Homework Assignment N°3

BML36
Thibault Douzon
Rajavarman Mathivanan

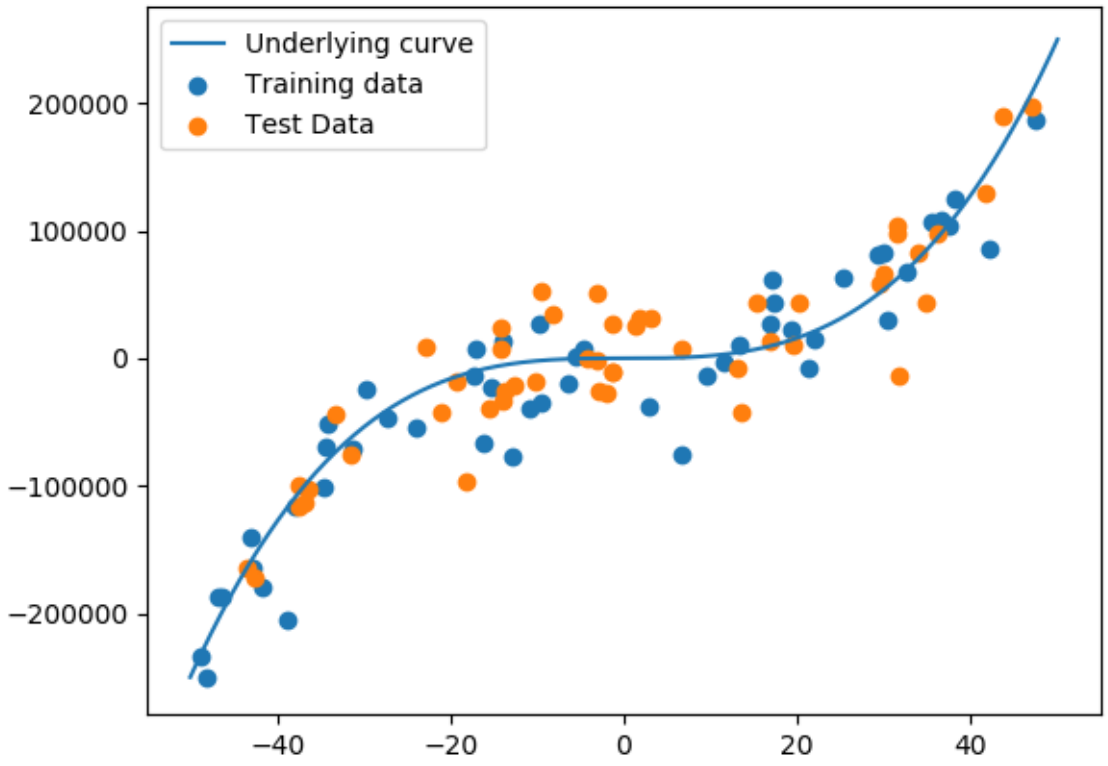
September 18th, 2018

Contents

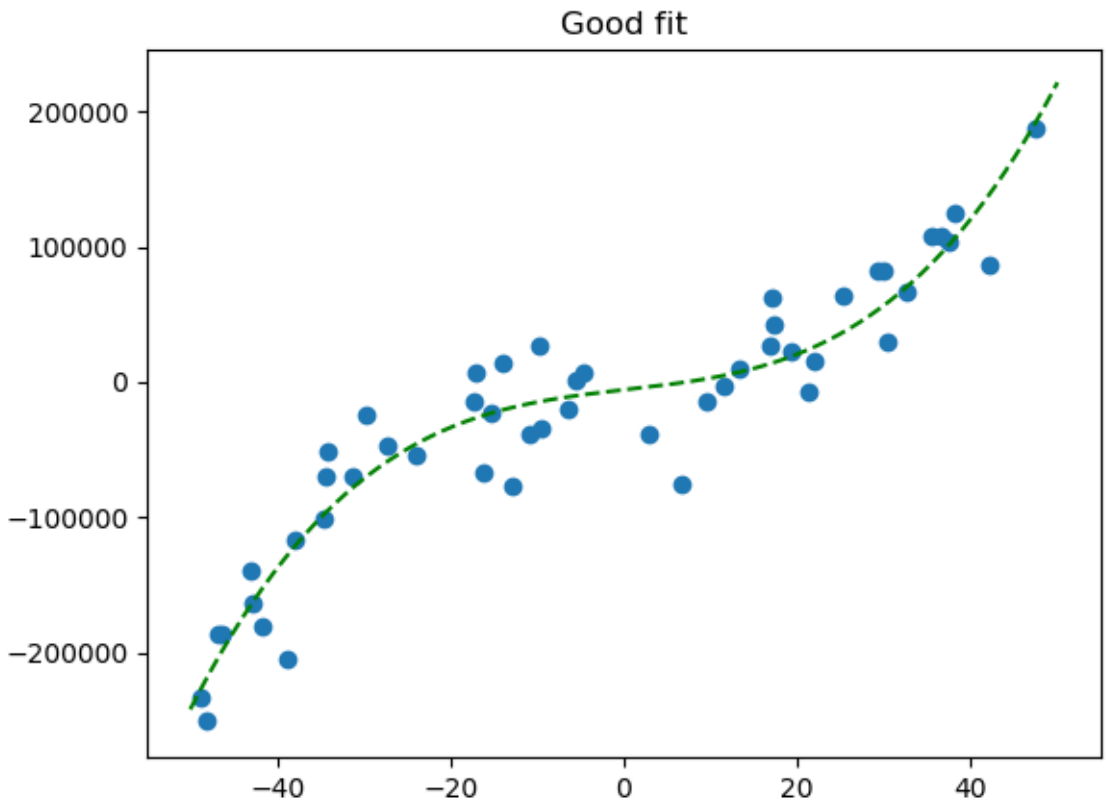
1 Exercise 1: K-CV, under & over-fitting

1.1 Part a

To show what are under and over-fitting, we will use polynomial regression with different degrees on sample data. We generate our data points using a degree 3 polyomial and adding gaussian noise. Here is the dataset:

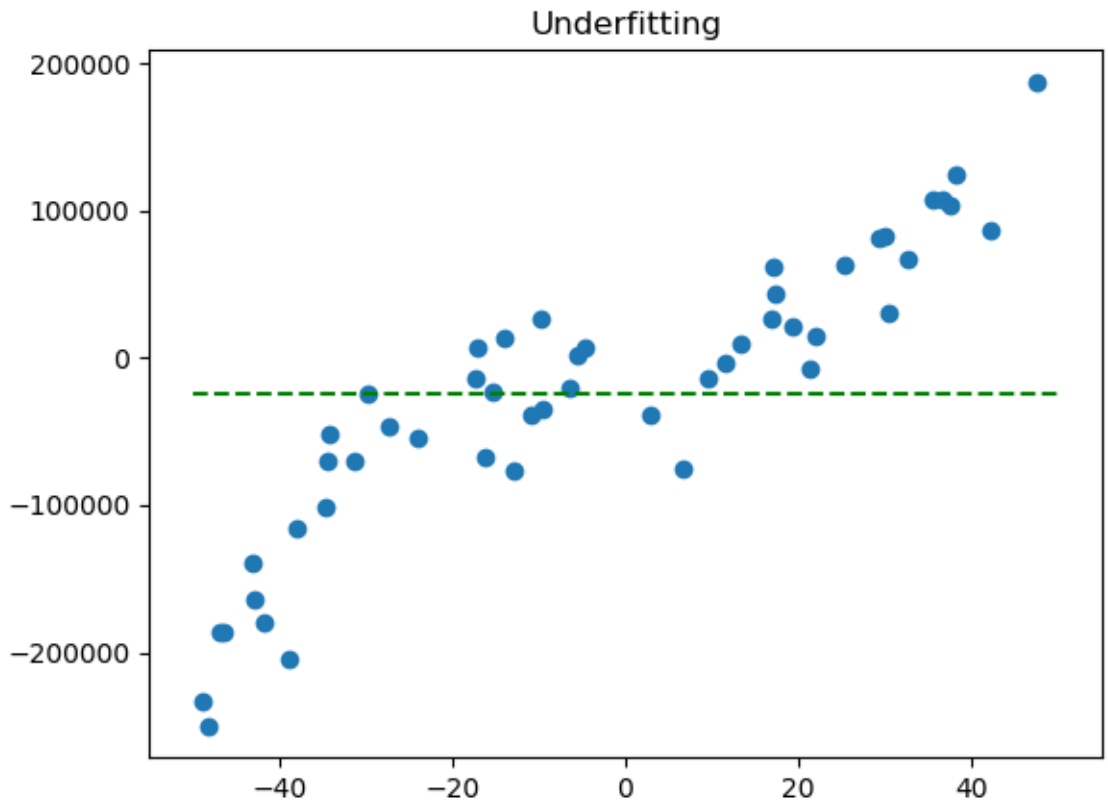


Our model minimizes the sum of the squared error. Because our underlying function is a degree 3 polynomial, the best model we computed is the following, also degree 3:



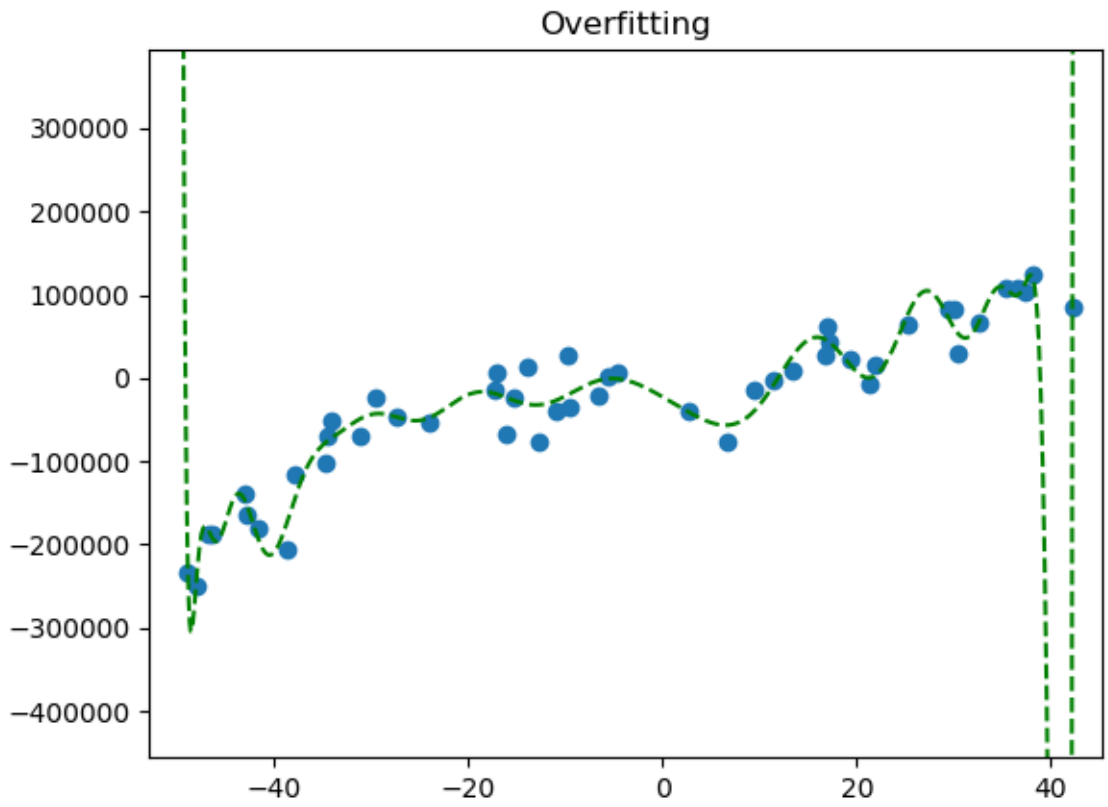
We can see that the green line follows accordingly our dataset, without trying to go through every points. We can expect this model to generalize well.

- ☐ Under-fitting comes when the chosen model is not complex enough to represent the learning data and the underlying function.
  An under-fit model suffer from high *bias* and high error on learning data.
  We would say from such a model that it does not fit enough to the data.
  An example is our degree 0 polynomial: an horizontal line can't describe correctly our underlying degree 3 polynomial:



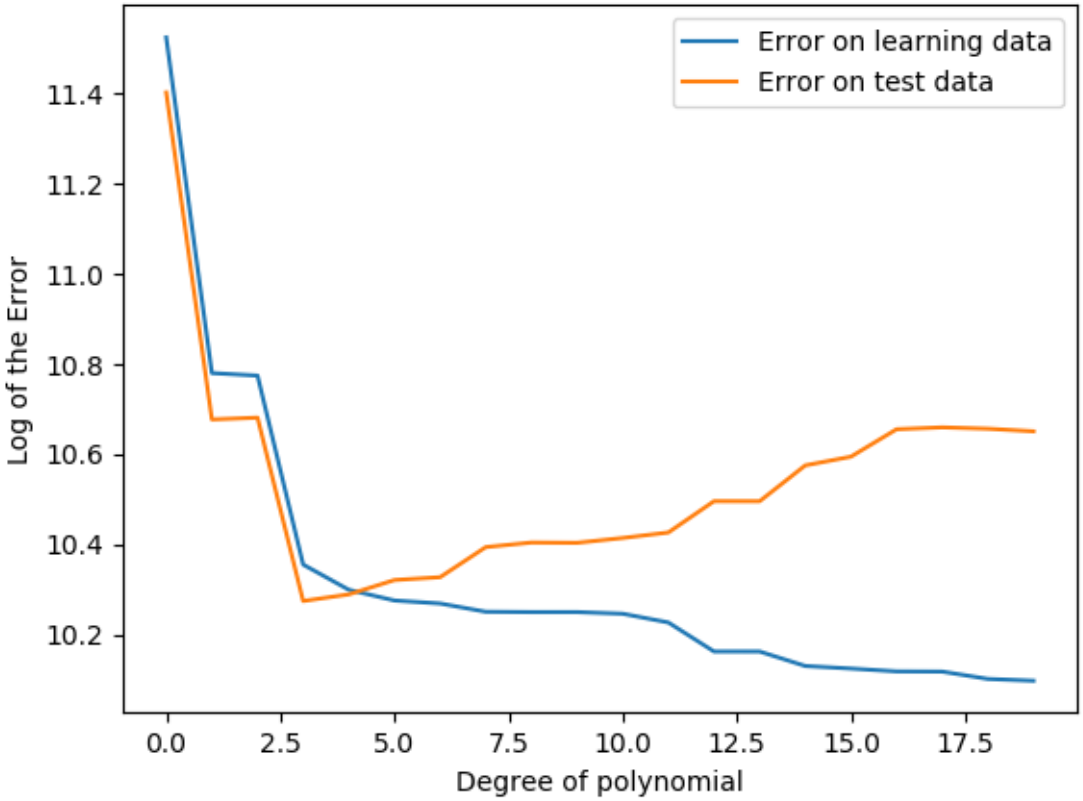
The x and y axis represent the x and y coordinates for our points such that $f(x) \approx y$.
We can see on the previous plot that the green line is close to points in the middle but makes a non negligeable error on extreme points.
When controlling the learning with a test dataset, we know we are under-fitting when both the error on the training and the test data is high: our model does not describe correctly the learning data and can't generalize to other data.

☐ Over-fitting comes when to model sticks too much to the learning data. In some sense, our model tries to interpret the noise present in the learning data as if it were significant.
An over-fit model suffer from high *variance* and high error on testing data.
An example of over-fit model is our degree 20 polynomial model:

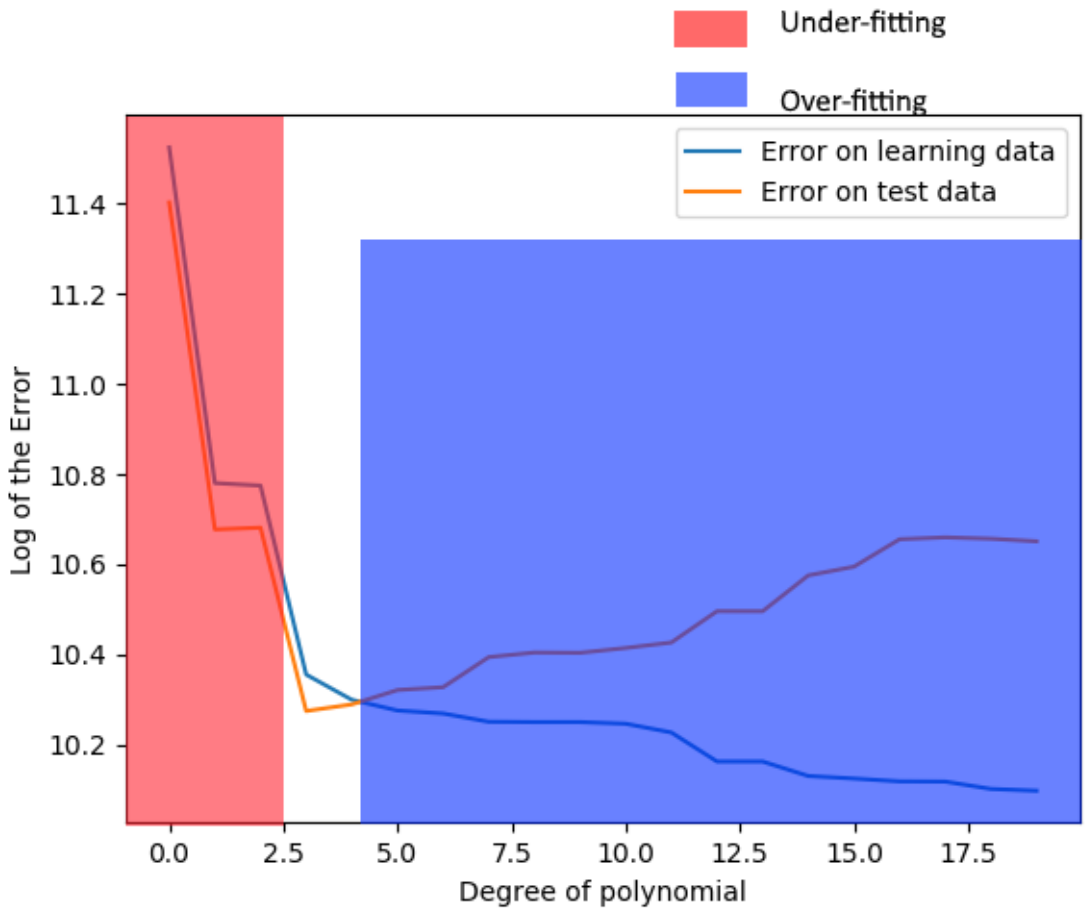


As before, this plot shows the learning data and the curve of the model. We can see the curve tries to pass through every points even those remote from the underlying curve (which suffer from high noise).
Another sign of over-fitting is very high value for parameters of the model (absolute value). This symptom is why we can use L1 or L2 regularization to prevent us from over-fitting.

To better see both phenomena in one figure, we can plot the error in term of the degree of our model. If we do so for both training and testing dataset, we get the following plot:



On this figure we can see that the error on learning data is globally always decreasing (very steeply at the beginning, then slowly) when the complexity of the model increases. But the curve of the error on the testing data does not follows the same pattern: it first decreases up to some degree and then increases again. We can divide the previous figure in two areas, one represents under-fitting and the other over-fitting. And between the two lays the models that may generalyze well:



As we said before, in the red area both curves decrease steeply: this is the under-fitting zone. In the blue one, error on learning data continue decreasing but slowly while error on testing data increases: this is the over-fitting zone.

1.2 Part b

It is not a good idea to simply divide our dataset in training/testing and test each model on the test set to select the best model because it could result in over-fitting and model the noise in the test set instead of the underlying model. Especially if we provide more data to the learning set than to the test set: the smaller the test set, the more noisy is our testing, and the more probable a model can fall in over-fitting.
Indeed, nothing tells us the the best model on the test set will generalize well and isn't over-fitting the test sample also.
To prevent us from over-fitting the data that aren't in the learning set, we need validation set, to finally evaluate the performance of our model on brand new data. This validation set will allow us to chose the best hyperparameters for our models and chose the best model between the trained ones.
The terms "validation" and "test" are very ambiguous and often interchanged and some call the validation set the holdout set.

WEIRD test / validation

livre -> validation = à chaque étape, si l'E augmente -> stop training
test = à la fin pour choisir le meilleur
diapos -> validation = ???
test = à chaque étape si l'E augmente -> stop

1.3 Part c

We use K-fold cross-validation because we want to use as much data as possible for training. But keeping away test and validations sets from our training set. K-CV allows us to use the whole dataset for training and still keep an eye on the over-fitting whereas data in the validation set will never be used for learning.
Pros:

- The whole dataset can be used for training. Very useful when we don't have many data.
- Can be used to determine the hyperparameters of our models: train all the different models with different parameters K times each and pick the best at the end (the one with the minimal mean of the error on the evaluation part remaining)
- It provides an accurate estimation of the performance of the model

Cons:

- The learning time is increased by a factor K: if we split our dataset in 10 parts, we will need 10 different trainings and thus it will last 10 times more to train.
- The number of folds we make is another parameter: when the dataset is small we can split in as many sets as the number of data, but it's too computer intensive when the dataset gets large.

2 Exercise 2: ROC curve

2.1 Part a

We can represent this situation with a table where a column represents what the model predicts (positive or negative) and the rows represents the reality of the data. In the heading, the probability that the inequation is true is between the parenthesis

	$x < \theta \rightarrow (\theta)$	$x > \theta \rightarrow (1-\theta)$
P	$\mathrm{TP} = \theta\mathrm{P}$	$\mathrm{FN} = (1-\theta)\mathrm{P}$
N	$\mathrm{FP} = \theta\mathrm{N}$	$\mathrm{TN} = (1-\theta)\mathrm{N}$

We deduce the following from the previous table:

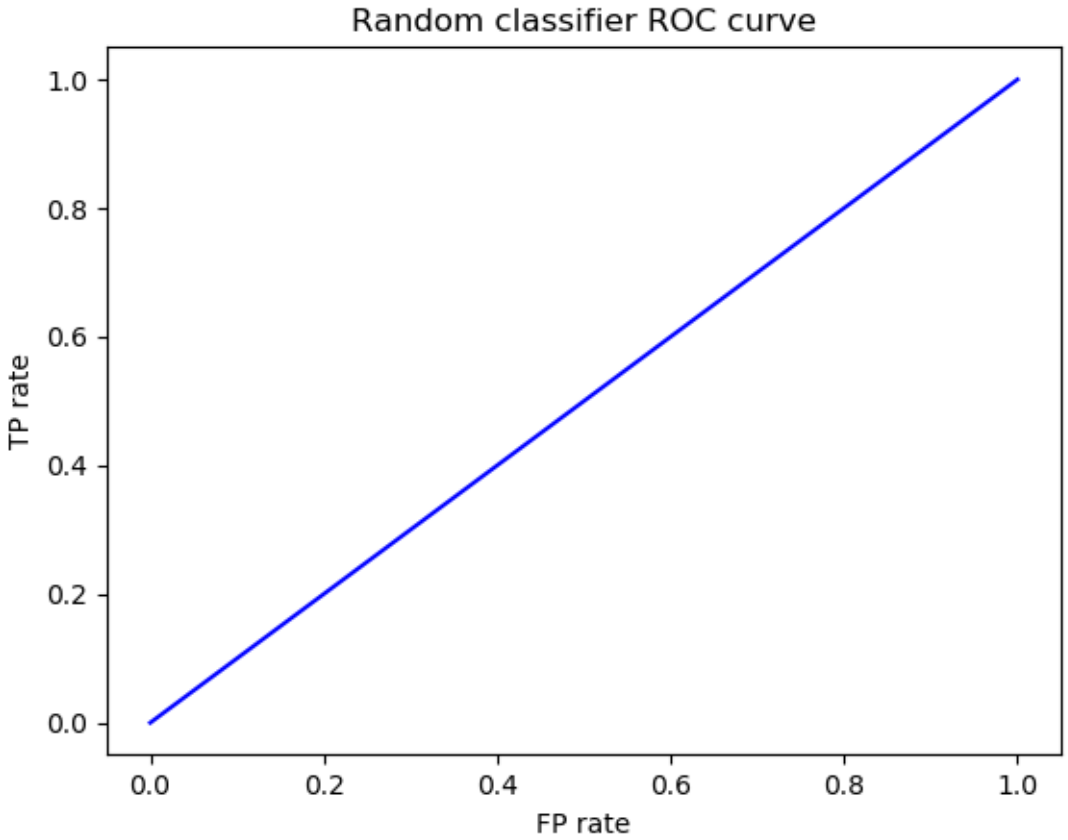
$$\mathrm{TP}_{\mathrm{rate}} = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}} = \frac{\theta\mathrm{P}}{\theta\mathrm{P}+(1-\theta)\mathrm{P}} = \theta$$

$$\mathrm{FP}_{\mathrm{rate}} = \frac{\mathrm{FN}}{\mathrm{FP}+\mathrm{TN}} = \frac{\theta\mathrm{N}}{\theta\mathrm{N}+(1-\theta)\mathrm{N}} = \theta$$

For a random classifier such as this one, TP and FP rate do not depend on P and N but only on θ.

2.2 Part b

From the previous question, we can draw the ROC curve of the classifier. Because we have the following identity $\text{TP}_{\text{rate}} = \text{FP}_{\text{rate}} = \theta$, when we plot the TP_{rate} in terms of the FP_{rate} we get the identity function (because both quantities are always equal to each other).



This is a well known result: a random classifier has a ROC curve that splits the ROC space in two.

2.3 Part c

Now it is easy to compute the AUC, this ROC curve is a line following the diagonal of the base square, thus the are under the curve is half the area of the square.

$$\text{AUC} = \frac{1}{2}$$

We could get to the same result using calculus: the underlying function is $f(x) = x$.
Calculus gives us the formula for the AUC:

$$\text{AUC} = \int_0^1 f(x)dx$$

$$\text{AUC} = \int_0^1 x dx = \left[\frac{x^2}{2}\right]_0^1$$

$$\text{AUC} = \frac{1}{2}$$

2.4 Part d

The equal error rate points lay on the $x = y$ curve, exactly like our random classifier. This ROC curve seems to only have two fixed points: $(0, 0)$ and $(1, 1)$