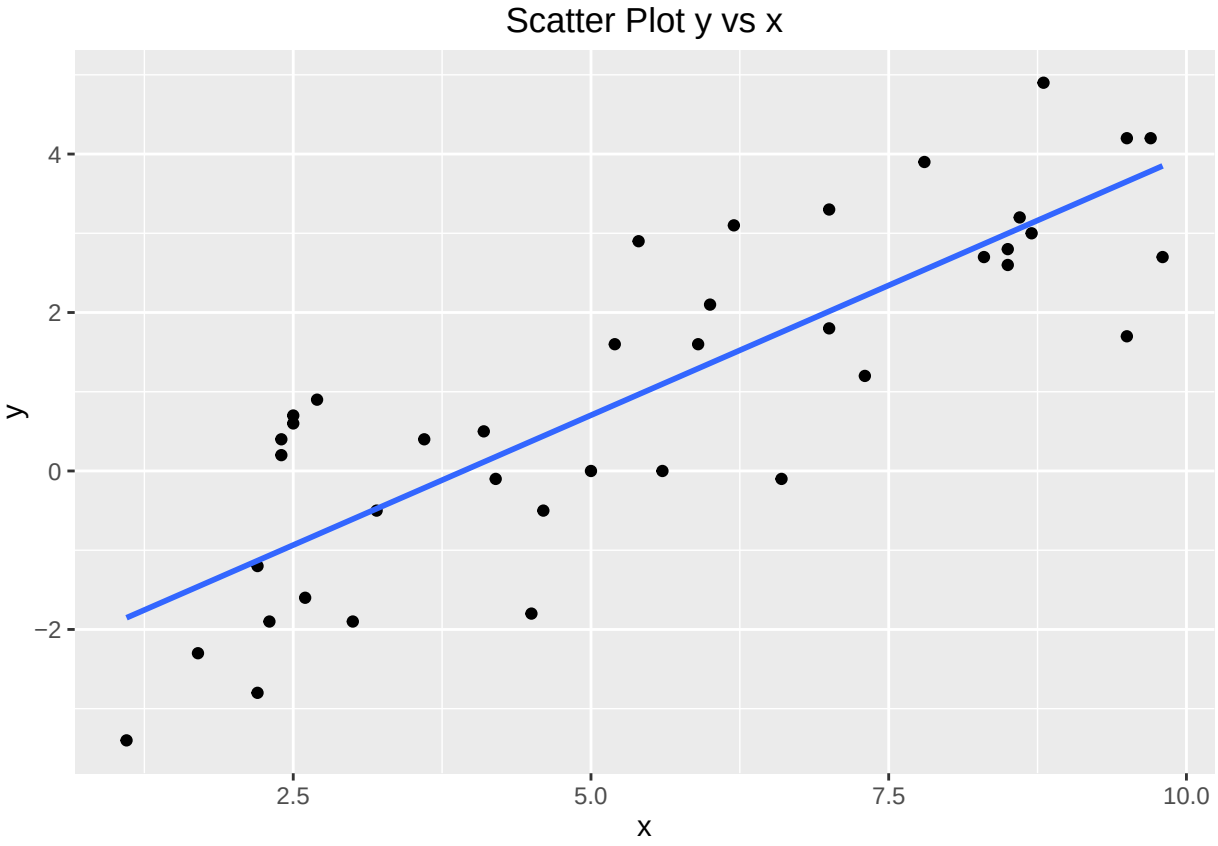
In this module we'll look at algorithms where we use linear combination of our features to predict outcome of interest, be it regression problem or classification problem.

Linear Regression

Lets start our discussion by looking at first 10 values of this small data .

y	x
0.9	2.7
1.2	7.3
3.1	6.2
0.6	2.5
4.2	9.5
1.7	9.5
-1.2	2.2
2.8	8.5
1.6	5.2
1.6	5.9

When we plot this in a simple scatter, it looks like there is a linear relationship between y and x.



Which means we can write prediction equation for y as linear combination of values which x takes.

$$\hat{y_i} \ = \ \beta_0 \ + \ \beta_1 * x_i$$

Now the error for each prediction that we make will be:

$$e_i \ = \ y_i \ - \ \hat{y_i} \ = (\ y_i \ - \ \beta_0 \ - \ \beta_1 * x_i)$$

As we discussed trying to minimize simple summation of these errors doesn't make sense because of positive and negative errors cancelling each other. We will instead minimize sum of square of errors that is :

$$SSE \ = \ \sum {e_i}^2 \ = \ \sum (y_i \ - \ \hat{y_i})^2 \ = \ \sum (\ y_i \ - \ \beta_0 \ - \ \beta_1 * x_i)^2 \tag{1}$$

You must be thinking; now that we have our loss function; we can simply apply gradient descent or similar optimization technique to estimate βs. However it turns out that this loss function is pretty friendly and we don't need to get into optimization techniques to find out optimal values of βs.

(1) can be written in matrix format like this :

$$SSE \ = \ \sum (Y \ - \ X\beta)^2$$

where Y , X and β are matrices as given below:

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ . \\ . \\ y_n \end{bmatrix}$$

$$X = \begin{bmatrix} 1 & x_{11} & x_{21} & . & . & x_{p1} \\ 1 & x_{12} & x_{22} & . & . & x_{p2} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ 1 & x_{1n} & x_{2n} & . & . & x_{pn} \end{bmatrix}$$

$$\beta \ = \ \begin{bmatrix} \beta_0 \\ \beta_1 \\ . \\ . \\ \beta_p \end{bmatrix}$$

I have taken liberty to extend the idea to more features which you see here as part of X matrix. x_{ij} represents j^{th} value of i^{th} variables/feature. All 1 column in X matrix represent multipliers of β_0.

Now we can get gradient of the above matrix form loss function w.r.t. β and then equate it to 0, values of β thus obtained will be optimal values of β. Goal of optimization algorithms is also obtain values of parameters at which loss function's gradient is zero, because that is where loss function achieves its minima. In case of the particular loss function that we are considering write now has direct solution to $\nabla C \ = \ 0$ equation.

Lets first see what is the gradient of the loss function $C = \ \sum (Y \ - \ X\beta)^2$. Keep in mind that y_i and x_{ij} here numbers/data points from the data. They are not parameters. They are observed values of the features.

$$\nabla C \ = \ -2X^T(Y \ - \ X\beta)$$

X^T means X-transpose. Equating this to zero gets us :

$$X^T X\beta \ = \ X^T Y$$

which eventually gives value of βs as:

$$\beta \ = \ (X^T X)^{-1}(X^T Y)$$

It might look scary now if you are not very familiar with matrices, but for all practical purposes this is a simple direct solution for values of βs for which $\sum (Y \ - \ X\beta)^2$ is minimum for given data matrices Y and X.

If you haven't realized yet , this framework works strictly with variables which are numeric. If your data has categorical values you can convert them to numbers if the values are in order and can be assigned numbers which do not change meaning of the data. If there is no order to the categories then you can make dummy variables which are nothing but new variables taking value 1/0 for each category of the categorical variable. Also if there are n distinct categories of a categorical variable, you need to create n-1 dummies only.[Reason for this will be discussed in the class]

Now that we are done building our model/prediction equation, we'd like to test its performance on the future data. oops! but we don't have any future data. What we can do instead is that break our original data into two parts [train and test], and build our model on train and test its performance on test data.

A popular measure of performance of regression models is RMSE which stands for Root Mean Squared Errors. Which can be calculated by getting errors in the test data and then take mean and then square root.

Regularisation

Mathematically, even if you include a random variable/feature in your model , it will end up having a coefficient. This result might not generalize very well and give bad performance results on the test data. To avoid this situation we can add a penalty on size of *beta*s in our cost function itself. Here is example of such a loss function with L2 penalty:

$$C \ = \ (Y \ - \ X\beta)^2 \ + \ \lambda \|\beta\|^2 \qquad (2)$$

The second term imposes a restriction on size of β, any feature which doesn't help much in bringing down the first term of the cost function gets their βs shrinked towards zero and thus do not affect our model much. The above form of linear regression is also known as Ridge Regression.

Another popular cost function with L1 penalty is as follows:

$$C \ = \ (Y \ - \ X\beta)^2 \ + \ \lambda \|\beta\| \qquad (2)$$

This is known as Lasso Regression. λ here is the weightage assigned to penalty on size of parameters. if $\lambda \to 0$ then these are same as simple linear regression that we looked at earlier. If $\lambda \to \infty$ then all the βs will have to be zero and we'll be left with no model, only average value as prediction.

Due to the difference in penalty assigned in Ridge and Lasso regression; there are few fundamental difference in both

Ridge has a closed form solution for βs.

$$\nabla C = -2X^T(Y \ - \ X\beta) + 2\lambda\beta \ = 0$$

this yields :

$$\beta \ = \ (X^T X + \lambda I)^{-1}(X^T Y)$$

Where I is identity matrix of appropriate dimensions. On the other hand; cost function for lasso is not differentiable and hence parameter estimation has to rely on numerical methods of optimization

Lasso reduces model by making few βs zero

Ridge regression cost function makes it so that βs will be shrinked to close to zero but will never be made exactly zero. Where as lasso regression might result in some *beta*s being reduced to exactly zero [it depends on the value of λ, higher the value of λ , more βs will be made zero].

Value of λ

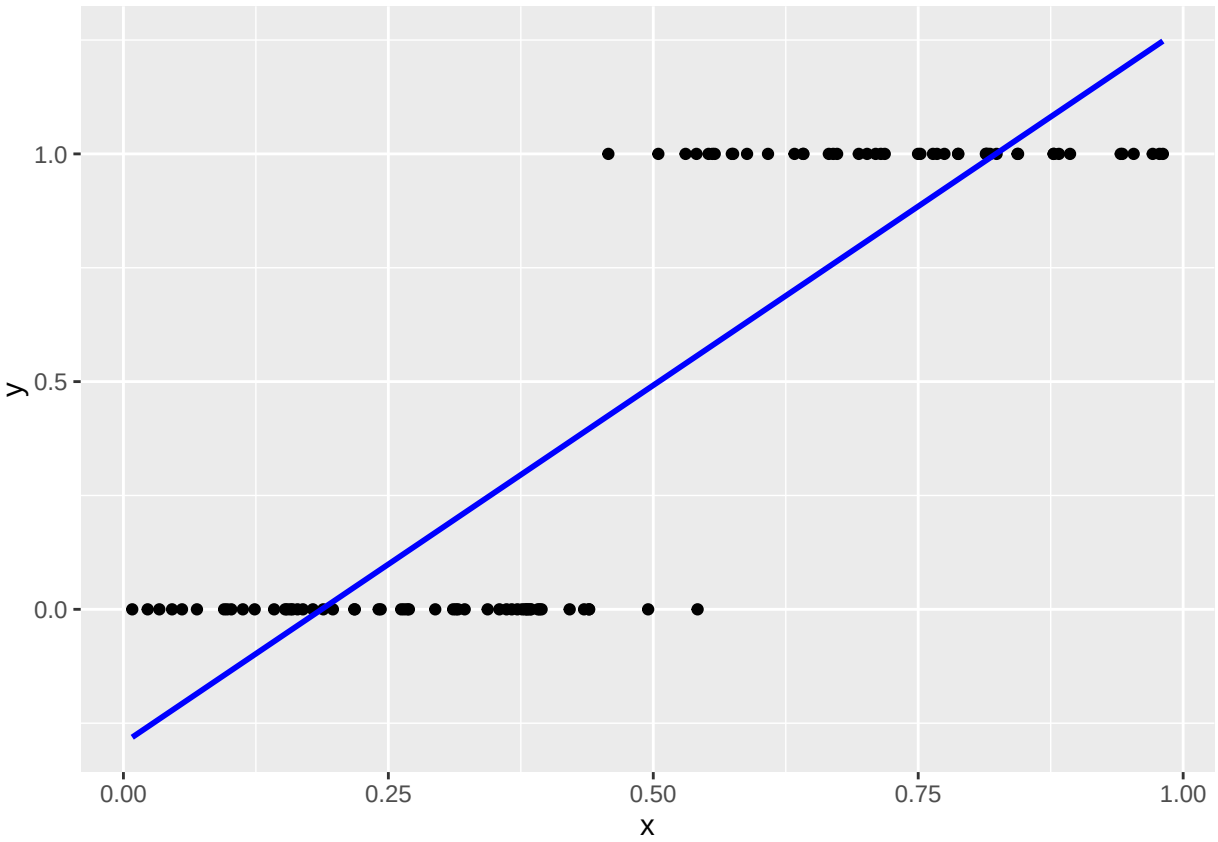
There is no formula for optimal value of lambda. We determine best value of λ using cross validation, this will be demonstrated in class .

Classification with Logistic Regression

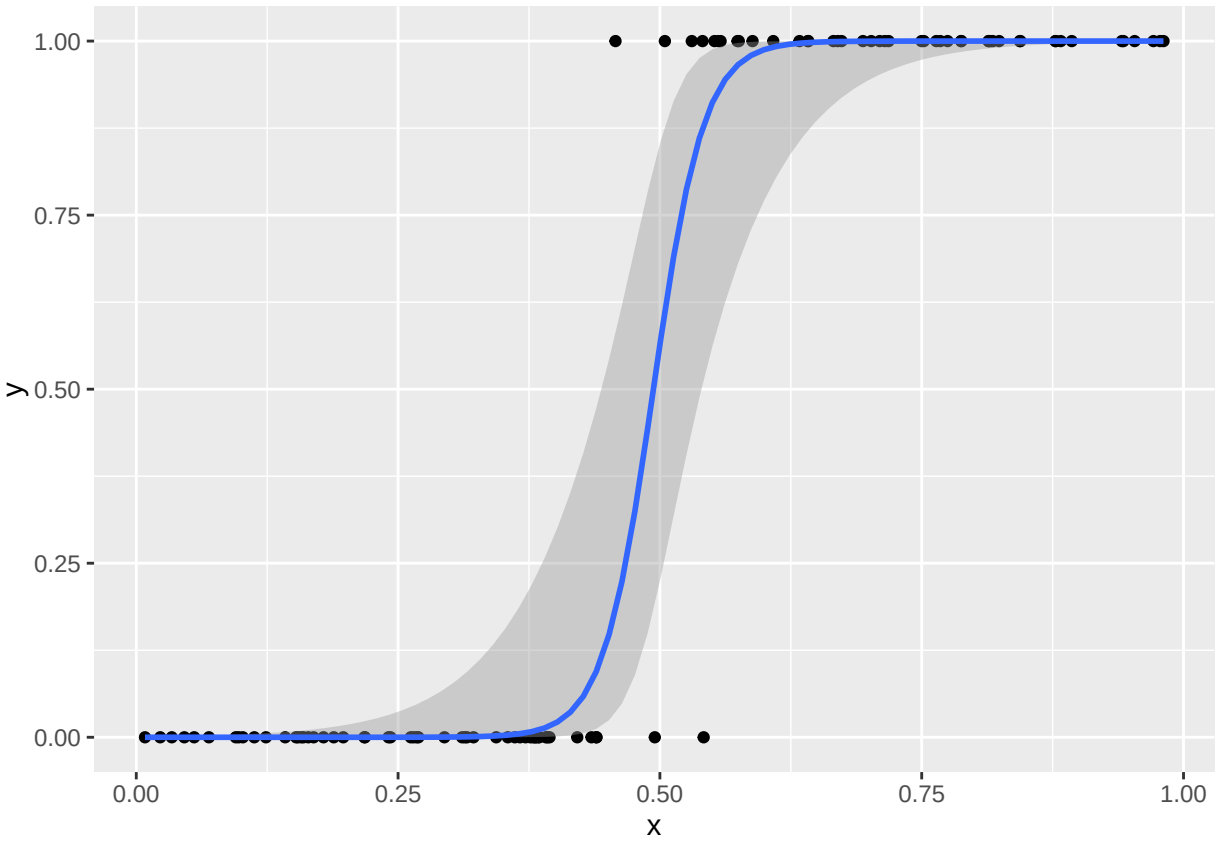
Note: Unfortunately 'regression' term is stuck with logistic 'regression'. Its a classification algorithm; Not a regression algorithm.

Now that we are done with that clarification. Lets look into classification with logistic regression.

You have seen how to model a continuous numeric response with linear regression technique. But in many business scenarios our target is binary. For example whether someone will buy my product , whether someone will default on the loan they have taken. Answer to all these and many other such questions is yes/no. We can convert that to 1/0 and then try to model them with linear regression technique but that doesn't give good results. Have a look.



You can see how badly this fails. what we really need is something like this.



Although to reach there we need to first develop some understanding regarding this. So far we have seen that linear regression approach fails at many levels. Lets look at these kind of problems from a different perspective. Consider this data regarding a hypothetical situation where we asked children of various age whether they are afraid of ghosts or not. Here are the results:

Age	Response
4	yes
4	yes
4	yes
4	yes
4	no
5	yes
5	yes
5	yes
5	no
5	no
6	yes
6	yes
6	no
6	no
6	no
7	yes
7	no
7	no
7	no
7	no

Now if someone asked you what might the response be if the child's age is 7. By looking at the table your guess would be "no". What you did there was to look at probability of response being "no" when age is 7.And you naturally guessed for "no" because that had higher probability *[chances]*

So instead of modeling y we should model P(y="yes") or P(y="no"). lets denote that by just p. Instead of $y = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 .... \beta_p * x_p$. we'd model this:

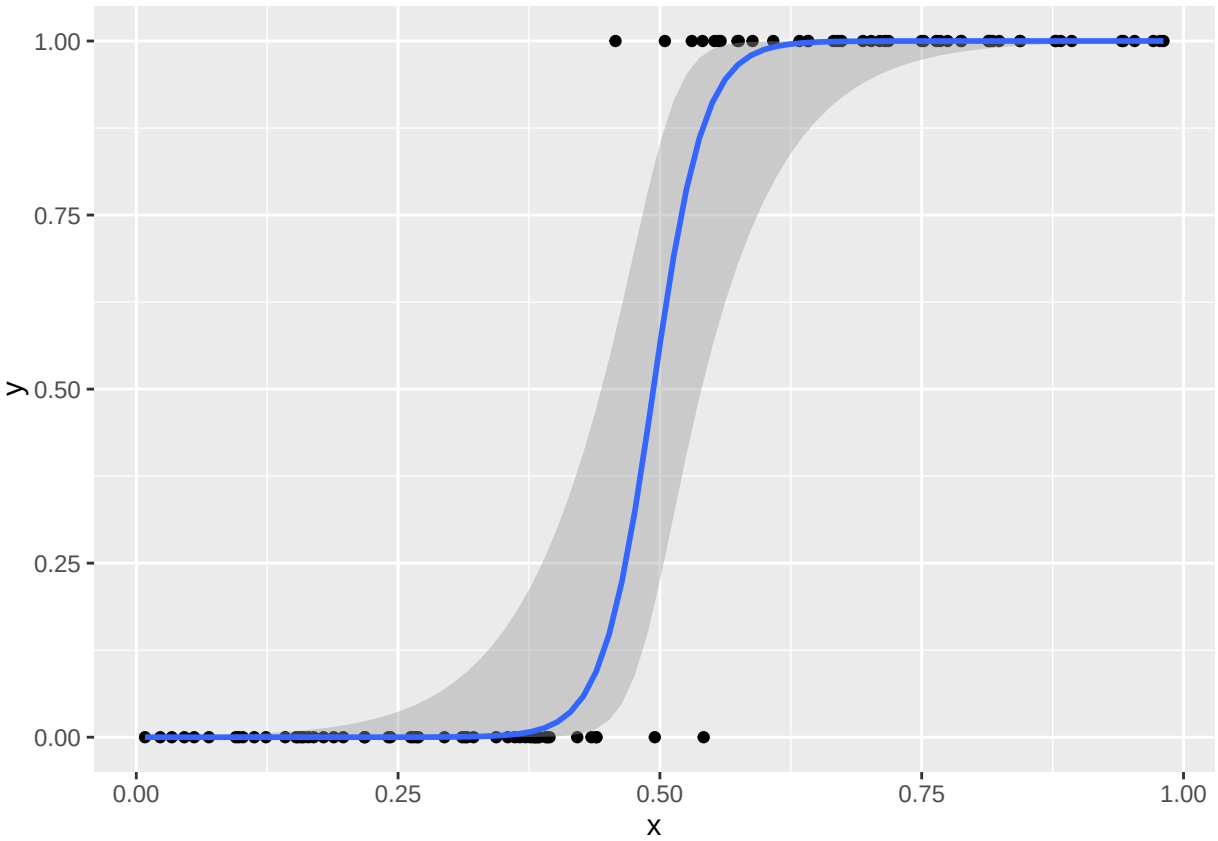
$$p = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 .... \beta_p * x_p$$

but this is problematic because right hand side in equation above can take values in the interval $(-\infty, +\infty)$ .where as probability p can take values in [0,1]. We need to transform this so that ranges match on both sides.

$$log(\frac{p}{1-p}) = \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 .... \beta_p * x_p$$

This takes care of the range mismatch issue. This transformation also provides one important characteristic to the probability p which you'll get as result. Remember in our hypothetical example we said you looked at high probability outcome and decided that'd be your prediction. Now imagine this to be at a large scale. Your prediction will still be binary, but you'll have to come up with a cutoff for this "p", so that above that cutoff your prediction will be 1 and below that your prediction will be 0.

You'd like this cutoff to be such that it enables you to make as least as possible miss-classifications. For this to happen , your "p" should be consistently high for one class and low for another , so that you can choose a nice cutoff for "p" in between which very well divides both the classes. Look at this curve again. Due to transformation mentioned above, we get our probabilities p to lie on this curve which enables us to get a good cutoff.



Now lets talk about those parameter estimation. The objective here is not to "correctly" estimate $log(\frac{p}{1-p})$. No. We want our parameters to take values which result in such a score *[probabilities or p]* which enables us to have a good cutoff. Meaning this "score" should be high for one class and low for another.

Lets say $P(y_i = 1|x_i) = M_i$

Now consider this :

$$L_i = M_i^{y_i} * (1 - M_i)^{1-y_i}$$

whenever $y_i = 1 : L_i = M_i$ and when $y_i = 0 : L_i = 1 - M_i$. Which means that when $y_i = 1$, L_i equals to probability of y being 1, when $y_i = 0$, L_i equals to probability of y being 0. You'd want your probabilities to match with real outcome. In other words you'd like to maximize L_i. Again we'd maximize a collective form of these L_i.

$$L = \prod_{i=1}^{n} L_i$$

We can maximize this L or we can minimize -2log(L) they will both result in same values of parameters.

I am going to skip some mathematical manipulation here . Logistic regression cost function with L2 penalty looks like this :

$$C \ = \ log(1 + exp(-\beta^T XY)) \ + \ \lambda \|\beta\|^2$$

Once you obtained probability scores from logistic regression algorithm, next step is decide a cutoff/threshold on this score so that you can finally arrive at binary prediction. For this we'll look at few performance metrics which can be used for choosing a cutoff.

Performance Metrics for Classification Problems

We understand that no ultimate model and cutoff is going to result in perfect predictions. There are going to be cases where you predicted class to be 1 and in realty it'll be 0 and vice versa. We can make a cross table for predicted and real results like this:

	Predicted=1	Predicted=0
Real=1	TP	FN
Real=0	FP	TN

Here TP = True Positive , Count of cases where real outcome was 1 and prediction was also 1

FP = False Positive , Count of cases where real outcome was 0 but prediction was 1

TN = True Negative , Count of cases where real outcome was 0 and prediction was also 0

FN = False Negative, Count of cases Where real outcome was 1 but prediction was 0

T = TP + FN = Count of all cases where real outcome was 1

N = TN + FP = Count of all cases where real outcome was 0

On the basis of these counts we can define following metrics

$$Accuracy = \frac{(TP + TN)}{(T + N)}$$

$$Miss - ClassificationError = \frac{(FP + FN)}{(T + N)} = 1 - Accuracy$$

Few other important measures are sensitivity,specificity,Precision and Recall. Which are defined as follows:

Sensitivity = Ability of the model to capture all positives.

$$Sensitivity = \frac{TP}{(TP + FN)}$$

Specificity = Ability of the model to capture all negatives.

$$Specificity = \frac{TN}{TN + FP}$$

Precision = How accurate the model's prediction for positives are

$$Precision = \frac{TP}{TP + FP}$$

Recall = Its same as sensitivity.

We can consider many cutoffs on the entire range of probability score [0,1] and calculate confusion matrix for each. Among all the cutoffs that we are considering we should chose one which gives us a good capturing ability of positives as well good accuracy. Next we discuss two such measures which can be used to decide cutoffs

KS

This is defined as follows :

$$KS \ = \ \frac{TP}{P} \ - \ \frac{FP}{N}$$

you can see that this focuses on capturing as many positives as possible but at the same time penalizes for miss-classifying Negatives as Positives. We can choose a cutoff which has max KS value. But this puts equal emphasis on misclassification of both positives and negatives , which might not be the case generally from the business perspective.

There is another measure which enables us to put different priority on either of the class, which we discuss next

F_β Score

This is defined as Follows:

$$F_\beta \ = \ \frac{(1 \ + \ \beta^2)(Precision * Recall)}{\beta^2 Precision + Recall}$$

This β is different from our parameters. Do not get confused with the similar symbol. In here when you use $\beta = 1$; its known as F1 score , which is basically harmonic mean of precision and recall and favors them both equally. Cutoff will be chosen where F_β score is highest.

When $\beta > 1$, cutoff with maximum F_β score favors Precision, where as $\beta < 1$ favors Recall more.

Performance Measure for score model : AUC score

We can consider many cutoffs across our score range of 0 to 1 and calculate Sensitivity and Specificity For all these cutoffs. When we plot these pairs of (Sensitivity , 1-Specificity) , the curve obtained is called ROC curve.

In the curve depiction you'll generally see , Sensitivity labeled as TPR[True Positive Rate] and (1-Specificity) labeled as FPR [False Positive Rate]

For an ideal model this ROC curve comes out to be the triangle as shown below in the image. AUC [Area under the curve] for this is 1, any other score model will have AUC less than 1. Closer to 1 it is better is your score model.

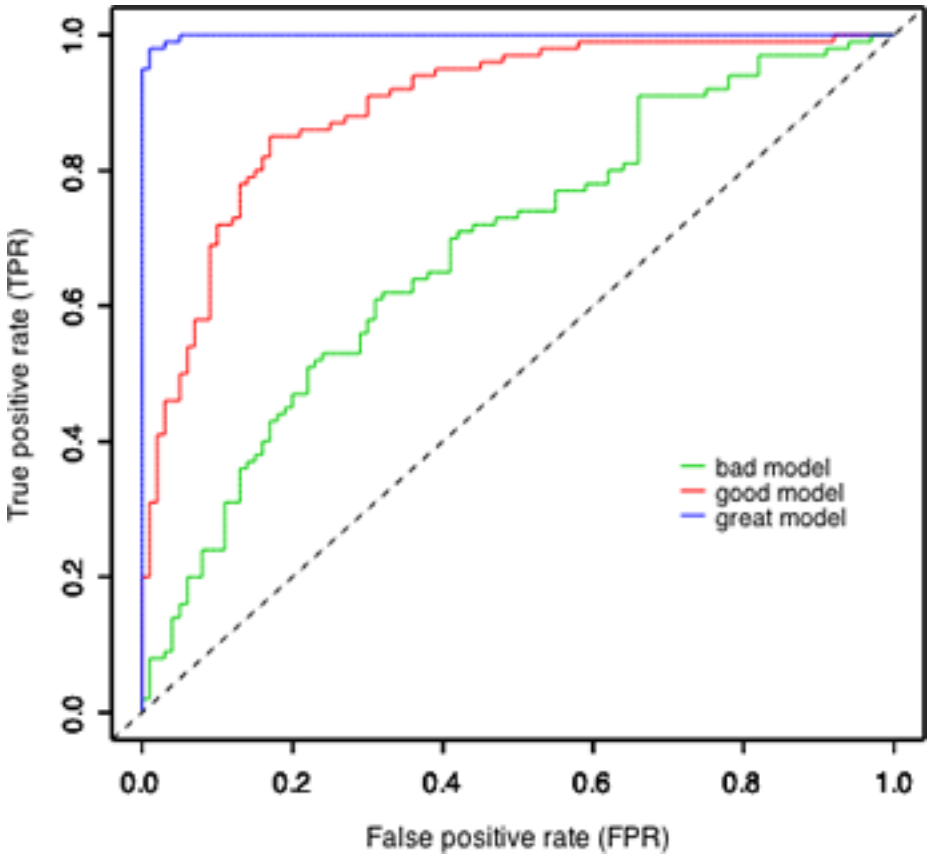


Figure 1: ROC Curve

We'll stop our discussion here. In case of any clarification, please take to QA forum.

Prepared by : Lalit Sachan

Contact : lalit.sachan@edvancer.in