Create Artificial Data With SMOTE

How you can leverage a simple algorithm to compensate for lack of data

Aashish Nair · Follow
Published in Towards Data Science
6 min read · Jan 24, 2022

Photo by Brett Jordan on Unsplash

Data imbalance is ubiquitous in machine learning. Real data rarely represents every class equally. In applications such as disease diagnosis, fraud detection, and spam classification, some classes will always be underrepresented.

This is a major obstacle for many machine learning related endeavors. After all, if you lack the data for a specific outcome, your model will not be able to predict that outcome adequately.

Thankfully, there are ways to circumvent this issue. If you can't acquire sufficient data for the minority class, why not just make some of your own?

Here, we introduce the Synthetic Minority Over-sampling Technique (SMOTE), a popular algorithm used to generate artificial data.

Why SMOTE?

Before covering the ins-and-outs of SMOTE, let's discuss the need for artificial data in the first place.

Why do we need to go through the trouble of creating our own data when we can utilize sampling techniques such as undersampling and oversampling?

Simply put, it is because these approaches come with significant flaws.

Undersampling entails evening the samples for each class by removing records from the majority class. This is sensible in theory, but utterly absurd when put into practice.

Data is precious.

Businesses go through considerable efforts to collect and store data for the sake of the subsequent analysis and model building that will rely on that data.

Try telling to your boss that you want to discard 80% of the data that your data engineers painstakingly procured for the sake of "balance". It won't end well.

Oversampling entails evening the samples for each class by increasing the minority class size by duplicating the minority class samples. This isn't a much better alternative since a model trained with such data will be prone to overfitting.

If discarding or duplicating samples is ineffective, the only remaining option is to create your own data.

SMOTE is one of the most popular algorithms used to achieve this.

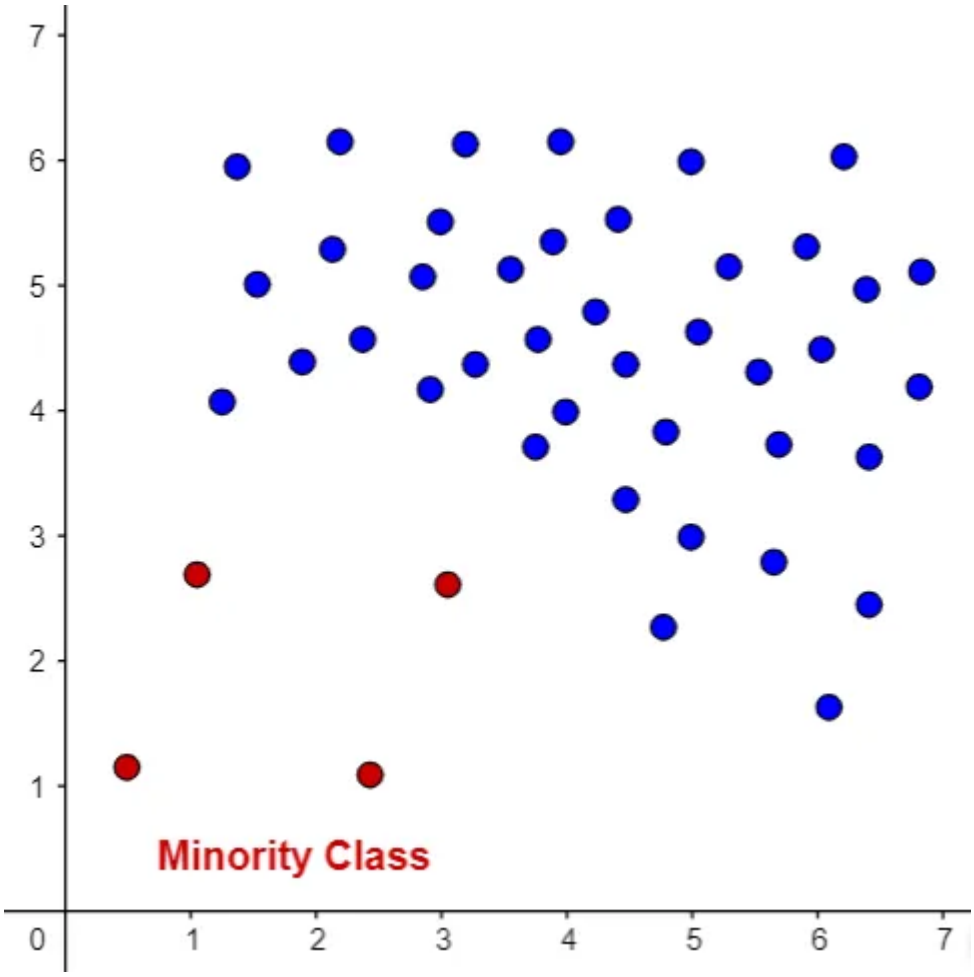
How SMOTE works

SMOTE's inner working seems abstract but is pretty simple.

To understand how SMOTE generates artificial data, it is better to recognize that each sample has its own representation in a feature space.

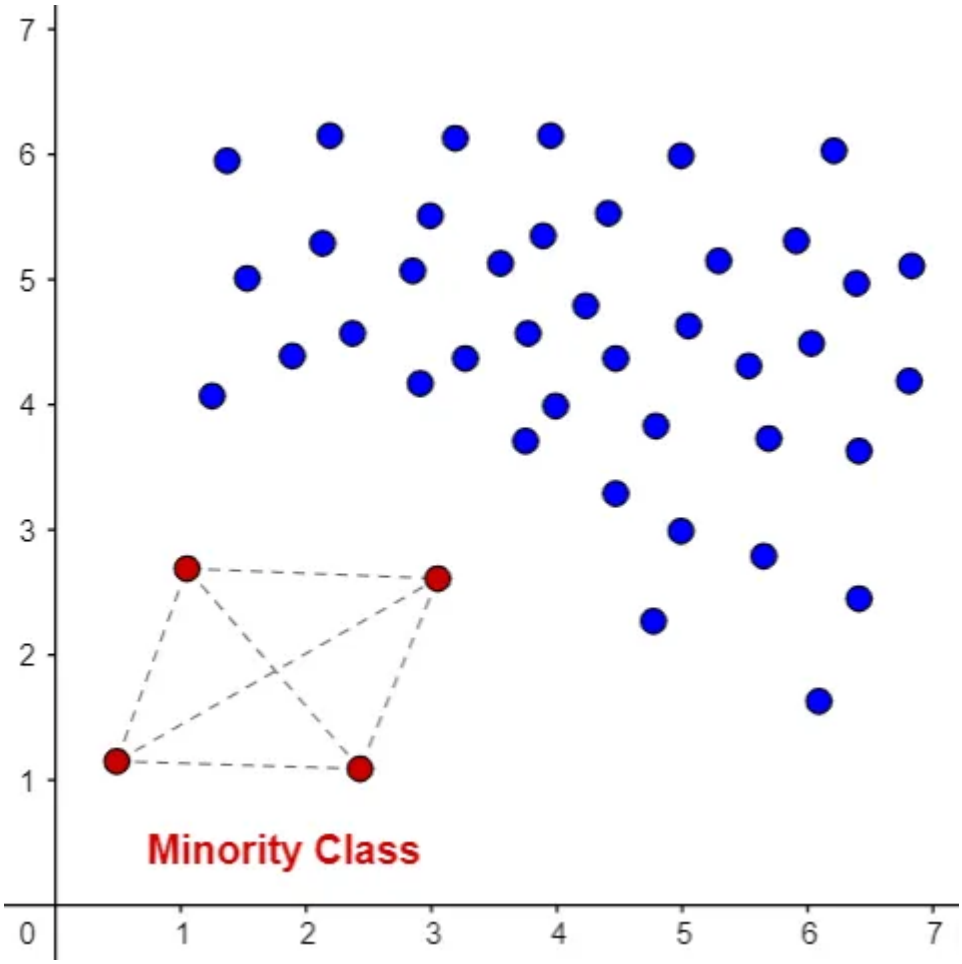
Let's use an example to demonstrate SMOTE. Below is a set of made-up data represented by vectors in a 2D feature space. The red dots represent the minority class.

Note: Most datasets usually have much higher dimensionality. We are using an oversimplified 2D feature space just for the sake of convenience.



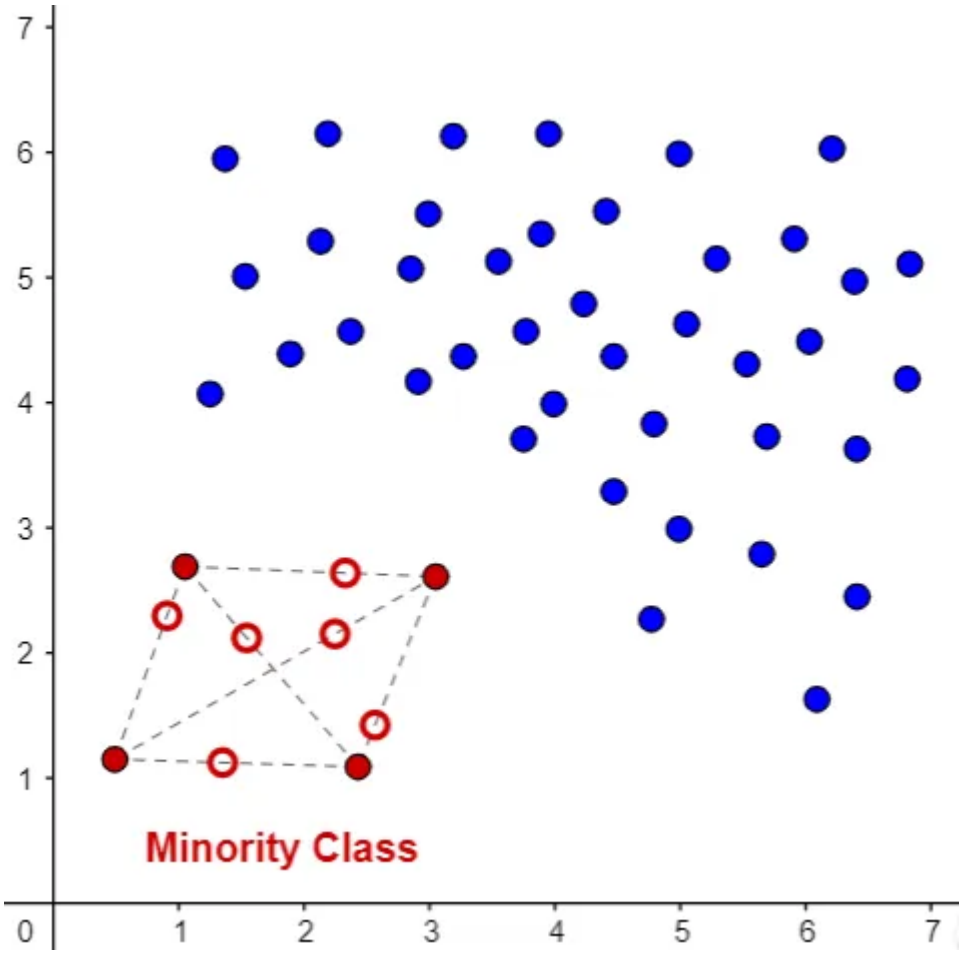
2D Feature Space (Created By Author)

First, SMOTE identifies the k nearest neighbors of the data points from the minority class (in this case, k=3).



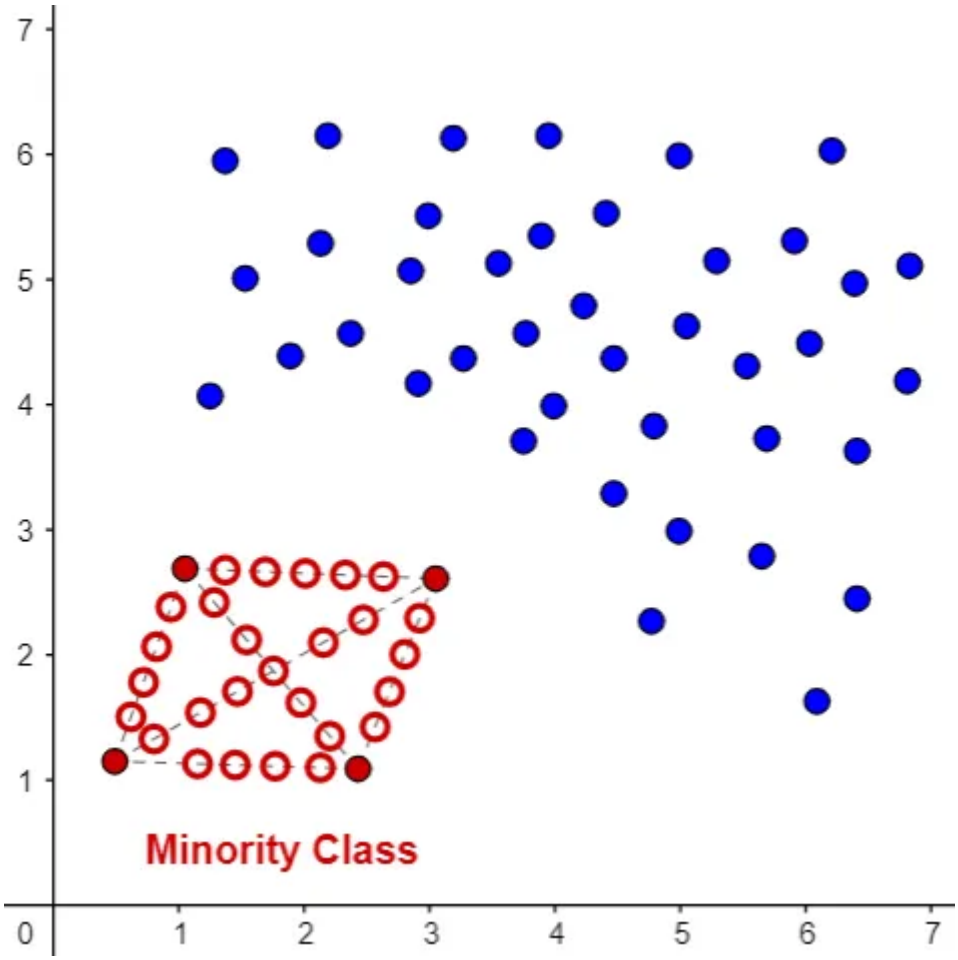
2D Feature Space (Created By Author)

Next, it creates a new point at a random location between all the neighbors. These new points represent artificial data that belong to the minority class.



2D Feature Space (Created By Author)

It will continue generating new data until the data imbalance is resolved.



2D Feature Space (Created By Author)

Drawbacks

SMOTE remedies imbalanced datasets without inviting the risks that come from undersampling or oversampling. However, this method has its own limitations.

Firstly, SMOTE runs the risk of including other classes when searching for neighbors of the minority class. This will result in the generation of data that does not adequately represent the minority class.

Secondly, SMOTE's algorithm is not reliable when used on data with high dimensionality.

Naturally, a lot of these weaknesses have been addressed in tons of research, where different variations of SMOTE are developed. However, that is beyond the scope of this article.

Caution

Photo by Goh Rhy Yan on Unsplash

SMOTE should only be used to augment *training* data. Your test dataset should remain untouched. Applying SMOTE to the entire dataset will result in data leakage.

It also means that the testing data that is used to evaluate the model will consist of artificial data. It may sound obvious, but you want the data that evaluates your model to be real. Otherwise, none of the reported evaluation metrics would mean anything.

Case Study

This time, let's see how SMOTE works when used on real data. For this demonstration, we use a dataset (copyright-free) providing customer information to build a model that determines whether or not they are interested in vehicle insurance. The raw data can be accessed here.

Below, we can see the variables in the dataset.

```
import pandas as pd

# load dataset
df_raw = pd.read_csv('aug_train.csv')

# show columns in dataset
df_raw.info()
```

load dataset hosted with ❤️ by GitHub view raw

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 382154 entries, 0 to 382153
Data columns (total 12 columns):
 #   Column                Non-Null Count   Dtype
---  ------                --------------   -----
 0   id                    382154 non-null  int64
 1   Gender                382154 non-null  object
 2   Age                   382154 non-null  int64
 3   Driving_License       382154 non-null  int64
 4   Region_Code           382154 non-null  float64
 5   Previously_Insured    382154 non-null  int64
 6   Vehicle_Age           382154 non-null  object
 7   Vehicle_Damage        382154 non-null  object
 8   Annual_Premium        382154 non-null  float64
 9   Policy_Sales_Channel  382154 non-null  float64
 10  Vintage               382154 non-null  int64
 11  Response              382154 non-null  int64
dtypes: float64(3), int64(6), object(3)
memory usage: 35.0+ MB
```

Code Output (Created By Author)

The target label is “Response”.

Prior to any model building, we need to preprocess the raw data.

```
# remove missing values
df = df_raw.dropna()

# remove id column
df = df.drop('id', axis=1)

# identify categorical variables
cat_var = [column for column in df if df[column].dtype=='object']

# one hot encode categorical variables
for col in df.columns:
    if df[col].dtype == 'object':
        dummies = pd.get_dummies(df[col])
        df = pd.concat([df, dummies], axis=1)
df = df.drop(cat_var, axis=1)

# create input and output data
target = 'Response'
X = df.drop(target, axis=1)
y = df[target]
```

preprocess dataset hosted with ❤ by GitHub view raw

A quick count shows us the number of records in each class.

```
from collections import Counter

# count the frequency of each class
count = Counter(y)
print(count)
```

count hosted with ❤ by GitHub view raw

```
Counter({0: 319553, 1: 62601})
```

Code Output (Created By Author)

The ratio of the size of the majority class to that of the minority class is about 5:1, a strong indicator of data imbalance.

First, let's build a random forest model to predict customer interest in vehicle insurance without SMOTE.

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestClassifier

# split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

# normalize the data
mms = MinMaxScaler()
X_train = mms.fit_transform(X_train)
X_test = mms.transform(X_test)

# generate predictions with the random forest classifier
rfc = RandomForestClassifier(random_state=42)
rfc.fit(X_train, y_train)
y_pred = rfc.predict(X_test)
```

build random forest model hosted with ❤ by GitHub view raw

We'll evaluate this model with the f-1 score metric.

```
from sklearn.metrics import f1_score
import numpy as np

# compute the f-1 score
f1 = np.round(f1_score(y_test, y_pred),2)
print('F-1 score of model without SMOTE: {}'.format(f1))
```

predictions with random forest model hosted with ❤ by GitHub view raw

```
F-1 score of model without SMOTE: 0.44
```

Code Output (Created By Author)

The model registers an f-1 score of 0.44, leaving room for improvement.

Next, we will repeat the same procedure, but after adding artificial data. We can do this in Python with the imblearn module's SMOTE.

This module allows us some flexibility when creating synthetic data. You get to control things such as:

- The ratio of the number of samples in the majority class to the number of samples in the minority class

- The classes that are targeted by SMOTE
- The number of nearest neighbors to consider when constructing samples

We will stick to the default parameters. This means that the algorithm considers the 5 nearest neighbors and will generate data until the number of minority samples and the number of majority samples are the same.

Note how SMOTE is only applied on the *training* set. As mentioned previously, it is very important not to generate artificial data for the testing set.

```
Counter({1: 239759, 0: 239759})
```

Code Output (Created By Author)

The data imbalance in the training data has now been addressed. So, let's build a random forest classifier with this data and see how it performs on the testing set.

```
F1-score of model with SMOTE: 0.52
```

Code Output (Created By Author)

The model registers an f-1 score of 0.52, proving to be a better predictor of customer interest in vehicle insurance compared to the model trained without artificial data.

Conclusion

Photo by Prateek Katyal on Unsplash

All in all, this brief overview of SMOTE should help you get familiar with the logic behind artificial data generation and its uses in machine learning applications.

SMOTE, though very applicable, has its limitations and drawbacks. Even now, a lot of research and investment is being put into exploring new variants.

If you are interested, you can check out the smote-variants package, which allows you to implement 85 different variants of the algorithm!

I wish you the best of luck in your data science endeavors!

References

1. Möbius. (2022). Learning From Imbalanced Insurance Data, Version 4. Retrieved January 22nd, 2022 from https://www.kaggle.com/arashnic/imbalanced-data-practice.

Data Science Machine Learning

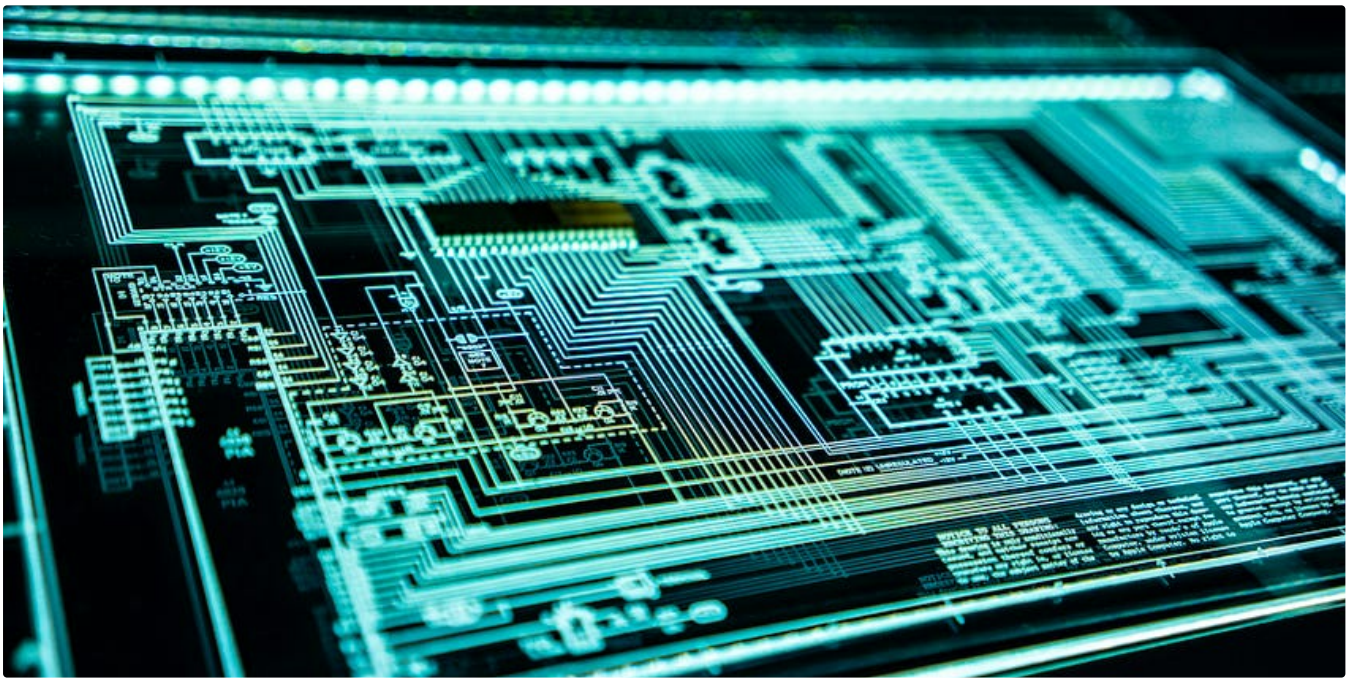
Follow

Written by Aashish Nair

1.1K Followers · Writer for Towards Data Science

Data Scientist aspiring to teach and learn through writing. Reach out to me on LinkedIn: www.linkedin.com/in/aashish-nair.

More from Aashish Nair and Towards Data Science



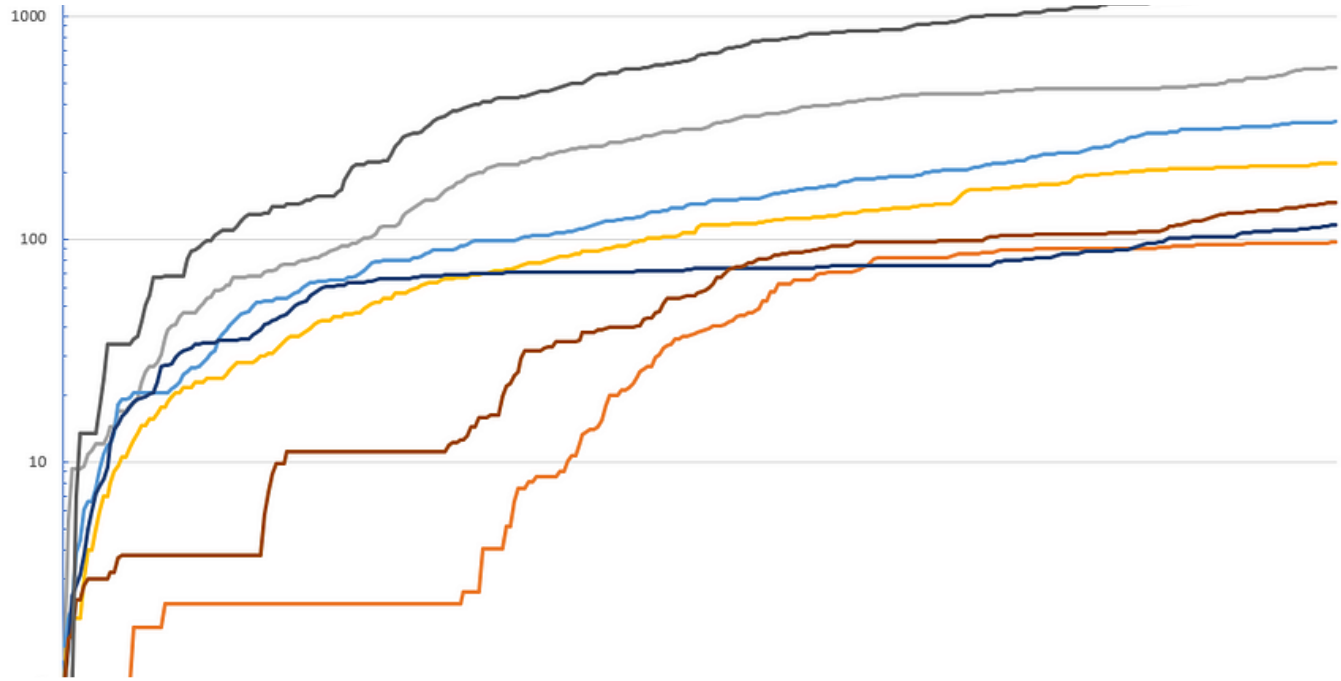
Aashish Nair in Towards Data Science

How to Build a Local Chatbot with Llama2 and LangChain

Overview and Implementation with Python

✨ · 6 min read · Oct 12

101 3



Pau Blasco i Roca in Towards Data Science

My Life Stats: I Tracked My Habits for a Year, and This Is What I Learned

I measured the time I spent on my daily activities (studying, doing sports, socializing, sleeping…) for 332 days in a row.

12 min read · Nov 21

4.5K 80

Anthony Alcaraz in Towards Data Science

Embeddings + Knowledge Graphs: The Ultimate Tools for RAG Systems

The advent of large language models (LLMs) , trained on vast amounts of text data, has been one of the most significant breakthroughs in...

✨ · 10 min read · Nov 14

1.1K 9

Aashish Nair in Towards Data Science

Targeting Multicollinearity With Python

Learn how Python's features help deal with this 8-syllable enigma with ease

6 min read · Dec 6, 2021

155 2

See all from Aashish Nair

See all from Towards Data Science

Recommended from Medium

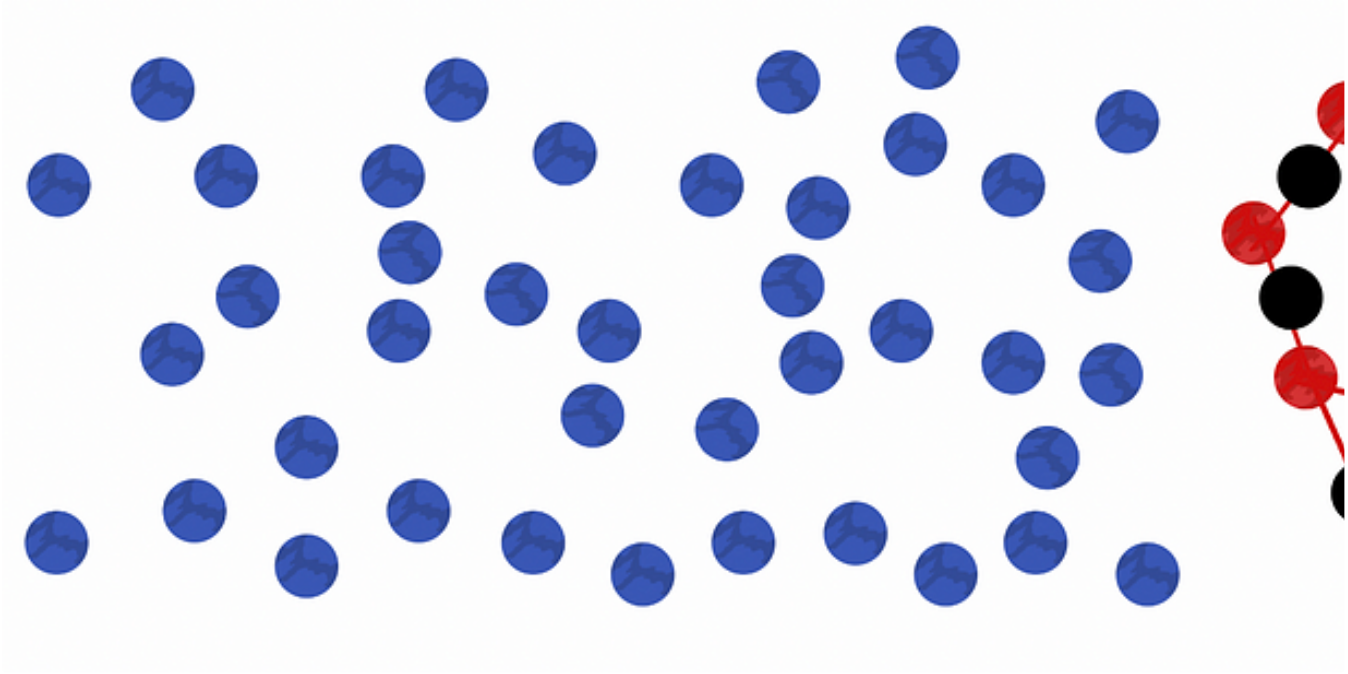
Indraneel Dutta Baruah

How to handle imbalanced data—Oversampling techniques

Let's learn about various versions of SMOTE and ADASYN, etc., and their Python implementation!

· 10 min read · Sep 24

4



Essam Wisam in Towards Data Science

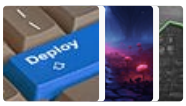
Class Imbalance: From SMOTE to BorderlineSMOTE1, SMOTE-NC and SMOTE-N

Exploring three algorithms to tackle the class imbalance problem

12 min read · Aug 30

25

Lists

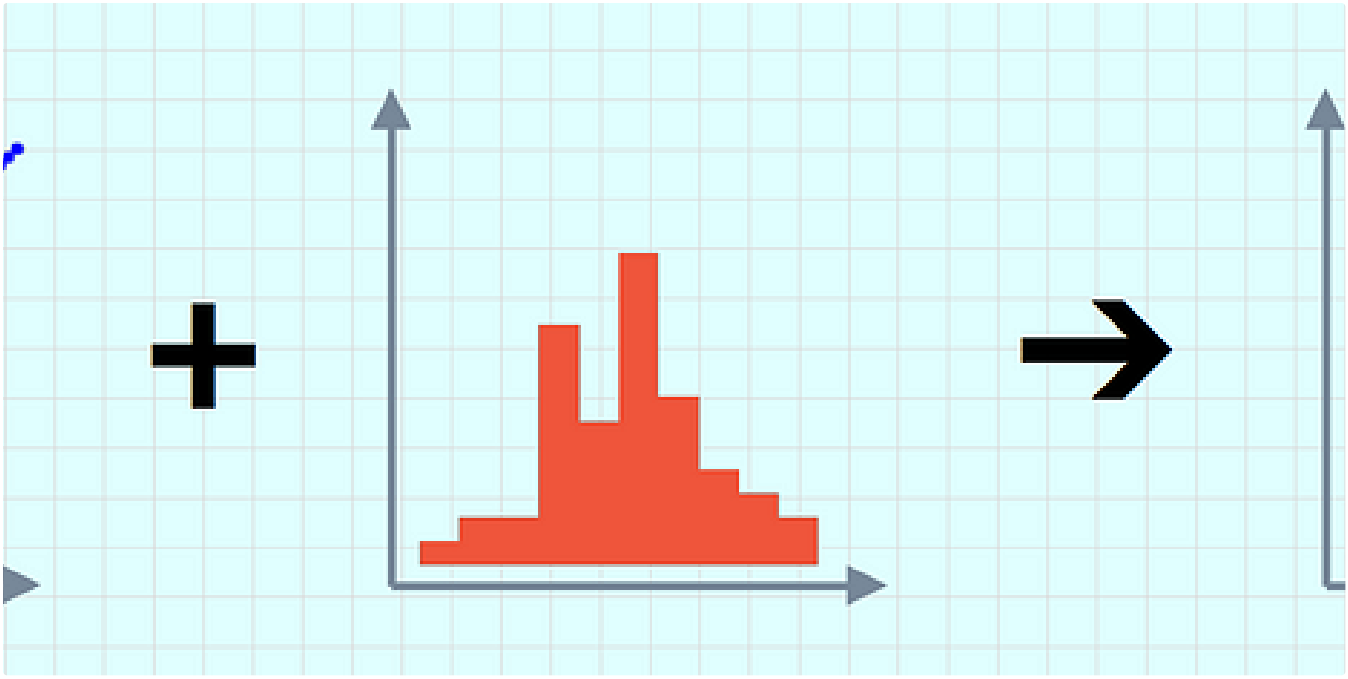


Predictive Modeling w/ Python
20 stories · 686 saves

Practical Guides to Machine Learning
10 stories · 783 saves

Natural Language Processing
976 stories · 469 saves

data science and AI
38 stories · 2 saves



MS Somanna

Guide to Adding Noise to your Data using Python and Numpy

In this article you'll learn why you should add noise to your otherwise perfect synthetic data, what are the types of noises you can add...

8 min read · Jul 22

73 1

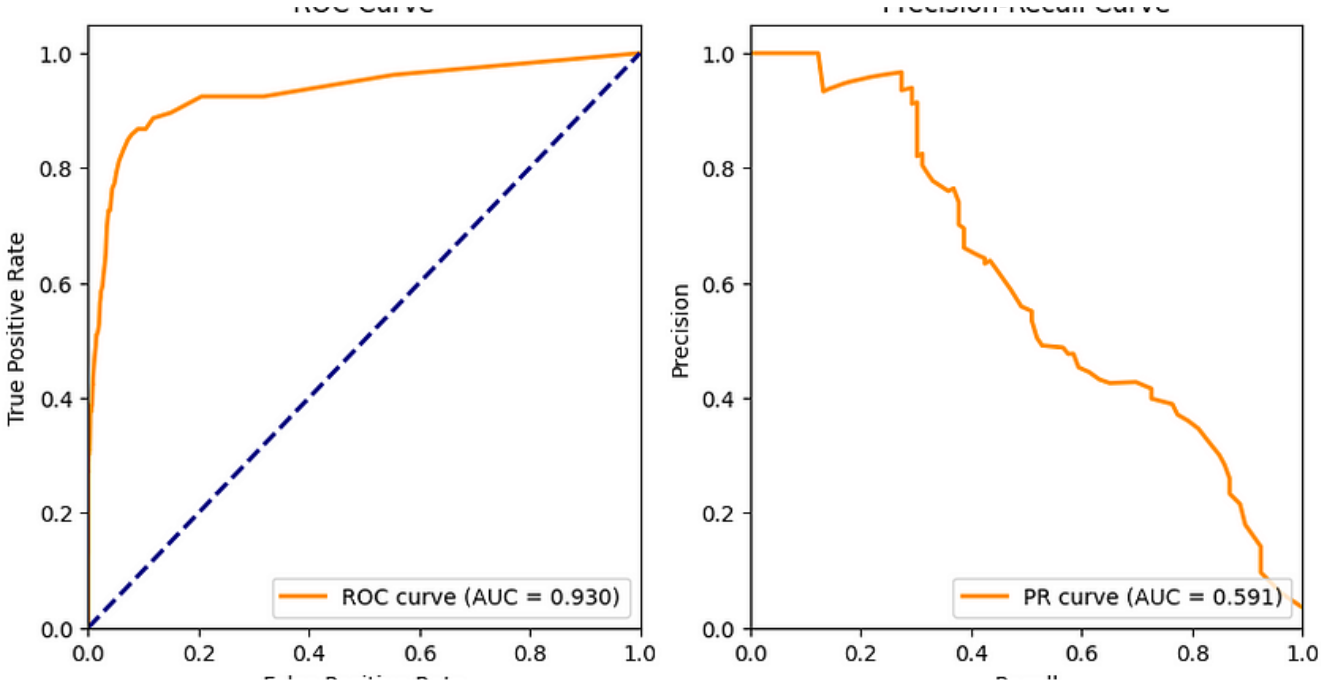
XQ in The Research Nest

Exploring GANs to Generate Synthetic Data

A simple step-by-step tutorial using the IRIS dataset

11 min read · Jul 14

6



Juan Esteban de la Calle

How and Why I Switched from the ROC Curve to the Precision-Recall Curve to Analyze My Imbalanced...

Machine learning is transforming how we tackle problems and make decisions in an array of sectors, from healthcare to finance. Binary...

5 min read · Jul 10

55

M. Masum, PhD

Imbalanced Data: Analyze First, Sample Later

Classification is a common task in machine learning, where the goal is to classify or label given data or observations. Specifically, in...

✨ · 6 min read · Oct 18

46