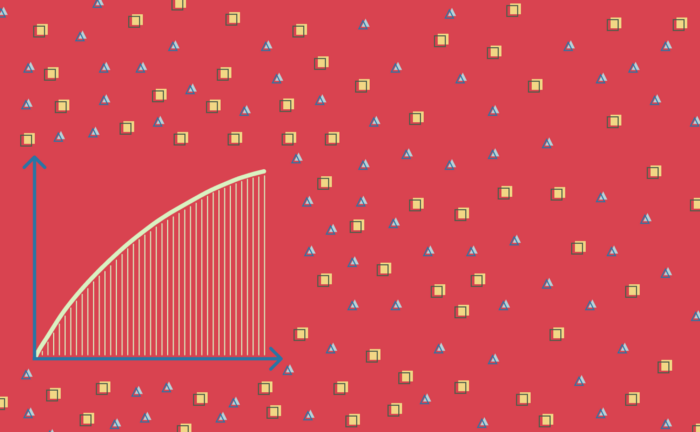
**ROC Analysis and the AUC — Area Under the Curve**

**Explained with a Real Life Example in Python**



(Image by Author)

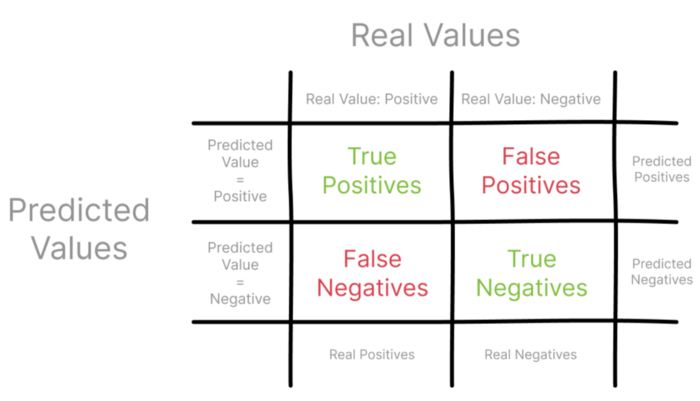
*Receiver Operating Characteristic Curve (ROC) analysis and the Area Under the Curve (AUC) are tools widely used in Data Science, borrowed from signal processing, to assess the quality of a model under different parameterizations, or compare performance of two or more models.*

*Traditional performance metrics, like precision and recall, rely heavily on positive observations. So instead, ROC and AUC use True Positive and False Positive Rates to assess quality, which take into account both positive and negative observations.*

The road from breaking down a problem and solving it with Machine Learning has multiple steps. At a high-level it involves data collection, cleaning and feature engineering, building the model and, last but not least, **evaluating model performance**.

When you’re evaluating the quality of a model, typically you use metrics like [precision and recall](https://en.wikipedia.org/wiki/Precision_and_recall), also referred to as confidence in the data mining field and sensitivity, respectively.

These metrics compare the predicted values to the real observation values, usually from a hold-out set, and are best visualized using a [confusion matrix](https://en.wikipedia.org/wiki/Confusion_matrix).



*Confusion Matrix (Image by Author)*

Let’s focus on **Precision** first, also referred to as Positive Predictive Value. Using the **confusion matrix**, you can construction Precision as the ratio of all the true positives over all predicted positives.

https://miro.medium.com/max/344/1*KG9Z7pL0gx0eDxvazaqxGg.png

**Recall**, which is also referred to as True Positive Rate, represents the ratio fo True Positives over all the Positives, observed and predicted.

https://miro.medium.com/max/327/1*1Y9klrS07lwKnYhj00124Q.png

Describing Precision and Recall using the different sets of observations in the confusion matrix, you can start to see how these metrics might provide a narrow view of model performance.

Something that stands out is the fact that Precision and Recall only focus on the positive examples and predictions[1], and don’t take into account any negative examples. Additionally, they don’t compare the performance of the model against a median-scenario, one that simply random-guesses.

After digging deeper into how Precision and Recall are calculated, you can start to see how these metrics might provide a narrow view of model performance.

To complement your model evaluation and rule out biases from Precision and Recall you can reach for a few robust tools in the Data Scientist’s toolkit: the **Receiver Operation Characteristic Curve (ROC)** analysis and its **Area Under the Curve (AUC)**.

**ROC Curve: from Signal Theory to Machine Learning**

ROC is as summary tool, used to visualize the trade-off between Precision and Recall[2].

This technique emerged in the field of signal detection theory, as part of the development of radar technology during World War II [3]. The name may be a bit confusing for those unfamiliar with signal theory, but it refers to readingradar signals by military radar operators, hence the *Receiver Operating* part of [Receiver Operating Characteristic Curve](https://en.wikipedia.org/wiki/Receiver_operating_characteristic#History).

Part of a radar operator’s job is to identify approaching enemy units on a radar, the key part, being able to literally distinguish signal, i.e., actual incoming units, from noise, e.g., static noise or other random interference. They are experts at determining what’s signal and what’s noise, to avoid charging at a supposed enemy unit when it’s either one of your own units or simply there’s nothing there.

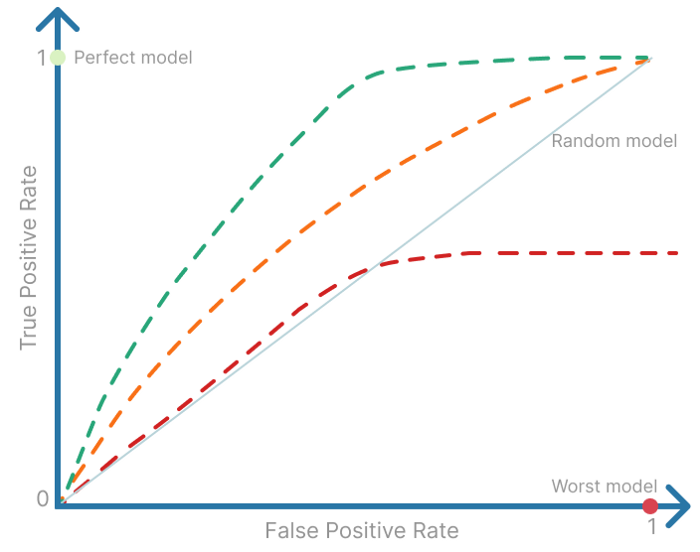
Right now you may be thinking *Hold on, this sounds like a familiar task!*

And indeed it is, this task is conceptually very similar to classifying an image as a cat or not, or detecting a patient developed a disease or not, while keeping a low false positive rate.

**ROC analysis** uses the ROC curve to determine how much of the value of a binary signal is polluted by noise, i.e., randomness[4]. It provides a summary of sensitivity and specificity across a range of operating points, for a continuous predictor[5].

The ROC curve is obtained by plotting the *False Positive Rate*, on the x-axis, against the *True Positive Rate*, on the y-axis.

Because the *True Positive Rate* is the probability of detecting a signal and *False Positive Rate* is the probability of a false alarm, ROC analysis is also widely used in medical studies, to determine the thresholds that confidently detect diseases or other behaviors[5].



Examples of different ROC curves (Image by author)

A perfect model will have both a *False Positive* and *True Positive Rate* equal to zero, so it will be a single operating point to the top left of the ROC plot. Whereas the worst possible model will have a single operating point on the bottom-left of the ROC plot, where the *False Positive Rate* is equal to one and *True Positive Rate* is equal to zero.

It [ROC Curve] provides a summary of sensitivity and specificity across a range of operating points, for a continuous predictor.

A random-guessing model, has a 50% chance of correctly predicting the result so, *False Positive Rate* will always be equal to the *True Positive Rate*. That’s why there’s a diagonal on the plot, representing that 50/50 chance of *detecting signal vs noise*.

**Using Area Under the Curve (AUC) to evaluate Machine Learning models**

Your parents have a cozy bed and breakfast and you, as a Data Scientist, set yourself up to the task of building a model that classifies their reviews as positive or negative.

To tackle this [Sentiment Analysis](https://en.wikipedia.org/wiki/Sentiment_analysis) task, you started off by using the [Multilayer Perceptron](https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141) and used accuracy and loss as a way to understand if was really good enough to solve your classification problem.

Knowing how **ROC analysis** is resistant to bias, and the fact that it’s used in Machine Learning to compare models or to compare different parameterizations of the same model, you want to see if the Multilayer Perceptron is actually a good model when it comes to classifying reviews from your parents’ bed and breakfast.

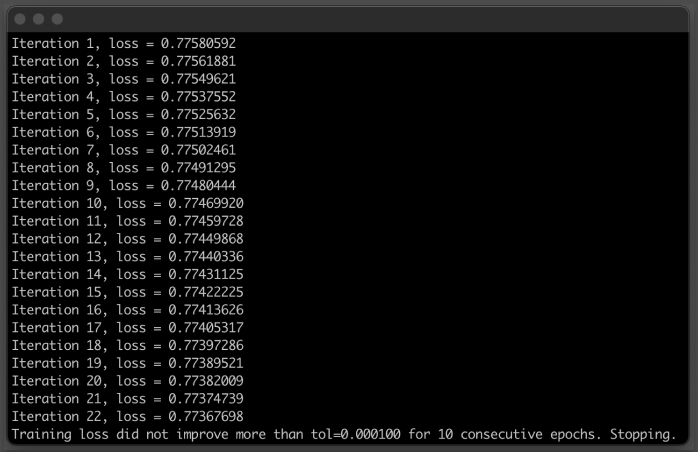
To rebuild the model, you take the corpus of reviews, then split it into training and testing and tokenize it.

from sklearn.model\_selection import train\_test\_split  
from sklearn.feature\_extraction.text import TfidfVectorizer  
  
corpus = [  
 'We enjoyed our stay so much. The weather was not great, but everything else was perfect.',  
 'Going to think twice before staying here again. The wifi was spotty and the rooms smaller than advertised',  
 'The perfect place to relax and recharge.',  
 'Never had such a relaxing vacation.',  
 'The pictures were misleading, so I was expecting the common areas to be bigger. But the service was good.',  
 'There were no clean linens when I got to my room and the breakfast options were not that many.',  
 'Was expecting it to be a bit far from historical downtown, but it was almost impossible to drive through those narrow roads',  
 'I thought that waking up with the chickens was fun, but I was wrong.',  
 'Great place for a quick getaway from the city. Everyone is friendly and polite.',  
 'Unfortunately it was raining during our stay, and there weren\'t many options for indoors activities. Everything was great, but there was literally no other oprionts besides being in the rain.',  
 'The town festival was postponed, so the area was a complete ghost town. We were the only guests. Not the experience I was looking for.',  
 'We had a lovely time. It\'s a fantastic place to go with the children, they loved all the animals.',  
 'A little bit off the beaten track, but completely worth it. You can hear the birds sing in the morning and then you are greeted with the biggest, sincerest smiles from the owners. Loved it!',  
 'It was good to be outside in the country, visiting old town. Everything was prepared to the upmost detail'  
 'staff was friendly. Going to come back for sure.',  
 'They didn\'t have enough staff for the amount of guests. It took some time to get our breakfast and we had to wait 20 minutes to get more information about the old town.',  
 'The pictures looked way different.',  
 'Best weekend in the countryside I\'ve ever had.',  
 'Terrible. Slow staff, slow town. Only good thing was being surrounded by nature.',  
 'Not as clean as advertised. Found some cobwebs in the corner of the room.',  
 'It was a peaceful getaway in the countryside.',  
 'Everyone was nice. Had a good time.',  
 'The kids loved running around in nature, we loved the old town. Definitely going back.',  
 'Had worse experiences.',  
 'Surprised this was much different than what was on the website.',  
 'Not that mindblowing.'  
]  
  
# 0: negative sentiment. 1: positive sentiment  
targets = [1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0]  
  
# Splitting the dataset  
train\_features, test\_features, train\_targets, test\_targets = train\_test\_split(corpus, targets, test\_size=0.25,random\_state=123)  
  
#Turning the corpus into a tf-idf array  
vectorizer = TfidfVectorizer(stop\_words='english', lowercase=True, norm='l1')

The [Multilayer Perceptron](https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141) model is ready to be trained.

from sklearn.neural\_network import MLPClassifier  
  
def buildMLPerceptron(train\_features, train\_targets, num\_neurons=2):  
 """ Build a Multi-layer Perceptron and fit the data  
 Activation Function: ReLU  
 Optimization Function: SGD, Stochastic Gradient Descent  
 Learning Rate: Inverse Scaling  
 """  
  
 classifier = MLPClassifier(hidden\_layer\_sizes=num\_neurons, max\_iter=35, activation='relu', solver='sgd', verbose=10, random\_state=762, learning\_rate='invscaling')  
 classifier.fit(train\_features, train\_targets)  
  
 return classifier  
  
train\_features = vectorizer.fit\_transform(train\_features)  
test\_features = vectorizer.transform(test\_features)  
  
# Build Multi-Layer Perceptron with 3 hidden layers, each with 5 neurons  
ml\_percetron\_model = buildMLPerceptron(train\_features, train\_targets, num\_neurons=5)

All set to train the model! When you run the code above you’ll see something like the following.



*Output of training the Multilayer Perceptron model. (Image by Author)*

To fully analyze the ROC Curve and compare the performance of the Multilayer Perceptron model you just built against a few other models, you actually want to calculate the [Area Under the Curve (AUC)](https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve), also referred to in literature as *c-statistic*.

The **Area Under the Curve (AUC)** has values between zero and one, since the curve is plotted on a 1x1 grid and, drawing a parallel with signal theory, it’s a measure of a signal’s detectability[6].

This is a very useful statistic, because it gives an idea of how well models can rank true observations as well as false observations. It’s actually a normalized version of the [Wilcoxon-Mann-Whitney sum of ranks test](https://en.wikipedia.org/wiki/Mann%E2%80%93Whitney_U_test#Area-under-curve_(AUC)_statistic_for_ROC_curves), which tests the null hypothesis where two samples of ordinal measurements are drawn from a single distribution [4].

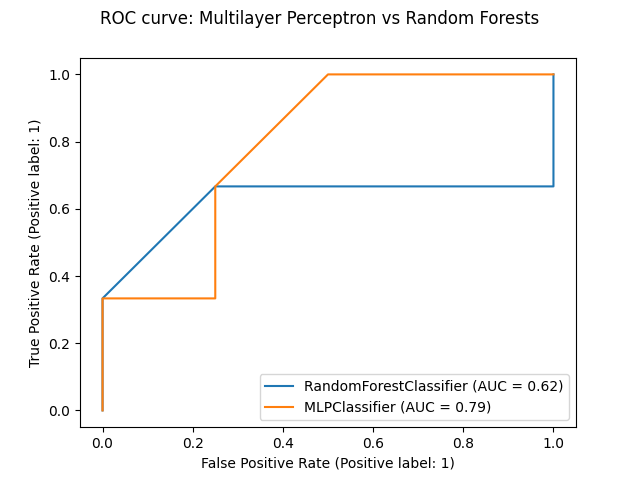
The c-statistic normalizes the number of pairs of one positive and one negative draws.

[…] drawing a parallel with signal theory, [the area under the curve] it’s a measure of a signal’s detectability.

To plot the ROC Curve and calculate the Area Under the Curve (AUC) you decided to use [SckitLearn’s RocCurveDisplay method](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.RocCurveDisplay.html#sklearn.metrics.RocCurveDisplay.from_estimator) and compare your [Multilayer Perceptron](https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141) to a Random Forests model, attempting to solve the same classification task.

import matplotlib.pyplot as plt  
from sklearn.ensemble import RandomForestClassifier  
from sklearn.metrics import roc\_auc\_score, RocCurveDisplay  
  
def plot\_roc(model, test\_features, test\_targets):  
 """  
 Plotting the ROC curve for a given Model and the ROC curve for a Random Forests Models  
 """  
  
 # comparing the given model with a Random Forests model  
 random\_forests\_model = RandomForestClassifier(random\_state=42)  
 random\_forests\_model.fit(train\_features, train\_targets)  
  
 rfc\_disp = RocCurveDisplay.from\_estimator(random\_forests\_model, test\_features, test\_targets)  
 model\_disp = RocCurveDisplay.from\_estimator(model, test\_features, test\_targets, ax=rfc\_disp.ax\_)  
 model\_disp.figure\_.suptitle("ROC curve: Multilayer Perceptron vs Random Forests")  
  
 plt.show()  
  
# using perceptron model as input  
plot\_roc(ml\_percetron\_model, test\_features, test\_targets)

The code above plots the ROC curves for your [Multilayer Perceptron](https://towardsdatascience.com/multilayer-perceptron-explained-with-a-real-life-example-and-python-code-sentiment-analysis-cb408ee93141) and the Random Forests model. It also calculates the Area Under the Curve (AUC) for both models.



ROC Plot for the Multilayer Perceptron vs a Random Forests model. (Image by Author)

**Conclusion**

From the ROC analysis plot and the value of the Area Under the Curve (AUC) for each model, you can see the overall AUC for your Multilayer Perceptron model, denoted in the plot as MLPClassifier, is slightly higher.

When compared to a Random Forests model attempting to solve the same task of classifying the sentiment of reviews for your parents’ bed and breakfast, the Multilayer Perceptron did a better job.

In this particular case, that’s also visible by how *close* the orange line starts getting to the top-left corner of the plot, where the True Positive Rate of the predictions is increasingly higher and, by opposition, the False Positive Rate is increasingly lower.

You can also see the Random Forests model is only slightly better than a Random Model, which would have an AUC equal to 0.5.

Hope you enjoyed learning about ROC analysis and the Area Under the Curve, two powerful techniques to compare Machine Learning models, using metrics that are more resistant to bias.

*Thanks for reading!*