KDnuggets

ROC Curve Explained

<= Previous post
Next post =>

Tags: Data Visualization, Metrics, Python, ROC-AUC

Learn to visualise a ROC curve in Python.

---

comments

By Zolzaya Luvsandorj, Data Scientist at iSelect

Area under the ROC curve is one of the most useful metrics to evaluate a supervised classification model. This metric is commonly referred to as ROC-AUC. Here, the ROC stands for Receiver Operating Characteristic and AUC stands for Area Under the Curve. In my opinion, AUROCC is a more accurate abbreviation but perhaps doesn't sound as nice. In the right context, AUC can also imply ROC-AUC even though it can refer to area under any curve.

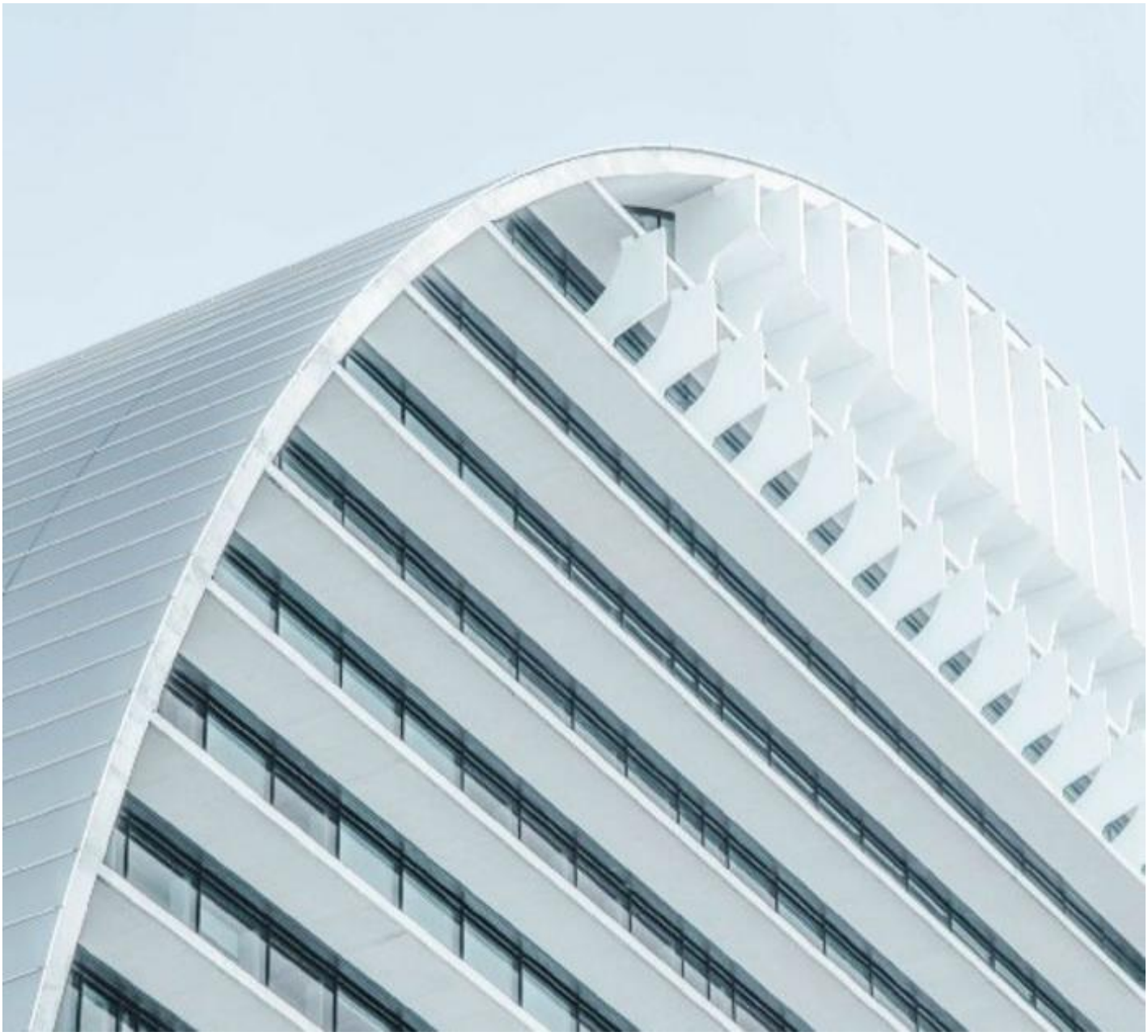


Photo by Joel Filipe on Unsplash

In this post, we will understand how the ROC curve is constructed conceptually, and visualise the curve in a static and interactive format in Python.

Understanding the curve

A ROC curve shows us the relationship between False Positive Rate (aka FPR) and True Positive Rate (aka TPR) across different thresholds. Let's understand what each of these three terms mean.

Firstly, let's start with a refresher on how a confusion matrix looks like:

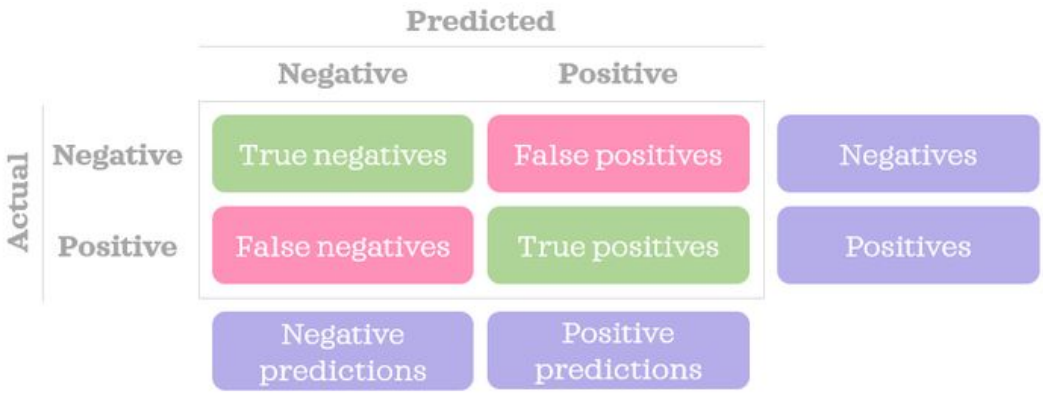


Image by author

Having refreshed our memory on confusion matrix, let' s look at the terms.

False Positive Rate

We can find the FPR using the simple formula below:

$$FPR = \frac{\text{False positives}}{\text{Negatives}}$$

FPR tells us the percentage of incorrectly predicted negative records.

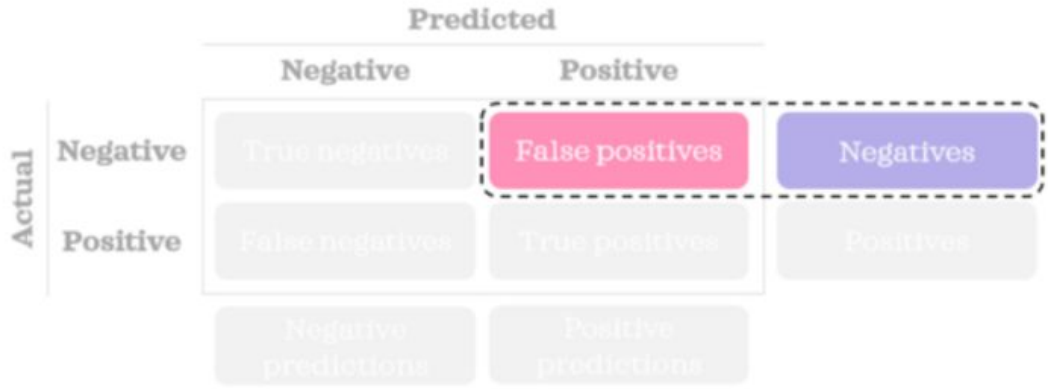


Image by author

True Positive Rate

We can find the TPR using the simple formula below:

$$TPR = \frac{\text{True positives}}{\text{Positives}}$$

TPR tells us the percentage of correctly predicted positive records. This is also known as Recall or Sensitivity.

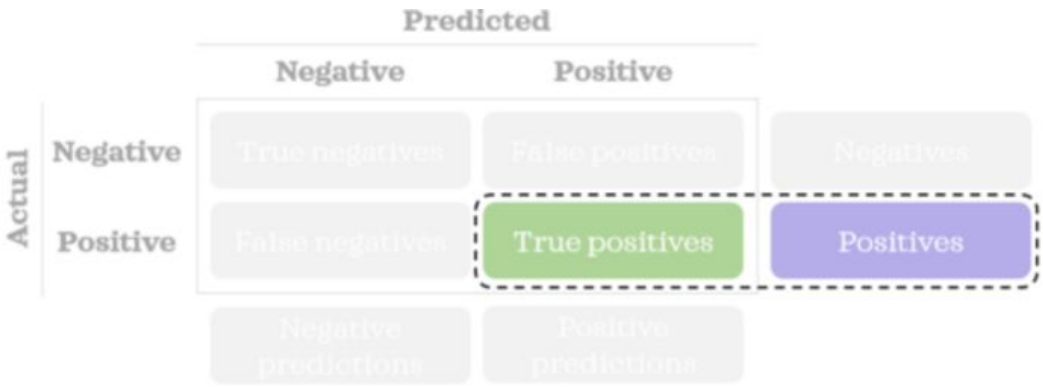


Image by author

Threshold

In general, a classification model can predict the probability of being a certain class for a given record. By comparing the probability value to a threshold value we set, we can classify the record into a class. In other words, you will need to define a rule similar to the following:

> If the probability of being positive is greater than or equal to the threshold, then a record is classified as a positive prediction; otherwise, a negative prediction.

In the small example below, we can see the probability scores for three records. Using two different threshold values (0.5 and 0.6), we classified each record into a class. As you can see, the predicted classes vary depending on the threshold value we choose.

id	Probability of being negative	Probability of being positive	Predicted class (threshold: 0.5)	Predicted class (threshold: 0.6)
a1	0.6	0.4	negative	negative
a2	0.5	0.5	positive	negative
a3	0.3	0.7	positive	positive

Image by author

When building a confusion matrix and calculating rates like FPR and TPR, we need predicted classes rather than probability scores.

ROC curve

Now that we know what FPR, TPR and threshold values are, it' s easy to understand what a ROC curve shows. When constructing the curve, we first calculate FPR and TPR across many threshold values. Once we have the FPR and TPR for the thresholds, we then plot FPR on the x-axis and TPR on the y-axis to get a ROC curve. That' s it! ✨

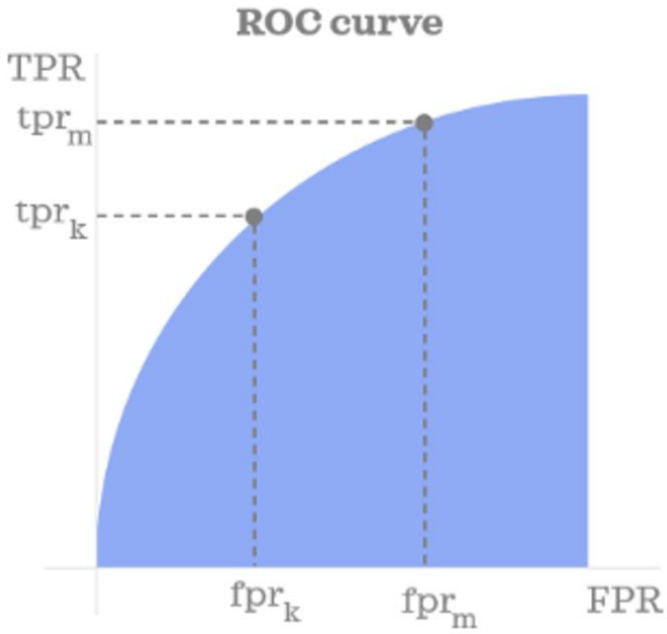


Image by author

Area under a ROC curve ranges from 0 to 1. A completely random model has an AUROCC of 0.5 which is represented by the dashed blue triangle diagonal line below. The further the ROC curve is from this line, the more predictive the model is.

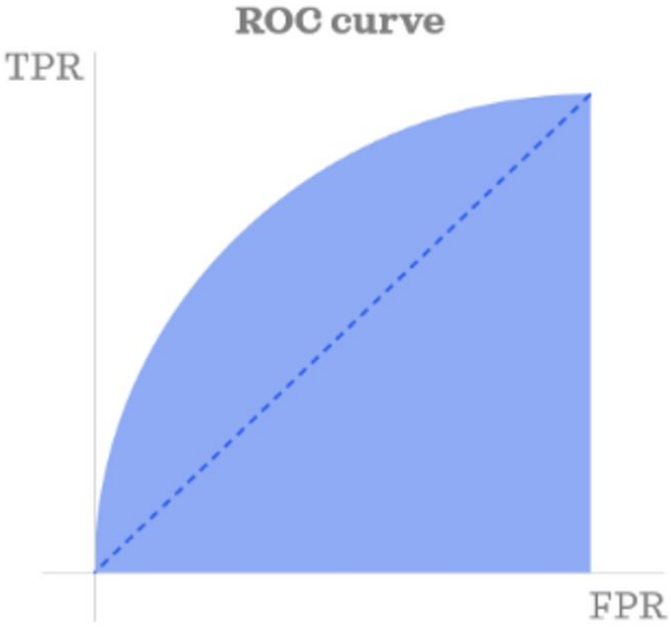


Image by author

Now, it’ s time to look at some code examples to consolidate our knowledge.

Build static ROC curve in Python

Let’ s first import the libraries that we need for the rest of this post:

```
import numpy as np
import pandas as pd
pd.options.display.float_format = "{:.4f}".formatfrom sklearn.datasets import load_breast_cancer
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, plot_roc_curveimport matplotlib.pyplot as plt
```

```
import seaborn as sns
import plotly.express as px
sns.set(palette='rainbow', context='talk')
```

Now we will build a function that will find us the number of false positives and true positives given the correct class, predicted probability of being a positive class and a threshold:

```
def get_fp_tp(y, proba, threshold):
    """Return the number of false positives and true positives."""
    # Classify into classes
    pred = pd.Series(np.where(proba>=threshold, 1, 0),
                     dtype='category')
    pred.cat.set_categories([0,1], inplace=True)
    # Create confusion matrix
    confusion_matrix = pred.groupby([y, pred]).size().unstack()\
                           .rename(columns={0: 'pred_0',
                                            1: 'pred_1'},
                                   index={0: 'actual_0',
                                          1: 'actual_1'})
    false_positives = confusion_matrix.loc['actual_0', 'pred_1']
    true_positives = confusion_matrix.loc['actual_1', 'pred_1']
    return false_positives, true_positives
```

Please note that you will be working with partitioned data sets (e.g. training, test) in reality. But we will not partition our data for simplicity in this post.

We will build a simple model on a toy dataset and get the probabilities of being positive (represented by a value of 1) for the records:

```
# Load sample data
X = load_breast_cancer()['data'][:,:2] # first two columns only
y = load_breast_cancer()['target']# Train a model
log = LogisticRegression()
log.fit(X, y)# Predict probability
proba = log.predict_proba(X)[:,1]
```

We will use 1001 different thresholds between 0 and 1 with increments of 0.001. In other words, threshold values will look something like 0, 0.001, 0.002, … 0.998, 0.999, 1. Let’ s find the FPR and TPR for the threshold values.

```
# Find fpr & tpr for thresholds
negatives = np.sum(y==0)
positives = np.sum(y==1)columns = ['threshold', 'false_positive_rate', 'true_positive_rate']
inputs = pd.DataFrame(columns=columns, dtype=np.number)
thresholds = np.linspace(0, 1, 1001)for i, threshold in enumerate(thresholds):
    inputs.loc[i, 'threshold'] = threshold
    false_positives, true_positives = get_fp_tp(y, proba, threshold)
    inputs.loc[i, 'false_positive_rate'] = false_positives/negatives
    inputs.loc[i, 'true_positive_rate'] = true_positives/positives
inputs
```

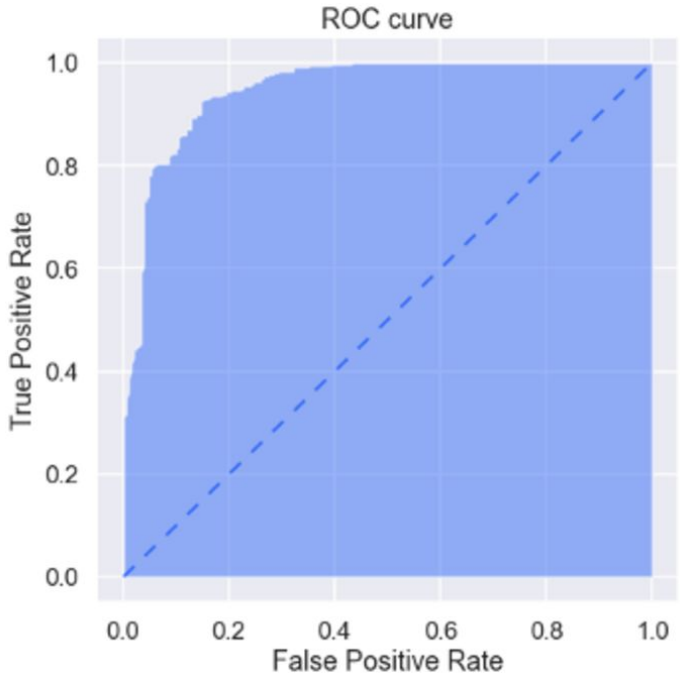
	threshold	false_positive_rate	true_positive_rate
0	0.0000	1.0000	1.0000
1	0.0010	0.8585	1.0000
2	0.0020	0.7877	1.0000
3	0.0030	0.7642	1.0000
4	0.0040	0.7075	1.0000
...	...	...	...
996	0.9960	0.0000	0.1176
997	0.9970	0.0000	0.0952
998	0.9980	0.0000	0.0644
999	0.9990	0.0000	0.0364
1000	1.0000	0.0000	0.0000

1001 rows × 3 columns

Data for the plot is ready. Let' s plot it:

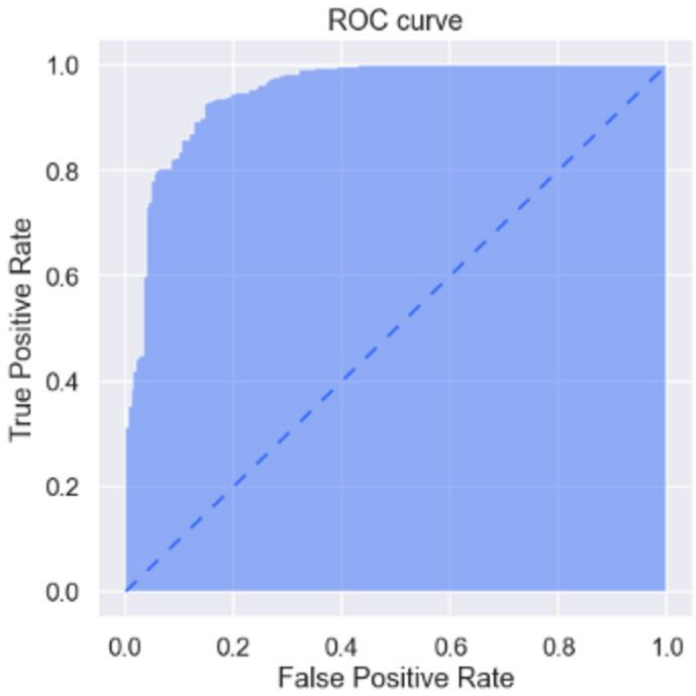
```
def plot_static_roc_curve(fpr, tpr):
    plt.figure(figsize=[7,7])
    plt.fill_between(fpr, tpr, alpha=.5)
    # Add dashed line with a slope of 1
    plt.plot([0,1], [0,1], linestyle=(0, (5, 5)), linewidth=2)
    plt.xlabel("False Positive Rate")
    plt.ylabel("True Positive Rate")
    plt.title("ROC curve");

plot_static_roc_curve(inputs['false_positive_rate'],
                      inputs['true_positive_rate'])
```



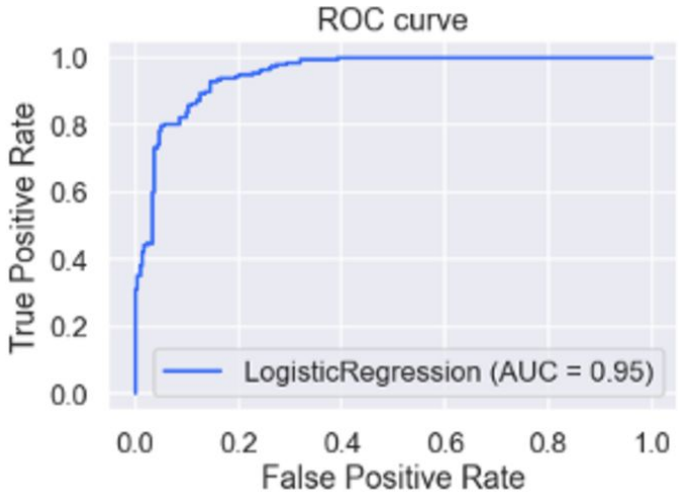
While building a custom function helps us understand the curve and its inputs, and control them better, we can also take advantage of sklearn's capabilities that are more optimised. For instance, we can get FPR, TPR and thresholds with a `roc_curve()` function. We can plot the data the same way using our custom plotting function:

```
fpr, tpr, thresholds = roc_curve(y, proba)
plot_static_roc_curve(fpr, tpr)
```



Sklearn also provides a `plot_roc_curve()` function which does all the work for us. All you need is a single line (adding title is optional):

```
plot_roc_curve(log, X, y)
plt.title("ROC curve"); # Add a title for clarity
```



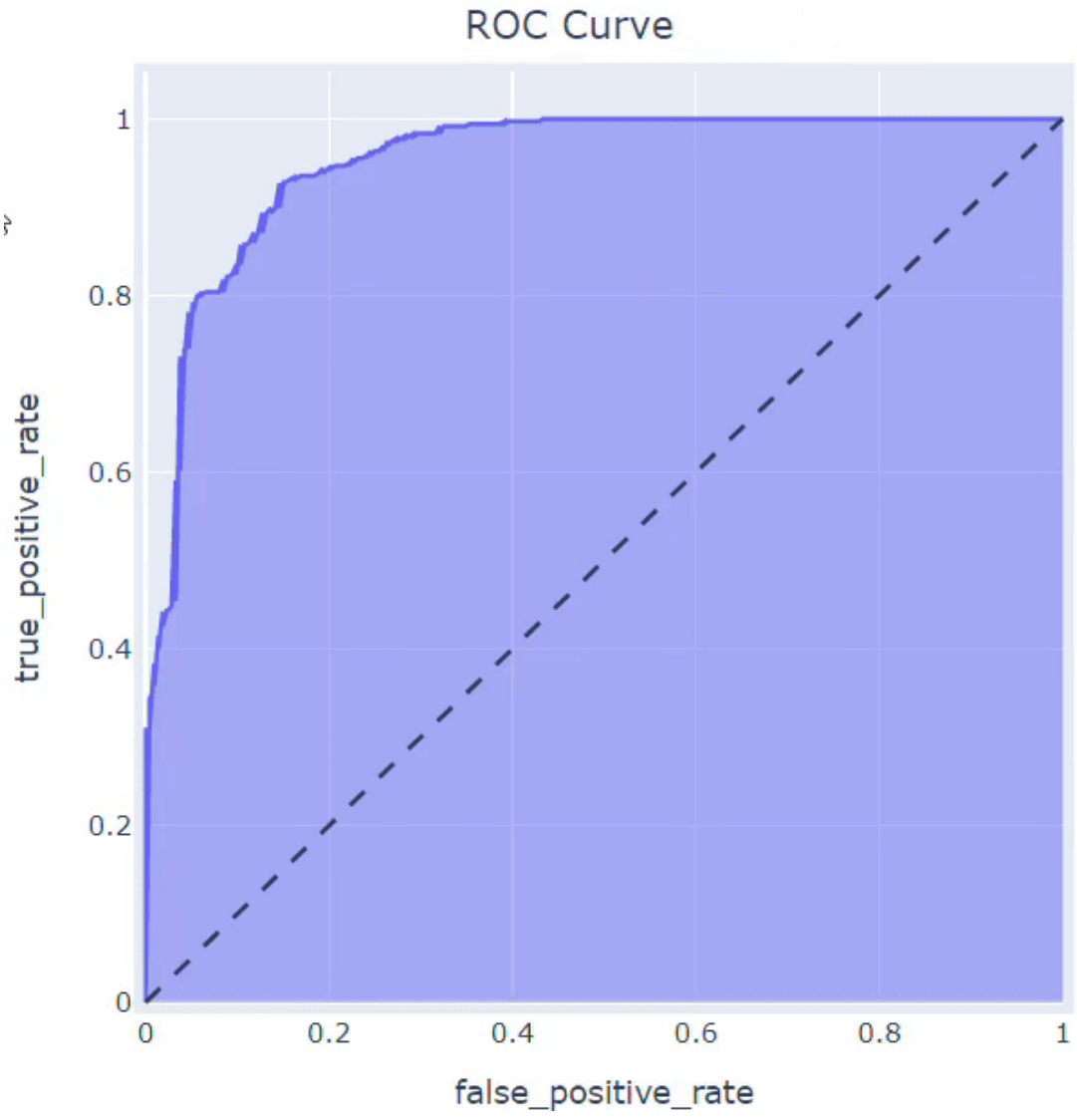
Plot interactive ROC curve in Python

When using static plots, it's hard to see the corresponding threshold value for different points across the curve. One option is

to inspect the `inputs` dataframe we created. Another option is to create an interactive version of the plot so that we can see the FPR and TPR alongside the corresponding threshold value when we hover over the graph:

```
def plot_interactive_roc_curve(df, fpr, tpr, thresholds):
    fig = px.area(
        data_frame=df,
        x=fpr,
        y=tpr,
        hover_data=thresholds,
        title='ROC Curve'
    )
    fig.update_layout(
        autosize=False,
        width=500,
        height=500,
        margin=dict(l=30, r=30, b=30, t=30, pad=4),
        title_x=.5, # Centre title
        hovermode = 'closest',
        xaxis=dict(hoverformat='.4f'),
        yaxis=dict(hoverformat='.4f')
    )
    hovertemplate = 'False Positive Rate=%{x}<br>True Positive Rate=%{y}<br>Threshold=%{customdata[0]:.4f}<extra></extra>'
    fig.update_traces(hovertemplate=hovertemplate)

    # Add dashed line with a slope of 1
    fig.add_shape(type='line', line=dict(dash='dash'), x0=0, x1=1, y0=0, y1=1)
    fig.show()plot_interactive_roc_curve(df=inputs,
                          fpr='false_positive_rate',
                          tpr='true_positive_rate',
                          thresholds=['threshold'])
```



The interactivity is quite useful, isn't it?

Hope you enjoyed learning how to build and visualise a ROC curve. Once you understand this curve, it' s easy to understand another related curve: Precision-Recall curve.

Thank you for reading this article. If you are interested, here are links to some of my other posts:

- Interesting Ways to Use Punctuations in Python
- 5 tips to learn Python from zero
- Introduction to Python Virtual Environment for Data Science
- Introduction to Git for Data Science
- Organise your Jupyter Notebook with these tips
- 6 simple tips for prettier and customised plots in Seaborn (Python)
- 5 tips for pandas users
- Writing advanced SQL queries in pandas

Bye for now 🏃‍♀️👋

Bio: Zolzaya Luvsandorj works as a Data Scientist at iSelect. Upon completing her BCom as a top student with multiple prestigious awards, Zolzaya worked as a Data Analyst in a consultancy firm for 3 years before moving on to her current role. She loves expanding her knowledge in data science, computer science and statistics and explaining data science concepts in simple words in her blogs.

Original. Reposted with permission.

Related:

- Get Interactive Plots Directly With Pandas
- How to create an interactive 3D chart and share it easily with anyone
- Metric Matters, Part 1: Evaluating Classification Models

<= **Previous post**
Next post =>

Top Stories Past 30 Days

Most Popular

1. **Data Scientists Will be Extinct in 10 Years**
2. **5 Tasks To Automate With Python**
3. **How to Generate Automated PDF Documents with Python**
4. **Pandas vs SQL: When Data Scientists Should Use Each Tool**
5. **Top 10 Data Science Projects for Beginners**

Most Shared

1. **Data Scientists Will be Extinct in 10 Years**
2. **Five types of thinking for a high performing data scientist**
3. **5 Lessons McKinsey Taught Me That Will Make You a Better Data Scientist**
4. **Analytics Engineering Everywhere**
5. **Semantic Search: Measuring Meaning From Jaccard to Bert**

Latest News

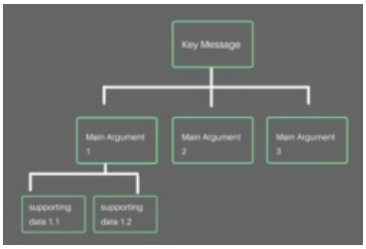
- eBook: How to use third-party data to make smarter deci...
- Relax! Data Scientists will not go extinct in 10 years,...
- How to Get Practical Data Science Experience to be Care...
- How to Build An Image Classifier in Few Lines of Code w...
- KDnuggets 21:n25, Jul 7: Data Scientists and ML Engineers A...
- ROC Curve Explained

Top Stories
Last Week

Most Popular

1. **5 Lessons McKinsey Taught Me That Will Make You a Better Data Scientist**
2. **What will the demand for Data Scientists be in 10 years? Will Data Scientists be extinct?**

3. **Add A New Dimension To Your Photos Using Python**
4. **Managing Your Reusable Python Code as a Data Scientist**
5. **Data Scientists are from Mars and Software Developers are from Venus**



Most Shared

1. **5 Lessons McKinsey Taught Me That Will Make You a Better Data Scientist**
2. **Managing Your Reusable Python Code as a Data Scientist**
3. **Data Scientists are from Mars and Software Developers are from Venus**
4. **How to Train a Joint Entities and Relation Extraction Classifier using BERT Transformer with spaCy 3**
5. **Add A New Dimension To Your Photos Using Python**

More Recent Stories

- ROC Curve Explained
- A Learning Path To Becoming a Data Scientist
- How To Transition From Data Freelancer to Data Entrepreneur (A...
- Top Stories, Jun 28 – Jul 4: 5 Lessons McKinsey Taught M...
- GitHub Copilot: Your AI pair programmer – what is all th...
- Data Scientists and ML Engineers Are Luxury Employees
- Predict Customer Churn (the right way) using PyCaret
- Semantic Search: Measuring Meaning From Jaccard to Bert
- High-Performance Deep Learning: How to train smaller, faster, ...
- Prepare Behavioral Questions for Data Science Interviews
- How to Use NVIDIA GPU Accelerated Libraries
- Learning Data Science Through Social Media
- 5 Lessons McKinsey Taught Me That Will Make You a Better Data Scientist **[Gold Blog]**
- Managing Your Reusable Python Code as a Data Scientist **[Silver Blog]**
- Ethics, Fairness, and Bias in AI
- From Scratch: Permutation Feature Importance for ML Interpreta...
- KDnuggets 21:n24, Jun 30: What will the demand for Data Sci...
- StreamSets DataOps Platform – Summer '21 Public Beta. ...
- Computational Complexity of Deep Learning: Solution Approaches
- Unleashing the Power of MLOps and DataOps in Data Science