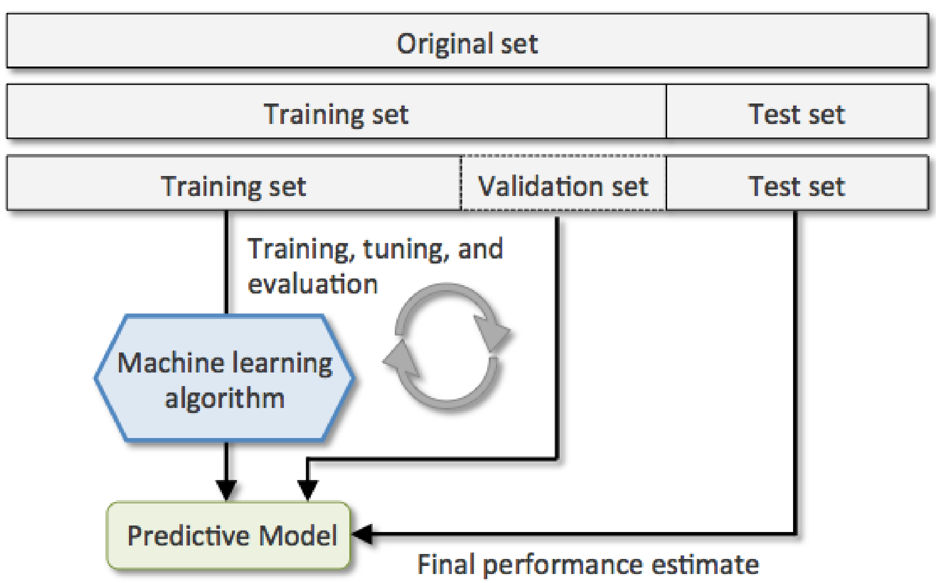
Evaluating Classifiers

1. Train, Test, and Validation
2. Data Leakage
3. Underfitting and Overfitting
4. Evaluation Metrics
5. Evaluation Curves

1. Train / Validation / Test



Training set is what the machine learning model use to learn. This sample needs to be representative of the population and it should not leak any information from the test sample.

Validation set is what you use to optimise your model parameters, and make a choice between different models. Since it will be used by the model(s), it should not leak any information from the test sample.

Test set is what you use to test how your model performs with unseen data. We need to be very strict about keeping the unseen property intact.

Naming Conventions

Train	Validation	Test
Train	Dev Set	Validation
Untitled	Cross Validation	Holdout
Untitled	Holdout	

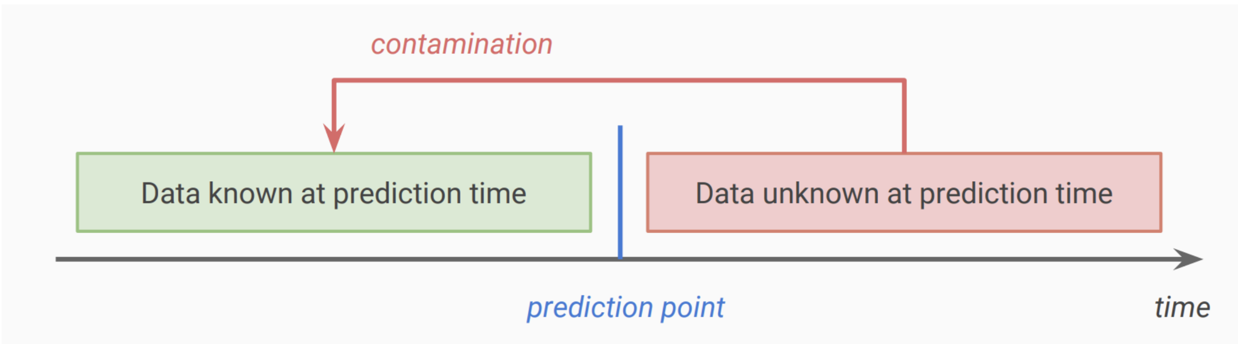
Note : Because you will find people calling these sets different names, when you are working with a team you have to be **explicit** about the naming conventions and what people mean when they refer to these sets

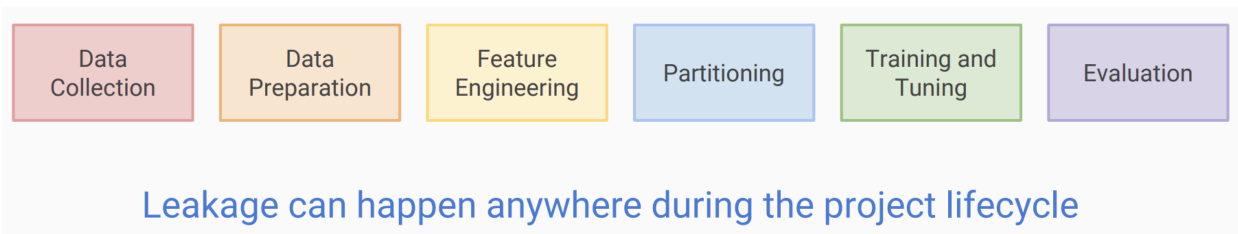
Split Size Conventions

Data Size	Train	Validation	Test
S (K's)	60	20	20
M (Hundred K's)	70	15	15
L (Millons)	80	10	10
XL (Hundred Millions or more)	90	5	5

Note: Care has to be taken with **imbalanced** data that the under represented samples are well covered in the validation and test samples when splitting the data. *More on this later*

2. Data Leakage





Leakage happens when we fail to preserve the unseen attribute of the test sample and the model already started learning something about it. Leakage can be very subtle and it can start to happen in all phases of the project life cycle and it will lead to overly optimistic results. Well established scientists and researchers sometimes miss subtle leakage sources sometimes too.

https://twitter.com/AndrewYNg/status/931026446717296640

3. Data

3.1. Training

We use the ChestX-ray14 dataset released by Wang et al. (2017) which contains 112,120 frontal-view X-ray images of 30,805 unique patients. Wang et al. (2017) annotate each image with up to 14 different thoracic pathology labels using automatic extraction methods on radiology reports. We label images that have pneumonia as one of the annotated pathologies as positive examples and label all other images as negative examples for the pneumonia detection task. We randomly split the entire dataset into 80% training, and 20% validation.

Before inputting the images into the network, we downscale the images to 224×224 and normalize based on the mean and standard deviation of images in the ImageNet training set. We also augment the training data with random horizontal flipping.

ples. For the pneumonia detection task, we randomly split the dataset into training (28744 patients, 98637 images), validation (1672 patients, 6351 images), and test (389 patients, 420 images). There is no patient overlap between the sets.

Pathology	Wang et al. (2017)	Yao et al. (2017)	CheXNet (ours)
Atelectasis	0.716	0.772	**0.8209**
Cardiomegaly	0.807	0.904	**0.9048**
Effusion	0.784	0.859	**0.8831**
Infiltration	0.609	0.695	**0.7204**
Mass	0.706	0.792	**0.8618**
Nodule	0.671	0.717	**0.7766**
Pneumonia	0.633	0.713	**0.7632**
Pneumothorax	0.806	0.841	**0.8932**
Consolidation	0.708	0.788	**0.7939**
Edema	0.835	0.882	**0.8932**
Emphysema	0.815	0.829	**0.9260**
Fibrosis	0.769	0.767	**0.8044**
Pleural Thickening	0.708	0.765	**0.8138**
Hernia	0.767	0.914	**0.9387**

Paper v1 (AUC)

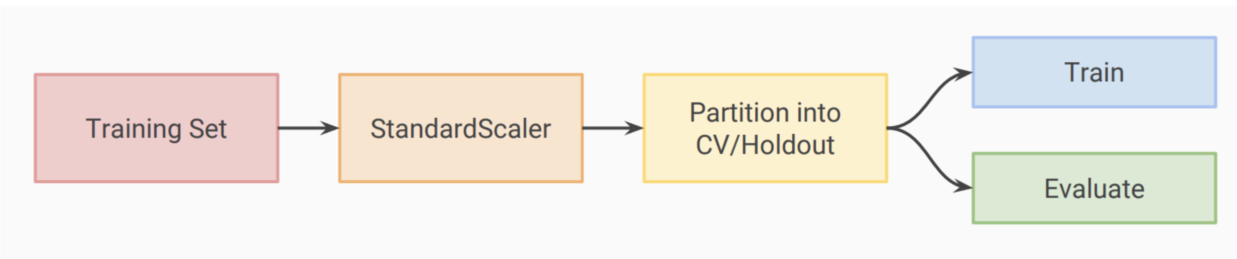
CheXNet (ours)
0.8094
0.9248
0.8638
0.7345
0.8676
0.7802
0.7680
0.8887
0.7901
0.8878
0.9371
0.8047
0.8062
0.9164

Paper v3 (AUC)

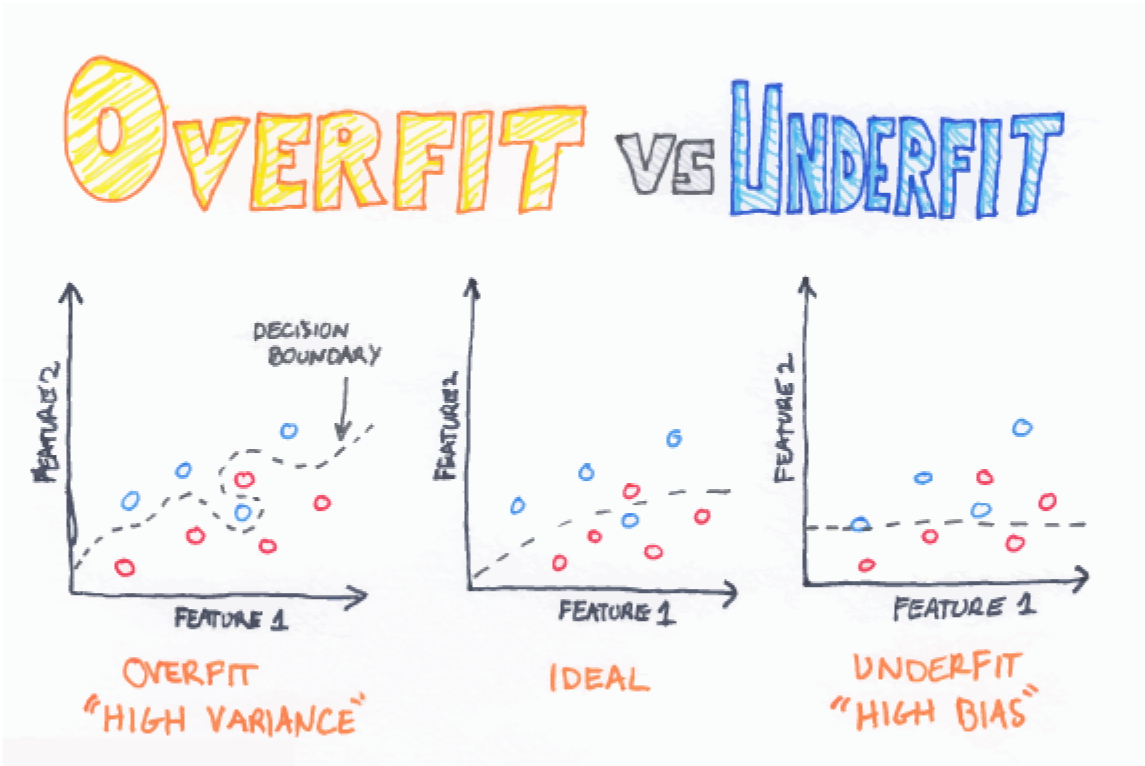
Example

EmployeeID	Title	ExperienceYears	MonthlySalaryGBP	AnnualIncomeUSD
315981	Data Scientist	3	5,000.00	78,895.44
4691	Data Scientist	4	5,500.00	86,784.98
23598	Data Scientist	5	6,200.00	97,830.35

Another example



3. Underfitting and Overfitting



Overfitting means that your model is too complex that it memorized the training data and cannot generalize well for all data

Underfiitting means that your model is too simples that it fails to capture the relationships needed for prediction

An ideal model is a somewhere in between ; Not too complex and not too simple but can generalize well for all data

How can we identify these problems ?

1. **Comparing training and test score** will tell you if you have a model that is **overfitting**. That means your training score will be much higher than your test score. Example 90% accuracy for training but 60% accuracy for test.

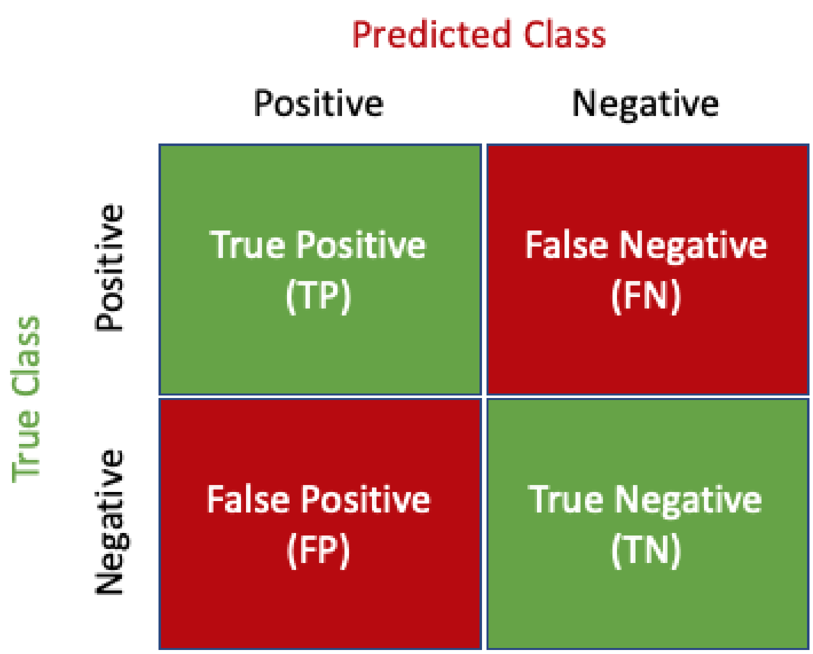
2. If you have a **low score for both training and test**, that means your model is **underfit** and you work more on the model to achieve better results

How can we calculate those scores and what kind of different score we should use to evaluate our models ?

- Next section

4. Evaluation Metrics

Confusion Matrix

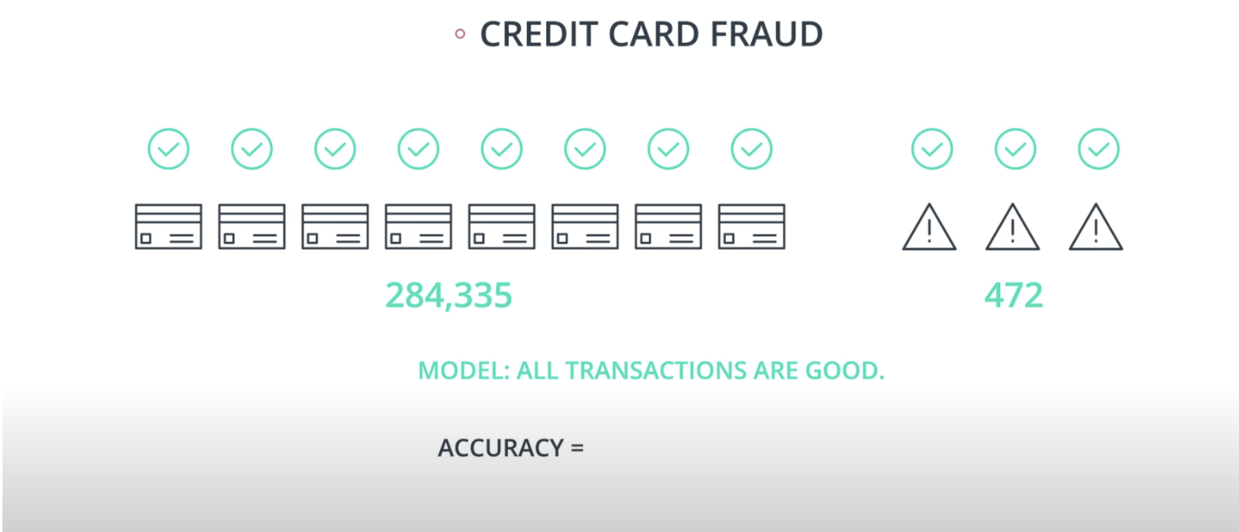


Using a confusion matrix, you woud have the count of all those values represented here and you can use that to diagnose the performance of your model.

Accuracy

$$acc = \frac{TN + TP}{TN + TP + FN + FP}$$

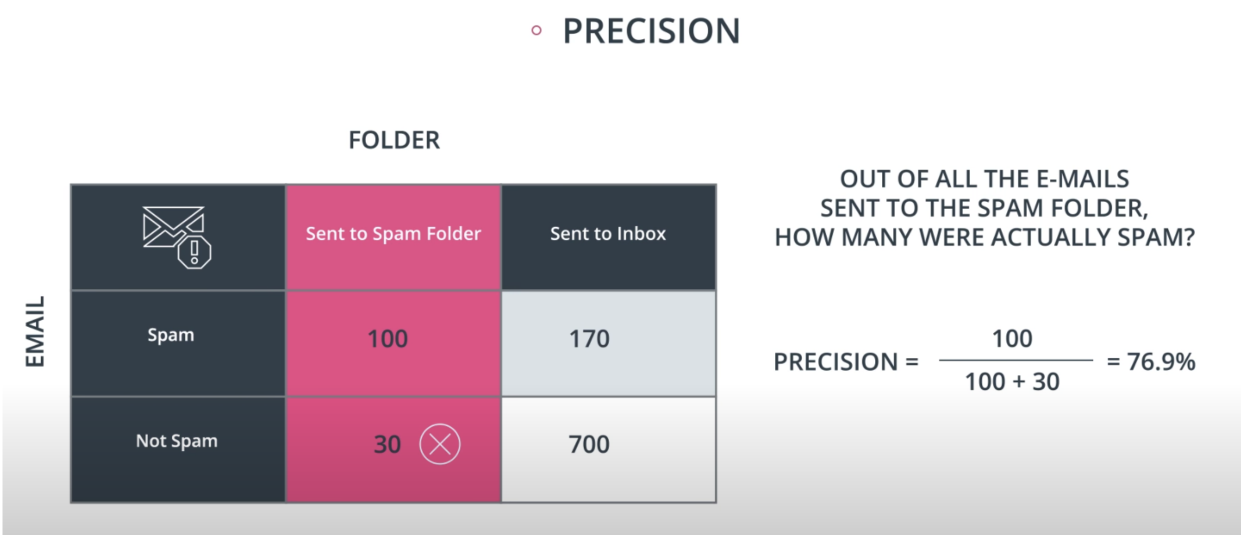
Quiz



What is the accuracy of a model that will always predict that the transaction is good ?

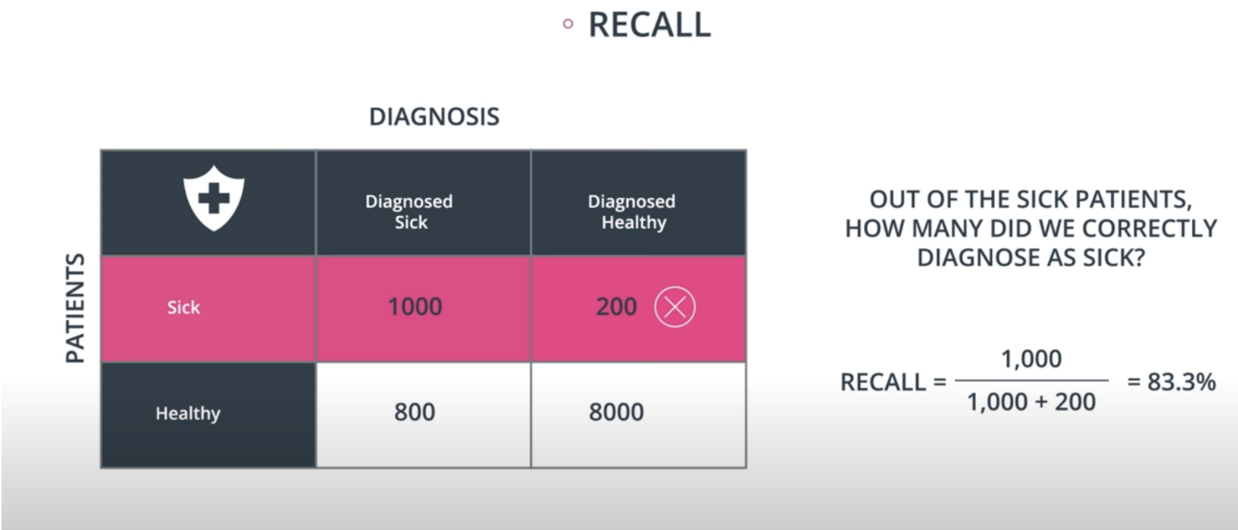
Answer

Precision



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

Recall



$$rec = \frac{TP}{TP + FN}$$

F1 Score

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

False Positive Rate

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

Evaluation Curves

ROC

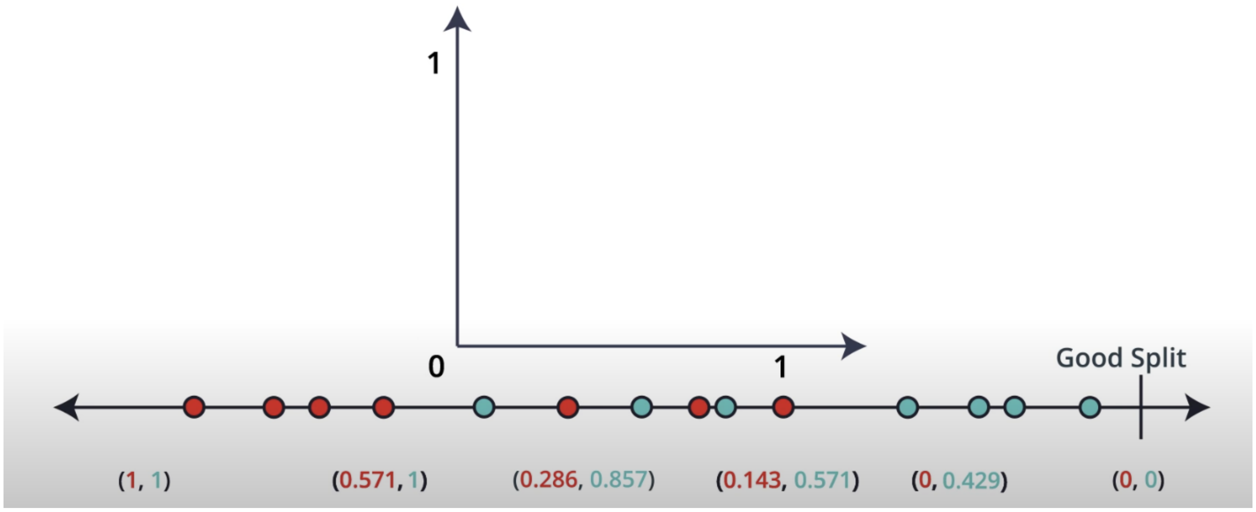
Receiver Operating Characteristic is a method used whereby we can use to evaluate the performance of the models and pick the right fit.

So for example if we can imagine the decision boundary split line moving on a linear line seperating positive and negative labels. We can calculate the true positives and the false positives as following:

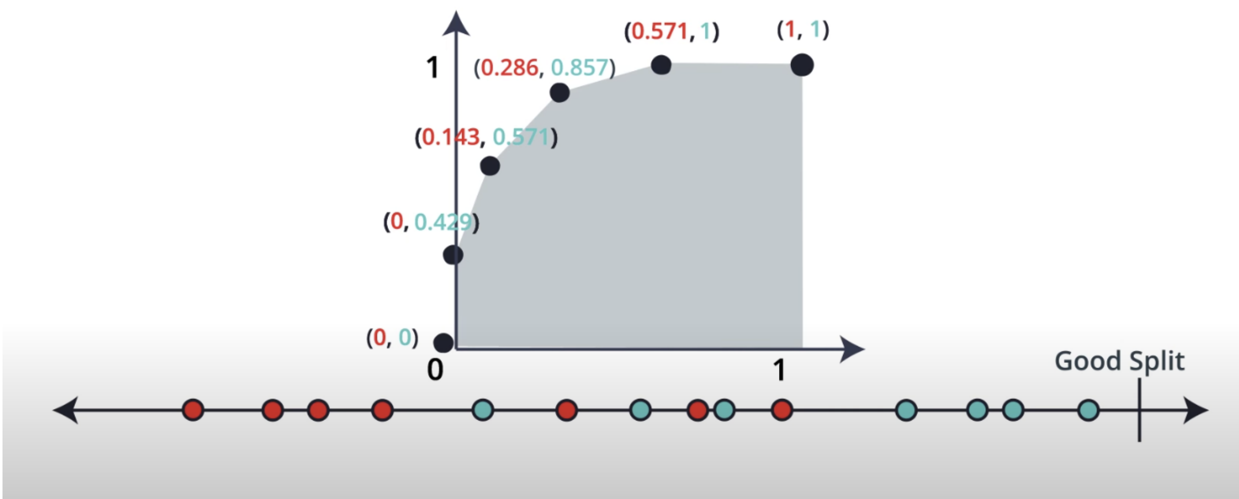
Now for the same approach we can change the threshold of classification to something like 0.01 for example and the results would look like this:

on the other end of the spectrum, we can change the threshold of classification to something like 0.99 for example and the results would look like this:

If we do this procedure for all the points like the following:



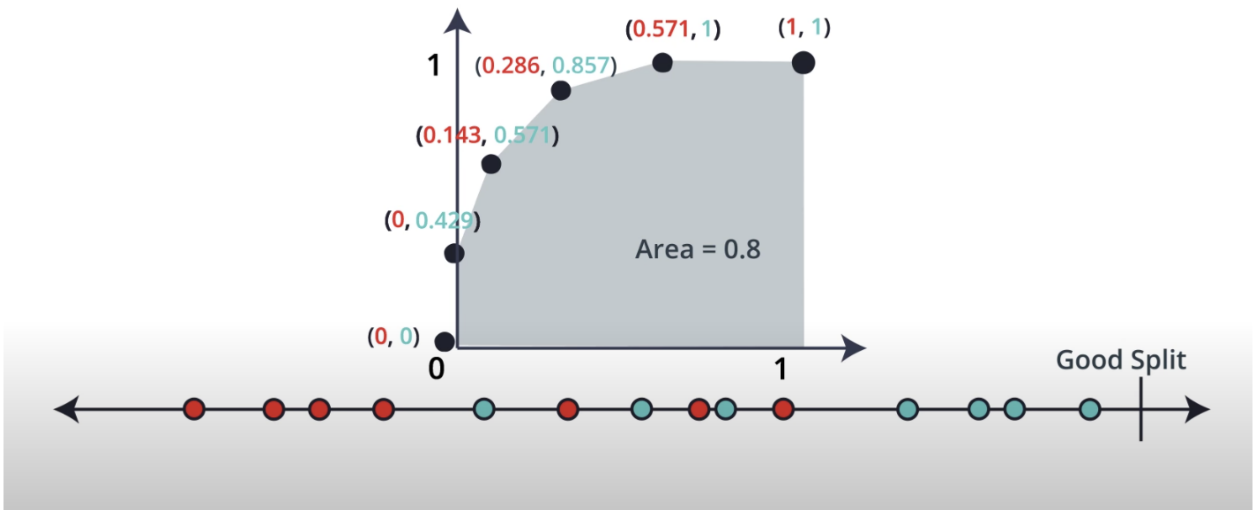
we can plot them on a graph that looks like this



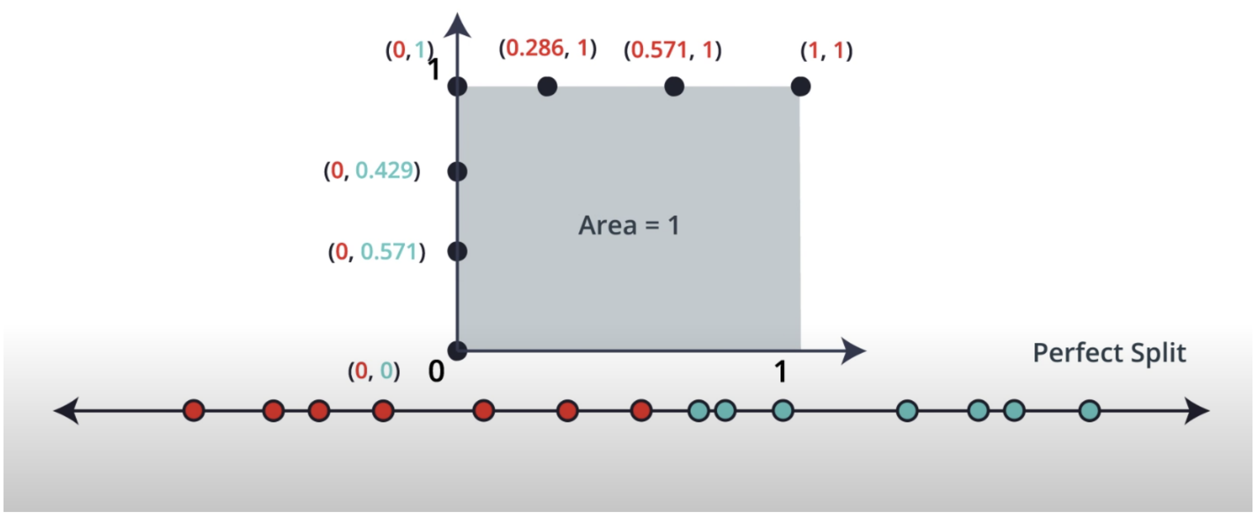
AUC

Now how can we compare different approaches against each other ?

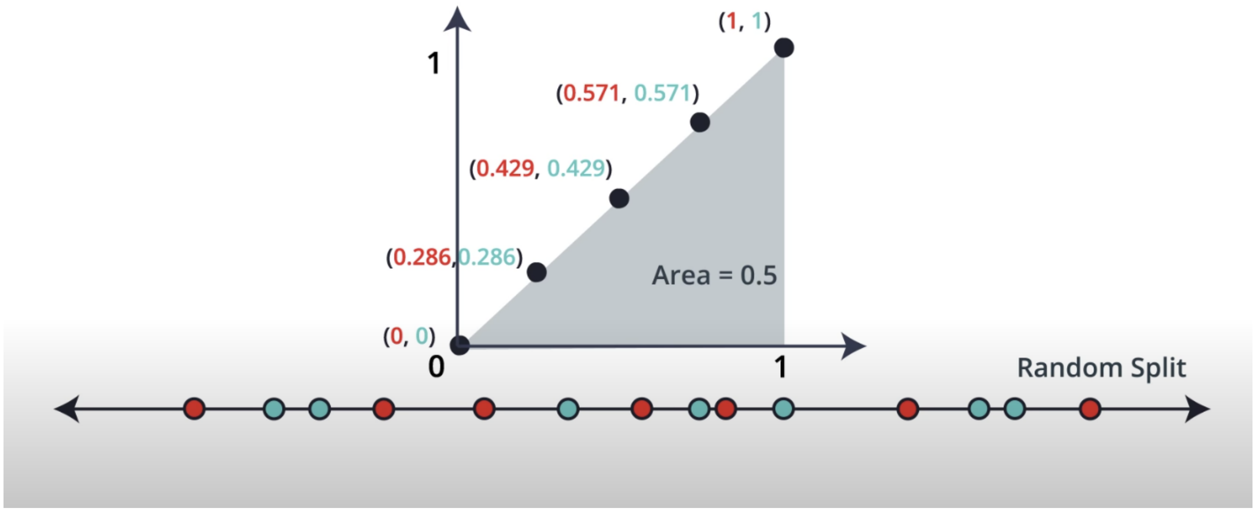
We compute the Area under Curve (AUC), so for example a good approach would have an AUC of 0.8 for example like the following:



and a perfect approach would have an AUC of 1 as following:



and a random split would have an AUC of 0.5 for example:

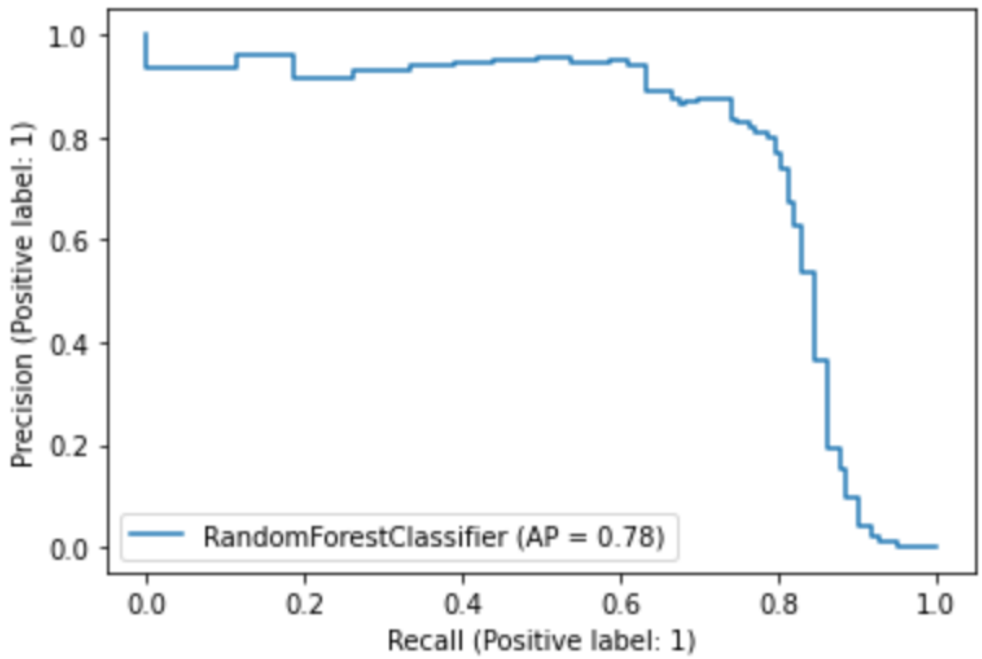


Precision-Recall Curve

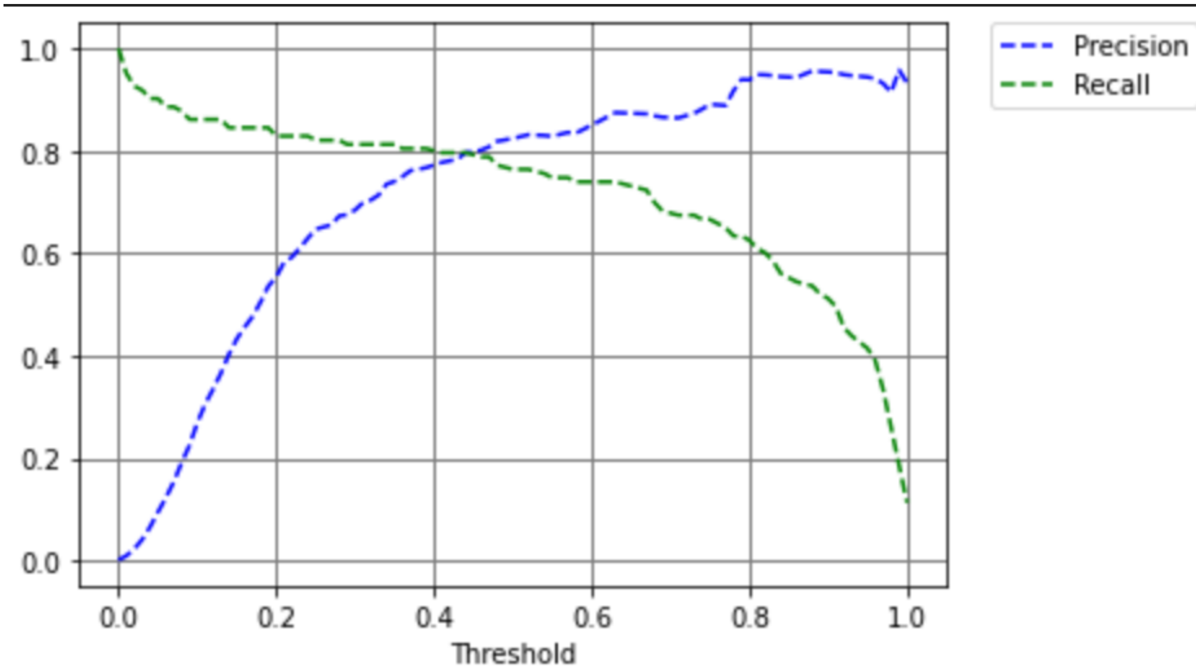
Another kind of evaluation method, which is used to pick a model while trading off precision and recall would be to

do the same procedure we did for the ROC curve but in this case, we would be comparing precision and recall versus each other

and also evaluating approaches by calculating area under the precision recall curve



we can also plot the precision and recall cuvres versus the threshold and pick the best model that serves our interest by choosing a threshold point.



Baseline Model

We should always compare the performance of our models to some established baselines. This is especially true for complicated machine learning solutions that require a lot of resources. When presenting a new approach we need to justify its high costs!

Sources for baselines can be:

- previous state-of-the-art models
- human performance on the same classification task
- heuristic rules that everyone can understand (such as: predict 0 if passenger is a third class male)

In case no better alternatives are available we can use these approaches:

- a random guess of the class outcome
- ‘predict’ the class that is most frequent in the training sample (the majority class)

sklearn offers some naive baseline classifiers with the DummyClassifier class