Classification Models Performance Evaluation — CAP Curve

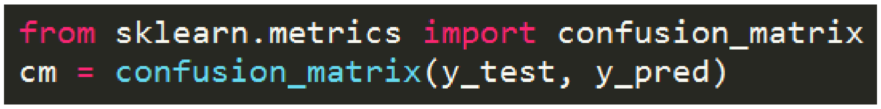
Alaa Khaled Follow
Aug 1, 2017 · 6 min read

Let's start by defining what is meant by Classification, Classification is the process of trying to assign something to one of the available groups. You may have 2 groups (Binary Classification) or more than 2 groups (Multi-class Classification).

Classification Algorithms includes: (Logistic Regression, K-Nearest Neighbor, Support Vector Machine, and Naive Bayes…etc)

For a data scientist it is really important to make sure how good is your classification model. There are some famous possible ways to evaluate your model. They can be listed as following:

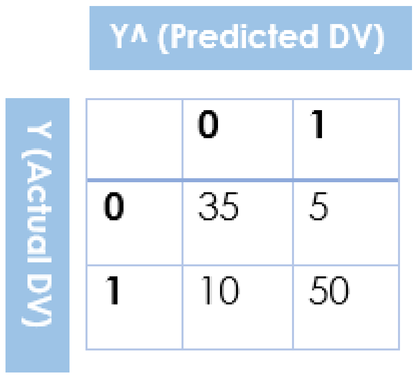
1. **Confusion Matrix—**It can be calculated easily using the Scikit-Learn Library implementation. You just have to feed it a vector that contains the predictions of your dependent variable ***y*** ^ and a vector of the actual values of your dependent variable ***y***



```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
```

scikit-learn confusion matrix module

Now after you calculated your confusion matrix which, will be a 2*2 matrix for any binary classification problem, the resulting matrix would be like this



Y^ (Predicted DV)

Y (Actual DV)

	0	1
0	35	5
1	10	50

confusion matrix

A confusion matrix consists of 4 values which are very valuable to know for every data scientist:

1. False Positive (FP)—instances that are incorrectly classified as Positive Class instances (0,1) = 5
2. False negative (FN)—instances that are incorrectly classified as Negative Class instances (1,0) = 10
3. True Positive (TP)—instances that are correctly classified as Positive Class instances (1,1) = 50
4. True Negative (TN)—instances that are correctly classified as Negative Class instances (0,0) = 35

Total instances of the dataset (training + test instances) = 5+10+50+35 = 100

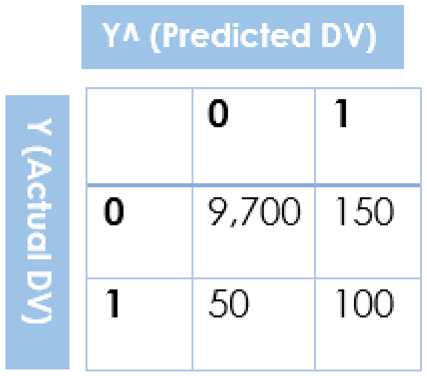
Now, we can calculate 2 important percentages:

1- Accuracy rate = Correct/ total = (50+35)/ 100 = 85%

2- Error rate = Incorrect/ total = (5+10)/ 100 = 15%

Now, I should celebrate after building such a classifier it has such a good accuracy rate, …or Should I ?!

Let's see, please follow me in the following scenario "You are a data scientist who works in a bank and has built a classification model to classify fraud and non-fraud transactions, you want to evaluate your model so you decided to calculate the confusion matrix and that was the result:”



Y^ (Predicted DV)

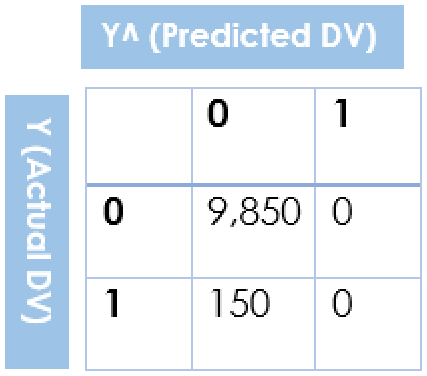
Y (Actual DV)

	0	1
0	9,700	150
1	50	100

confusion matrix — Fraud detection

By analyzing the confusion matrix we can say that we have a pretty good classifier with Accuracy rate = 9,800/ 10,000 = 98%

But The data scientist have a pretty weird idea that he wants to try; The idea is to stop the classifier from classifying any of the transactions as fraud (positive class ‘1’) then, he calculated the new confusion matrix and it was as the following:



Y^ (Predicted DV)

Y (Actual DV)

	0	1
0	9,850	0
1	150	0

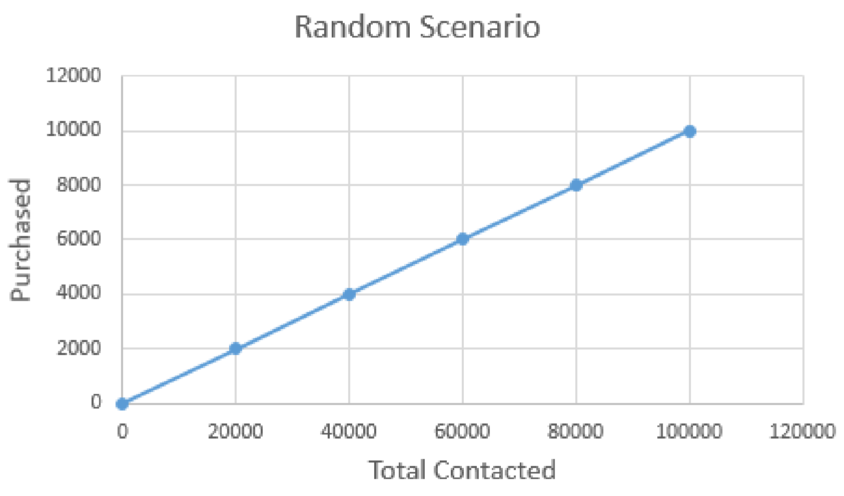
The classifier Accuracy rate = 9,850/ 10,000 = 98.5% which means there is a 0.5% increase in the accuracy rate although the classifier is not working properly!

And that is called **Accuracy Trap** So we definitely say that measuring accuracy rate is not enough to answer the question ‘How good is your classifier?!'

The solution is to try another measurement method which is

2. Cumulative Accuracy Profile (CAP) Curve—It is a more robust method to assist our machine model. To understand the intuition behind it, You have to follow me in the following scenarios:

Scenario#1—Imagine that you as a data scientist work in a company that want to promote its new product so they will send an email with their offer to all the customers and usually 10% of the customer responses and actually buys the product so they though that that will be the case for this time and that scenario is called the *Random Scenario*.



Scenario#2—You still work in the same company but this time you decided to do it in a more systematic way:

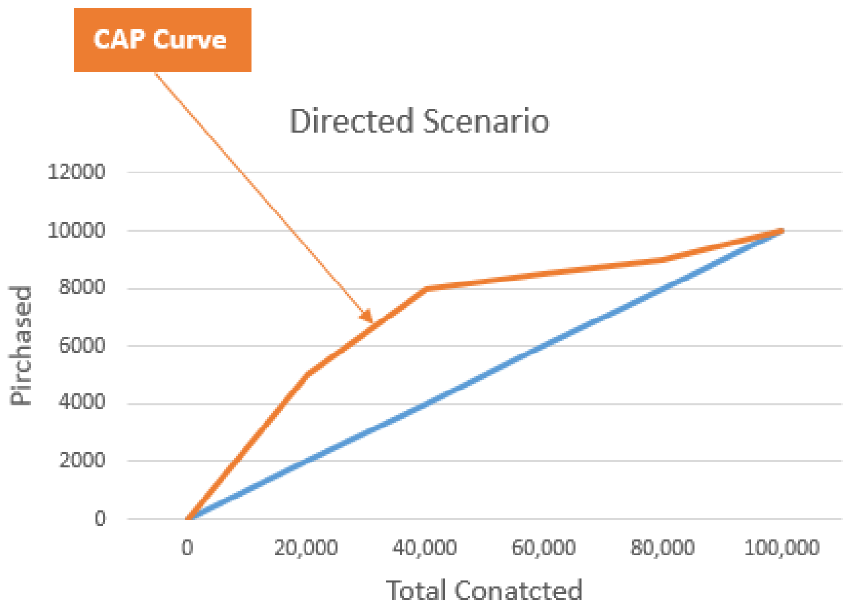
1. Inspect your historical data and take a group of customers who actually bought the offer and try to extract those information [browsing device type (mobile or laptop), Age, Salary, Savings]

2. Measure those factors and try to discover which of them affects the number of Purchased products or in other words fit the data to a Logistic Regression model.

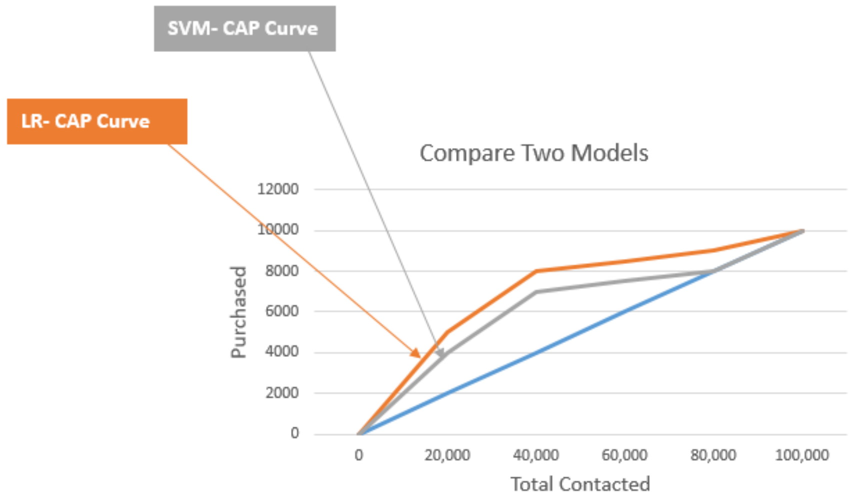
3. Make a prediction of which customers are more likely to purchase the product.

4. Then specially target those people which you predicted are more likely to buy the product.
5. Then by measuring the response of those targeted group represented in that curve 'CAP Curve'.

We definitely can notice the improvement; when you contacted 20,000 targeted customers you got about 5,000 positive responses where in scenario#1 by contacting the same number of customers, you got only 2,000 positive responses.



So, the idea here is to compare your model to the random scenario and you can take it to the next level by building another model maybe a Support Vector Machine (SVM)/ Kernel SVM model to compare it with your current logistic regression model.

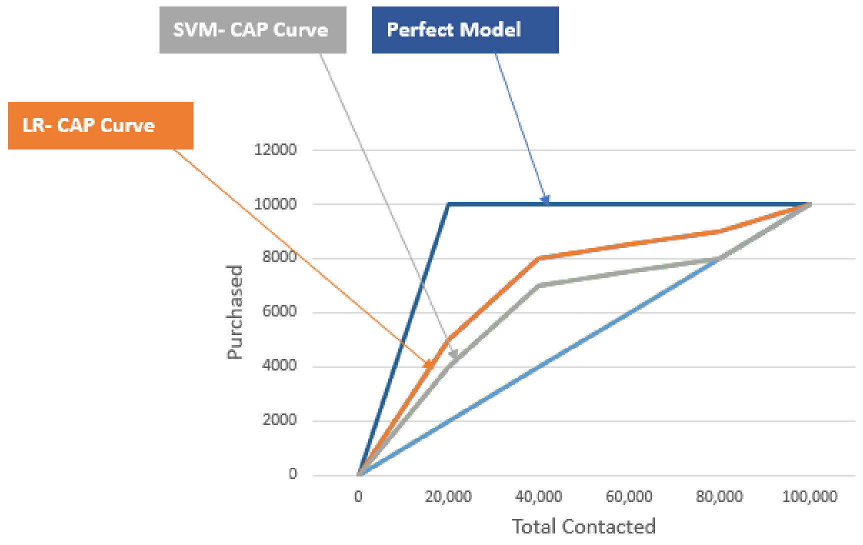


But, How to analyze the resulting graph ?

> *The better your model, the larger will be the area between its CAP curve and the random scenario straight line.*

Hypothetically we can draw the so called The **Perfect Model** which represents a model which is kind of impossible to build unless you have some sort of a Crystal Ball . It shows that when sending the offer to 10,000 possible customer you got a perfect positive response where all contacted people bought the product.

Plotting such a hypothetical model will help us as a reference to evaluate your models CAP curves.



There are 2 approaches to analyze the previous graph:

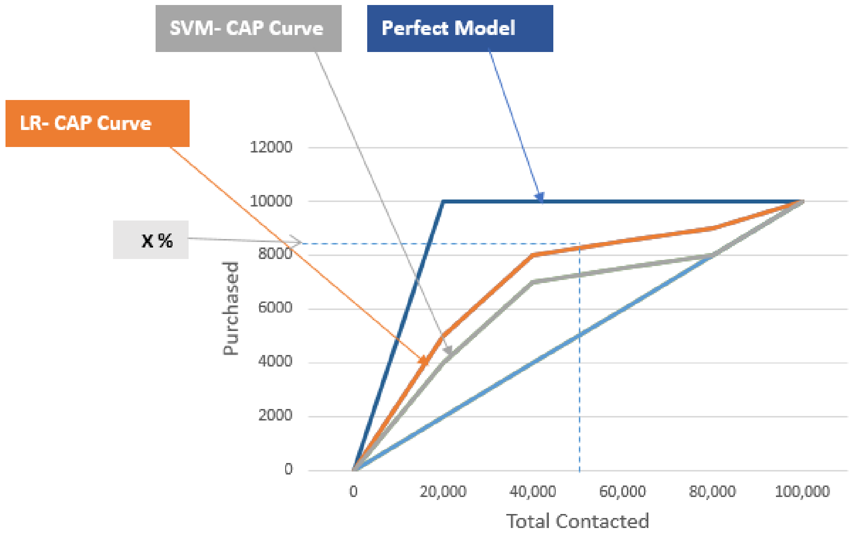
First —

1. Calculate area under the Perfect Model Curve (***aP***)
2. Calculate area under the Perfect Model Curve (***aR***)
3. Calculate Accuracy rate(**AR**) = ***aR/ aP***; as (**AR**)~1 (The better is your model) and as (**AR**)~0 (The worse is your model)

Second —

1. Draw a line from the 50% point (50,000) in the Total Contacted axis up to the Model CAP Curve
2. Then from that intersection point, Project it to the Purchased axis
3. This X% value represents how good your model is:

- If **X < 60%** /(6000) then you have a rubbish model
- If **60% < X < 70%** /(7000) then you have a poor model
- If **70% < X < 80%** /(8000) then you have a good model
- If **80% < X < 90%**/ (9000) then you have a very good model
- If 9**0% < X < 100%** / (10,000) then your model is too good to be true! what I mean is that, this usually happens due **Overfitting** which is definitely not a good thing as your model will be good in classifying only the data it is trained on but very poor with new unseen instances.



References

Machine Learning A to Z Hands On Python and R In Data Science Course from Udemy—Which is a very good course BTW!