Cheat Sheet for Statistical Analysis in Python

Statistics is complicated. Why not simplify it a little bit?

Killian Kahalley · Follow

Published in Geek Culture

4 min read · Jun 23, 2021

As a fairly new data scientist coding in Python, I've had to do quite a bit of statistical analysis on a bunch of data. Whether it be running hypothesis tests, determining confidence intervals, or plotting normal curves, it seems like there are thousands of things to remember when trying to accurately look at data to make predictions and correlations. So, to keep fellow data scientists free from the overwhelming feeling I had when I was learning all of these new tools, I've compiled a cheat sheet of sorts to reference when you need to make common statistical analyses.

Since Python comes built-in with many statistical analysis tools and mathematical functions, we will be using its base packages for most of these tests.

You skipping all the messy statistical math

Generally Useful Functions

Before running any of these tests, you must run:

- **from math import ***
- **from itertools import ***
- **from collections import ***

This allows you to use all the tools in each of these packages in python without calling them, such as typing factorial(~) instead of math.factorial(~).

> **factorial(#)**

This function returns the factorial of the number that you give it. For example, if you gave it the number 10, it would calculate 10 * 9 * 8*… *1 and give you the number 3,628,800. This function is very useful for finding the number of combinations in a set and lots more.

> **permutations (iterable, n)**

This function creates all permutations of length n for whatever you put in it. It could be a list of numbers, a string, or anything else that can be iterated over (used in a for loop). For example, if you were to use "ABC" as the iterable, and 3 and your 'n', you would receive this result:

```
('A', 'B', 'C'),
('A', 'C', 'B'),
('B', 'A', 'C'),
('B', 'C', 'A'),
('C', 'A', 'B'),
('C', 'B', 'A')
```

> **Counter (iterable)**

This function takes an iterable and returns each element of that iterable with a count for how many times it appeared. This is very useful for making frequency tables, or for finding the mode of a set. For example, Counter([‘a’,’a’,’b’,’c’,’c’,’c’]) returns:

Counter({‘a’: 2, ‘b’: 1, ‘c’: 3})

Statistically Significant Functions (hehe)

In order to utilize most of Python’s pre-loaded statistical functions, you must import SciPy, and more specifically, the ‘stats’ module from SciPy:

- **from scipy import stats (or *)**

From there, most statistical calculations you need to run can be accessed. These next few functions are all going to be explained briefly:

> **stats.norm.ppf (proportion)**

- This function gives a z-score for what proportion you give it. For example, if you give it 0.95, it will give you the z-score that includes 95% of the values.

> **stats.norm.cdf (z-score)**

- Does the opposite of the ppf function. You give it a z-score, and it gives you the proportion of values below that z-score. If you have a z-score for a specific value, you can plug that into this formula to see the proportion of values smaller than it.

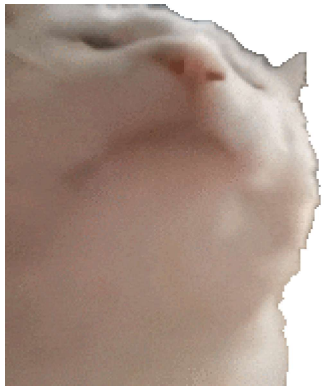
> **stats.t.sf (t-score, df)**

- This gives the area to the right of whatever t-score you specify with the degrees of freedom. This is useful for finding the proportion of values larger than the t-score you give it for your data set (essentially 1 - cdf).

> **statts.ttest_ind (x, y, equal_var = True/False)**

- While this one may look complicated, it’s actually pretty simple. You give it two sets of data, one x and one y, and then specify whether the two have the same variance or not. It runs a 2-sample T-test on the means of the two data sets and gives you the t-score and the p-value for the difference of those means.

Bonus Functions!



> **np.random.normal (mean, std, size)**

- This creates random samples of whatever size you give it from a normal distribution, centered around the mean you give it and with the standard deviation you give it. Useful for generating samples for other statistical tests when you know the population mean and standard deviation.

> **stats.skew(x)**

- Gives you the skew for whatever list or data you enter into it. If 0, then you have no skew. If >0, you have a right skew. If <0, you have a left skew.

> **stats.kurtosis(x)**

- Gives you the kurtosis for the data entered. If =0, the data is normal-shaped. If >0, the data is tall and skinny. If <0, the data is short and fat. (for more info on kurtosis, click here)

> **np.random.binomial (trials, probability of success)**

- Randomly generates the number of successes with a given number of trials and a probability of success

Finally… The Holy Grail…

> **import statsmodel.api as sm**
>
> **model = sm.OLS(data1, sm.add_constant(data2))**
>
> **model.summary()**

These few lines of code will give you lots of statistical data between two different sets of data. You can pass specific columns from DataFrames as the two data sets, and that's what I personally use it the most for. This summary chart gives you data on the data's skew, kurtosis, the p-value of the difference in means, the R-squared value, the equation for the trendline, and so much more. This statistical modeling technique is key when comparing two different sets of data.

That's it! Use these cheat codes responsibly!

Python Statistics Coding Inference Cheatsheet

K

Follow

Written by Killian Kahalley

6 Followers · Writer for Geek Culture

Student at Birmingham-Southern College

More from Killian Kahalley and Geek Culture

Killian Kahalley

How Does Sleep Affect Testing Performance?

An introductory study into my own sleep data

4 min read · Aug 5, 2021

52 1

---

parse_expenses.py

```
import datetime

def parse_expenses(expenses_string):
    """Parse the list of expenses and return the list of triples (date, value, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2016-01-02 -34.01 USD
        2016-01-03 2.59 DKK
        2016-01-03 -2.72 EUR
    """
    expenses = []
    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
                         float(value),
                         currency))
    return expenses
```

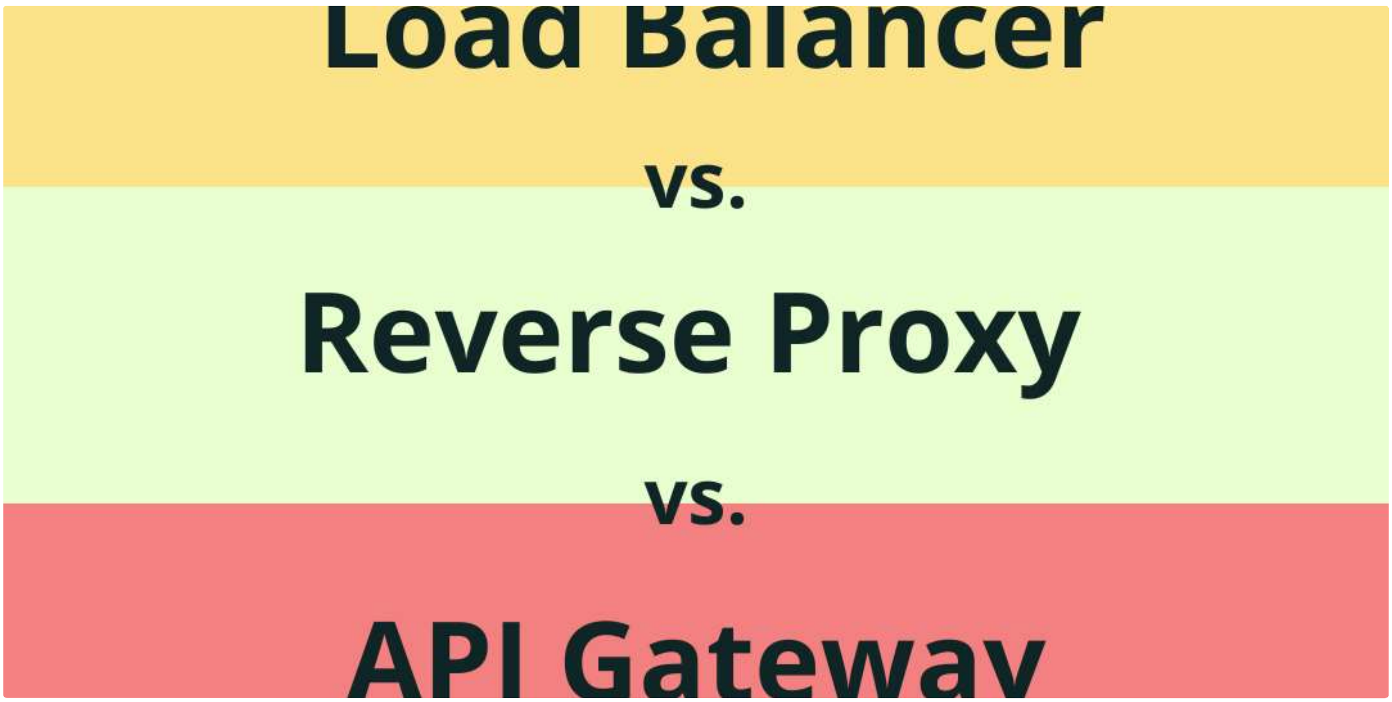
Jacob Bennett in Geek Culture

The 5 paid subscriptions I actually use in 2023 as a software engineer

Tools I use that are cheaper than Netflix

· 4 min read · Mar 26

4K 35



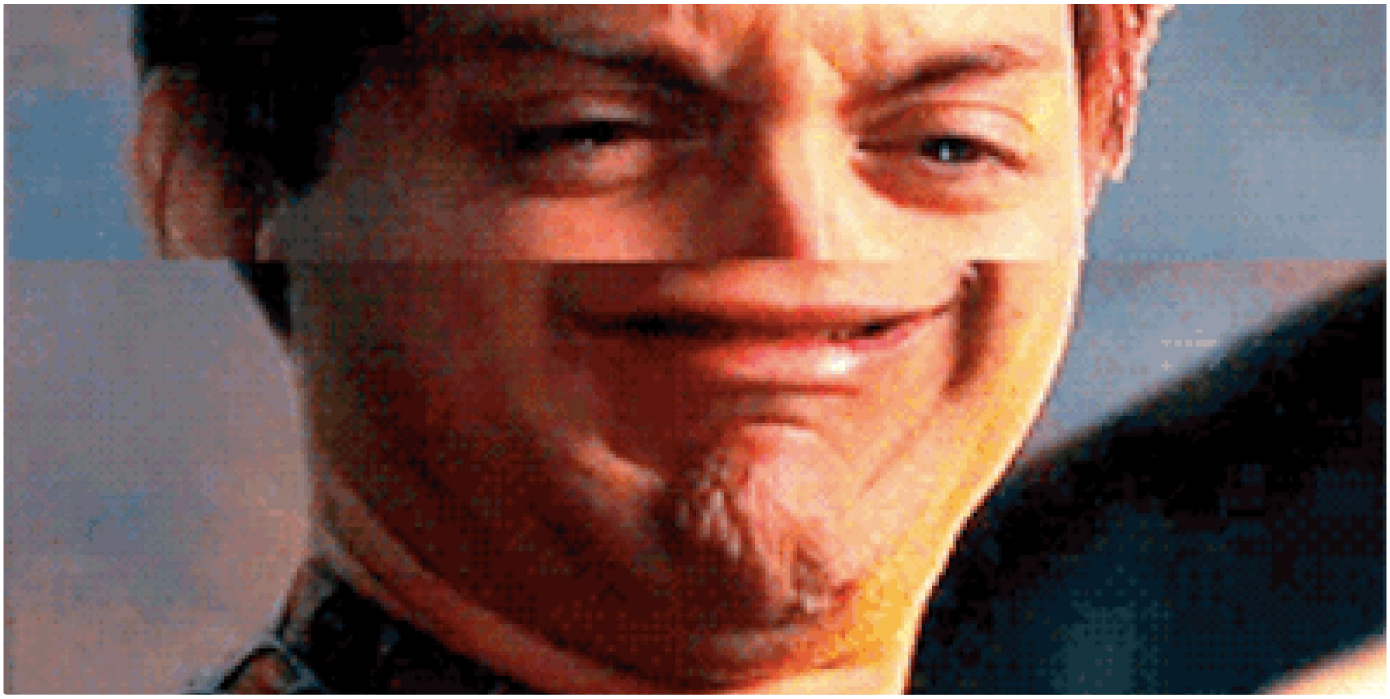
Arslan Ahmad in Geek Culture

Load Balancer vs. Reverse Proxy vs. API Gateway

Understanding the Key Components for Efficient, Secure, and Scalable Web Applications.

12 min read · May 17

1K 5



K Killian Kahalley

Paper has a place in Data Science

Writing your notes on paper can turn you into an elite programming machine.

5 min read · Jun 3, 2021

See all from Killian Kahalley

See all from Geek Culture

Recommended from Medium

Matt Chapman in Towards Data Science

The Portfolio that Got Me a Data Scientist Job

Spoiler alert: It was surprisingly easy (and free) to make

· 10 min read · Mar 25

4.1K 71

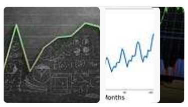
Youssef Hosni in Level Up Coding

13 SQL Statements for 90% of Your Data Science Tasks

Structured Query Language (SQL) is a programming language designed for managing and manipulating relational databases. It is widely used by...

· 15 min read · Feb 27

Lists



Predictive Modeling w/ Python

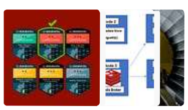
18 stories · 228 saves

Coding & Development

11 stories · 95 saves

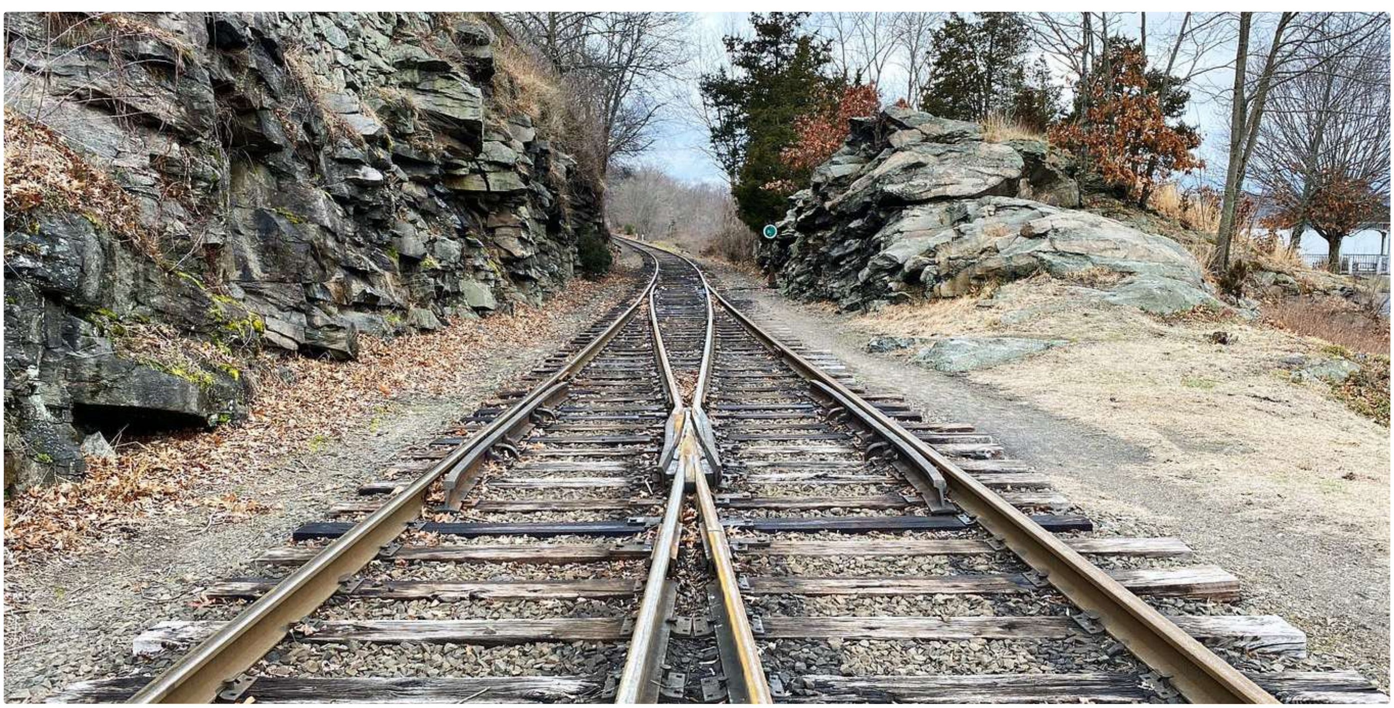
Practical Guides to Machine Learning

10 stories · 238 saves



New_Reading_List

174 stories · 57 saves



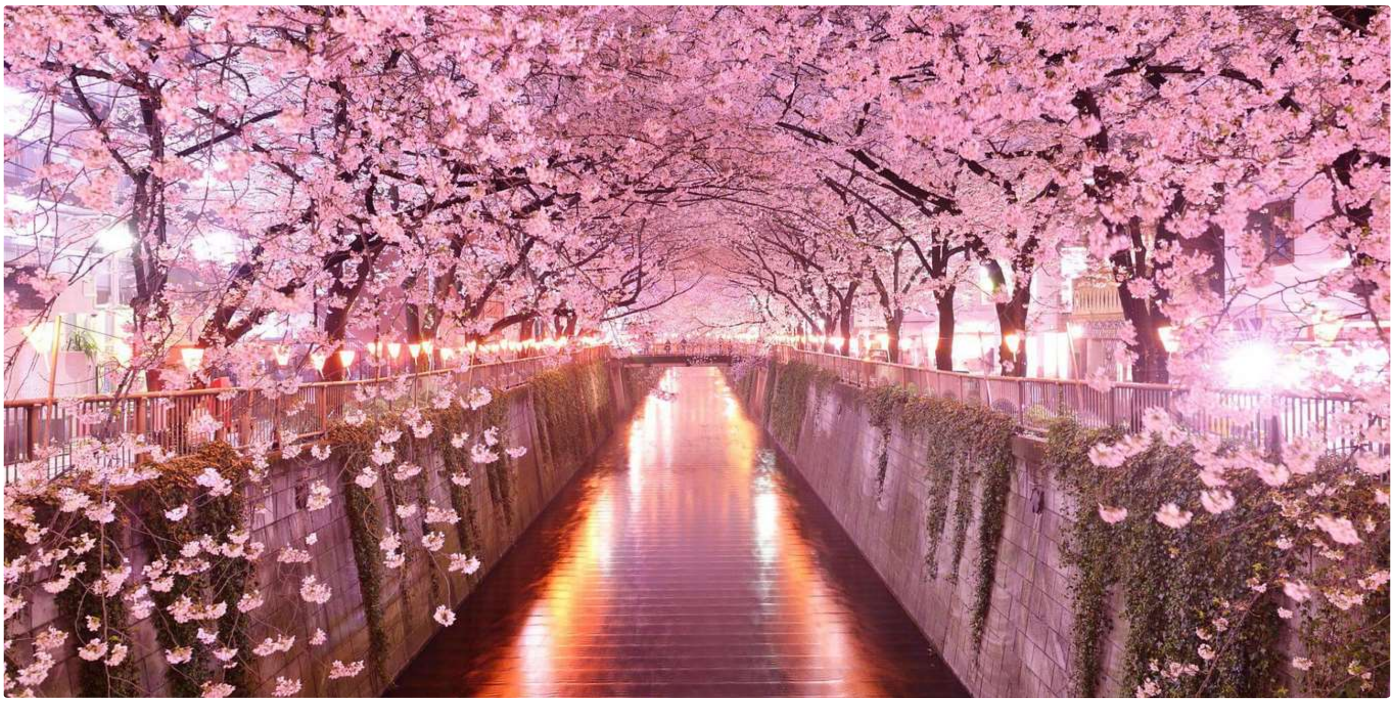
Cornellius Yudha Wijaya in Towards AI

3 Pandas Functions for DataFrame Merging

Learn how Pandas merging functions work with code examples

· 14 min read · 4 days ago

274 4



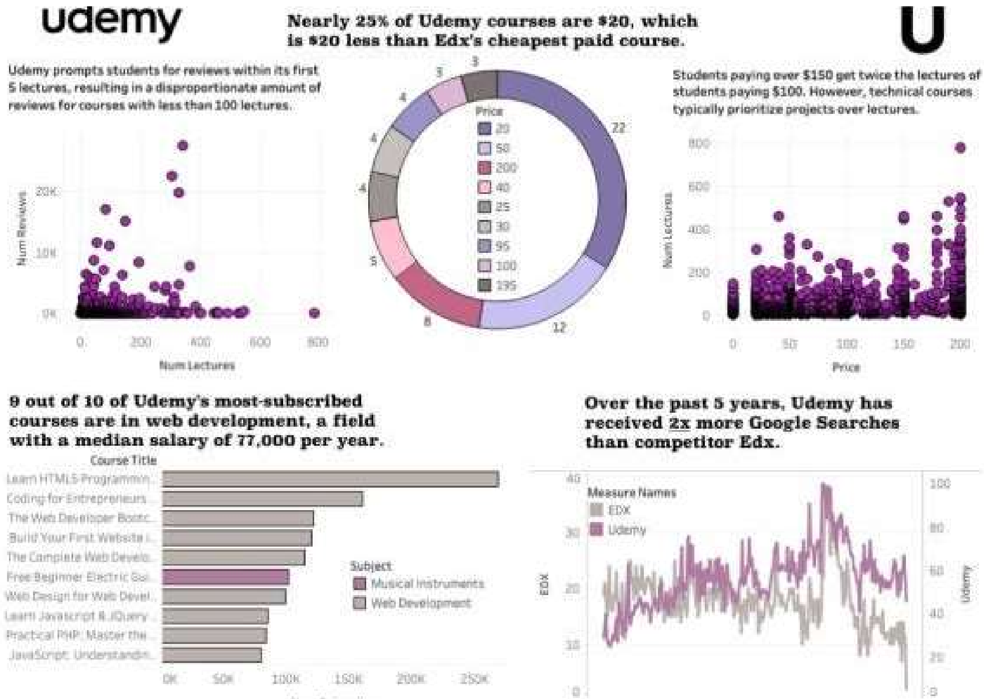
Yang Zhou in TechToFreedom

9 Fabulous Python Tricks That Make Your Code More Elegant

Pythonic is a synonym for elegant

· 5 min read · Nov 14, 2022

2.4K 23



Zach Quinn in Pipeline: Your Data Engineering Resource

Creating The Dashboard That Got Me A Data Analyst Job Offer

A walkthrough of the Udemy dashboard that got me a job offer from one of the biggest names in academic publishing.

· 9 min read · Dec 5, 2022

1.3K 20

```
commit ffcf2c01b7ef612893529cef188cc1961ed64521 (HEAD -> master, origin/master, origin/bors/staging, origin/HEAD)
Merge: fc991bf81 5159211da
Author: iohk-bors[bot] <43231472+iohk-bors[bot]@users.noreply.github.com>
Date:   Tue Nov 8 17:44:34 2022 +0000

    Merge #4563

    4563: New p2p topology file format r=coot a=coot

    Fixes #4559.


    Co-authored-by: Marcin Szamotulski <coot@coot.me>
    Co-authored-by: olgahryniuk <67585499+olgahryniuk@users.noreply.github.com>

commit fc991bf814891a9349f22cf278632d39b04d4628
Merge: 5633d1c05 5cd94d372
Author: iohk-bors[bot] <43231472+iohk-bors[bot]@users.noreply.github.com>
Date:   Tue Nov 8 13:07:58 2022 +0000

    Merge #4613

    4613: Update building-the-node-using-nix.md r=CarlosLopezDeLara a=CarlosLopezDeLara

    Build the cardano-node executable. No default configuration.

    Co-authored-by: CarlosLopezDeLara <carlos.lopezdelara@iohk.io>

commit 5159211da7a644686a973e4fb316b64ebb1aa34c
Author: olgahryniuk <67585499+olgahryniuk@users.noreply.github.com>
Date:   Tue Nov 8 13:25:10 2022 +0200
```

Jacob Bennett in Level Up Coding

Use Git like a senior engineer

Git is a powerful tool that feels great to use when you know how to use it.

· 4 min read · Nov 15, 2022

8.7K 87

See more recommendations