

Introduction to Machine Learning

We'll start our discussion with one thing that people tend to forget over time. Whatever we are learning is about solving business problems. Lets start with; where do we start after we are given a business problem to work on in context of machine learning

Converting Business Problems to Data Problems

At the end of the day; whatever we are doing here is about money. To put that more appropriately, we are developing these techniques to solve business problems. Real business problems do not come conveniently all dressed up as a data problem in general. For example consider this :

- A bank is making too many losses because of defaulters on retail loans

Its a genuine business problem but it isn't a data problem yet on the face of it. So, what is a data problem then. A data problem is a business problem expressed in terms of the data [potentially]available in the business process pipeline.

It majorly has two components:

- Response/Goal/Outcome
- Set of factor/features data which affects our goal/response/outcome

Lets look at the loan default problem and find these components.

- Outcome is loan default, which we would like to predict when considering giving loan to a prospect.
- What factors could help us in doing that. Banks collect a lot of information on a loan application such as financial data, personal information. In addition to that they also make queries regarding credit history to various agencies. We could use all these features/information to predict whether a customer is going to default on their loan or not and then reconsider our decision of granting loans depending on the result.

Here are few more business problems which you can try converting to data problems :

- A marketing campaign is causing spam complaints from the existing customers
- Hospitals can not afford to have additional staff all year round which are needed only when patient intakes become higher than a certain amount
- An e-commerce company wants to know how it should plan for the budget on cloud servers
- How many different election campaign strategies should a party opt for
- What kind of toys should lego launch in India

Three kinds of Problems

If you went through the business problems mentioned above, you'd have realized there are mainly three kind of problems which can then be clubbed into two categories :

1. Supervised

2. Unsupervised

Supervised problems are the problems which have explicit outcomes. Such as default on loan, Required Number of Staff, Server Load etc. Within these , you can see separate kinds.

1. Regression
2. Classification

Regression problems are those where outcome is a continuous numeric value e.g. Sales , Rainfall , Server Load (values over a range with technically infinite unique values possible as outcome and have an ordinal relationship [e.g. 100 is twice as much as to 50]).

Classification problem on the other hand have their outcome as categories [e.g.: good/bad/worse; yes/no ; 1/0 etc], with limited number of defined outcomes .

Unsupervised problems are those where there is no explicit measurable outcome associated with the problem. You just need to find general pattern hidden in the measured factors. Finding different customer segments or electoral segments or finding latent factors in the data comes under such problems.

Now categories of problems can be formally grouped like this:

1. Supervised
 1. Regression
 2. Classification
2. Unsupervised

Our focus in this module will be over **Supervised** problems with an existing outcome which we are trying to predict in context of a regression or a classification problem

Data Problem to Mathematical Problem

Lets discuss key takeaways from the discussion above; which will help us in converting a data problem (of supervised kind) to its mathematical representation.

- We want to make use of historical data to extract pattern so that we can build a predictive model for future/new data
- We want our solution to be as accurate as possible

Now, what do we mean by pattern ? In mathematical terms; we are looking for a function which takes input (the values of factors affecting my response/outcome) and outputs the value of outcome (which we call predictions)

Regression Problem

We'll denote our prediction for i^{th} observation as \hat{y}_i and real value of the outcome in the historical data given to us as y_i . And this function that we talk about is denoted by f . Inputs collectively are denoted as X_i

$$\hat{y}_i = f(X_i)$$

It isn't really possible to have a function which will make perfect predictions [in fact it is possible but not good to have a function which makes perfect prediction, its called over-fitting ; we'll keep that discussion for later] .

Our predictions are going to have errors . We can calculate those errors easily by comparing our predictions with real outcomes .

$$e_i = y_i - \hat{y}_i$$

We would want this error to be as small as possible across all observations. One way to represent the error `across all observation` will be average error for the entire data . However simple average is going to be meaning less , because errors for different observations might have different signs; some negative some positive. Overall average error might as well be zero, but that doesn't mean individual errors don't exist. We need to come up with something for this error `across all observation` which doesn't consider sign of errors . There are couple of ideas that we can consider .

- Mean Absolute Error $\frac{1}{n} \sum_{i=1}^n |e_i|$
- Mean Squared Error $\frac{1}{n} \sum_{i=1}^n e_i^2$

These here are called **Cost Functions**, another popular name for the same is Loss Function . They are also mentioned as just Cost/Loss . In many places , these are considered as averages but simple sum [Sum of absolute errors or sum of squared errors] . It doesn't really matter because difference between them is division by a constant (number of observations); it doesn't affect our pattern extraction or function estimation for prediction.

Among the two that we mentioned here , Mean Squared Error is more popular in comparison to Mean Absolute Error due to its easy differentiability. How does that matter? It'll be clear once we discuss `Gradient Descent` .

Lets take a pause here and understand , what do these cost functions represent? These represent our business expectation of model being as accurate as possible in a mathematically quantifiable way .

We would want our prediction function $f(X_i)$ to be such that , for the given historical data, Mean Squared Error (or whatever other cost function you are considering) is as small as possible .

Classification Problem

It was pretty straight forward, as to what do we want to predict in Regression Problem. However it isnt that simple for classification problem as you might expect . Consider the following data on customer's Age and their response to a product campaign for an insurance policy.

Age	Outcome	Age	Outcome
20	NO	30	NO
20	NO	30	NO
20	NO	30	YES
20	NO	30	YES
20	YES	30	YES
25	NO	35	NO
25	NO	35	YES
25	NO	35	YES
25	YES	35	YES
25	YES	35	YES

If I asked you , what will be the outcome if somebody's age is 31, according to the data shown above . Your likely answer is **YES** as you see that majority of the people with increasing age have said **YES** .

If I further asked you whats the outcome for somebody with age 34, your response will again be **YES** . Now, whats the difference between these two cases . You can easily see that outcome is **more likely** to be **YES** when the age is higher . This tells us that , we are not really interested in absolute predictions **YES/NO** ; but in the probability of the outcome being **YES/NO** for given inputs.

More formally we are interested in this : $P(y_i = YES|X_i)$

YES/NO at the end of the day are just labels. We'll consider them to be 1/0 to make our life mathematically easy; as will be evident in some time. Also to keep the notation concise , we'll use just p_i instead of $P(y_i = 1|X_i)$

Now that we have figure out that we want to predict p_i , Lets see how do we write that against our prediction function $f(X_i)$

range of p_i being a probability is between 0 to 1. However $f(X_i)$ theoretically can be anything between $-\infty$ to $+\infty$. it wont make sense to write $p_i = f(X_i)$

we need to apply some transformation on any one side to ensure that the ranges match. sigmoid or logit function is one such transformation which is popular in use.

$$p_i = \frac{1}{1 + e^{-f(X_i)}}$$

another format [just rearranged terms] for the same is

$$\log\left(\frac{p_i}{1 - p_i}\right) = f(X_i)$$

Finally, we are clear about what we want to predict and how it is written against our prediction function . We can now start discussing what are our business expectation from this and how do we represent the same in form of a cost/loss function .

We'll ideally want the probability of outcome being 1 to 100% when outcome in reality is 1 , and probability of outcome being 1 to be 0% when outcome in reality is 0 . But, as usual , perfect solution is practically not possible. Closest compromise will be that p_i is as close to 1 as possible when outcome in reality is 1 and it is as close to 0 as possible when outcome in reality is 0.

$$\begin{aligned} p_i \uparrow & \quad \forall \quad y_i = 1 \\ p_i \downarrow & \quad \forall \quad y_i = 0 \end{aligned}$$

since p_i takes value in the range $[0,1]$, we can say that when p_i goes close to 0, it implies that $1 - p_i$ goes close to 1. So the above expression can be written as :

$$\begin{aligned} p_i \uparrow & \quad \forall \quad y_i = 1 \\ (1 - p_i) \uparrow & \quad \forall \quad y_i = 0 \end{aligned}$$

This is still at intuition level, we need to find a way to convert this idea into a mathematical expression which can be used as a cost function . Consider this expression :

$$L_i = p_i^{y_i} (1 - p_i)^{(1-y_i)}$$

Expression L_i is known as likelihood because it gives you the probability of the real outcome that you get. This expression takes value p_i when $y_i = 1$ and takes value $(1 - p_i)$ when $y_i = 0$. Using the idea that we devised above, we can rewrite it like this :

$$\begin{aligned} L_i \uparrow & \quad \forall \quad y_i = 1 \\ L_i \uparrow & \quad \forall \quad y_i = 0 \end{aligned}$$

This implies that we want the likelihood to be as high as possible irrespective of what the outcome (1/0) is. Another way to express that is , that we want likelihood to align with whatever the real outcome is . As earlier we are not concerned with likelihood of a single observation, we are interested in it at over all level. Since likelihood is nothing but a probability , if we want to calculate likelihood of entire data, it'll be nothing but multiplication of all individual likelihoods.

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

This is our cost function which we need to maximise . we want such an $f(X_i)$ for which L is as high possible for the given historical data. However this is not the form in which this is used . Since optimizing a quantity where individual terms are multiplied with each other is a very difficult problem to solve we'll instead use $\log(L)$ as our cost function , which makes individual terms getting summed up instead .

$$\log(L) = \sum_{i=1}^{i=n} L_i$$

To make this a standard minimization problem , instead of maximizing $\log(L)$, minimization of $-\log(L)$ is considered

Finally , cost function for classification problem is $-\log(L)$, also known as **negative log likelihood**

other forms of the same which you'll get to see :

$$-\log(L) = -\sum_{i=1}^{i=n} [y_i \log(p_i) + (1 - y_i) \log(1 - p_i)]$$

this can also be written in terms of prediction function as follows :

$$-\log(L) = -\sum_{i=1}^{i=n} [y_i f(X_i) - \log(1 + e^{f(X_i)})]$$

Classification Model Evaluation and Probability Scores to Hard Classes

We discussed earlier that prediction model for classification output probabilities as outcome. In many case that'll suffice as a way of scoring our observation in terms of most likely to least likely. However in many cases we'd eventually need to convert those probability scores to hard classes.

For example you might build a model to detect whether a patients symptoms amount to them having certain disease or not. Say your model gives a probability for the patient having that particular disease. Now when a doctor goes to use your model, they need to arrive at a decision whether to start the patient on the needed medication or not.

The way is to figure out a cut-off/threshold for the probability scores where we can say that, obs with higher score than this will be classified as 1 and others will be classified as 0. Now, the question becomes; how do we come up with this cut-off.

Confusion Matrix and associated measurements

Irrespective of what cut-off we choose, the hard class prediction rule is fixed. If probability score is higher than the cutoff then the prediction is 1 otherwise 0 [given, we are predicting probability of outcome being 1]. Any cut-off decision results in some of our predictions being true and some of them being false. Since there are are only two hard classes, we'll have 4 possible cases .

- When we predict 1 [+ve] but in reality the outcome is 0[-ve] : False Positive
- When we predict 0 [-ve] but in reality the outcome is 1[+ve] : False Negative
- When we predict 1 [+ve] and in reality the outcome is 1[+ve] : True Positive
- When we predict 0 [-ve] and in reality the outcome is 0[-ve] : True Positive

This is generally displayed in a matrix format known as confusion matrix .

	Positive_predicted	Negative_predicted
Positive_real	TP	FN
Negative_real	FP	TN

TP : Number of true positive cases

TN : Number of true negative cases

FP : Number of false positive cases

FN : Number of false negative cases

Using these figures we can come up with couple of popular measurements in context of classification

How accurate our predictions are

$$\text{Accuracy} = \frac{TP+TN}{\text{Total obs}}$$

How well we are capturing/recalling positive cases

$$\text{Sensitivity or Recall} = \frac{TP}{TP+FN}$$

How well we are capturing negative cases

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

How accurately we have been able to predict positive cases

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

Some might suggest that we can take any one of them, and measure for all scores, and decides that score to be our cutoff where the taken measurement is highest for the training data. However none of these above mentioned measurements take care of our business requirement of good separation between two classes. Here are the issues associated with these individually if we consider them as candidate for determining cutoffs.

Accuracy This works only if our classes are roughly equally present in the data. However generally this is not the case in many business problems. For example, consider campaign response model. Typical response rate is 0.5-2%. Even if predict that none of the customers are going to subscribe to our campaign; accuracy will be in the range 98-99.5%, quite misleading.

Sensitivity or Recall We can arbitrarily make Sensitivity/Recall 100% by predicting all the cases as positive. Which is kinda useless because it doesn't take into account that labeling lot of negative cases as positive should be penalized too.

Specificity We can arbitrarily make Specificity 100% by predicting all the cases as Negative. Which is kinda useless because it doesn't take into account that labeling lot of positive cases as negative should be penalized too.

Precision We can make precision very high by keep cut-off too high and thus predicting very few sure shot cases as positive, but in that scenario our recall will be down to dumps.

So it turns out that, these measures are a good look into how our model is doing, but none of them taken individually can be used to determine proper cut-off. Following are some proper measures which give equal weight to goal of capturing as many positives as possible and at the same time; not labeling too many negative cases as positives.

$$KS = \frac{TP}{TP+FN} - \frac{FP}{TN+FP} = S_p + S_n - 1$$

You can see that KS will be highest when we recall is high but at the same time we are not labeling a lot of negative cases as positive. We can calculate KS for all the scores and chose that score as our ideal cut-off for which KS is maximum.

There is another measure which lets you give different weights for your precision or recall. There can be cases in real business problems where you'd want to give more importance to recall over precision or otherwise. Consider a critical test which determines whether someone has a particular aggressively fatal disease or not. In this case we wouldn't want to miss out on positive cases and wouldn't really mind some of the negative cases being labeled as positive.

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

Notice that value of β here will determine; how much and what are we going to give importance to. For $\beta=1$, equal importance is given to both precision and recall. When $1 > \beta \rightarrow 0$, F_{β} score favors Precision and results in high probability score being chosen as cutoff. On the other hand when $\beta > 1$ and as $\beta \rightarrow \infty$ it favors Recall and results in low probability scores being chosen as cutoff.

Now these let us find a proper cutoff given a probability score model. However so far we haven't discussed how do asses, how good the score is itself. Next section is dedicated to the same.

ROC curve and AUC Score

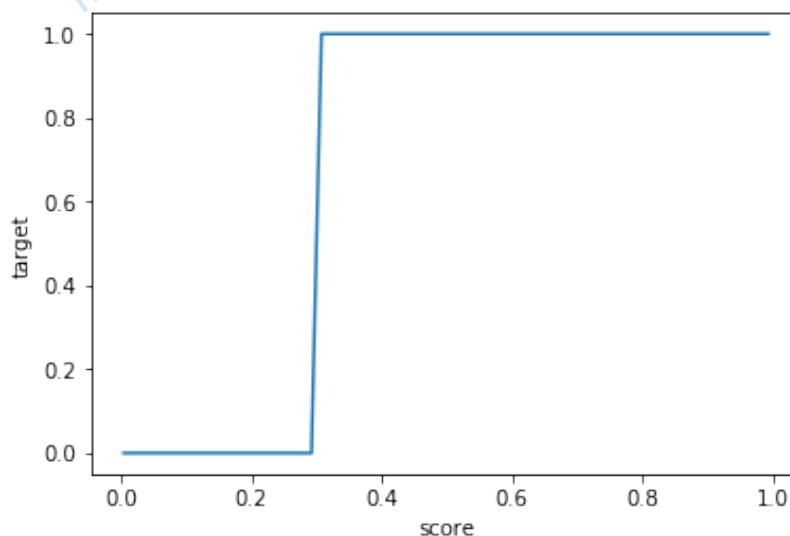
When we asses performance of a classification probability score, we try to see how it stacks up with an ideal scenario. For an ideal score, there will exist a clean cutoff; meaning, there will be no overlap between two classes when chose that cut-off. There will be no False Positives Or False Negatives

Consider a hypothetical scenario where we have an ideal prob score like given below.

```
1 d=pd.DataFrame({'score':np.random.random(size=100)})
2 d['target']=(d['score']>0.3).astype(int)
```

```
1 import seaborn as sns
```

```
1 sns.lineplot(x='score',y='target',data=d)
```

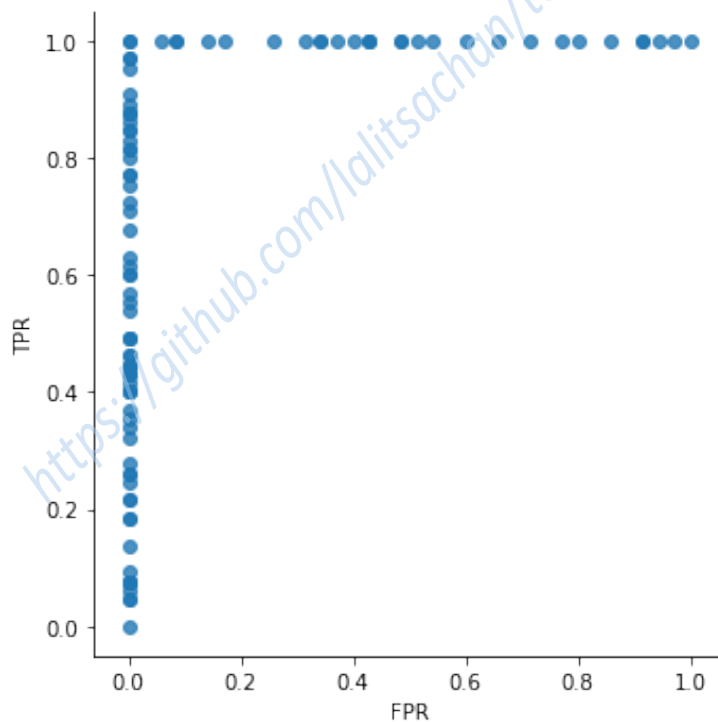


For this ideal scenario, we are going to consider many cutoffs between 0-1 and calculate **True Positive Rate** [Same as Sensitivity] and **False Positive Rate** [Same as 1-Specificity]. And plot those . The resultant plot that we'll get is known as ROC Curve . Lets see how that looks

```

1  TPR=[]
2  FPR=[]
3  real=d['target']
4  for cutoff in np.linspace(0,1,100):
5      predicted=(d['score']>cutoff).astype(int)
6      TP=((real==1)&(predicted==1)).sum()
7      FP=((real==0)&(predicted==1)).sum()
8      TN=((real==0)&(predicted==0)).sum()
9      FN=((real==1)&(predicted==0)).sum()
10
11     TPR.append(TP/(TP+FN))
12     FPR.append(FP/(TN+FP))
13
14 temp=pd.DataFrame({'TPR':TPR,'FPR':FPR})
15
16 sns.lmplot(y='TPR',x='FPR',data=temp,fit_reg=False)

```



Its perfectly triangular , and area under the curve is 1 for this ideal scenario, however lets see what happens if we make it not so ideal scenario and add some overlap. [we'll flip some targets in the same data]

```

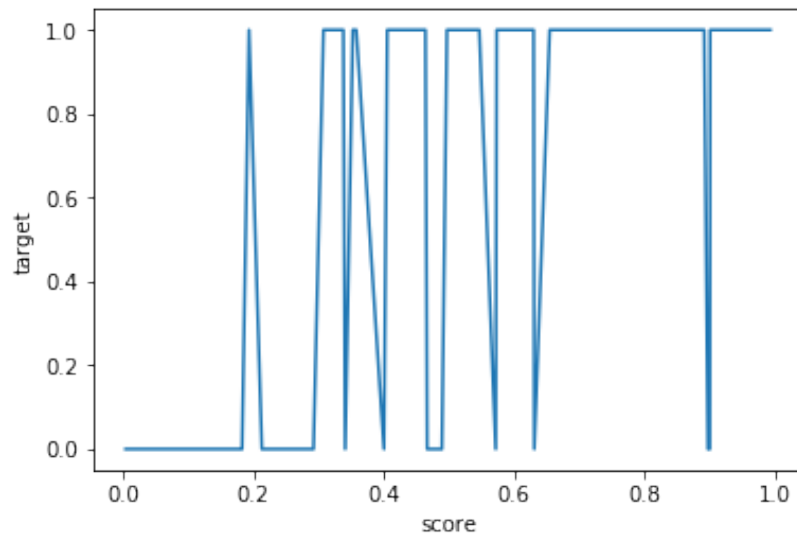
1  inds=np.random.choice(range(100),10,replace=False)
2  d.iloc[inds,1]=1-d.iloc[inds,1]

```

```

1  sns.lineplot(x='score',y='target',data=d)

```

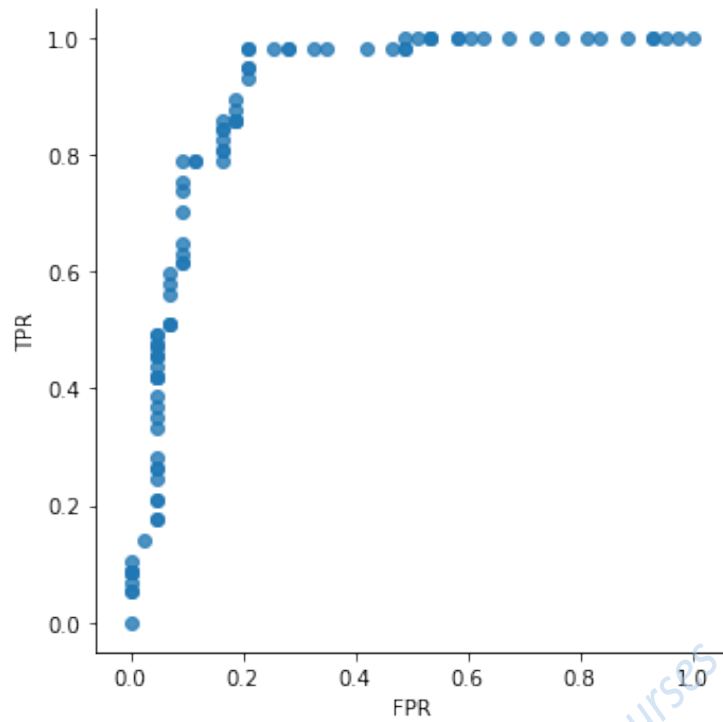


you can see that there is lot of overlap now and , there can not exist a clear cutoff. Lets see how the ROC curve looks for the same

```

1  TPR=[ ]
2  FPR=[ ]
3  real=d['target']
4  for cutoff in np.linspace(0,1,100):
5      predicted=(d['score']>cutoff).astype(int)
6      TP=((real==1)&(predicted==1)).sum()
7      FP=((real==0)&(predicted==1)).sum()
8      TN=((real==0)&(predicted==0)).sum()
9      FN=((real==1)&(predicted==0)).sum()
10
11     TPR.append(TP/(TP+FN))
12     FPR.append(FP/(TN+FP))
13
14 temp=pd.DataFrame({'TPR':TPR, 'FPR':FPR})
15
16 sns.lmplot(y='TPR',x='FPR',data=temp,fit_reg=False)

```



You can try introducing more overlap and you'll see the ROC curve move away from the ideal scenario. Area Under the Curve(AUC Score) will become less than one, more close to 1 it is more close to ideal scenario it is, better is your model .