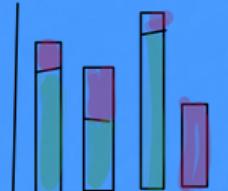
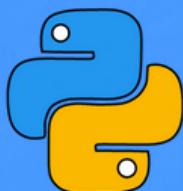


علم البيانات

200 نصيحة في بايثون وعلم البيانات



kabbani-books.com

ترجمة:

د. علاء طعيمة



Daily Dose of
Data Science



avichawla.substack.com

بِمَهْ تَعَالَى

علم البيانات

أكثر من 200 نصيحة في علم البيانات وبأيّثون

ترجمة:

د. علاء طعينة

مقدمة المؤلف

يتطلب كونك عالم بيانات خبرة في العديد من المجالات. يجب أن تكون جيداً في استخدام الأدوات المناسبة، مثل NumPy و Pandas و Sklearn وما إلى ذلك.

هذه الأدوات لا غنى عنها لدوره حياة تطوير العديد من المشاريع التي تعتمد على البيانات، مما يجعلها مهارات أساسية لبدء / الحفاظ على مهنتي في علم البيانات.

علاوة على ذلك، يعد SQL أمراً محورياً لجميع أدوار علم البيانات تقريباً اليوم. بالإضافة إلى ذلك، يعد سرد البيانات أمراً ضرورياً بنفس القدر لنقل النتائج والأفكار بشكل فعال إلى جمهور أوسع.

لتبسيط رحلة علم البيانات هذه وجعلها تبدو أقل صعوبة ويمكن الوصول إليها بشكل أكبر، كان Avi Chawla يشارك نصائح يومية لأكثر من 200 يوم.

وبعد إكمال 200 يوم، قام بإنشاء [أرشيف PDF كامل](#)، والذي يسرد جميع المنشورات التي كتبها.

لقد حاولت قدر المستطاع ان اترجم هذه النصائح في مجال بايثون وعلم البيانات مع الشرح المناسب والكافي، ومع هذا يبقى عملاً بشرياً يحتمل التقصص، فإذا كان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدينا الإلكتروني . alaa.taima@qu.edu.iq

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخل في مجالات علم البيانات ومساعدة القارئ العربي على تعلم هذا المجال. اسأل الله التوفيق في هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصين في مجال علم البيانات. ونرجو لك الاستمتاع مع الكتاب ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية

العراق

المحتويات

[1] دليل مرئي للدرج الاشتقافي العشوائي والدفعات الصغيرة والدفعي	A Visual Guide to Stochastic, Mini-batch, and Batch Gradient Descent	16.....
[2] الفرق الأقل شهرة بين حلقة for وList Comprehensions	A Lesser-Known Difference Between For-Loops and List Comprehensions	19.....
[3] قيود PCA التي يتجاهلها الكثير في كثير من الأحيان	The Limitation of PCA Which Many Folks Often Ignore	21.....
[4] طرق السحر: جوهرة قيمة من البرمجة كيانية التوجه لباليتون	Magic Methods: An Underrated Gem of Python OOP	24.....
[5] تصنیف خوارزمیات الانحدار التي لا يكلف الكثيرون أنفسهم عناء تذكرها	The Taxonomy Of Regression Algorithms That Many Don't Bother To Remember	27.....
[6] نهج تم تجاهله بشدة لتحليل إطارات بيانات Pandas A Highly Overlooked Approach To Analysing Pandas DataFrames	Pandas A Highly Overlooked Approach To Analysing Pandas DataFrames	29.....
[7] تصور التغيير في الترتيب بمزور الوقت باستخدام مخططات Bump Visualise The Change In Rank Over Time With Bump Charts	Bump Visualise The Change In Rank Over Time With Bump Charts	31.....
[8] استخدام هذه التقنية البسيطة حتى لا تعاني أبداً مع TP و TN و FP و FN مرة أخرى	Use This Simple Technique To Never Struggle With TP, TN, FP and FN Again	33.....
[9] المفهوم الخاطئ الأكثر شيوعاً حول العمليات الداخلية في الباندا	The Most Common Misconception About Inplace Operations in Pandas	35.....
[10] أنشئ تطبيقات ويب أنيقة مباشرة من Jupyter Notebook باستخدام Mercury	Build Mercury Jupyter Notebook Right From Jupyter Notebook with Mercury	37.....
[11] كن عالم بيانات ثنائى اللغة مع هذه الباندا الترجمات	Become A Bilingual Data Scientist SQL With These Pandas to SQL Translations	39.....
[12] ميزة أقل شهرة من Sklearn لتدريب النماذج على مجموعات البيانات الكبيرة	A Lesser-Known Feature of Sklearn To Train Models on Large Datasets	41.....
[13] سطر واحد بسيط لإنشاء مخططات Matplotlib ذات المظهر الاحترافي	A Simple One-Liner to Create Professional Looking Matplotlib Plots	44.....
[14] تجنب هذا الخطأ المكلف عند فهرسة إطار البيانات	Avoid This Costly Mistake When Indexing A DataFrame	46.....
[15] إشارات سطر أوامر لتشغيل سكريبتات باليتون بشكل أكثر مرونة	Command Line Flags To Run Python Scripts More Flexibly	49.....
[16] تنفس KMeans: أفضل وأسرع بديل لـ KMeans	Breathing KMeans: A Better and Faster KMeans Alternative to KMeans	51.....

[17] كم عدد الأبعاد التي يجب عليك تقليل بياناتك إليها عند استخدام PCA	How Many Dimensions Should You Reduce Your Data To When Using PCA	54.....
[18] تم شحن ميتو للتو مع الذكاء الاصطناعي!	Mito Just Got Supercharged With AI!	57.....
[19] كن حذراً قبل رسم أي استنتاجات باستخدام الإحصائيات الموجزة	Be Cautious Before Drawing Any Conclusions Using Summary Statistics	59.....
[20] استخدم كائنات بايثون المخصصة في سياق منطقي	Use Custom Python Objects In A Boolean Context	61.....
[21] دليل مرئي لتقنيات أخذ العينات في التعلم الآلي	A Visual Guide To Sampling Techniques in Machine Learning	63.....
[22] ربما تم إعطاؤك معلومات غير كاملة حول ثبات الصف	You Were Probably Given Incomplete Info About A Tuple's Immutability	66.....
[23] خدعة بسيطة تعمل على تحسين جودة مخططات Matplotlib بشكل كبير	A Simple Trick That Significantly Improves The Quality of Matplotlib Plots	68.....
[24] دليل مرئي ومبسط للغاية لـ PCA	A Visual and Overly Simplified Guide to PCA	70.....
[25] عزز نواة Jupyter الخاصة بك مع ipyflow	Supercharge Your Jupyter Kernel With ipyflow	73.....
[26] ميزة أقل شهرة لإنشاء المخططات باستخدام Plotly	A Lesser-known Feature of Plotly	75.....
[27] قيود المسافة الإقليدية التي كثيراً ما يتجاهلها الكثيرون	The Limitation Of Euclidean Distance Which Many Often Ignore	77.....
[28] تصور تأثير معلمة التنظيم	Visualizing The Impact of Regularization Parameter	80.....
[29] AutoProfiler: ملف تعريف إطار البيانات الخاص بك تلقائياً أثناء عملك	Automatically Profile Your DataFrame As You Work	82.....
[30] القليل من الجهد الإضافي يمكن أن يحول بشكل كبير مهاراتك في سرد القصص	A Little Bit Of Extra Effort Can Hugely Transform Your Storytelling Skills	84.....
[31] ميزة مخفية سيئة في بايثون لا يعرفها الكثير من المبرمجين	A Nasty Hidden Feature of Python That Many Programmers Aren't Aware Of	86.....
[32] تصور تفاعلي لشجرة قرار مع مخطط سانكي	Interactively Visualise A Decision Tree With A Sankey Diagram	89.....
[33] استخدم المدرجات التكراري بحذر. إنها مضللة للغاية!	Use Histograms with Caution. They Are Highly Misleading	91.....
[34] ثلاثة طرق بسيطة (فورية) تجعل مخططات التشتت خالية من الفوضى	Three Simple Ways To (Instantly) Make Your Scatter Plots Clutter Free	94.....
[35] نقطة (عالية) مهمة يجب مراعاتها قبل استخدام KMeans في المرة القادمة	A (Highly) Important Point to Consider Before You Use KMeans Next Time	97.....

[36] لماذا يجب تجنب إلحاد الصفوف بإطار بيانات	Why You Should Avoid Appending Rows To
100.....	A DataFrame
[37] يحتوي Matplotlib على العديد من الأحجار الكريمة المخفية. هنا واحد منهم	Matplotlib Has
102.....	Numerous Hidden Gems. Here's One of Them
[38] شيء غير بديهي حول قواميس بايثون	A Counterintuitive Thing About Python
104.....	Dictionaries
[39] ربما تكون أسرع طريقة لتنفيذ كود بايثون الخاص بك	Probably The Fastest Way To
106.....	Execute Your Python Code
[40] هل أنت متأكد من أنك تستخدم مصطلحات Pandas الصحيحة؟	Are You Sure You Are Pandas
108.....	?Using The Correct Pandas Terminologies
[41] هل عدم توازن الفئة دائمًا مشكلة كبيرة يجب التعامل معها؟	Is Class Imbalance Always a Big Problem to Deal With
111.....	?Big Problem to Deal With
[42] حيلة بسيطة تجعل الخرائط الحرارية أكثر أناقة	A Simple Trick That Will Make Heatmaps
113.....	More Elegant
[43] مقارنة مرئية بين المجاميع المحلية والقائمة على الكثافة	A Visual Comparison Between Locality and Density-based Clustering
115.....	
[44] لماذا لا نطلق عليه التصنيف اللوجستي بدلاً من ذلك؟	Why Don't We Call It Logistic
117.....	?Classification Instead
[45] شيء نموذجي حول أشجار القرار يتجاهله كثيرون في كثير من الأحيان	A Typical Thing About Decision Trees Which Many Often Ignore
119.....	
[46] تحقق دائمًا من متغير الإخراج قبل استخدام الانحدار الخطى	Always Validate Your Output
121.....	Variable Before Using Linear Regression
[47] حقيقة غير بديهية حول دوال بايثون	A Counterintuitive Fact About Python Functions
123.....	
[48] لماذا من المهم خلط مجموعة البيانات عشوائياً قبل تدريب نموذج التعلم الآلي	Why Is It Important To Shuffle Your Dataset Before Training An ML Model
125.....	
[49] قيود الخريطة الحرارية التي تبطئ تحليل بياناتك	The Limitations Of Heatmap That Are Slowing Down Your Data Analysis
127.....	
[50] قيود ارتباط بيرسون الذي يتجاهلها كثيرون في كثير من الأحيان	The Limitation Of Pearson Correlation Which Many Often Ignore
129.....	
[51] لماذا يُنصح عادةً بوضع بذور للمولدات العشوائية؟	Why Are We Typically Advised to Set Seeds for Random Generators
130.....	
[52] تقنية تم الاستخفاF بها لتحسين تصورات البيانات الخاصة بك	An Underrated Technique To Improve Your Data Visualizations
131.....	

أداة بدون كود لإنشاء المخططات والجداول المحورية في Jupyter [53]	132..... Charts and Pivot Tables in Jupyter
إذا لم تكن قادراً على برمجة نهج موجه، فجرّب هذا [54]	133..... .Vectorized Approach, Try This
لماذا يُنصح عادةً بعدم التكرار مطلقاً عبر إطار بيانات؟ [55]	135..... ?Never Iterate Over A DataFrame
قد يكون اللالعب بالكائنات القابلة للتغيير في لغة بايثون مثيراً في بعض الأوقات [56]	136..... Manipulating Mutable Objects In Python Can Get Confusing At Times
يمكن لهذا التعديل الصغير أن يعزز بشكل كبير من وقت تشغيل This Small Tweak KMeans [57]	138..... Can Significantly Boost The Run-time of KMeans
معظم مبرمجي بايثون لا يعرفون هذا عن البرمجة كائنية التوجّه في بايثون [58]	140..... Programmers Don't Know This About Python OOP
من قال إن Matplotlib Cannot إنشاء مخططات تفاعلية؟ [59]	142..... ?Create Interactive Plots
لا تقم بإنشاء مخططات شريطية فوضوية. بدلاً من ذلك، جرب المخططات الفقاعية! [60]	144..... !Create Messy Bar Plots. Instead, Try Bubble Charts
يمكنك إضافة قائمة كمفتاح قاموس (فنياً)! [61]	146..... !(Technically)
غالباً ما يهمل معظم مستخدمي التعلم الآلي هذا أثناء استخدام الانحدار الخطى ML [62]	148..... Folks Often Neglect This While Using Linear Regression
35 مكتبة بايثون مخفية تعتبر جواهر مطلقة [63]	150..... Absolute Gems
استخدم المخططات الصندوقية بحذر! قد تكون مضللة. [64]	151..... They May Be Misleading
تقنيّة تم الاستخفاف بها لإنشاء مخططات بيانات أفضل [65]	153..... Create Better Data Plots
اضافة إطار بيانات Pandas الذي ينتظرها كل عالم بيانات [66]	154..... Extension Every Data Scientist Has Been Waiting For
عزّز Shell مع بايثون باستخدام Xonsh [67]	155..... Supercharge Shell With Python Using Xonsh
لا يعرف معظم مستخدمي سطر الأوامر هذه الحيلة الرائعة حول استخدام الترمinal [68]	156..... Command-line Users Don't Know This Cool Trick About Using Terminals
حيلة بسيطة لتحقيق أقصى استفادة من الجداول المحورية في Pandas [69]	157..... Make The Most Out of Pivot Tables in Pandas
لماذا لا تقدم بايثون تغليف برمجة كائنية التوجّه حقيقي [70]	158..... OOP Encapsulation

[71] لا تقلق أبداً بشأن تحليل الأخطاء مرة أخرى أثناء قراءة ملف CSV باستخدام Pandas	Pandas Never Worry About Parsing Errors Again While Reading CSV with Pandas
[72] طريقة مثيرة للاهتمام وغير معروفة لإنشاء مخططات باستخدام Pandas	Pandas An Interesting and Lesser-Known Way To Create Plots Using Pandas
[73] لا يعرف معظم مبرمجي بايثون هذا عن حلقات-for	Most Python Programmers Don't Know This About Python For-loops
[74] كيفية تفعيل التحميل الزائد للدوال في بايثون	How To Enable Function Overloading In Python
[75] قم بإنشاء تلميحات مفيدة أثناء كتابة كود Pandas الخاص بك	Generate Helpful Hints As You Write Your Pandas Code
[76] طرق تسريع NumPy Methods 25x With Bottleneck	Speedup NumPy Methods 25x With Bottleneck
[77] تصور تحول البيانات لشبكة عصبية Network	Visualizing The Data Transformation of a Neural Network
[78] لا تقم أبداً بإعادة بناء كودك يدوياً مرة أخرى. بدلاً من ذلك، استخدم Sourcery!	Never Refactor Your Code Manually Again. Instead, Use Sourcery!
[79] ارسم البيانات التي تبحث عنها في ثوانٍ	Draw The Data You Are Looking For In Seconds
[80] نمط مخططات Matplotlib لجعلها أكثر جاذبية	Style Matplotlib Plots To Make Them More Attractive
[81] تسريع إدخال / إخراج باركيه من Pandas بمقدار خمس مرات	Speed-up Parquet I/O of Pandas by 5x
[82] أداة مفتوحة المصدر لتعزيز سير عمل Pandas الخاص بك	Open-Source Tools to Supercharge Your Pandas Workflow
[83] توقف عن استخدام طريقة الوصف في Pandas. بدلاً من ذلك، استخدم .Describe	Stop Using .Skimpy Pandas. Instead, use .The Describe Method in Pandas.
[84] الطريقة الصحيحة لطرح تحديثات المكتبة في بايثون	The Right Way to Roll Out Library Updates in Python
[85] أسطر بسيطة واحدة لمعاينة شجرة القرار باستخدام Sklearn	Simple One-Liners to Preview a Decision Tree Using Sklearn
[86] توقف عن استخدام طريقة الوصف في Pandas. بدلاً من ذلك، استخدم Summarytools	Stop Summarytools Pandas. Instead, use Summarytools
[87] لا تبحث أبداً في نوتبوك Jupyter يدوياً مرة أخرى للعثور على الكود الخاص بك	Never Search Jupyter Notebooks Manually Again To Find Your Code

F-strings Are Much More Versatile Than You Think	[88]
183.....	
Is This The Best Animated Guide لـ KMeans على الإطلاق؟	[89]
184.....?To KMeans Ever	
An Effective Yet Underrated Technique To Improve Model Performance	[89]
185.....	
Create Data Plots Right From The Terminal	[90]
187.....	
Make Your Matplotlib Plots More Professional	[91]
188.....	
37 مكتبة بايثون المخفية هي جواهر مطلقة	[92]
189.....Absolute Gems	
Preview Your README GitHub README الخاص بك محلياً بأسلوب	[93]
190.....File Locally In GitHub Style	
NumPy و Pandas بارجاع قيم مختلفة للانحراف المعياري. لماذا؟	[94]
191.....?Return Different Values for Standard Deviation. Why	
تصور تاريخ Commit مع الانيميشن الجميل	[95]
192.....Repo With Beautiful Animations	
Perfplot: Measure, Visualize and Compare Run-time With Ease	[96]
193.....	
يمكن أن توفر لك أداة واجهة المستخدم الرسومية هذه ساعات من العمل اليدوي	[97]
195.....Tool Can Possibly Save You Hours Of Manual Work	
كيف يمكنك تحديد التكرارات الضبابية في بيانات تحتوي على مليون سجل؟	[98]
197.....?Identify Fuzzy Duplicates In A Data With Million Records	
أوقف معاينة إطارات البيانات الخام. بدلاً من ذلك، استخدم .Raw DataFrames. Instead, Use DataTables	[99]
199.....	
A Single Line That Will Make Your Python Code Faster	[100]
200.....	
قم بتحميم سحابة الكلمات في بايثون	[101]
202.....	
كيفية ترميز الميزات الفئوية مع العديد من الأصناف؟	[102]
203.....?Features With Many Categories	
خريطة التقويم كبدائل أغنى للمخطط الخطى	[103]
204.....Line Plot	
10 أدوات آلية من تحليل البيانات الاستكشافية ستوفر عليك ساعات من العمل (الشاق)	[104]
205.....Automated EDA Tools That Will Save You Hours Of (Tedious) Work	

Why KMeans May Not Be KMeans خوارزمية التجميع المناسبة دائمًا	105
206.....The Apt Clustering Algorithm Always	
Converting Python To LaTeX Has LaTex إلى Python يمكن تحويله ببساطة	106
208..... Possibly Never Been So Simple	
(10) مخطط الكثافة كبديل أغنی لمخطط التشتت Density Plot As A Richer Alternative to Scatter Plot	107
209.....	
30 مكتبة بايثون (بشكل كبير) لتعزيز إنتاجية علم البيانات الخاصة بك Python Libraries	108
211..... to (Hugely) Boost Your Data Science Productivity	
109 سطر واحد Sklearn لتوسيع البيانات التركيبية Sklearn One-liner to Generate Synthetic Data	
212.....	
قم بتسمية بياناتك بنقرة زر Label Your Data With The Click Of A Button	110
214. Analyze A Pandas DataFrame Without Code بدون كود Pandas	111
112 سطر واحد بايثون لإنشاء مخططات مرسومة يدوياً Python One-Liner To Create Sketchy Hand-drawn Plots	
215.....	
70 مرة أسرع عن طريق تغيير سطر واحد فقط من التعليمات البرمجية Pandas x Faster	113
217..... Pandas By Changing Just One Line of Code	
114 دليل تفاعلي لإتقان Pandas دفعـة واحدة An Interactive Guide To Master Pandas In One Go	
219.....	
115 اجعل التدوين النقطي أكثر قوة في بايثون Make Dot Notation More Powerful in Python	
220.....	
116 أروع اختراق لـ Jupyter Notebook Hack Jupyter Notebook	
222.....	
117 قم بإنشاء مخطط الفقاعي متحرك في بايثون Create a Moving Bubbles Chart in Python	
223.....	
118 استخدم Scikit-Learn API على نماذج PyTorch Skorch: Use Scikit-learn API PyTorch	
224..... on PyTorch Models	
119 تقليل استخدام الذاكرة لطاربيانات Pandas بنسبة 90% Reduce Memory Usage Of A Pandas DataFrame By 90%	
226.....	
120 طريقة أنيقة لأداء مهام إيقاف التشغيل في بايثون An Elegant Way To Perform Shutdown Tasks in Python	
228.....	
121 تصوـر اتجاهـات بحـث Google Search لـعام 2022 باستخدـام بايثـون Trends of 2022 using Python	
230.....	
122 قـم بإنشـاء مـخطط شـريط السـباق في باـيثـون Create A Racing Bar Chart In Python	
232.....	
123 تسـريع Pandas Apply 5x مع NumPy Speed-up Pandas Apply 5x with NumPy	
233 مـرات مع Pandas Apply 5x	123

أداة على الإنترنت بدون كود لاستكشاف وفهم الشبكات العصبية	124
235..... To Explore and Understand Neural Networks	
ما هي طرق الكلاس ومتى يتم استخدامها؟	125
236..... ?Them	
اجعل Sklearn KMeans أسرع 20 مرة	126
237..... Make Sklearn KMeans 20x times faster	
تسريع NumPy 20x مع Numexpr	127
238..... Speed-up NumPy 20x with Numexpr	
ميزة أقل شهرة لتطبيق طريقة في Pandas	128
239..... In Pandas	
طريقة أنيقة لأداء ضرب المصفوفة	129
240..... An Elegant Way To Perform Matrix Multiplication	
قم بإنشاء إطار بيانات Pandas من Dataclass	129
241..... Create Pandas DataFrame from Dataclass Dataclass	
إخفاء السمات أثناء طباعة كائن من Dataclass Object	130
242..... Hide Attributes While Printing A Dataclass	
القائمة: الصنف :: المجموعة : List : Tuple :: Set?	131
الفرق بين Dot و Matmul في NumPy	132
244..... Difference Between Dot and Matmul in NumPy	
قم بتشغيل SQL في Jupyter To Pandas لتحليل إطار بيانات	133
246..... Analyze A Pandas DataFrame	
إعادة هيكلة الكود الآلي مع المصادر Sourcery	134
247... Automated Code Refactoring With Sourcery	
إضافة سمات إلى تهيئة عملية نشر كائن Post_init Dataclass	135
249..... To A Dataclass Object Post Initialization	
تبسيط دوالك بدوالك جزئية	136
250..... Simplify Your Functions With Partial Functions	
عندما لا يجب استخدام طريقة head() في Pandas	137
251..... Method In Pandas	
DotMap: بديل أفضل لقاموس بايثون	138
252..... Dictionary	
منع الاستيرادات السيئة مع all_ في بايثون	139
254..... Prevent Wild Imports With all_	
ثلاث نصائح أقل شهرة لقراءة ملف CSV باستخدام Pandas	140
256..... For Reading a CSV File Using Pandas	
أفضل تنسيق ملف لتخزين إطار بيانات Pandas	141
257..... DataFrame	

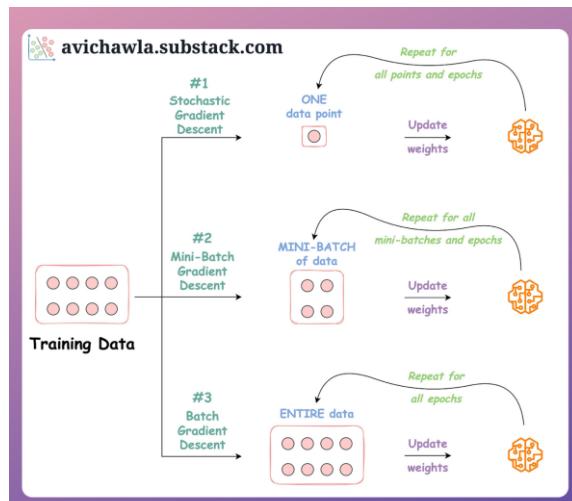
Debugging Made Easy With PySnooper	تصحيح الأخطاء أصبح سهلاً مع PySnooper	142
PySnooper258	
Lesser-Known Feature of the Merge Pandas	الميزة الأقل شهرة لطريقة الدمج في Pandas	143
Pandas259.....Method in Pandas	
The Best Way to Use Apply() in Pandas	أفضل طريقة لاستخدام Apply() في Pandas	144
Pandas260.....	
Deep Learning Network Debugging	أصبح تصحيح أخطاء شبكة التعلم العميق أمراً سهلاً	145
Made Easy261.....	
Don't Print NumPy	لا تطبع مصفوفات NumPy! استخدم Lovely-NumPy بدلاً من ذلك.	146
Arrays!263.....Use Lovely-NumPy Instead	
Performance Comparison of Python 3.11 and 3.10	مقارنة أداء بايثون 3.11 وبايثون 3.10	147
Python 3.10265.....	
View Documentation in Jupyter Notebook	عرض الوثائق في Jupyter Notebook	148
Jupyter Notebook266.....	
A No-code Tool To Understand Your Data Quickly	أداة بدون كود لفهم بياناتك بسرعة	149
.....267.....		
?Why 256 is 256 But 257 is not 257 ؟	لماذا 256 هو 256 ولكن 257 ليس 257	150
257.....		
Make a Class Object Behave Like a Function	اجعل كائن الكلاس يتصرف مثل الدالة	151
.....271.....		
Lesser-known feature of Pickle Files	ميزة أقل شهرة لملفات Pickle	152
Pickle273.....	
Dot Plot: A Potential Alternative to Bar Plot	المخطط النقطي: بديل محتمل للمخطط الشريطي Bar Plot	153
.....275.....		
Why Correlation (and Other) Statistics Can Be Misleading	لماذا يمكن أن يكون الارتباط (والإحصائيات الأخرى) مضللة.	154
.....277.....		
Supercharge value_counts()	عزز طريقة value_counts() في Pandas مع	155
Pandas279.....Method in Pandas With Sideload	
.....281.....		
Write Your Own Flavor Of Pandas	اكتب نكهة Pandas الخاصة بك	156
GitHub Copilot	: مساعد الترميز AI الذي يجب عليك استخدامه عبر	157
CodeSquire283.....CodeSquire: The AI Coding Assistant You Should Use Over GitHub Copilot	
Vectorization Does Not Always Guarantee Better	لا يضمن التوجيه دائمًا أداءً أفضل	158
.....285.....		
Performance		
In Defense of Match-case Statements in Python	دفاعاً عن حالات match-case بلغة بايثون	159
.....286.....		
Enrich Your Notebook With Interactive Controls	إثراء النوتبوک الخاص بك مع عناصر تحكم تفاعلية	160
.....288.....		

Get Notified When Jupyter Cell Has Executed Jupyter	161
290.....	
Data Analysis Using No-Code Jupyter	162
291..... Pandas In Jupyter	
293.. Using Dictionaries In Place of If-conditions If	163
Clear Cell Output In Jupyter	164
295..... Notebook During Run-time	
A Hidden Feature of Describe Method In Pandas Pandas	165
297.....	
Use Slotted Class To Improve Slotted Class لتحسين كود بايثون الخاص بك	166
299..... Your Python Code	
Stop Analysing Raw Tables.	167
301..... أوقف تحليل الجداول الخام، استخدم التصميم بدلاً من ذلك! !Use Styling Instead	
Explore CSV Data Right From The CSV مباشرة بيانات CSV من الترمinal	168
302..... Terminal	
304..... Generate Your Own Fake Data In Seconds	169
Import Your Python Package as a Module	170
306.....	
308..... Specify Loops and Runs In %timeit %timeit	171
Waterfall Charts: A Better Alternative to Line/Bar Plot	172
309.....	
Hexbin Plots As A Richer Alternative to Scatter Plots	173
311.....	
Importing Modules Made Easy with Pyforest	174
313.....	
Analyse Flow Data With Sankey Diagrams	175
315.....	
Feature Tracking Made Simple In Sklearn Transformers	176
317.....	
319.. Lesser-known Feature of f-strings in Python	177
Don't Use time.time() To Measure Execution Time	178
320.....	
Now You Can Use OpenAI API مع واجهة برمجة تطبيقات DALL-E	179
321..... DALL-E With OpenAI API	

1) مخطط الانحدار الخطي متعدد الحدود أصبح سهلاً مع Seaborn	180 323.....Regression Plot Made Easy With Seaborn
2) استرجع المخرجات المحسوبة مسبقاً في Jupyter Notebook	181 325.....Computed Output In Jupyter Notebook
3) موازاة "Pandas Apply()" With Swifter	182 326.....Swifter With Pandas Apply()
4) إنشاء إطار بيانات خالي من المشاكل باستخدام الحافظة	183 327.....By Using Clipboard
5) قم بتشغيل دليل مشروع بايثون كسكريبت	184 328.....Run Python Project Directory As A Script
6) فحص تدفق البرنامج باستخدام IceCream	185 329...Inspect Program Flow with IceCream
7) لا تقوم بإنشاء أعمدة شرطية في Pandas مع Apply	186 330.....Don't Create Conditional Columns Apply in Pandas with Apply
8) التخطيط الجميل مع Pandas	187 332.....Pretty Plotting With Pandas
9) قم ببناء نماذج أساسية بسهولة مع Sklearn	188 334.....Build Baseline Models Effortlessly With Sklearn
10) تتبع الأخطاء بدقة باستخدام Python 3.11	189 335.....Fine-grained Error Tracking With Python 3.11
11) اعثر على الكود الخاص بك مختبئاً في بعض النوتبوک من Jupyter	190 337.....Hiding In Some Jupyter Notebook With Ease
12) أعد تشغيل الكيرنل دون فقد المتغيرات	191 338.....Restart the Kernel Without Losing Variables
13) كيف تقرأ عدة ملفات CSV بكافأة	192 339.....How to Read Multiple CSV Files Efficiently
14) ارسم بأنقة حدود قرار المصنف	193 341.....Elegantly Plot the Decision Boundary of a Classifier
15) طريقة أنيقة لاستيراد المقاييس من Sklearn	194 342.....An Elegant Way to Import Metrics From Sklearn
16) تكوين Sklearn ل لإخراج إطار بيانات DataFrame	195 343.....Pandas Configure Sklearn To Output Pandas DataFrame
17) عرض شريط التقدم مع Apply() في Pandas	196 344.....Display Progress Bar With Apply() in Pandas
18) تعديل دالة أثناء وقت التنفيذ	197 345.....Modify a Function During Run-time
19) مخطط الانحدار أصبح سهلاً مع Plotly	198 346.....Regression Plot Made Easy with Plotly
20) الانحدار الخطي متعدد الحدود باستخدام NumPy	199 347.....Polynomial Linear Regression with NumPy

200) قم بتعديل نوع بيانات الأعمدة المتعددة مرة واحدة	Alter the Datatype of Multiple Columns at Once
349.....	
201) نوع البيانات لمعالجة أعمدة القيم المفقودة في Pandas	Datatype For Handling Missing Values in Pandas
350.....	
202) قم بموازنة Pandas مع Pandarallel	351..... Parallelize Pandas with Pandarallel
203) لماذا يجب خزن إطار البيانات في ملف CSV	Why you should not dump DataFrames to CSV
352.....	
204) حفظ الذاكرة مع مولدات بايثون	354..... Save Memory with Python Generators
205) لا تستخدم print() لتصحيح التعليمات البرمجية الخاصة بك.	Don't use print() to debug your code
355.....	
206) ابحث عن كود بايثون غير المستخدم بسهولة	356 Find Unused Python Code With Ease
207) عرف نوع البيانات الصحيح للأعمدة الفئوية	Define the Correct DataType for Categorical Columns
357.....	
208) نقل المتغيرات بين Notebooks	Transfer Variables Between Jupyter Notebooks
358.....	
209) لماذا يجب عليك قراءة ملفات CSV مع Pandas	Why You Should Not Read CSVs with Pandas
359.....	
209) تعديل كود بايثون أثناء وقت التنفيذ	360..... Modify Python Code During Run-Time
210) التعامل مع البيانات المفقودة مع Missingno	362. Handle Missing Data With Missingno

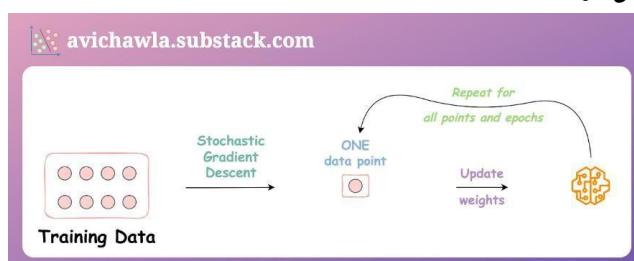
1) دليل مرئي للتدرج الاشتقافي العشوائي والدُفعات الصغيرة A Visual Guide to Stochastic, Mini-batch, and Batch Gradient Descent



❖ التدرج الاشتقافي **Gradient descent** هي خوارزمية تحسين مستخدمة على نطاق واسع لتدريب نماذج التعلم الآلي . machine learning models

يمثل التدرج الاشتقافي العشوائي Stochastic والدُفعات الصغيرة mini-batch والدُفعي batch ثلاثة أشكال مختلفة من التدرج الاشتقافي، وتمتاز بعدد نقاط البيانات المستخدمة لتحديث أوزان النموذج .iteration model weights

❖ التدرج الاشتقافي العشوائي **Stochastic gradient descent**: قم بتحديث أوزان الشبكة باستخدام نقطة بيانات واحدة في كل مرة.

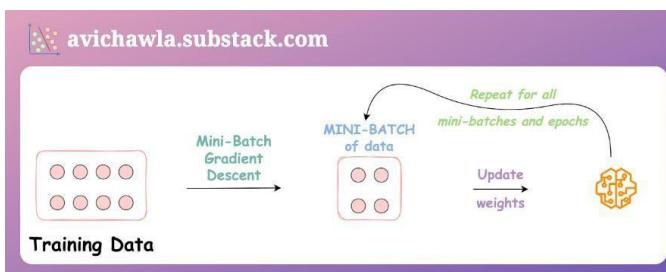


- ❖ المزايا:
- أسهل في الذاكرة.

- يمكن أن تقارب converge بشكل أسرع في مجموعات البيانات الكبيرة ويمكن أن تساعد في تجنب الحدود الدنيا المحلية local minima بسبب التذبذبات oscillations.

العيوب:

- الخطوات الصادمة يمكن أن تؤدي إلى تقارب أبطأ وتتطلب المزيد من الضبط للمعلمات hyperparameters الفائقة.
 - مكلف حسابياً بسبب التحديثات المتكررة.
 - يفقد ميزة العمليات الموجة vectorized operations.
- الدرج الاشتقaci ذو الدفعات الصغيرة Mini-batch gradient descent:** قم بتحديث أوزان الشبكة باستخدام بعض نقاط بيانات في كل مرة.



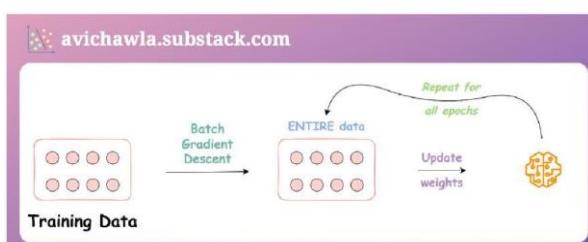
المزايا:

- أكثر كفاءة من الناحية الحسابية من الدرج الاشتقaci الدفعي بسبب فوائد التوجيه.
- تحديثات أقل ضوضاء من الدرج الاشتقaci العشوائي.

العيوب:

- يتطلب ضبط حجم الدفعه batch size.
- قد لا يتقارب مع الحد الأدنى العالمي global minimum إذا لم يتم ضبط حجم الدفعه جيداً.

- الدرج الاشتقaci ذو الدفعات Batch gradient descent:** قم بتحديث أوزان الشبكة باستخدام البيانات بالكامل مرة واحدة.



المزايا:

- اتخاذ خطوات أقل صادمة نحو الحدود الدنيا العالمية.

- يمكن الاستفادة من التوجيه .vectorization
- ينتج تقارباً أكثر استقراراً.

❖ العيوب:

- يفرض قيود الذاكرة لمجموعات البيانات الكبيرة.
 - بطيء حسابياً حيث يتم حساب العديد من التدرجات، ويتم تحديث جميع الأوزان مرة واحدة.
- دورك الآن: ما هي بعض المزايا / العيوب الأخرى التي يمكنك التفكير فيها؟ اسمحوا لي أن أعرف.

المقالة:

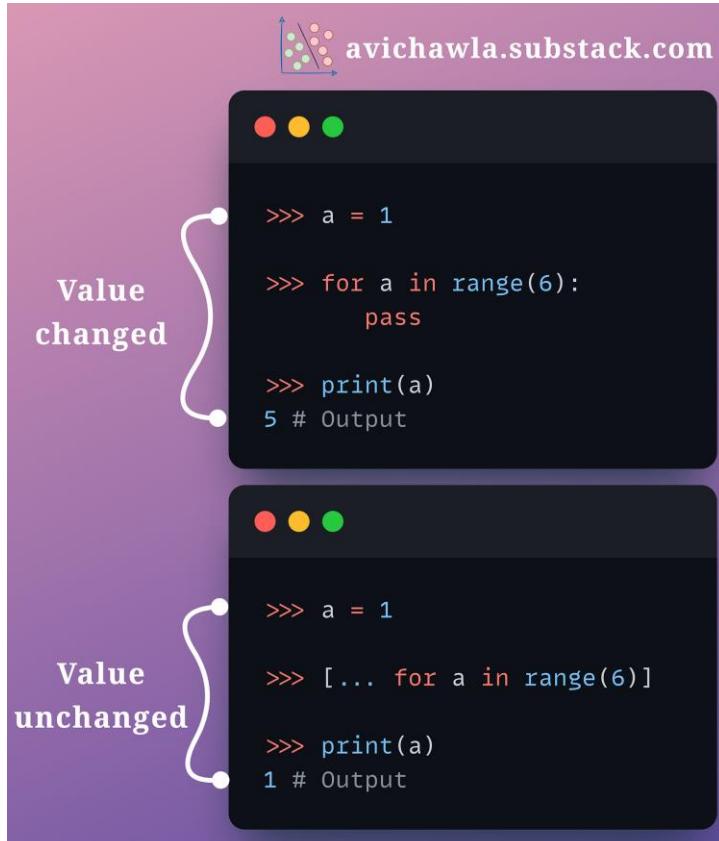
<https://avichawla.substack.com/p/a-visual-guide-to-stochastic-mini>

ال코드:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Machine%20Learning/Gradient-Descent-Visual-
Guide.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Gradient-Descent-Visual-Guide.ipynb)

2) الفرق الأقل شهرة بين حلقة for و List Comprehensions

A Lesser-Known Difference Between For-Loops and List Comprehensions



في الكود أعلاه، قامت حلقة for-loop بتحديث المتغير الحالي (a)، لكن قائمة الفهم list comprehension لم تقم بذلك. يمكنك تخمين لماذا؟ اقرأ المزيد لتعرف.

يتم التعامل مع متغير الحلقة بشكل مختلف في for-loops و list comprehensions .

تقوم حلقة for-loop بتسريب متغير الحلقة إلى النطاق المحلي. بمعنى آخر، بمجرد انتهاء الحلقة، لا يزال بإمكانك الوصول إلى متغير الحلقة.

يمكنا التحقق من ذلك أدناه:

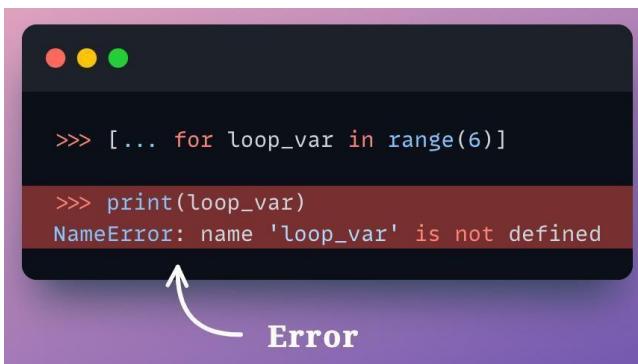


```
>>> for loop_var in range(6):
...     ...
>>> print(loop_var) ## Loop variable accessible
5
```

↑
No error

في مقتطف الكود الرئيسي أعلاه، نظرًا لوجود متغير الحلقة (a) بالفعل ، تمت الكتابة فوقه في كل تكرار. list comprehension لا تعمل بهذه الطريقة. بدلاً من ذلك، يظل متغير الحلقة دائمًا محليًا في comprehension. لا يتم تسريبه للخارج أبدًا.

يمكنا التتحقق من ذلك أدناه:



```
>>> [... for loop_var in range(6)]
>>> print(loop_var)
NameError: name 'loop_var' is not defined
```

↑
Error

هذا هو السبب في أن المتغير الحالي (a)، والذي تم استخدامه أيضًا داخل قائمة الفهم، ظل دون تغيير. حدد استيعاب القائمة متغير الحلقة (a) محليًا في نطاقه.

دوريك الآن: ما هي بعض الاختلافات الأخرى التي تعرفها بين الحلقات التكرارية وقائمة الفهم؟

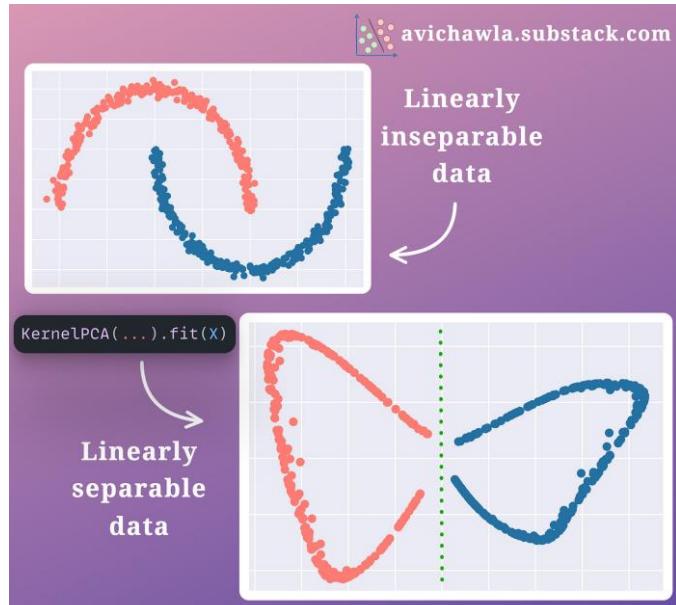
المقالة:

<https://avichawla.substack.com/p/a-lesser-known-difference-between>

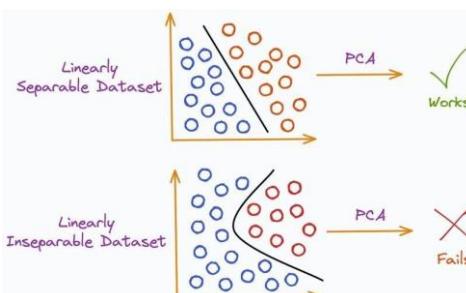
الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/for-loop-and-list-comprehension-diff.ipynb>

3) قيود PCA التي يتجاهلها الكثير في كثير من الأحيان Limitation of PCA Which Many Folks Often Ignore



تخيل أن لديك مجموعة بيانات تصنيف classification dataset. إذا كنت تستخدم PCA لتقليل الأبعاد، فمن المفترض بطبيعتها أن بياناتك قابلة للفصل خطياً linearly separable. ولكن قد لا يكون هذا هو الحال دائماً. وبالتالي، ستفشل PCA في مثل هذه الحالات.

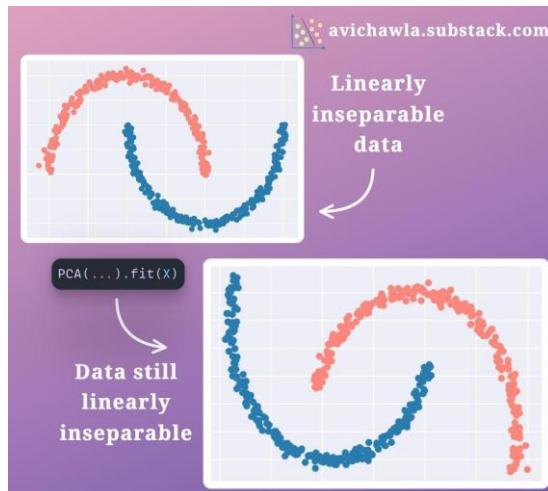


إذا كنت ترغب في قراءة كيفية عمل PCA، فإني أوصي بشدة بقراءة إحدى مشاركاتي السابقة: [دليل مرئي ومبسط للغاية لـ PCA](#).

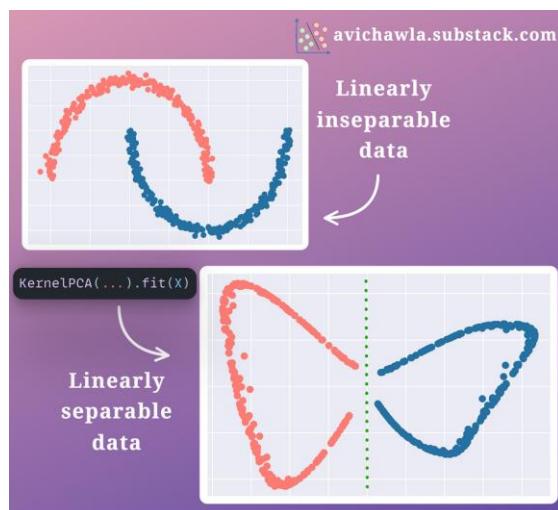
لحل هذه المشكلة، نستخدم kernel trick (أو KernelPCA). الفكرة هي: قم بإسقاط البيانات إلى مساحة أخرى باستخدام دالة kernel، حيث تصبح البيانات قابلة للفصل خطياً.

تطبيق خوارزمية PCA القياسية على البيانات المحولة.

على سبيل المثال، في الصورة أدناه، البيانات الأصلية لا يمكن فصلها خطياً. لا يؤدي استخدام PCA مباشرة إلى أي نتائج مرغوبة.



ولكن كما ذكر أعلاه، يحول KernelPCA أولاً البيانات إلى مساحة قابلة للفصل خطياً ثم يطبق PCA، مما ينتج عنه مجموعة بيانات قابلة للفصل خطياً.



يوفر Sklearn غلباً KernelPCA wrapper لـ `kernel`, يدعم العديد من دوال الشائعة الاستخدام. يمكنك الحصول على مزيد من التفاصيل هنا: [Sklearn Docs](#)

بعد قولـي هذا، تجدر الإشارة أيضـاً إلى أن وقت تشغيل PCA هو تكعيبـي cubic بالنسبة إلى عدد أبعـاد البيانات.

$$\text{Runtime : } O(nd^2 + d^3)$$

d : dimensions

n : samples

عندما نستخدم KernelPCA، عادةً، يتم إسقاط البيانات الأصلية (في أبعـاد n) إلى مساحة أبعـاد جديدة أعلى (بأبعـاد $m < n$). لذلك، فهو يزيد من وقت التشغيل الإجمالي overall run-time لـ PCA.

المقالة:

<https://avichawla.substack.com/p/the-limitation-of-pca-which-many>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Kernel-PCA-vs-PCA.ipynb>

4) طرق السحر: جوهرة قيمة من البرمجة كيانية التوجه Magic Methods: An Underrated Gem of Python OOP



avichawla.substack.com

Magic Method	Syntax	Usage/Description
<code>__new__</code>	<code>__new__(cls, *args, **kwargs):</code>	Invoked before <code>__init__</code> to allocate memory to object
<code>__init__</code>	<code>__init__(self, *args, **kwargs):</code>	Invoked after <code>__new__</code> to initialise the object
<code>__str__</code>	<code>__str__(self):</code>	Invoked when <code>str(obj)</code> or <code>print(obj)</code> is used
<code>__int__</code>	<code>__int__(self):</code>	Invoked when <code>int(obj)</code> is used
<code>__len__</code>	<code>__len__(self):</code>	Invoked when <code>len(obj)</code> is used
<code>__call__</code>	<code>__call__(self, *args, **kwargs):</code>	Invoked when class object is called as a function: <code>obj()</code>
<code>__getitem__</code>	<code>__getitem__(self, key):</code>	Invoked when object is indexed: <code>obj[key]</code>
<code>__setitem__</code>	<code>__setitem__(self, key, value):</code>	Invoked when object is indexed and value is set: <code>obj[key]=value</code>
<code>__delitem__</code>	<code>__delitem__(self, key):</code>	Invoked when object's index is deleted: <code>del obj[key]</code>
<code>__contains__</code>	<code>__contains__(self, item):</code>	Invoked when the <code>in</code> operator is used: <code>item in obj</code>
<code>__bool__</code>	<code>__bool__(self):</code>	Invoked when object is used in boolean context: <code>if obj</code> or <code>bool(obj)</code>
<code>__iter__</code>	<code>__iter__(self):</code>	Invoked when object is iterated: <code>for x in obj</code>
<code>__eq__</code>	<code>__eq__(self, other):</code>	Invoked when <code>==</code> operator is used to compare two objects: <code>obj1 == obj2</code>
<code>__ne__</code>	<code>__ne__(self, other):</code>	Invoked when <code>!=</code> operator is used to compare two objects: <code>obj1 != obj2</code>
<code>__add__</code>	<code>__add__(self, other):</code>	Invoked when two objects are added: <code>obj1 + obj2</code>
<code>__mul__</code>	<code>__mul__(self, other):</code>	Invoked when two objects are multiplied: <code>obj1 * obj2</code>
<code>__abs__</code>	<code>__abs__(self):</code>	Invoked to compute absolute value of object: <code>abs(obj)</code>
<code>__neg__</code>	<code>__neg__(self):</code>	Invoked when unary operator <code>-</code> is used on an object: <code>-obj</code>
<code>__invert__</code>	<code>__invert__(self):</code>	Invoked when <code>~(tilde)</code> operator is used to invert an object: <code>~obj</code>

الطرق السحرية (وتسمى أيضًا **dunder methods**) هي طرق خاصة تم تحديدها داخل تطبيق كلاس بایثون .Python class

في ملاحظة جانبية، كلمة "Dunder" هي اختصار لـ "Double Underscore" وهي مثبتة مسبقاً ومثبتة بشرط سفلية مزدوجة، مثل `__len__` و `__str__` وغيرها الكثير.

توفر الطرق السحرية مرونة هائلة في تحديد سلوك كائنات الكلاس في سيناريوهات معينة.

على سبيل المثال، لنفترض أننا نريد تحديد سلوك مخصص لإضافة كائنين من كلاسنا (`obj1 + obj2`).

من الأمور الواضحة والمباشرة القيام بذلك عن طريق تحديد طريقة `method`، على سبيل المثال `add_objects()`، وتمرير الكائنين كمعامل `argument` لها.



يبينما ستنجح الطريقة المذكورة أعلاه، فإن استدعاء طريقة ما بشكل صريح لإضافة كائنين ليس أنيقاً مثل استخدام عامل التشغيل `+`:



هذا هو المكان الذي تأتي فيه الطرق السحرية. في المثال أعلاه، سيسمح لك تنفيذ الطريقة السحرية `add` بإضافة الكائنين باستخدام عامل `+` بدلاً من ذلك.

وبالتالي، تسمح لنا الطرق السحرية بجعل كلاستنا أكثر سهولة وأسهل في التعامل معها.

نتيجة لذلك، يعد الوعي بها أمراً بالغ الأهمية لتطوير خطوط أنابيب أنيقة وبديهية.

تلخص الصورة المرئية حوالي 20 طريقة سحرية شائعة الاستخدام في بايثون.

دورك الآن: ما هي الأساليب السحرية الأخرى التي سترجها هنا؟ أي منها تستخدم أكثر؟ اسمحوا لي أن أعرف.

المقالة:

<https://avichawla.substack.com/p/magic-methods-an-underrated-gem-of>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/20-Magic-Methods.ipynb>

٥) تصنيف خوارزميات الانحدار التي لا يكلف الكثيرون أنفسهم عناء تذكرها

The Taxonomy Of Regression Algorithms That Many Don't Bother To Remember

تسمح لنا خوارزميات الانحدار Regression algorithms بمنزلة العلاقة بين متغير تابع independent variable ومتغير واحد أو أكثر من المتغيرات المستقلة dependent variable.



avichawla.substack.com

Regression Type	Description	Equation/Loss Function
Linear Regression	Simple Linear Regression	$\hat{y} = wx + b$ $Loss = \sum \frac{(y - \hat{y})^2}{n}$
	Polynomial Linear Regression	$\hat{y} = w_1x + w_2x^2 + \dots + b$ $Loss = \sum \frac{(y - \hat{y})^2}{n}$
	Multiple Linear Regression	$\hat{y} = w_1x_1 + w_2x_2 + \dots + b$ $Loss = \sum \frac{(y - \hat{y})^2}{n}$
Regularized Regression	Ridge Regression	$Loss = \sum \frac{(y - \hat{y})^2}{n} + \lambda \sum_{i=1}^n w_i^2$
	Lasso Regression	$Loss = \sum \frac{(y - \hat{y})^2}{n} + \lambda \sum_{i=1}^n w_i $
	Elastic Net	$Loss = \sum \frac{(y - \hat{y})^2}{n} + \lambda((1 - \alpha) * \sum_{L2} w_i^2 + \alpha * \sum_{L1} w_i)$
Categorical Probability	Logistic Regression	$P(X) = \frac{1}{1 + e^{-(w_1x_1 + w_2x_2 + \dots + b)}}$
	Multinomial Logistic Regression (or Softmax Regression)	$P(Y = k X) = \frac{e^{score_k}}{\sum_{j=1}^K e^{score_j}}$

بعد تقديم معلمات نموذج الانحدار، يمكننا الحصول على نظرة ثاقبة حول كيفية تأثير التغييرات في أحد المتغيرات على الآخر.

نظرًا لاستخدامها على نطاق واسع في علم البيانات data science، فإن الوعي بأشكالها المختلفة أمر بالغ الأهمية لنقل الخوارزمية التي تستخدمها بدقة.

فيما يلي ثمانية من أكثر خوارزميات الانحدار القياسية الموضحة في سطر واحد:

❖ الانحدار الخطى Linear Regression

- الانحدار الخطى البسيط Simple linear regression: متغير واحد مستقل (x) ومتغير تابع واحد (y).

- الانحدار الخطي متعدد الحدود: Polynomial Linear Regression ميزات كثيرة Polynomial features ومتغير واحد تابع (y).
 - الانحدار الخطي المتعدد: Multiple Linear Regression ميزات عشوائية Arbitrary features.
 - ❖ الانحدار المنتظم Regularized Regression.
 - انحدار الlassو Lasso Regression: الانحدار الخطي مع تنظيم L1.
 - انحدار ريدج Ridge Regression: الانحدار الخطي مع تنظيم L2.
 - الشبكة المرنة Elastic Net: الانحدار الخطي مع تنظيم كلا من L1 و L2.
 - ❖ توقع الاحتمالية الفئوية Categorical Probability Prediction.
 - الانحدار اللوجستي Logistic Regression: توقع احتمالية النتائج الثنائية.
 - الانحدار اللوجستي متعدد الحدود (أو Multinomial Logistic Regression) انحدار Softmax : توقع احتمالات فئوية متعددة.
- دورك الآن: ما هي خوارزميات الانحدار الأخرى التي ستدرجها هنا؟

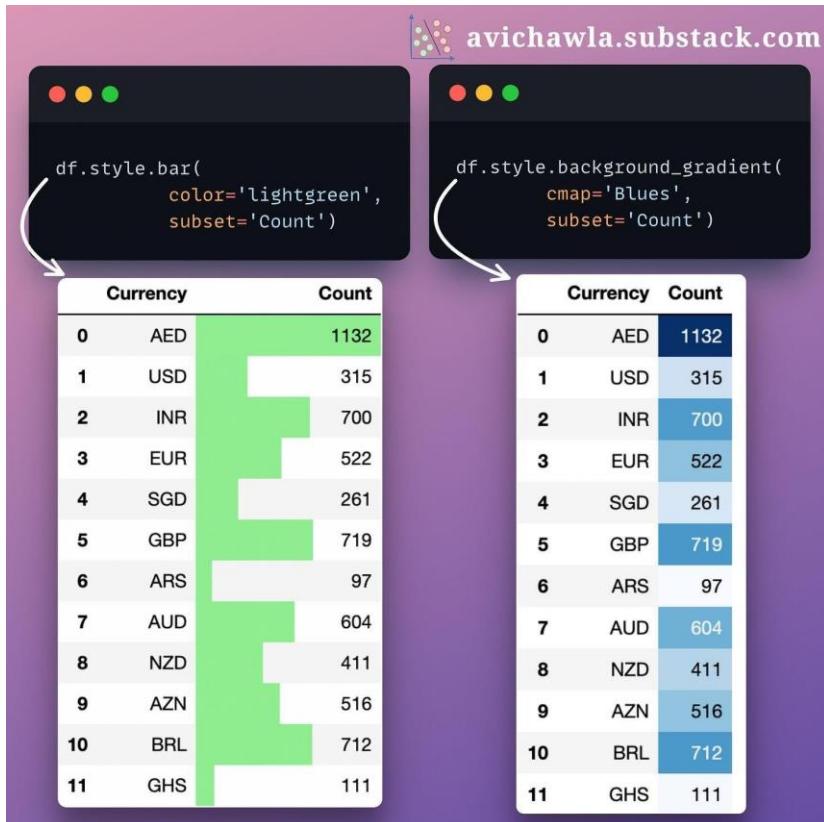
المقالة:

<https://avichawla.substack.com/p/the-taxonomy-of-regression-algorithms>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Regression-Algo-Taxonomy.ipynb>

6) نهج تم تجاهله بشدة لتحليل إطارات بيانات Pandas A Highly Overlooked Approach To Analysing Pandas DataFrames



بدلاً من معالجة إطارات البيانات DataFrame الأولية، يمكن أن يجعل التصميم تحليل البيانات أسهل وأسرع بكثير. إليك الطريقة.

Jupyter IDE هو قائم على الويب. يتم تقديم أي شيء تطبعه باستخدام HTML و CSS .
هذا يعني أنه يمكنك تصميم مخرجاتك بعدها طرق مختلفة.

لتصميم إطارات بيانات Pandas، استخدم Styling API (df.style). نتيجة لذلك، يتم تقديم DataFrame بتصميم محدد.

اقرأ المزيد هنا: [التوثيق](#).

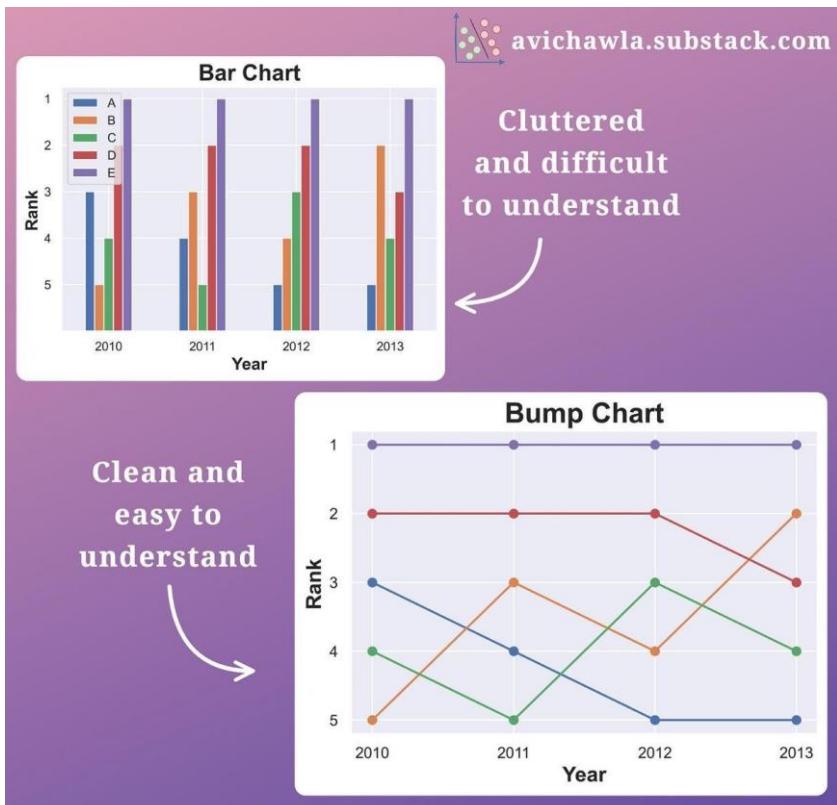
المقالة:

<https://avichawla.substack.com/p/a-highly-overlooked-approach-to-analysing>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Pandas/Style-DF.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Style-DF.ipynb)

7) تصور التغيير في الترتيب بمرور الوقت باستخدام مخططات Bump Visualise The Change In Rank Over Time With Bump Charts



عند تصور التغيير في الترتيب بمرور الوقت rank over time، قد لا يكون استخدام المخطط الشريطي bar chart مناسباً. بدلاً من ذلك، جرب bump charts.

يتم استخدامها بشكل محدد لتصور ترتيب العناصر المختلفة بمرور الوقت.

على عكس المخطط الشريطي الشائع الاستخدام، فهي واضحة وأنيقة وسهلة الفهم.

دورك الآن: ما هي بعض البدائل الأخرى للمخططات الشريطية التي يمكنك تجربتها في مثل هذه الحالات؟ اسمحوا لي أن أعرف.

ابحث عن الكود الخاص بإنشاء bump chart في بايثون هنا: [Notebook](#)

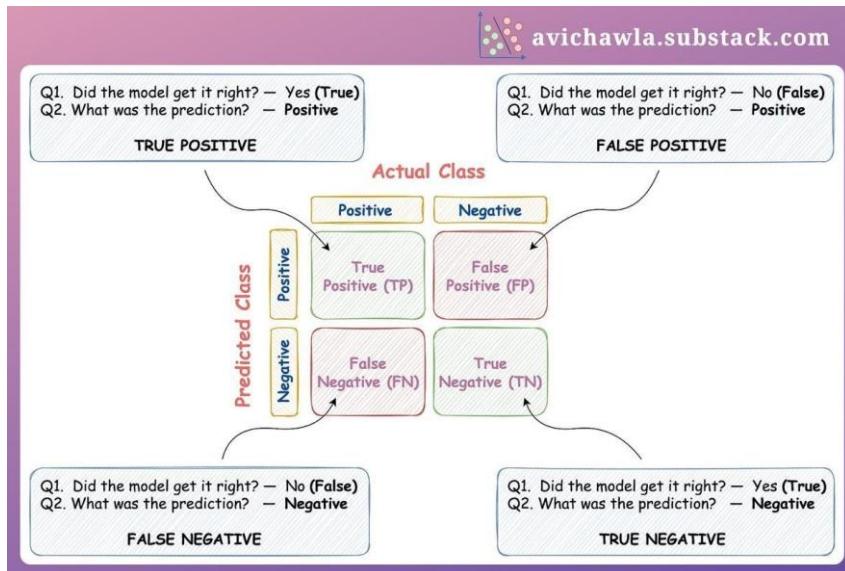
المقالة:

<https://avichawla.substack.com/p/a-simple-one-liner-to-create-professional>

المصدر:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/LovelyPlots-Professional-Matplotlib.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/LovelyPlots-Professional-Matplotlib.ipynb)

8) استخدم هذه التقنية البسيطة حتى لا تعاني أبداً مع TP و FN و FP و TN Use This Simple Technique To Never Struggle With TP, TN, FP and FN Again



هل غالباً ما تكافح لتسمية تنبؤات النموذج مثل TP و TN و FP و FN وفهمها؟ إذا كانت الإجابة بنعم، فإليك دليل بسيط لمساعدتك.

عند تصنيف أي توقع، اسأل نفسك سؤالين:

هل فهمها النموذج بشكل صحيح؟ الإجابة: نعم (أو صواب) / لا (أو خطأ).

ما هي الفئة المتوقعة predicted class؟ الجواب: إيجابي / سلبي.

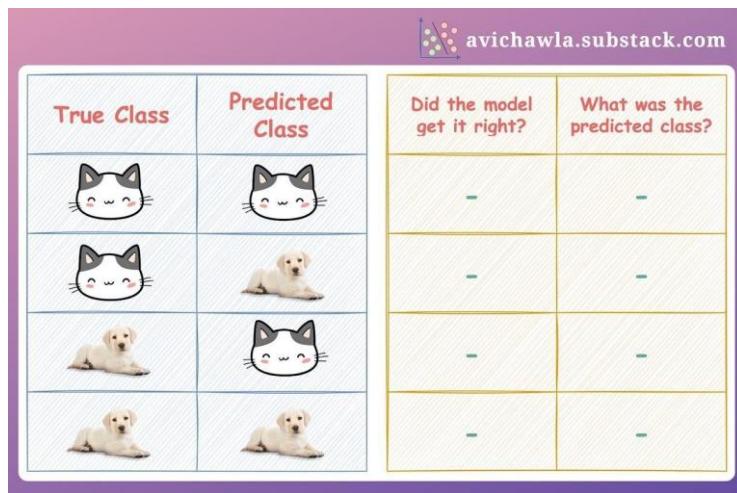
بعد ذلك، اجمع الإجابتين السابقتين للحصول على التسمية النهائية.

على سبيل المثال، لنفترض أن الفئة الفعلية والمتوتر كأن إيجابياً.

هل فهمها النموذج بشكل صحيح؟ الجواب نعم (أو صحيح).

ما هي الفئة المتوقعة؟ الجواب إيجابي. التسمية النهائية: **TRUE POSITIVE**.

كترين، حاول تسمية النتائج التالية. اعتبر فئة "Cat" على أنها فئة إيجابي و "الكلب" على أنها سلبية.



avichawla.substack.com

True Class	Predicted Class	Did the model get it right?	What was the predicted class?
		-	-
		-	-
		-	-
		-	-

9) المفهوم الخاطئ الأكثر شيوعاً حول العمليات الداخلية في الباندا The Most Common Misconception About Inplace Operations in Pandas



avichawla.substack.com

Method	Run-time	
	<i>inplace=False</i>	<i>inplace=True</i>
<code>df.replace()</code>	140 μs	244 μs (Slow)
<code>df.sort_values()</code>	374 μs	450 μs (Slow)
<code>df.reset_index()</code>	35 μs	10 μs (Fast)
<code>df.drop()</code>	200 μs	262 μs (Slow)
<code>df.fillna()</code>	90 μs	222 μs (Slow)
<code>df.dropna()</code>	750 μs	1088 μs (Slow)
<code>df.drop_duplicates()</code>	856 μs	1058 μs (Slow)
<code>df.rename()</code>	151 μs	152 μs (Equal)

غالباً ما يقوم مستخدمو Pandas بتعديل DataFrame في مكانهم `inplace` متوقعين أداءً أفضل. ومع ذلك، قد لا تكون فعالة دائماً. إليكم السبب.

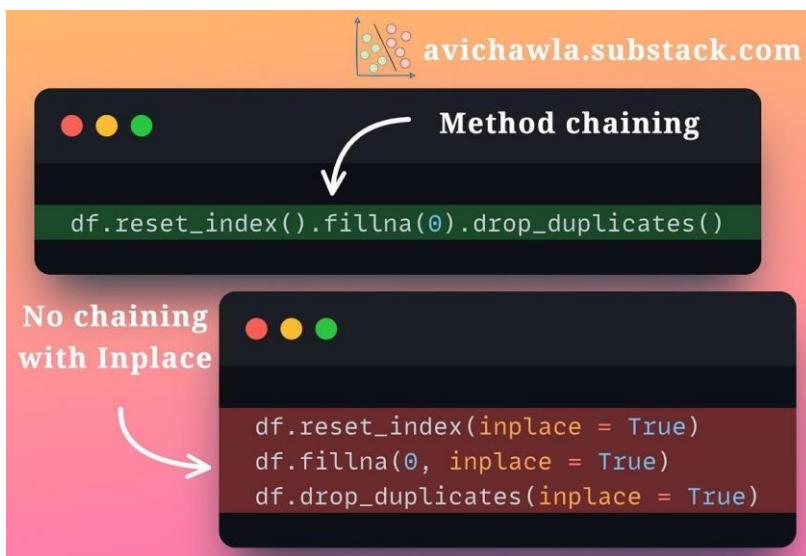
تقارن الصورة وقت التنفيذ للعمليات الموضعية `inplace` وغير الموضعية `non-in-place`. في معظم الحالات، تكون العمليات في الموقع بطيبة.

لماذا؟

خلافاً لل اعتقاد الشائع، فإن معظم العمليات الموضعية لا تمنع إنشاء نسخة جديدة. كل ما في الأمر أن inplace يعيد النسخة إلى نفس العنوان.

ولكن أثناء هذه المهمة، تقوم Pandas بإجراء بعض الفحوصات الإضافية (SettingWithCopy) للتأكد من تعديل DataFrame بشكل صحيح. قد تكون هذه، في بعض الأحيان، عملية مكلفة. ومع ذلك، بشكل عام، ليس هناك ما يضمن أن تكون العملية في الموقع أسرع.

علاوة على ذلك، لا تسمح العمليات الموضعية inplace بسلسل عمليات متعددة، مثل هذا:



المقالة:

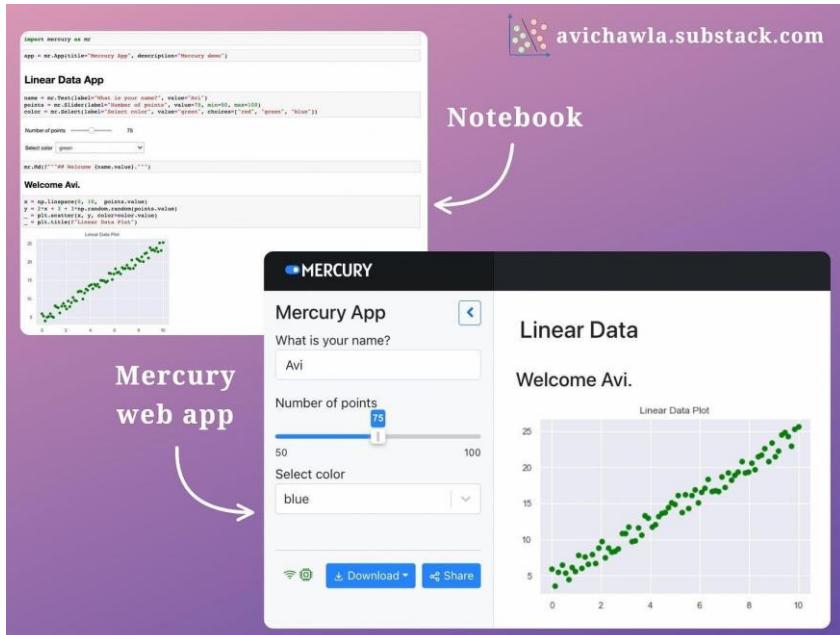
<https://avichawla.substack.com/p/the-most-common-misconception-about>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/inplace-noninplace-runtime.ipynb>

10) أنشئ تطبيقات ويب أنيقة مباشرة من Jupyter Build Elegant Web Apps Mercury Notebook Right From Jupyter Notebook with Mercury

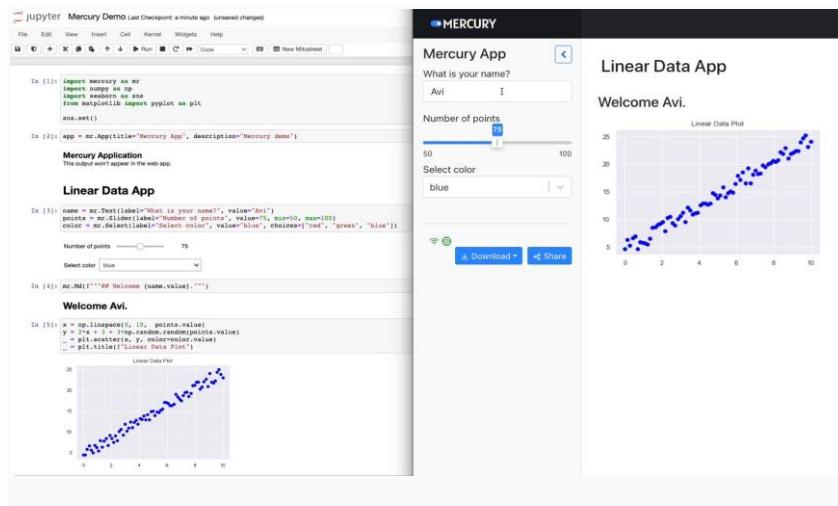
يعد استكشاف ومشاركة رؤى البيانات باستخدام Jupyter أمرًا شائعًا جدًا لأصحاب البيانات. ومع ذلك، يعد التطبيق التفاعلي أفضل لمن لا يهتمون بكتورك ويهمون بنتائجك.



في حين أن إنشاء العروض التقديمية أمر ممكن، إلا أنه قد يستغرق وقتاً طويلاً. أيضاً، على المرء أن يغادر Jupyter Notebook

بدلاً من ذلك، جرب Jupyter Notebook. إنها أداة مفتوحة المصدر تقوم بتحويل Jupyter Notebook الخاص بك إلى تطبيق ويب في أي وقت من الأوقات. وبالتالي، يمكنك إنشاء تطبيق الويب دون مغادرة Notebook.

يتم عرض عرض توضيحي سريع أدناه:



علاوة على ذلك، فإن جميع تحسينات Jupyter Notebook يتم الرجوع إليها فورًا في تطبيق Mercury على عكس streamlit المعتمد على نطاق واسع، يمكن أن تكون تطبيقات الويب التي تم إنشاؤها باستخدام Mercury.

- تم تصديره بتنسيق .PDF / HTML
- تم عرضه كعرض تقديمي مباشر.
- مؤمن بالمصادقة لتنقية الوصول.

المقالة:

<https://avichawla.substack.com/p/build-elegant-web-apps-right-from>

الكود:

<https://github.com/ChawlaAvi/Mercury-Web-App>

11) كن عالم بيانات ثنائي اللغة مع هذه الباندا لترجمات SQL Become A Bilingual Data Scientist With These Pandas to SQL Translations



avichawla.substack.com

Operation	Pandas	SQL
Read CSV	<code>pd.read_csv(file)</code>	<code>LOAD DATA INFILE 'data.csv' INTO TABLE table FIELDS TERMINATED BY '' LINES TERMINATED BY '\n' IGNORE 1 ROWS;</code>
Print first 10 (or k) rows	<code>df.head(10)</code>	<code>SELECT * FROM table LIMIT 10;</code>
Dimensions	<code>df.shape</code>	<code>SELECT count(*) FROM table;</code>
Datatype	<code>df.dtypes</code>	<code>DESCRIBE table;</code>
Filter Data	<code>df[df.column>10]</code>	<code>SELECT * FROM table where column>10;</code>
Select column(s)	<code>df.column</code>	<code>SELECT column FROM table;</code>
Sort	<code>df.sort_values("column")</code>	<code>SELECT * FROM table ORDER BY column;</code>
Fill NaN	<code>df.column.fillna(0)</code>	<code>UPDATE table SET column=0 WHERE column IS NULL;</code>
Join	<code>pd.merge(df1, df2, on = "col", how = "inner")</code>	<code>SELECT * FROM table1 JOIN table2 ON (table1.col = table2.col);</code>
Concatenate	<code>pd.concat((df1, df2))</code>	<code>SELECT * FROM table1 UNION ALL table2;</code>
Group	<code>df.groupby("column"). agg_col.mean()</code>	<code>SELECT column, avg(agg_col) FROM table GROUP BY column;</code>
Unique values	<code>df.column.unique()</code>	<code>SELECT DISTINCT column FROM table;</code>
Rename column	<code>df.rename(columns = {"old_name": "new_name"})</code>	<code>ALTER TABLE table RENAME COLUMN old_name TO new_name;</code>
Delete column	<code>df.drop(columns = ["column"])</code>	<code>ALTER TABLE table DROP COLUMN column;</code>

يعد كل من Pandas و SQL أدوات قوية لعلماء البيانات للعمل مع البيانات.

يمكن استخدام SQL و Pandas معاً لتنظيم مجموعات البيانات الكبيرة و تحويلها و تحليلها، وإنشاء خطوط أنابيب ونماذج معقدة.

وبالتالي، فإن الكفاءة في كلا الإطارين يمكن أن تكون ذات قيمة كبيرة لعلماء البيانات.

يصور هذا الشكل بعض العمليات الشائعة في Pandas والترجمات المقابلة لها في SQL.

لدي مدونة مفصلة عن ترجمة Pandas إلى SQL مع العديد من الأمثلة. اقرأها هنا:

[blog](#)

دورك الآن: ما هي ترجمات Pandas إلى SQL الأخرى التي سترجحها هنا؟

المقالة:

<https://avichawla.substack.com/p/become-a-bilingual-data-scientist>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandas-to-SQL.ipynb>

12) ميزة أقل شهرة من Sklearn لتدريب النماذج على مجموعات البيانات الكبيرة

A Lesser-Known Feature of Sklearn To Train Models on Large Datasets



```

from sklearn.linear_model import SGDClassifier
clf = SGDClassifier(...)

clf.fit(X, y)

```

Failed due to
memory
constraints

```

# 1. Load data in chunks (say, 1000 rows at once).
data = pd.read_csv("data.csv", chunksize = 1000)

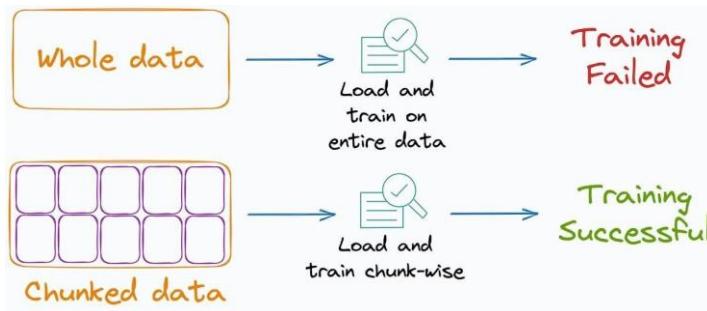
# 2. Train from mini-batches using partial_fit.
for batch in data:
    clf.partial_fit(batch["X"],
                    batch["y"],
                    classes = [0, 1, 2])

```

من الصعب تدريب النماذج باستخدام sklearn عندما يكون لديك الكثير من البيانات. قد يؤدي هذا غالباً إلى حدوث أخطاء في الذاكرة حيث يتم تحميل البيانات بالكامل في الذاكرة. ولكن إليك ما يمكن أن يساعد.

تطبق Sklearn واجهة برمجة التطبيقات (API) للعديد من الخوارزميات، والتي توفر التعلم الإضافي .incremental learning

كما يوحي الاسم، يمكن للنموذج التعلم بشكل تدريجي من مجموعة صغيرة من الأمثلة. هذا يمنع قيود الذاكرة المحدودة حيث يتم تحميل حالات قليلة فقط في الذاكرة مرة واحدة.



كما هو موضح في الصورة الرئيسية ، يأخذ `clf.partial_fit()` البيانات بالكامل ، وبالتالي ، قد يؤدي إلى حدوث أخطاء في الذاكرة . ولكن ، تحميل أجزاء من البيانات واستدعاء طريقة `clf.partial_fit()` يمنع ذلك ويوفر تدريجياً سلساً .

تذكر أيضًا أنه أثناء استخدام `partial_fit API` ، قد لا تحتوي الدفعات الصغيرة على مثيلات لجميع الفئات (خاصة الدفعات الصغيرة الأولى) . وبالتالي ، لن يتمكن النموذج من التعامل مع الفئات الجديدة / غير المرئية في الدفعات الصغيرة اللاحقة . لذلك ، يجب عليك تمرير قائمة بجميع الفئات الممكنة في معلمة `clf` .

بعد قولـي هذا ، تجدر الإشارة أيضـاً إلى أنه ليس كل مقدر `sklearn` يطبق `partial_fit API` . هـا هي القائـمة:

- Classification

- `sklearn.naive_bayes.MultinomialNB`
- `sklearn.naive_bayes.BernoulliNB`
- `sklearn.linear_model.Perceptron`
- `sklearn.linear_model.SGDClassifier`
- `sklearn.linear_model.PassiveAggressiveClassifier`

- Regression

- `sklearn.linear_model.SGDRegressor`
- `sklearn.linear_model.PassiveAggressiveRegressor`

- Clustering

- `sklearn.cluster.MiniBatchKMeans`

- Decomposition / feature Extraction

- `sklearn.decomposition.MiniBatchDictionaryLearning`
- `sklearn.cluster.MiniBatchKMeans`

وـعـ ذلك ، فـمن المؤـكـد أنـ الأـمـرـ يـسـتحقـ الاـسـتـكـشـافـ لـعـرـفـةـ ماـ إـذـاـ كـانـ يـمـكـنـكـ الاـسـتـفـادـةـ مـنـهـ .

المقالة:

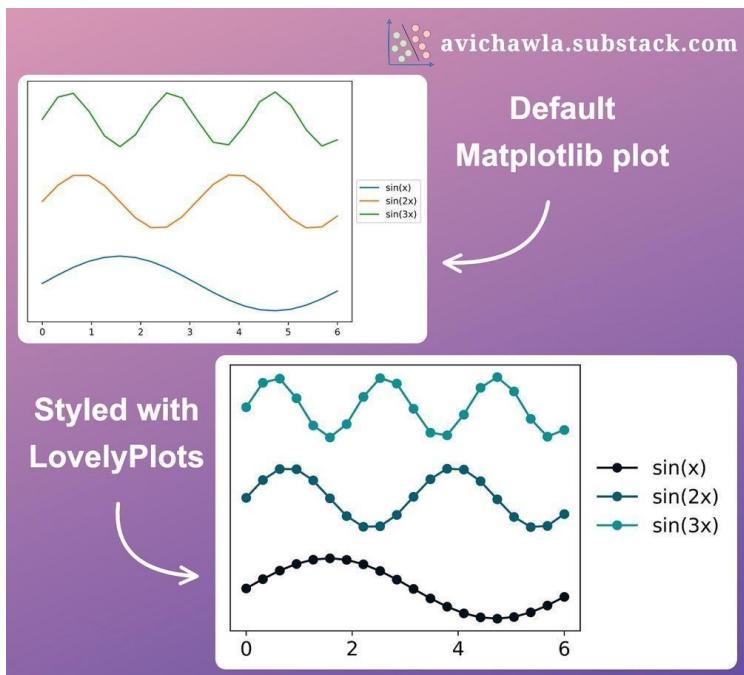
<https://avichawla.substack.com/p/a-lesser-known-feature-of-sklearn>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Sklearn/Sklearn-on-Large-Datasets.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Sklearn/Sklearn-on-Large-Datasets.ipynb)

13) سطر واحد بسيط لإنشاء مخططات ذات المظهر الاحترافي

A Simple One-Liner to Create Professional Looking Matplotlib Plots



يبدو التصميم الافتراضي لمخططات matplotlib بسيطاً جدًا في بعض الأحيان. إليك كيف يمكنك جعلها جذابة.

لإنشاء مخططات ذات مظهر احترافي للعرض التقديمية أو التقارير أو الأوراق العلمية، جرب LovelyPlots.

يوفر العديد من أوراق الأنماط لتحسين مظهرها الافتراضي، ببساطة عن طريق إضافة سطر واحد فقط من التعليمات البرمجية.

لتنزيل LovelyPlots، قم بتشغيل الأمر التالي:

```
pip install LovelyPlots
```

بعد ذلك، قم باستيراد مكتبة matplotlib، وقم بتغيير النمط على النحو التالي: (ليس عليك استيراد LovelyPlots في أي مكان)

```
import matplotlib.pyplot as plt  
plt.style.use(style) ## change to the style provided by  
LovelyPlots
```

اطبع قائمة بجميع الأنماط الممكنة على النحو التالي:

```
plt.style.available
```

البدء: [مستودع LovelyPlots](#)

المقالة:

<https://avichawla.substack.com/p/visualise-the-change-in-rank-over>

الكود:

<https://avichawla.substack.com/p/visualise-the-change-in-rank-over>

14) تجنب هذا الخطأ المكلف عند فهرسة إطار البيانات

This Costly Mistake When Indexing A DataFrame

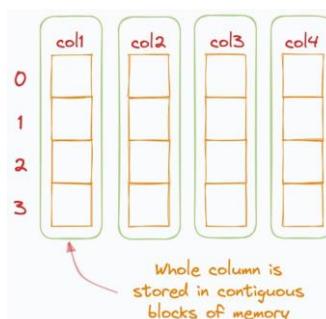
عند فهرسة إطار بيانات، يعد اختيار ما إذا كنت تريد تحديد عمود أول أو شريحة صف أول أمراً مهماً جدًا من منظور وقت التنفيذ.

The screenshot shows a Jupyter Notebook cell with the code `df.shape` which returns `(32768000, 9)`. Below it, two timing results are shown:

- First column then row:** `%timeit df["col"]["row"]` with a result of `2.96 µs ± 7.17 ns per loop`.
- First row then column:** `%timeit df.loc["row"]["col"]` with a result of `45.4 µs ± 384 ns per loop`.

A bracket on the right side groups the two timing results and points to the text: "Selecting a column first is over 15x faster".

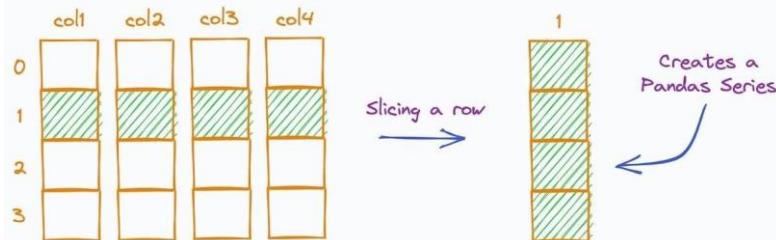
كما هو موضح أعلاه، فإن تحديد العمود الأول أسرع بـ 15 مرة من تقطيع slicing الصف الأول. لماذا؟ كما تحدثت من قبل، فإن Pandas DataFrame هو هيكل بيانات عمود رئيسي. وبالتالي، يتم تخزين العناصر المترتبة في عمود بجانب بعضها البعض في الذاكرة.



نظرًا لأن المعالجات فعالة مع كتل الذاكرة المجاورة، فإن الوصول إلى عمود يكون أسرع بكثير من الوصول إلى صف (اقرأ المزيد عن هذه في إحدى مشاركاتي السابقة [هنا](#)).

ولكن عندما تقوم بتقسيم صف أولاً، يتم استرداد كل صف من خلال الوصول إلى كتل غير متجاورة من الذاكرة، مما يجعله بطبيأً.

أيضاً، بمجرد تجميع جميع عناصر الصف، يحولها Pandas إلى سلسلة، وهي عبارة عن overhead آخر.



يمكننا التتحقق من هذا التحويل أدناه:

```

>>> df
   A   B
0  1  2
1  3  4

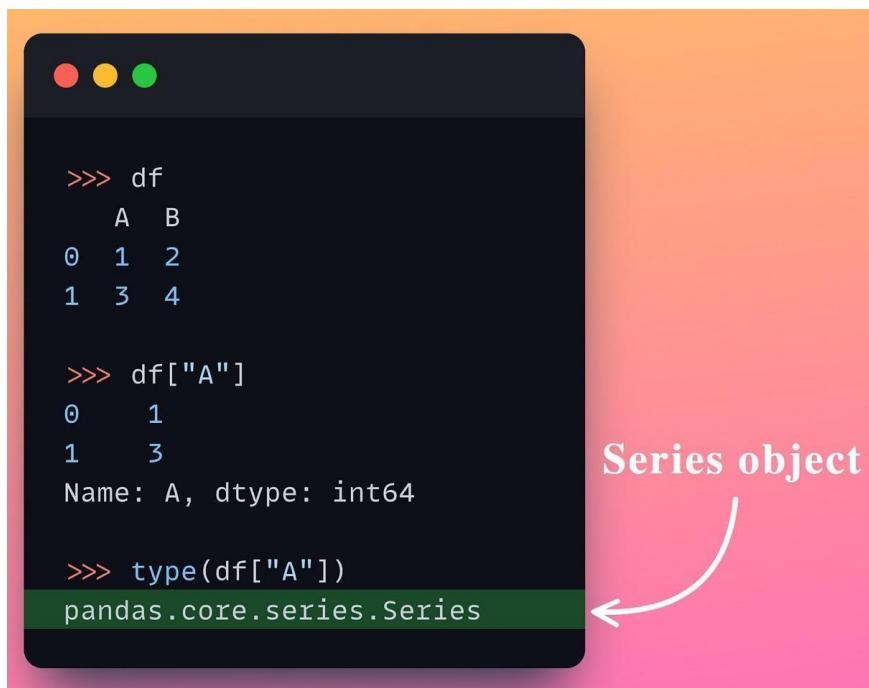
>>> df.loc[0]
A    1
B    2
Name: 0, dtype: int64

>>> type(df.loc[0])
pandas.core.series.Series

```

Series object

بدلاً من ذلك، عندما تحدد عموداً أولاً، يتم استرداد العناصر عن طريق الوصول إلى كتل متجاورة من الذاكرة، وهذا أسرع بكثير. أيضاً، العمود هو بطبيعته سلسلة Pandas. وبالتالي، لا توجد نفقات تحويل متضمنة كما هو مذكور أعلاه.



```

>>> df
   A   B
0  1  2
1  3  4

>>> df["A"]
0    1
1    3
Name: A, dtype: int64

>>> type(df["A"])
pandas.core.series.Series

```

Series object

بشكل عام، من خلال الوصول إلى العمود الأول، نتجنب الوصول إلى ذاكرة الوصول غير المجاورة، وهو ما يحدث عندما نصل إلى الصفر الأول.

هذا يجعل تحديد العمود أولاً أسرع من تقسيم الصفر الأول في عمليات الفهرسة.

إذا كنت محتاراً بشأن معنى التحديد selecting والفهرسة indexing والتقطيع slicing والترشيح filtering، فإنك يجب أن تقرأ ذلك:

<https://avichawla.substack.com/p/are-you-sure-you-are-using-the-correct>

المقالة:

<https://avichawla.substack.com/p/avoid-this-costly-mistake-when-indexing>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandas-Correct-Indexing-Order.ipynb>

١٥) ٩ إشارات سطر أوامر لتشغيل سكريپتات بايثون بشكل أكثر مرونة ٩ Command Line Flags To Run Python Scripts More Flexibly

Python Command Line Flags

Python Flag	Description	Usage
python -c	Run a single python command	python -c "print('Hello')"
python -i	Run interactive Python shell after running a script	python -i script.py
python -O	Ignore assert statements	python -O script.py
python -OO	Ignore assert statements and docstrings	python -OO script.py
python -W	Ignore warnings	python -W script.py
python -m	Run a module as a script	python -m my_package.my_module
python -v	Enable verbose mode. Prints more information about what interpreter is doing	python -v script.py
python -x	Ignore the first line of the script (often the shebang line)	python -x script.py
python -E	Ignore all Python Environment variables	python -E script.py



avichawla.substack.com

عند استدعاء سكريپت بايثون، يمكنك تحديد options/flags المختلفة. يتم استخدامها لتعديل سلوك مترجم بايثون عندما يقوم بتشغيل سكريپت أو وحدة نمطية.

فيما يلي ٩ من أكثر الخيارات شيوعاً:

- ❖ **- python -c**: قم بتشغيل أمر بايثون واحد. مفید لتشغيل سطر واحد بسيط أو اختبار مقتطفات التعليمات البرمجية.
 - ❖ **- i - python**: قم بتشغيل السكريبت كالمعتاد وادخل إلى الوضع التفاعلي بدلاً من إنتهاء البرنامج. مفید لتصحیح الأخطاء حيث يمكنك التفاعل مع الكائنات التي تم إنشاؤها أثناء البرنامج.
 - ❖ **- O - python**: تجاهل عبارات التأکید (هذه هي الأبجدية "O"). مفید لتحسين التعليمات البرمجية عن طريق إزالة کود التصحیح.
 - ❖ **- OO - python**: تجاهل عبارات التأکید وتتجاهل سلاسل المستندات. مفید لمزيد من تحسین الكود عن طريق إزالة سلاسل التوثيق.
 - ❖ **- W - python**: تجاهل كافة التحذیرات. يفیدی تحويل التحذیرات مؤقتاً والتركيز على التطوير.
 - ❖ **- m - python**: قم بتشغيل وحدة نمطية ك스크ريبت.
 - ❖ **- v - python**: ادخل إلى الوضع المطول verbose mode. مفید لطباعة معلومات إضافية أثناء تنفيذ البرنامج.
 - ❖ **- x - python**: تخطي السطر الأول. مفید لإزالة خطوط shebang أو التعليقات الأخرى في بداية السكريبت.
 - ❖ **- E - python**: تجاهل جميع متغيرات بيئه بايثون. مفید لضمان سلوك برنامج متسبق من خلال تجاهل متغيرات البيئة التي قد تؤثر على تنفيذ البرنامج.
- أی منها فاتني؟ اسمحوا لي أن أعرف.

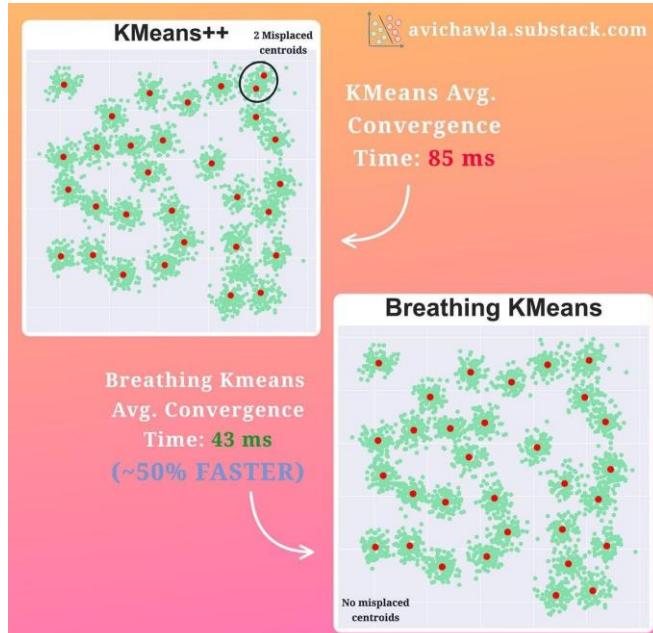
المقالة:

<https://avichawla.substack.com/p/9-command-line-flags-to-run-python>

الکود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Python-Flags.ipynb>

16) تنفس KMeans: أفضل وأسرع بديل ل Breathing KMeans: A Better and Faster Alternative to KMeans



أداء KMeans يعتمد كلياً على خطوة تهيئة النقطة الوسطى centroid initialization. وبالتالي، من المحتمل جدًا الحصول على مجموعات غير دقيقة.

على الرغم من أن Kmeans ++ يقدم تهيئة النقطة الوسطى، فإنه لا يضمن دائمًا تقاربًاvergence دقيقًا (اقرأ [كيف يعمل Kmeans ++ في مشاركتي السابقة](#)). هذا صحيح بشكل خاص عندما يكون عدد الكتل clusters مرتفعاً. هنا، قد يساعد تكرار الخوارزمية. لكنه يقدم overhead غير ضرورية في وقت التنفيذ.

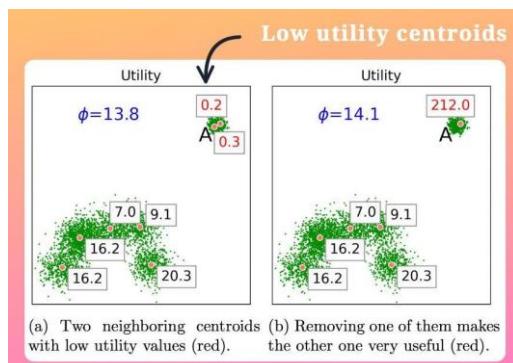
بدلاً من ذلك، يعد Breathing KMeans بديلاً أفضل هنا. وإليك كيف يعمل:

- ❖ **الخطوة 1:** تهيئة النقطة الوسطى k وتشغيل KMeans دون تكرار. بعبارة أخرى، لا تعيد تشغيله بتقنية مختلفة. فقط قم بتشغيله مرة واحدة.
- ❖ **الخطوة 2:** **Breathe in step :** أصف نقطة وسطى جديدة m جديدة وقم بتشغيل KMeans مع النقاط الوسطى $(k+m)$ دون تكرار.

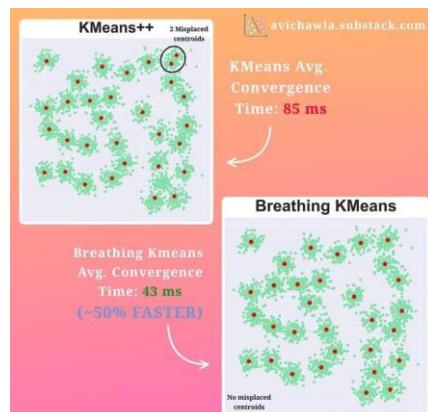
- ❖ الخطوة 3: **Breathe out step**: إزالة النقط الوسطى m من النقط الوسطى $(k+m)$ الموجودة.
- ❖ قم بتشغيل KMeans مع النقط الوسطى k المتبقية دون تكرار.
- ❖ الخطوة 4: إنقاص m بمقدار 1.
- ❖ الخطوة 5: كرر الخطوات من 2 إلى 4 حتى $m=0$.

يدخل **Breathe in** في خطوة النقط الوسطى الجديدة بالقرب من النقط الوسطى مع أكبر الأخطاء. خطأ النقطة الوسطى هو مجموع المسافة التربيعية للنقط تحت تلك النقطة الوسطى.

خطوة **Breathe out** تزيل النقطتين الوسطى مع فائدة منخفضة low utility. فائدة النقطة الوسطى تناسب مع بعده عن النقطة الوسطى الأخرى. الحدس هو أنه إذا كانت النقطتين الوسطى قريبة جدًا، فمن المحتمل أن تقع في نفس المجموعة. وبالتالي، سيتم تعين قيمة فائدة منخفضة لكليهما، كما هو موضح أدناه.



مع دورات التنفس المتكررة هذه، يوفر التنفس KMeans حلًا أسرع وأفضل من KMeans. في كل دورة، يتم إضافة النقط الوسطى الجديدة في موقع "جيدة"، وتم إزالة النقط الوسطى مع فائدة منخفضة. في الصورة أدناه، أنتج KMeans++ نقطتين في غير محلهما.



ومع ذلك، فقد جمعت تقنية Breathing KMeans البيانات بدقة، مع تحسن بنسبة 50% في وقت التشغيل.

يمكنك استخدام Breathing KMeans عن طريق تثبيت مكتبة مفتوحة المصدر، bkmeans، على النحو التالي:

```
pip install bkmeans
```

بعد ذلك، قم باستيراد المكتبة وتشغيل خوارزمية التجميع:



```
import numpy as np
from bkmeans import BKMeans

# generate random data set
X=np.random.rand(1000,2)

# create BKMeans instance
bkm = BKMeans(n_clusters=100)

# run the algorithm
bkm.fit(X)
```

في الواقع، ترث فئة BKMeans من فئة KMeans من sklearn. لذلك يمكنك تحديد معلمات أخرى واستخدام أي من الطرق الأخرى على كائن BKMeans حسب الحاجة.

المقالة:

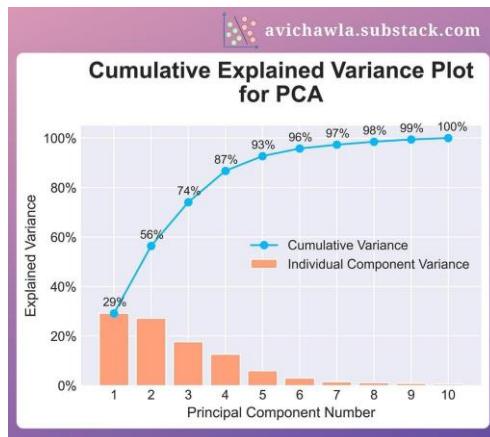
<https://avichawla.substack.com/p/breathing-kmeans-a-better-and-faster>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Breathing-KMeans.ipynb>

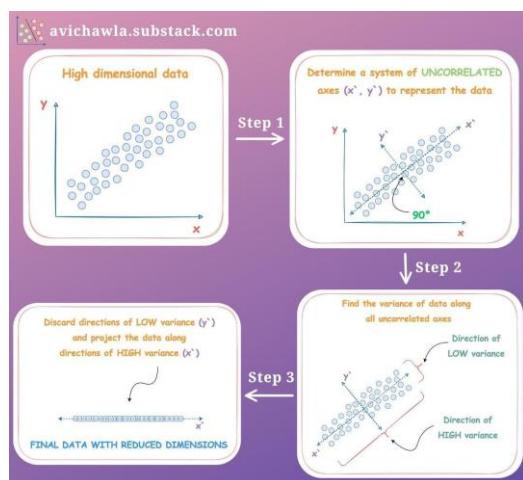
17) كم عدد الأبعاد التي يجب عليك تقليل بياناتك إليها عند استخدام PCA ؟ How Many Dimensions Should You Reduce Your Data To When Using PCA

عند استخدام PCA، قد يكون من الصعب تحديد عدد المكونات components التي يجب الاحتفاظ بها. ومع ذلك، إليك رسم plot يمكن أن تساعد بشكل كبير.



ملاحظة: إذا كنت لا تعرف كيفية عمل PCA، فلا تتردد في قراءة المنشور المفصل الخاص بي: [دليل مرئي لـ PCA](#).

مع ذلك، إليك تحدث سريع خطوة بخطوة. لا تتردد في تخطي هذا الجزء إذا كنت تتذكر منشور PCA الخاص بي.



الخطوة 1. خذ مجموعة بيانات عالية الأبعاد (y, x) في الصورة أعلاه) وقم بتمثيلها بمحاور غير مرتبطة (x, y) في الشكل أعلاه). لماذا غير مرتبط x, y ؟

هذا للتأكد من أن البيانات ليس لها ارتباط صفرى zero correlation على طول أبعادها وأن كل بُعد جديد يمثل تباينه الفردي.

على سبيل المثال، نظراً لارتباط البيانات الممثلة على طول (y, x ، فإن التباين على طول x يتأثر بانتشار البيانات على طول y .

بدلاً من ذلك، إذا قمنا بتمثيل البيانات على طول (y, x ، فإن التباين على طول x لا يتأثر بانتشار البيانات على طول y .

يتم تحديد الفضاء أعلىه باستخدام المتجهات الذاتية .eigenvectors

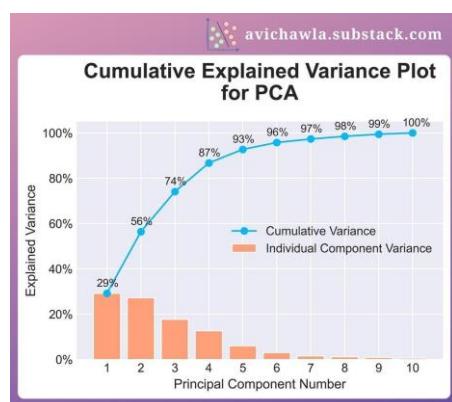
الخطوة 2. أوجد التباين على طول جميع المحاور غير المرتبطة (y, x). تشير القيمة الذاتية eigenvector إلى التباين .eigenvalue

الخطوة 3. تجاهل المحاور ذات التباين المنخفض. كم عدد الأبعاد التي يجب تجاهلها (أو الاحتفاظ بها) هي معلمة فائقة hyperparameter، والتي ستناقشها أدناه. اعرض البيانات على طول المحاور المحتجزة .retained axes

عند تقليل الأبعاد، يكون الغرض هو الاحتفاظ بما يكفي من التباين في البيانات الأصلية.

نظراً لأن كل مكون رئيسي principal component يشرح قدرًا من التباين، فإن التخطيط التراكمي للتباين من حيث المكونات يمكن أن يساعد في تحديد المكونات التي لها أكبر قدر من التباين.

وهذا ما يسمى مخطط التباين الموضح التراكمي .cumulative explained variance



على سبيل المثال، لنفترض أننا نعتزم الاحتفاظ بحوالي 85٪ من تباين البيانات. يصور المخطط أعلاه بوضوح أن اختزال البيانات إلى أربع مكونات سيؤدي إلى ذلك.

أيضاً، كما هو متوقع، تمثل المكونات العشرة معًا تبايناً بنسبة 100٪ في البيانات.

إنشاء هذا المخطط أمر بسيط للغاية في بايثون. ابحث عن الكود هنا: [PCA-CEV Plot](#).

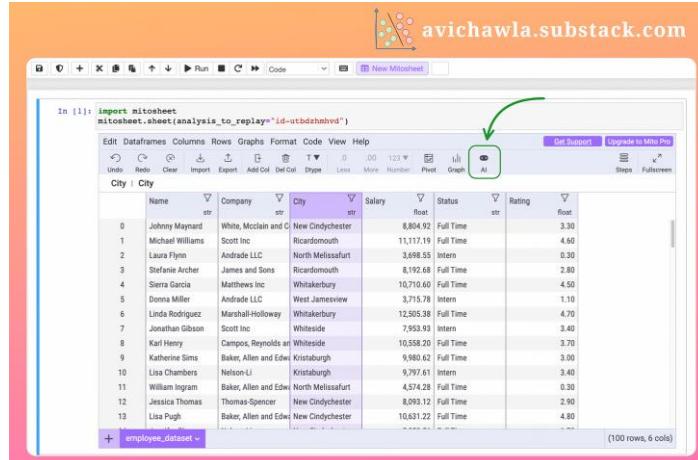
المقالة:

<https://avichawla.substack.com/p/how-many-dimensions-should-you-reduce>

ال코드:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Machine%20Learning/PCA-Dimensions-
Hyperparameter.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/PCA-Dimensions-Hyperparameter.ipynb)

18) تم تعزيز ميتو للتو مع الذكاء الاصطناعي! Got Supercharged With AI!



أنا شخصياً من أشد المعجبين بأدوات تحليل البيانات بدون كود. إنها مفيدة للغاية في التخلص من التعليمات البرمجية المتكررة عبر المشاريع - وبالتالي زيادة الإنتاجية productivity.

ومع ذلك، غالباً ما تكون معظم أدوات عدم وجود تعليمات برمجية محدودة من حيث الوظائف التي تدعمها. وبالتالي، فإن المرونة عادة ما تمثل تحدياً كبيراً أثناء استخدامها.

Mito هي أداة مفتوحة المصدر مذهلة تسمح لك بتحليل بياناتك داخل واجهة جدول بيانات في Jupyter دون كتابة أي كود.

علاوة على ذلك، قامت Mito مؤخراً بشحن واجهة جداول البيانات الخاصة بها باستخدام الذكاء الاصطناعي. نتيجة لذلك، يمكنك الآن تحليل البيانات الموجودة في نوتبوك باستخدام اوامر نصية text.prompts.

أحد أروع الأشياء حول استخدام Mito هو أن كل تعديل في جدول البيانات يولد تلقائياً كود بايثون المكافئ. هذا يجعل من الملائم إعادة إنتاج التحليل لاحقاً.



يمكنك تثبيت Mito باستخدام Pip كما يلي:

```
python -m pip install mitosheet
```

بعد ذلك، لتنشيطه في Jupyter، قم بتشغيل الأمرين التاليين:

```
python -m jupyter nbextension install --py --user  
mitosheetpython -m jupyter nbextension enable --  
py --user mitosheet
```

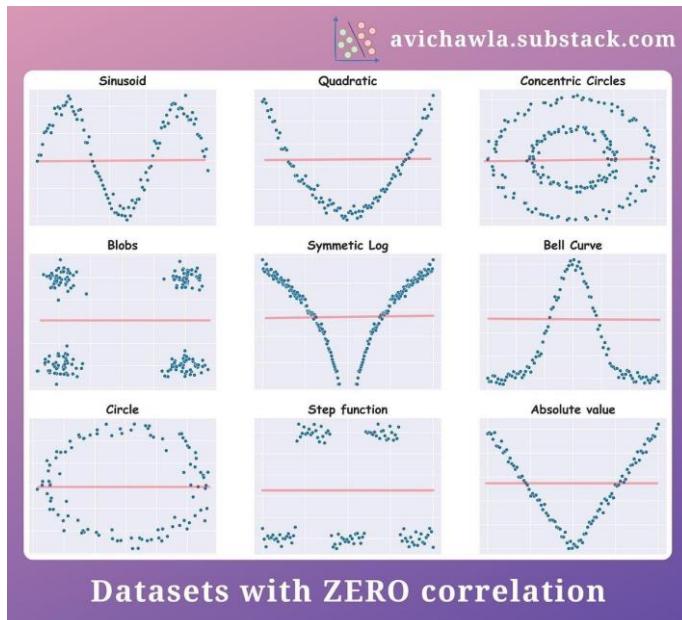
المقالة:

<https://avichawla.substack.com/p/mito-just-got-supercharged-with-ai>

ال코드:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Cool%20Tools/Text-to-Code-Jupyter.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/Text-to-Code-Jupyter.ipynb)

19) كن حذراً قبل رسم أي استنتاجات باستخدام الإحصائيات Be Cautious Before Drawing Any Conclusions Using Summary Statistics



أثناء تحليل البيانات، قد يميل المرء إلى استخلاص استنتاجات بناءً على إحصائياته فقط. ومع ذلك، قد تنقل البيانات الفعلية قصة مختلفة تماماً.

إليك صورة مرئية تصور تسعةمجموعات بيانات تقريباً. ارتباط صفرى zero correlation بين المتغيرين. لكن إحصاء الملخص (ارتباط بيرسون Pearson correlation في هذه الحالة) لا يعطي فكرة عما يوجد داخل البيانات.

علاوة على ذلك، يمكن أن تكون إحصاءات البيانات مدفوعة بشدة بالقيم المتطرفة outliers. لقد غطيت هذافي منشور سابق [هنا](#).

وبالتالي، لا يمكن التأكيد على أهمية النظر إلى البيانات بما فيه الكفاية. إنه يحميك من استخلاص استنتاجات خاطئة، والتي كان من الممكن أن تتوصل إليها بخلاف ذلك بالنظر إلى الإحصائيات وحدها.

على سبيل المثال، في مجموعة البيانات الجيبية sinusoidal dataset أعلاه، قد يجعلك ارتباط بيرسون تعتقد أنه لا يوجد ارتباط بين المتغيرين. ومع ذلك، تذكر أنه يحدد فقط مدى العلاقة الخطية بينهما. اقرأ المزيد حول هذافي واحدة أخرى من مشاركاتي السابقة [هنا](#).

وبالتالي، إذا كان هناك أي علاقة غير خطية أخرى (تربيعية quadratic، جيبية sinusoid، أسيّة exponential، إلخ)، فسوف تفشل في قياس ذلك.

المقالة:

<https://avichawla.substack.com/p/be-cautious-before-drawing-any-conclusions>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Statistics/Zero-Correlation-Datasets.ipynb>

(20) استخدام كائنات بايثون المخصصة في سياق منطقي Use Custom Python Objects In A Boolean Context

```

avichawla.substack.com

without_bool.py
class Cart:
    def __init__(self):
        self.items = []

    # No __bool__ method

my_cart = Cart()

if my_cart:
    print("Cart Not Empty")
else:
    print("Cart Empty")

"Cart Not Empty" # Output

Object of
custom class
evaluated to
True by default

with_bool.py
class Cart:
    def __init__(self):
        self.items = []

    def __bool__(self):
        return len(self.items)>0

my_cart = Cart()

if my_cart:
    print("Cart Not Empty")
else:
    print("Cart Empty")

"Cart Empty" # Output

Object
evaluated
to False

```

في السياق المنطقي boolean context، تقيم بايثون دائمًا كائنات objects مخصصة إلى True. لكن هذا قد لا يكون مرغوبًا في جميع الحالات. إليك كيفية تجاوز هذا السلوك.

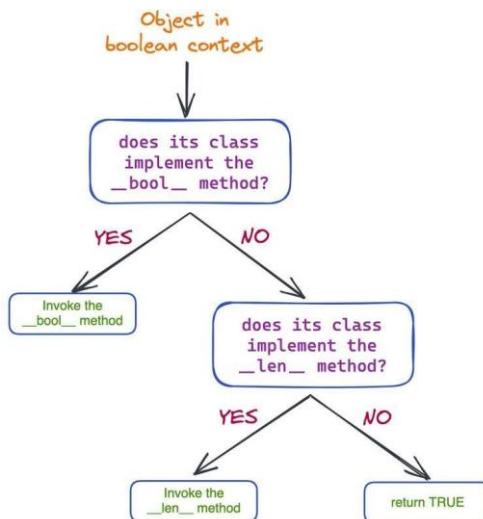
تُستخدم طريقة `__bool__` لتحديد سلوك كائن عند استخدامه في سياق منطقي. نتيجة لذلك، يمكنك تحديد شروط واضحة لتحديد مصداقية الكائن.

يتيح لك هذا استخدام كائنات الفتة بطريقة أكثر مرونة وبديهية.

كما هو موضح أعلاه، بدون طريقة `__bool__` (`without_bool.py`)، يتم تقييم الكائن إلى True. لكن تنفيذ طريقة `__bool__` يتيح لنا تجاوز هذا السلوك الافتراضي (`with_bool.py`).

بعض التفاصيل المفيدة الإضافية

عندما نستخدم أي كائن (سواء تم إنشاء مثيل له من فئة مخصصة أو مدمجة in-built) في سياق منطقي، فإليك ما تفعله بايثون:



أولاً، تتحقق بايثون من طريقة `bool` في تطبيق فئتها. إذا وجدت، يتم استدعاؤه. إذا لم يكن الأمر كذلك، فستتحقق بايثون من طريقة `len`. إن وجد، تم استدعاء `len`. وإلا، فإن بايثون ترجع `True`.

يوضح هذا السلوك الافتراضي للكائنات التي تم إنشاء مثيل لها من فئة مخصصة. نظراً لأن فئة `Cart` لم تنفذ الطريقة `bool` ولا الطريقة ، فقد تم تقييم كائن `Cart` إلى `True`.

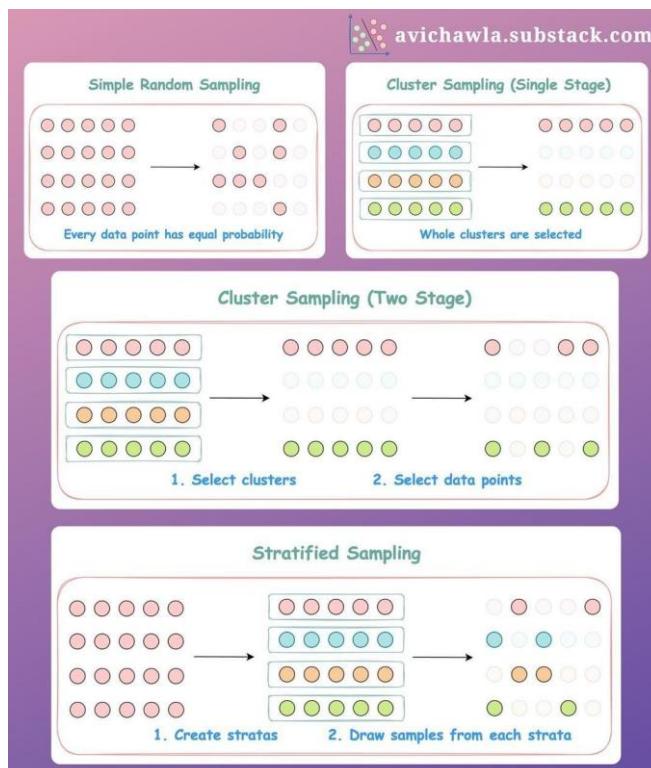
المقالة:

<https://avichawla.substack.com/p/use-custom-python-objects-in-a-boolean>

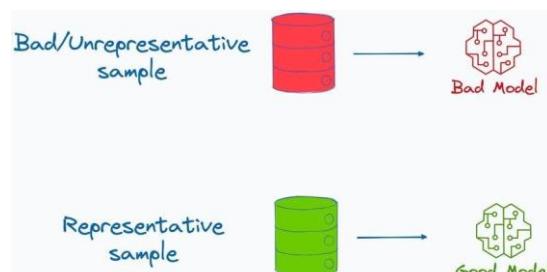
الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Class-Object-Boolean-Context.ipynb>

21) دليل مرئي لتقنيات أخذ العينات في التعلم الآلي Guide To Sampling Techniques in Machine Learning



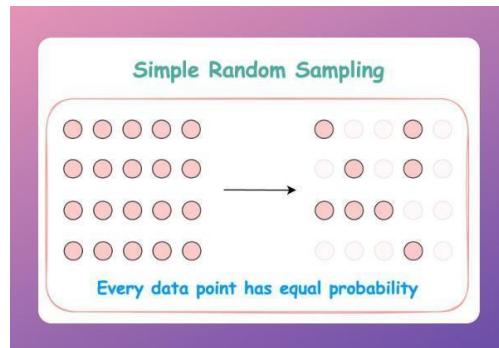
عندما تتعامل مع كميات كبيرة من البيانات، يُفضل غالباً سحب عينة أصغر نسبياً وتدريب نموذج. لكن أي خطأ يمكن أن تؤثر سلباً على دقة نموذجك.



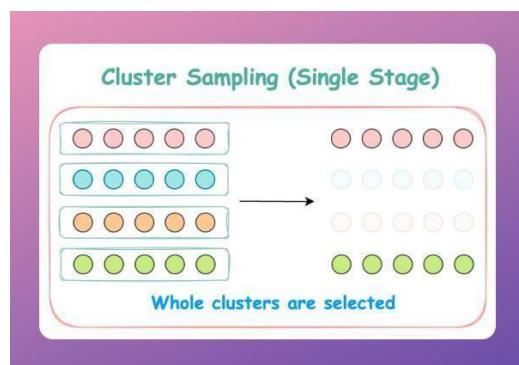
هذا يجعل أخذ العينات جانباً حاسماً في تدريب نماذج التعلم الآلي.

فيما يلي بعض التقنيات الشائعة الاستخدام التي يجب على المرء أن يعرفها:

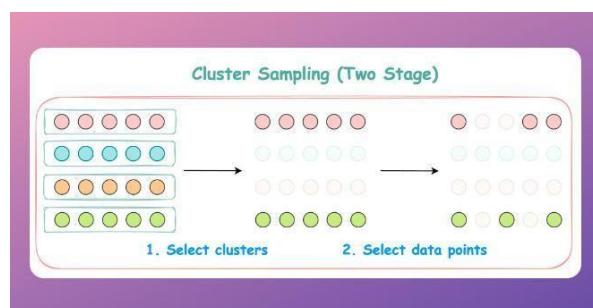
- أخذ العينات العشوائية البسيطة Simple random sampling: لكل نقطة بيانات احتمال متساوٍ لاختيارها في العينة.



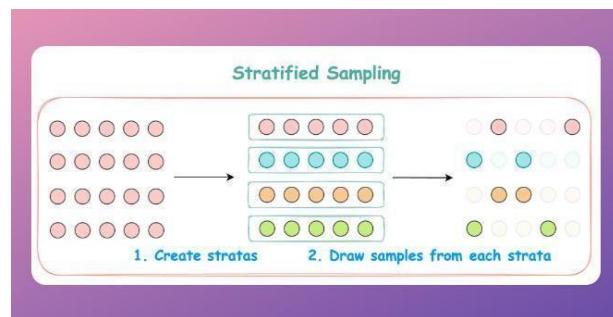
- أخذ العينات العنقودية (مرحلة واحدة) Cluster sampling (single-stage): قسم البيانات إلى مجموعات (عناقيد clusters) وحدد بعض مجموعات كاملة.



- أخذ العينات العنقودية (مراحلتان) Cluster sampling (two-stage): قسم البيانات إلى مجموعات، وحدد بعض مجموعات، واختر نقاطاً منها بشكل عشوائي.



- أخذ العينات الطبقية **Stratified sampling**: قسم نقاط البيانات إلى مجموعات متجانسة (بناءً على العمر والجنس وما إلى ذلك)، وحدد النقاط بشكل عشوائي.



ما هي بعض تقنيات أخذ العينات الأخرى التي تلجم إليها عادة؟

المقالة:

<https://avichawla.substack.com/p/a-visual-guide-to-sampling-techniques>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Sampling-Techniques.ipynb>

(22) ربما تم إعطاؤك معلومات غير كاملة حول ثبات الصف You Were Probably Given Incomplete Info About A Tuple's Immutability



عندما نقول إن الصف tuple غير قابلة للتغيير immutable، يعتقد العديد من مبرمجي بايثون أن القيم الموجودة داخل tuple لا يمكن أن تتغير. ولكن هذا ليس صحيحا.

يقتصر ثبات الصف immutability of a tuple فقط على هوية الأشياء التي تمتلكها، وليس قيمتها.

عبارة أخرى، لنفترض أن المجموعة تحتوي على كائنين لهما المعرفان 1 و 2. يقول الثبات immutability إن مجموعة المعرفات المشار إليها بواسطة المجموعة (وترتيبها) لا يمكن أن تتغير أبداً.

ومع ذلك، لا توجد قيود بحيث لا يمكن تعديل الكائنات الفردية ذات المعرفات 1 و 2.

وبالتالي، إذا كانت العناصر الموجودة داخل المجموعة كائنات قابلة للتغيير، فيمكنك بالفعل تعديليها.

وطالما بقيت مجموعة المعرفات كما هي، فلن يتم انتهاء ثبات المجموعة.

هذا يفسر الشرح أعلاه. نظراً لأن الإلحاد append عملية داخلية، فإن مجموعة المعرفات لم تتغير. وهكذا، فإن بايثون لم ترفع أي خطأ.

يمكننا أيضًا التتحقق من ذلك عن طريق طباعة مجموعة معرفات الكائنات المشار إليها داخل المجموعة قبل وبعد عملية الإلحادق:



The screenshot shows a Jupyter Notebook cell with the following Python code:

```
>>> my_tuple = (1, [2, 3])
>>> id(my_tuple[0]), id(my_tuple[1])
(583145, 434810)
>>> my_tuple[1].append(4)
>>> id(my_tuple[0]), id(my_tuple[1])
(583145, 434810)
```

A red curly brace on the left side of the code block is labeled "Same IDs", pointing to the two calls to `id()`. This indicates that the list at index 1 of the tuple did not change its memory address, thus proving that tuples are immutable.

كما هو موضح أعلاه، فإن المعرفات السابقة واللاحقة للإلحادق هي نفسها. وبالتالي، لا يتم انتهاء الثبات.

المقالة:

<https://avichawla.substack.com/p/you-were-probably-given-incomplete>

الكود:

<https://avichawla.substack.com/p/you-were-probably-given-incomplete>

(23) خدعة بسيطة تعمل على تحسين جودة مخططات A Simple Trick That Makes Matplotlib Plots Significantly Improve The Quality of Matplotlib Plots



غالباً ما تظهر مخططات Matplotlib باهتة dull وضبابية blurry، خاصة عند القياس أو التكبير. ومع ذلك، إليك حيلة بسيطة لتحسين جودتها بشكل ملحوظ.

يتم تقديم مخططات Matplotlib كصورة بشكل افتراضي. وبالتالي، فإن أي قياس / تكبير يشوه جودتها بشكل كبير.

بدلاً من ذلك، قم دائمًا بتقديم المخطط كرسم متوجه قابل للتطوير (SVG) .scalable vector graphic (SVG) كما يوحى الاسم، يمكن تحرجيمها دون المساس بجودة المخطط.

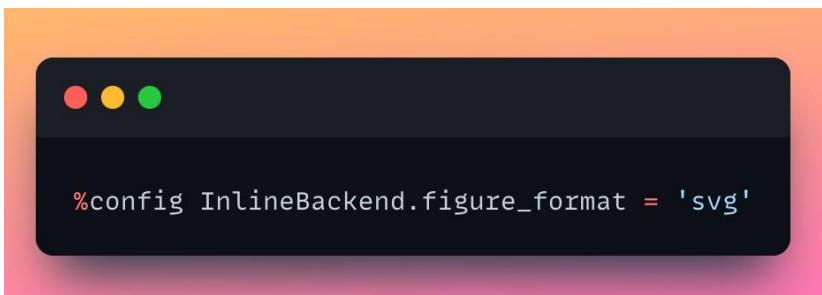
كما هو موضح في الصورة أعلاه، فإن المخطط التي يتم عرضها على أنها SVG تتفوق بوضوح وتكون أكثر وضوحاً بشكل ملحوظ من المخطط الافتراضي.

يتتيح لك الكود التالي تغيير تنسيق العرض إلى SVG. إذا لم يكن الاختلاف واضحًا في الصورة أعلاه، فإينني أوصي بتجربته بنفسك وملاحظة الفرق.



```
from matplotlib_inline.backend_inline import set_matplotlib_formats
set_matplotlib_formats('svg')
```

بدلاً من ذلك، يمكنك أيضًا استخدام الكود التالي:



```
%config InlineBackend.figure_format = 'svg'
```

ملاحظة. إذا كانت هناك فرصة لأنك لا تعرف ما الذي يتم تصويره في مخطط الشريط أعلاه، فراجع مقطع فيديو YouTube [هذا](#) بواسطة [Numberphile](#).

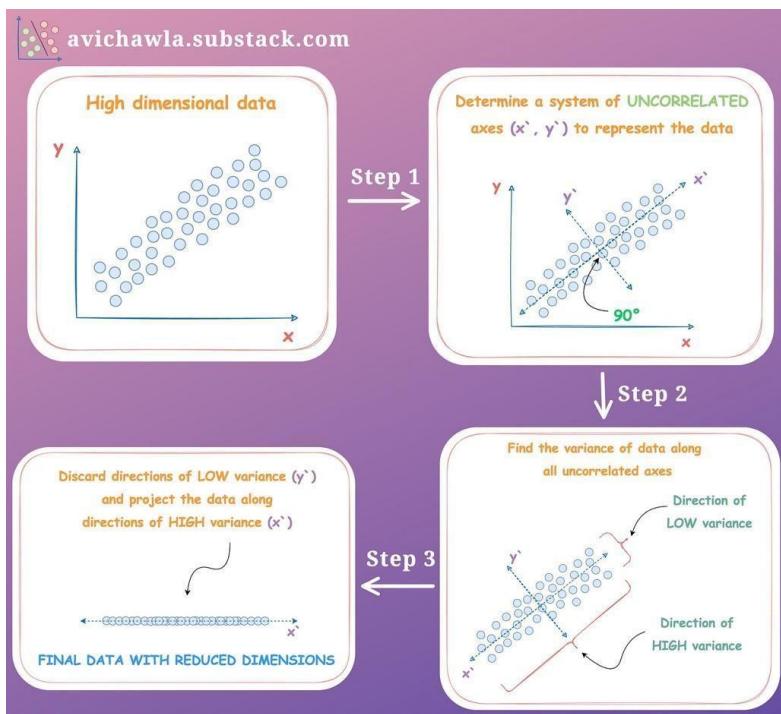
المقالة:

<https://avichawla.substack.com/p/a-simple-trick-that-significantly>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/SVG-Matplotlib-Plots.ipynb>

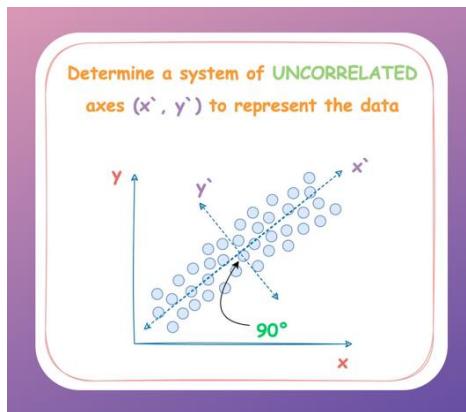
24) دليل مرئي ومبسط للغاية لـ PCA Simplified Guide to PCA



غالباً ما يكافح العديد من الأشخاص لفهم الجوهر الأساسي لتحليل المكون الرئيسي principal component analysis (PCA)، والذي يستخدم على نطاق واسع لتقليل الأبعاد dimensionality reduction. إليك دليل مرئي مبسط يصور ما يجري تحت الغطاء.

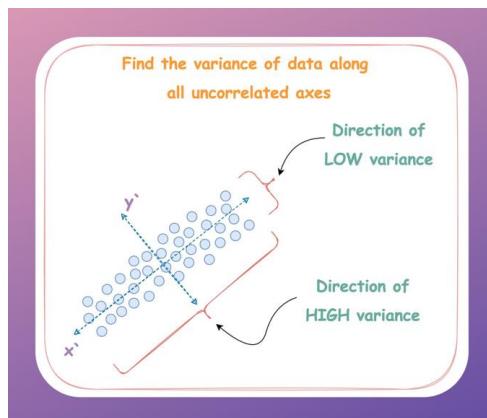
في الخلاصة، مع تقليل الأبعاد، يكون الهدف هو الاحتفاظ بأكبر قدر ممكн من التباين variation في البيانات.

بادئ ذي بدء، نظراً لأن البيانات قد تحتوي على ميزات مرتبطة correlated features، فإن الخطوة الأولى هي تحديد نظام إحداثيات جديد بمحاور متعامدة orthogonal axes. هذه مساحة حيث جميع الأبعاد غير مرتبطة uncorrelated.



يتم تحديد المساحة أعلاه باستخدام المتجهات الذاتية eigenvectors للبيانات.

بعد ذلك، نجد تباين بياناتنا على طول هذه المحاور غير المتراطة. يتم تمثيل التباين من خلال القيم الذاتية eigenvalues.

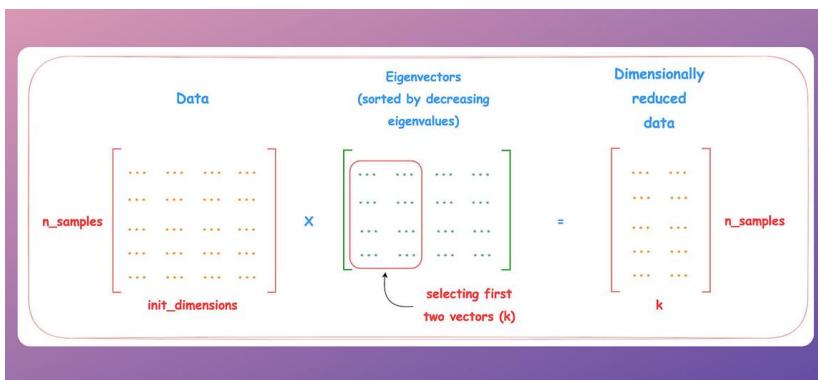


بعد ذلك، نقرر عدد الأبعاد التي نريد أن تحتوي عليها بياناتنا

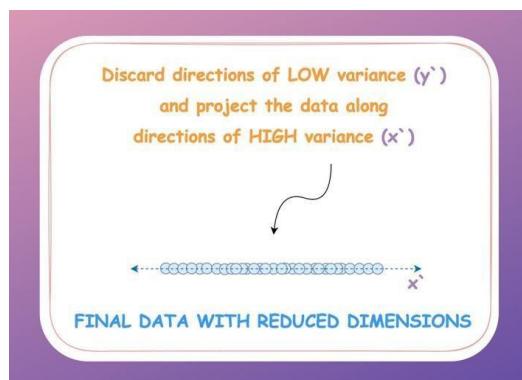
بعد التخفيض post-reduction (معلمة فائقة hyperparameter)، قل اثنين. نظرًا لأن هدفنا هو الاحتفاظ بأكبر قدر ممكن من التباين، فإننا نختار اثنين من المتجهات الذاتية ذات أعلى قيم ذاتية.

قد تسأل لماذا أعلى؟ كما ذكر أعلاه، يتم تمثيل التباين على طول المتجه الذاتي من خلال قيمته الذاتية. وبالتالي، فإن اختيار أعلى قيمتين من القيم الذاتية يضمن لنا الاحتفاظ بأقصى قدر من التباين في البيانات الإجمالية.

أخيرًا، يتم تحويل البيانات باستخدام ضرب مصفوفة matrix multiplication بسيط بأعلى متجهين، كما هو موضح أدناه:



بعد تقليل أبعاد مجموعة البيانات ثنائية الأبعاد 2D dataset المستخدمة أعلاه، نحصل على ما يلي.



هذه هي الطريقة التي يعمل بها PCA. آمل ألا تبدو هذه الخوارزمية مرعبة مرة أخرى.

المقالة:

<https://avichawla.substack.com/p/a-visual-and-overly-simplified-guide>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/PCA-Guide.ipynb>

25) عزز نواة Jupyter الخاصة بك مع ipyflow

هذا اختراق رائع لجوبيرت Jupyter تعلمته مؤخرًا.

أثناء استخدام Jupyter، يجب أن تكون قد لاحظت أنه عند تحديث متغير، يجب إعادة تنفيذ جميع الخلايا التابعة له يدوياً.

أيضاً، في بعض الأحيان، أليس من الصعب تحديد التسلسل الدقيق لعمليات تنفيذ الخلايا التي أنتجت مخرجات؟

هذا أمر ممل ويمكن أن يستغرق وقتاً طويلاً إذا كان تسلسل الخلايا التابعة طويلاً.

لحل هذه المشكلة، جرب **ipyflow**. إنها نواة جيدة لـ Jupyter، والتي تتبع العلاقة بين الخلايا والمتغيرات.

```
In [1]: import numpy as np
In [2]: %flow mode reactive
In [3]: x = 10 ## Updating x automatically executes its dependents
In [4]: y = np.sin(x) ## Dependent on x
z = np.cos(x) ## Dependent on x
In [5]: output = y**2 + z**2 ## Dependent on y and z
output
```

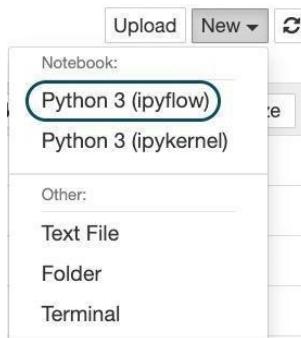
Automatic Execution of Dependent Cells

Export Code

```
In [ ]: from ipyflow import code
print(code(output))
```

وهكذا، في أي وقت، يمكنك الحصول على الكود المقابل لإعادة بناء أي كود. علاوة على ذلك، يتبع الأمر السحري الخاص به إعادة تنفيذ تلقائي للخلايا التابعة إذا تم تحديث متغير. كما هو موضح في العرض التوضيحي أعلاه، يؤدي تحديث المتغير `x` تلقائياً إلى تشغيل الخلايا التابعة له.

لاحظ أن نواة مختلفة عن النواة الافتراضية في Jupyter. وبالتالي، بمجرد تثبيت **ipyflow**، حدد النواة التالية أثناء تشغيل نوتبوك جديد:



ابحث عن مزيد من التفاصيل هنا: [ipyflow](#)

المقالة:

<https://avichawla.substack.com/p/supercharge-your-jupyter-kernel-with>

الكود:

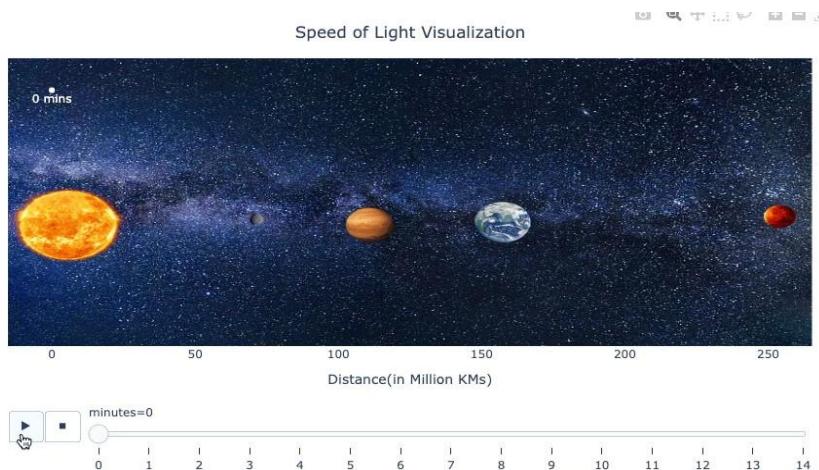
<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/Supercharge-Jupyter-Kernel.ipynb>

A Plotly أقل شهرة لإنشاء المخططات باستخدام Plotly

Lesser-known Feature of Creating Plots with Plotly

يتبع Plotly كثيراً عندما يتعلق الأمر بإنشاء أنواع مختلفة من المخططات. بينما يفضل العديد من الأشخاص للتفاعل، يمكنك أيضاً استخدامه لإنشاء مخططات متحركة مترددة .animated plots

إليكم تصوراً متحركاً animated visualization يوضح الوقت الذي يستغرقه الضوء للوصول إلى الكواكب المختلفة بعد مغادرة الشمس.



العديد من الدوال في دعم الرسوم المتحركة باستخدام Plotly

معلمات animation_group و anim_frame

تعتمد الفكرة الأساسية وراء إنشاء مخطط متحرك على رسم البيانات إطاراً واحداً في كل مرة.

على سبيل المثال، ضع في اعتبارك أننا نظمنا البيانات إطاراً تلو الآخر، كما هو موضح أدناه:

The figure shows a Jupyter Notebook cell with two tables. The top table is titled "planets" and has columns "planets", "x_position", "y_position", and "frame_id". It contains rows for Sun, Mercury, Venus, Earth, Mars, and Light. The bottom table is also titled "planets" and has the same columns. It contains rows for Sun, Mercury, Venus, Earth, Mars, and Light, with frame_id values of 1 and 2 respectively. The table for frame 2 is labeled "2nd Frame". An arrow points to the "Location in each frame" row, which is the last row of the table.

planets	x_position	y_position	frame_id
0 Sun	0.0	0.0	0
1 Mercury	76.0	0.0	0
2 Venus	118.0	0.0	0
3 Earth	156.0	0.0	0
4 Mars	256.0	0.0	0
5 Light	0.0	0.2	0

planets	x_position	y_position	frame_id
6 Sun	0.0	0.0	1
7 Mercury	76.0	0.0	1
8 Venus	118.0	0.0	1
9 Earth	156.0	0.0	1
10 Mars	256.0	0.0	1
11 Light	18.0	0.2	1

...

Location in each frame

الآن، إذا استدعاينا طريقة التبعثر scatter method باستخدام وسيطة anim_frame، فسوف ترسم البيانات إطاراً بإطار، مما يؤدي إلى ظهور رسم متحرك.



```
import plotly.express as px

>>> px.scatter(df,
               x="x_position",
               y="y_position",
               color = "planets",
               animation_frame="frame_id")
```

في استدعاء الدالة أعلاه، سيتم رسم البيانات المقابلة لـ frame_id = 0. سيتم استبدال هذا بالبيانات مع frame_id = 1 في الإطار التالي، وهكذا.

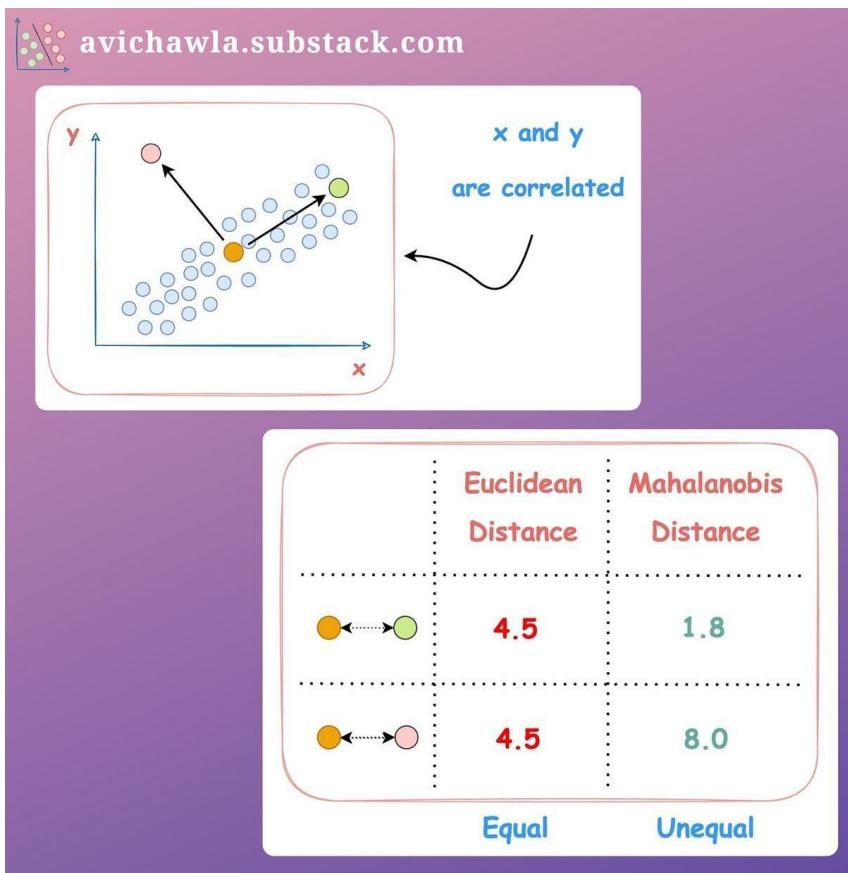
المقالة:

<https://avichawla.substack.com/p/a-lesser-known-feature-of-creating>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Animated-Plotting-With-Plotly.ipynb>

27) قيود المسافة الإقليدية التي كثيراً ما يتجاهلها الكثيرون The Limitation Of Euclidean Distance Which Many Often Ignore



المسافة الإقليدية Euclidean distance هي مقياس مسافة شائع الاستخدام. ومع ذلك، فإن قيودها غالباً ما تجعلها غير قابلة للتطبيق في العديد من مواقف البيانات.

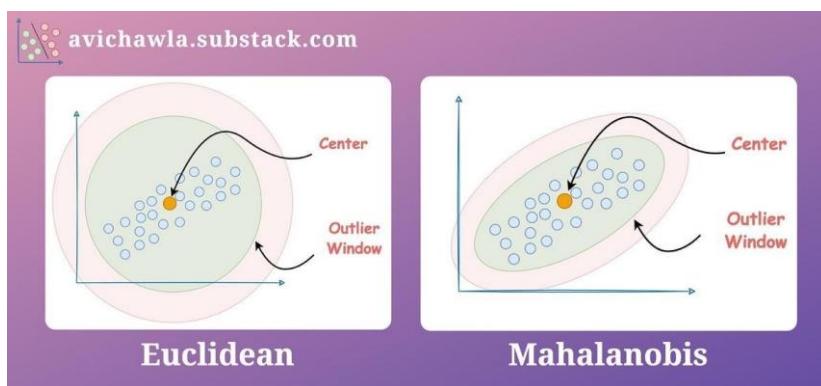
تفترض المسافة الإقليدية محاور مستقلة independent axes، والبيانات موزعة بشكل كروي إلى حد ما. ولكن عندما تكون الأبعاد مترابطة correlated، قد يتخرج عن الإقليدية نتائج مضللة misleading results.

تعد مسافة ماهالانوبيس Mahalanobis distance بديلاً ممتازاً في مثل هذه الحالات. إنه مقاييس مسافة متعدد المتغيرات يأخذ في الاعتبار توزيع البيانات.

نتيجة لذلك، يمكنه قياس مدى بعد نقطة البيانات عن التوزيع، وهو الأمر الذي لا تستطيع المسافة الأقلية القيام به.

كما هو موضح في الصورة أعلاه، يعتبر المسافة الأقلية أن النقاط الوردية والخضراء متساوية البعد عن النقطة المركزية. لكن مسافة ماهالانوبيس تعتبر النقطة الخضراء أقرب، وهذا صحيح بالفعل، مع مراعاة توزيع البيانات.

يتم استخدام مسافة ماهالانوبيس بشكل شائع في مهام الكشف الخارجية. كما هو موضح أدناه، في حين أن المسافة الأقلية تشكل حدوداً دائرية للقيم المتطرفة outliers، فإن ماهالانوبيس، بدلاً من ذلك، يأخذ في الاعتبار التوزيع — مما ينتج حداً أكثر عملية.



بشكل أساسي، تسمح مسافة ماهالانوبيس للبيانات ببناء نظام إحداثيات لنفسها، تكون فيه المحاور مستقلة .orthogonal ومتعدمة independent

من الناحية الحسابية، تعمل على النحو التالي:

الخطوة 1: تحويل الأعمدة إلى متغيرات غير مرتبطة.

الخطوة 2: قياس المتغيرات الجديدة لجعل تباينها يساوي 1.

الخطوة 3: ابحث عن المسافة الإقليدية في نظام الإحداثيات الجديد هذا، حيث تحتوي البيانات على وحدة تباين.

لذا في النهاية، نصل إلى الإقليدية. ومع ذلك، لاستخدام الإقليدية، نقوم أولاً بتحويل البيانات للتأكد من أنها تمثل لافتراضات.

رياضياً يحسب على النحو التالي:

$$D^2 = (x - \mu)^T \cdot C^{-1} \cdot (x - \mu)$$

- x : صفوف مجموعة البيانات الخاصة بك (الشكل: $(n_samples * n_dimensions)$)
 - μ : متوسط الأبعاد الفردية (الشكل: $1 * (n_dimensions)$)
 - C^{-1} : معكوس مصفوفة التغاير covariance matrix (الشكل: $* (n_dimensions)$)
 - D^2 : مربع مسافة Mahalanobis (الشكل: $(n_samples * n_samples)$)
- يمكنك العثور على مزيد من المعلومات هنا: [مستندات Scipy](#)

المقالة:

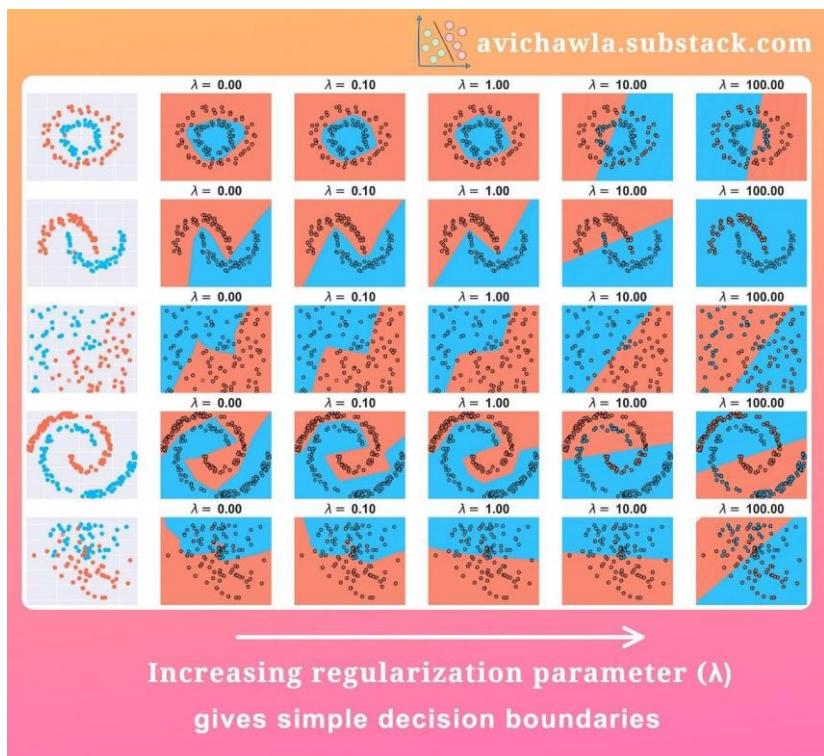
<https://avichawla.substack.com/p/the-limitation-of-euclidean-distance>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Mahalanobis-Distance.ipynb>

28) تصوّر تأثير معلمة التنظيم

Regularization Parameter



يشيع استخدام التنظيم Regularization لمنع الضبط الزائد overfitting. تصوّر الصورة المرئية أعلاه حدود القرار التي تم الحصول عليها في مجموعات البيانات المختلفة من خلال تغيير معلمة التنظيم regularization parameter.

كما هو موضح، تؤدي زيادة المعلمة إلى حد القرار مع انحناءات أقل. وبالمثل، فإن تقليل المعلمة ينتج حدوداً أكثر تعقيداً للقرار.

ل لكن هل تسائلت يوماً عما يدور وراء الكواليس؟ لماذا زيادة المعلمة تفرض حدوداً أبسط للقرار؟

لفهم ذلك، ضع في اعتبارك معادلة دالة التكلفة cost function أدناه (هذا من أجل الانحدار regression)، لكن الفكرة تبقى كما هي بالنسبة للتصنيف).

من الواضح أن التكلفة تزداد خطياً linearly مع المعلمة λ .

$$\begin{aligned}
 \text{Cost Function} &= \text{Loss} + \text{L2 Weight Penalty} \\
 &= \underbrace{\sum_{i=1}^M (y_i - \sum_{j=1}^N x_{ij} w_j)^2}_{\text{Squared Error}} + \lambda \underbrace{\sum_{j=1}^N w_j^2}_{\text{L2 Regularization Term}}
 \end{aligned}$$

**Higher the value of λ ,
higher the penalty**

الآن، إذا كانت المعلمة عالية جدًا، تصبح العقوبة penalty أعلى أيضًا. وبالتالي، لتقليل تأثيرها على دالة التكلفة الإجمالية، تضطر الشبكة إلى الاقتراب من الأوزان الأقرب إلى الصفر.

يصبح هذا واضحًا إذا قمنا بطباعة الأوزان النهائية لأحد النماذج، على سبيل المثال واحدة في أسفل اليمين (مجموعة البيانات الأخيرة، النموذج الأخير).

All weights close to zero

```
In [17]: clf.coefs_
Out[17]: array([[ 8.35476806e-06, -1.29066987e-05,  1.49535843e-05,
   8.43964067e-06,  5.46943218e-06,  1.18557175e-05,
   1.01037005e-05,  3.70503012e-06,  2.12142850e-06,
  -9.78452613e-06],
 [-1.35980250e-05,  1.52132934e-05,  3.30938991e-06,
  7.41538247e-07,  1.68626879e-05,  1.14315983e-05,
  6.64292409e-07, -1.40798113e-06,  1.31551207e-05,
  2.52379486e-05]])
```

إن وجود أوزان أصغر يؤدي إلى إلغاء العديد من الخلايا العصبية، مما ينبع عن شبكته أبسط بكثير. هذا يمنع العديد من التحولات المعقّدة التي كان يمكن أن تحدث لو لا ذلك.

المقالة:

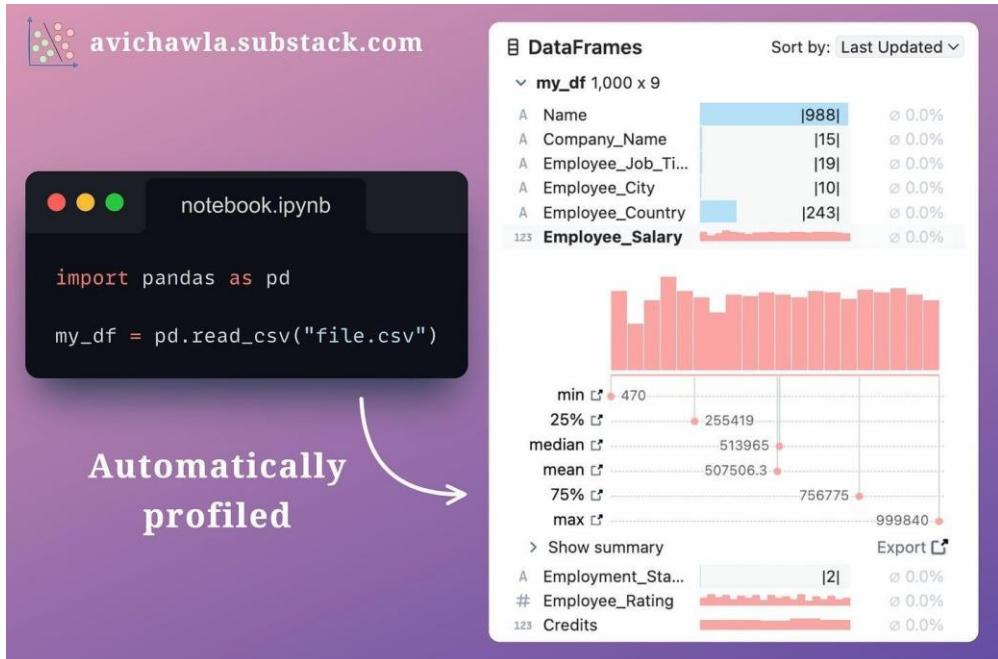
<https://avichawla.substack.com/p/visualising-the-impact-of-regularisation>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/What-Does-Regularization-Do.ipynb>

AutoProfiler (29): ملف تعريف إطار البيانات الخاص بك تلقائياً

AutoProfiler: Automatically Profile Your DataFrame As You Work



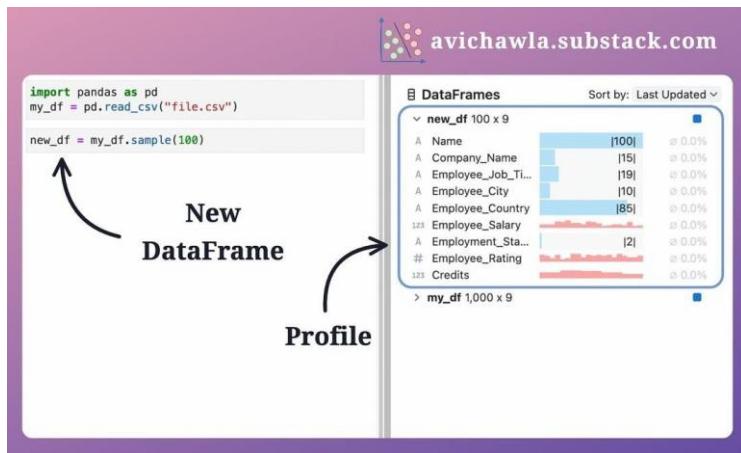
Pandas AutoProfiler: تلقائياً ، يعمل بروفایل لاطارات بيانات DataFrame بانداس في كل تنفيذ دون أي كود.

Pandas AutoProfiler: تلقائياً ، فإن إطارات بيانات PANDAS في كل تنفيذ ، دون أي كود.

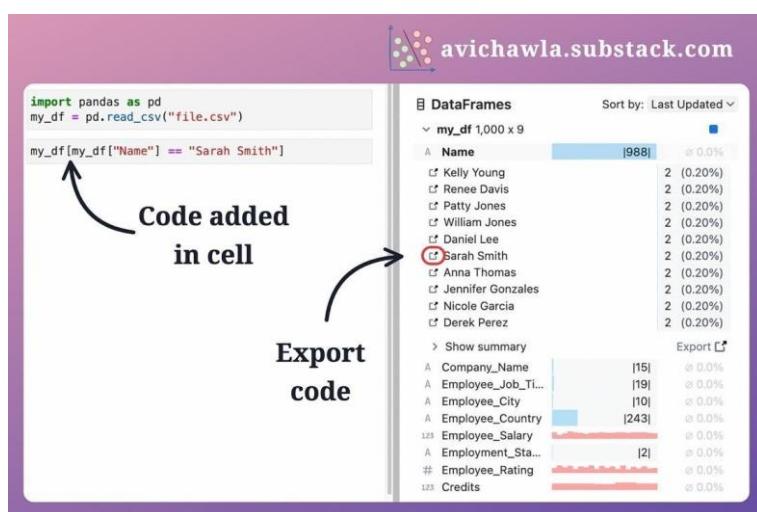
AutoProfiler هي أداة تحليل إطار بيانات DataFrame مفتوحة المصدر في Jupyter. يقرأ النوتوبوك الخاص بك ويقوم تلقائياً بعرض كل إطار بيانات في ذاكرتك أثناء تغييرها.

بمعنى آخر ، إذا قمت بتعديل إطار بيانات موجود ، فسيقوم برنامج AutoProfiler تلقائياً بتحديث البرنامج المقابل له.

أيضاً ، إذا قمت بإنشاء إطار بيانات جديد (على سبيل المثال من إطار بيانات موجود) ، فسيقوم AutoProfiler تلقائياً بتوفير ذلك أيضاً ، كما هو موضح أدناه:



تضمن معلومات البروفایلینك Profiling info توزيع الأعمدة وإحصائيات الملخص والإحصائيات الفارغة وغير ذلك الكثير. علاوة على ذلك، يمكنك أيضاً إنشاء الكود المقابل، مع ميزة التصدر الخاصة به.



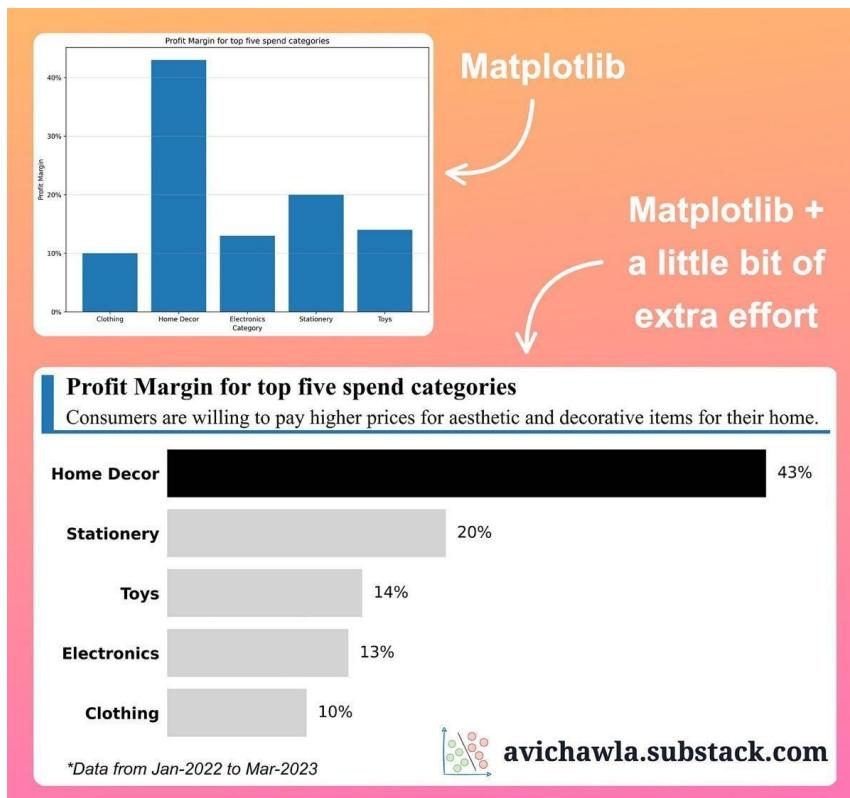
المقالة:

<https://avichawla.substack.com/p/autoprofiler-automatically-profile>

الكتاب:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/DataFrame-Auto-Profile.ipynb>

(30) القليل من الجهد الإضافي يمكن أن يحول بشكل كبير مهاراتك في سرد القصص A Little Bit Of Extra Effort Can Hugely Transform Your Storytelling Skills



يتم التقليل من أهمية Matplotlib عندما يتعلق الأمر بإنشاء مخططات ذات مظهر احترافي. ومع ذلك، فهي قادرة تماماً على القيام بذلك. على سبيل المثال، ضع في اعتبارك المخططات أدناه.

نعم، تم إنشاء كلاهما باستخدام matplotlib. لكن القليل من التنسيق يجعل المخطط الثاني أكثر إفادة وجاذبية وسهولة في المتابعة.

يساعد العنوان والعنوان الفرعي القصة بشكل كبير. كما أن الحاشية السفلية تحتوي على معلومات إضافية مهمة، والتي لا يمكن رؤيتها في أي مكان في المخطط الأساسي.

أخيراً، يلفت الشريط الغامق انتباه المشاهد على الفور وينقل أهمية الفئة.

إذن ما هي الرسالة هنا؟

لكي تكون راوياً جيداً للبيانات good data storyteller، تأكد من أن مخاططتك يتطلب الحد الأدنى من الجهد من المشاهد. وبالتالي، لا تتردد في بذل هذا الجهد الإضافي. هذا ينطبق بشكل خاص على البيئات المهنية.

في بعض الأحيان، قد يكون من الجيد أيضاً التأكد من أن تصوراتك تنقل القصة الصحيحة، حتى لو تم عرضها في غيابك.

المقالة:

<https://avichawla.substack.com/p/a-little-bit-of-extra-effort-can>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Professional-Plots-With-Matplotlib.ipynb>

31) ميزة مخفية سيئة في بايثون لا يعرفها الكثير من المبرمجين A Nasty Hidden Feature of Python That Many Programmers Aren't Aware Of

Mutable Default Parameter

```
def add_subject(name, subject, subjects=[]):
    subjects.append(subject)
    return {'name': name, 'subjects': subjects}

>>> add_subject('Joe', 'Maths')
>>> add_subject('Bob', 'Maths')
>>> add_subject('Roy', 'Maths')
```

Output:

```
{'name': 'Joe', 'subjects': ['Maths']}
{'name': 'Bob', 'subjects': ['Maths', 'Maths']}
{'name': 'Roy', 'subjects': ['Maths', 'Maths', 'Maths']}
```

Appended to
the same list

avichawla.substack.com

ربما تكون قابلية التغيير Mutability في بايثون من أكثر المفاهيم التي يساء فهمها وتتجاهلها. توضح الصورة أدناه مثلاً يكافح العديد من مبرمجي بايثون (خاصة المبتدئين منهم) لفهمه.

هل يمكنك فهمها؟ إذا لم يكن كذلك، دعونا نفهم ذلك.

يتم تقييم المعلمات الافتراضية للدالة في الوقت الذي يتم فيه تحديد الدالة. بمعنى آخر، لا يتم تقييمها في كل مرة يتم فيها استدعاء الدالة (كما هو الحال في C++).

وبالتالي، بمجرد تحديد دالة، يخزن كائن الدالة Function object المعلمات الافتراضية في سمة الإعدادات الافتراضية `_defaults` الخاصة بها. يمكننا التحقق من ذلك أدناه:

```

def my_function(a=1, b=2, c=3):
    pass

>>> my_function.__defaults__
(1, 2, 3)

```

وبالتالي، إذا قمت بتحديد معلمة افتراضية قابلة للتغيير mutable في دالة وقمت بتغييرها، فإنك تقوم عن غير قصد وبدون قصد بتعديل المعلمة لجميع الاستدعاءات المستقبلية لهذه الدالة.

هذا موضح في العرض التوضيحي أدناه. بدلاً من إنشاء قائمة جديدة عند كل استدعاء دالة، تلحق بايثون العنصر بالنسخة نفسها.

```

def add_subject(*args):
    ...

>>> add_subject.__defaults__
([],)

>>> add_subject('Joe', 'Maths')
>>> add_subject.__defaults__
(['Maths'],)

>>> add_subject('Bob', 'Maths')
>>> add_subject.__defaults__
(['Maths', 'Maths'],)

>>> add_subject('Roy', 'Maths')
>>> add_subject.__defaults__
(['Maths', 'Maths', 'Maths'],)

```

إذن ما الذي يمكننا فعله لتجنب ذلك؟

بدلاً من تحديد معلمة افتراضية قابلة للتغيير في تعريف الدالة، استبدلها ب None. إذا لم تستقبل الدالة قيمة مقابلة أثناء استدعاء الدالة، فقم بإنشاء كائن قابل للتغيير داخل الدالة.

هذا مرض أدناه:

The screenshot shows a Jupyter Notebook cell with the title "Replace mutable parameter". The code defines a function `add_subject` that takes `name`, `subject`, and an optional `subjects` parameter set to `None`. If `subjects` is `None`, it creates an empty list. Then it appends the `subject` to `subjects` and returns a dictionary with `name` and `subjects`. The output shows three calls to `add_subject` with 'Maths' as the subject, resulting in three dictionaries where each has a single 'Maths' subject.

```
Replace mutable parameter
def add_subject(name, subject, subjects=None):
    if subjects is None:
        # Create if no value was received
        subjects = []
    subjects.append(subject)
    return {'name': name, 'subjects': subjects}

>>> add_subject('Joe', 'Maths')
>>> add_subject('Bob', 'Maths')
>>> add_subject('Roy', 'Maths')
```

Output:

```
{'name': 'Joe', 'subjects': ['Maths']}
{'name': 'Bob', 'subjects': ['Maths']}
{'name': 'Roy', 'subjects': ['Maths']}
```

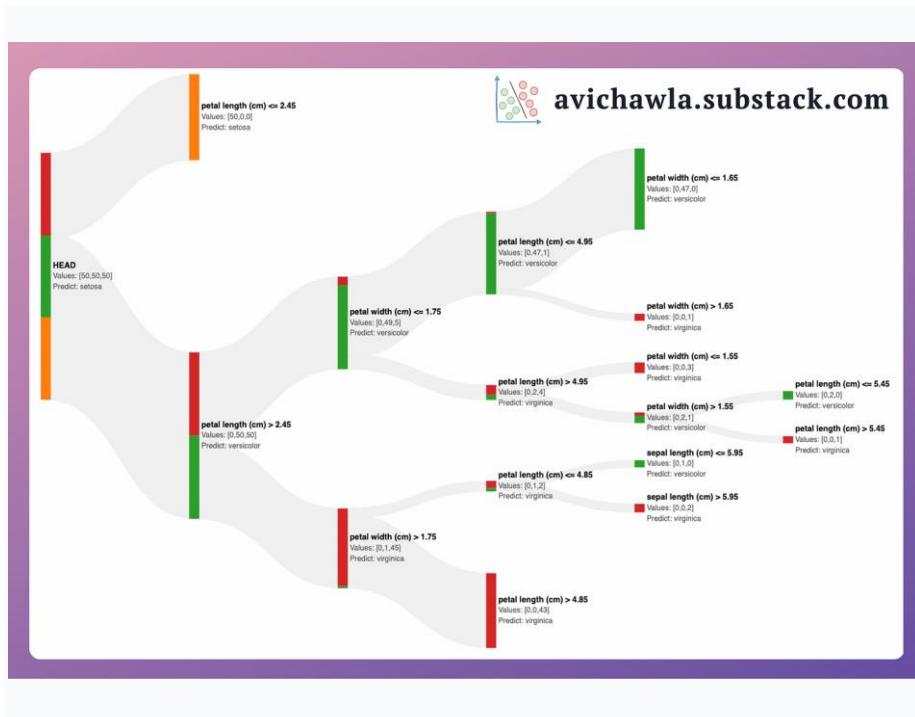


avichawla.substack.com

كما هو موضح أعلاه، نقوم بإنشاء قائمة جديدة new list إذا لم تلتقي الدالة أي قيمة عند استدعائها. يتبع لك هذا تجنب السلوك غير المتوقع لتحول نفس الكائن.

(32) تصور تفاعلي لشجرة قرار مع مخطط سانكي

Interactively Visualise A Decision Tree With A Sankey Diagram

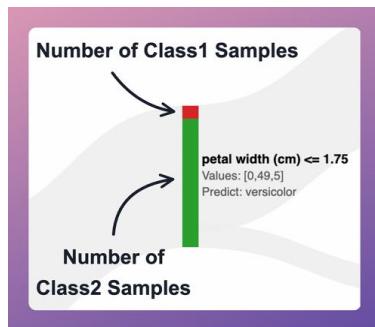


في إحدى منشوراتي السابقة، أوضحت سبب تجاوز أشجار قرارات `sklearn` للبيانات بمعالماتها الافتراضية (اقرأ [هنا](#) إذا كنت ترغب في الرجوع).

لتجنب ذلك، يوصى دائمًا بتحديد قيم المعلمة `hyperparameter` الفائقة المناسبة. يتضمن ذلك أقصى عمق للشجرة، وعينات دقيقة في عقد الأوراق، إلخ.

ولكن غالباً ما يتم تحديد قيم المعلمات الفائقة هذه باستخدام التجربة والخطأ trial-and-error، والتي يمكن أن تكون مملة بعض الشيء وتستغرق وقتاً طويلاً.

يتيح لك مخطط سانكي Sankey أعلاه تصور تفاعلي لتنبؤات شجرة القراري في كل عقدة. أيضاً، يتم ترميز عدد نقاط البيانات من كل فئة بالحجم على جميع العقد، كما هو موضح أدناه.



هذا يعطي على الفور تقديرًا لشوائب impurity العقدة. بناءً على ذلك، يمكنك أن تقرر بصرياً تقليل prune الشجرة.

على سبيل المثال، في شجرة القرار الكاملة الموضحة أدناه، يبدو أن تقليل الشجرة على عمق اثنين أمر معقول.



بمجرد حصولك على تقدير تجاري لقيم المعلمات الفائقة هذه، يمكنك تدريب شجرة قرار جديدة. بعد ذلك، قم بقياس أدائها على البيانات الجديدة لمعرفة ما إذا كانت شجرة القرار معممة أم لا.

المقالة:

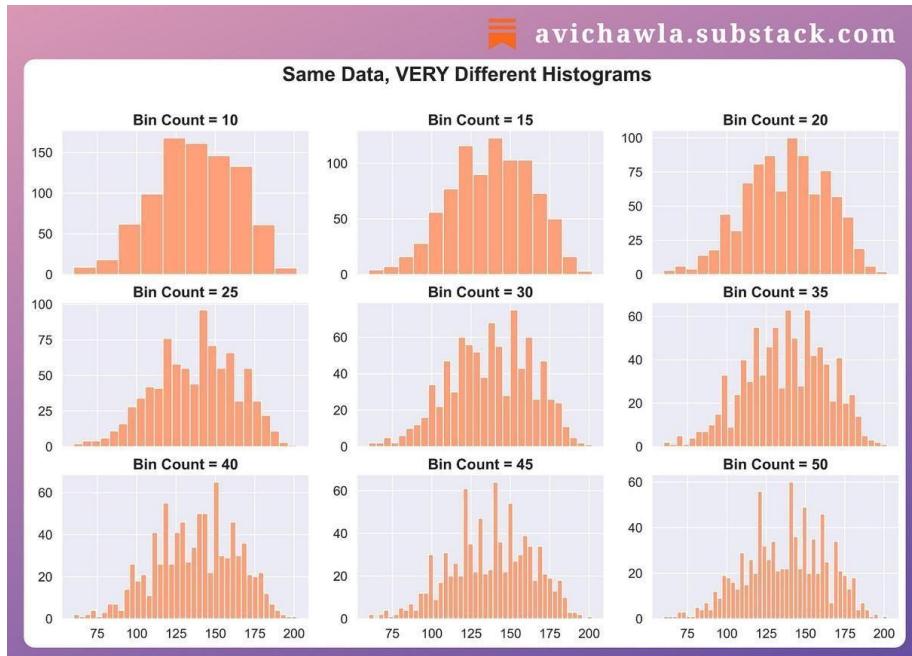
<https://avichawla.substack.com/p/interactively-visualise-a-decision>

الكود:

<https://avichawla.substack.com/p/interactively-visualise-a-decision>

(33) استخدم المدرجات التكراري بحذر. إنها مضللة للغاية!

Use Histograms with Caution. They Are Highly Misleading



تُستخدم المدرجات التكرارية Histograms بشكل شائع لتصور البيانات data visualization. لكنها قد تكون مضللة في بعض الأحيان. إليكم السبب.

تقسم المدرجات التكرارية البيانات إلى حاويات bins صغيرة وتمثل تكرار كل حاوية.

وبالتالي، فإن اختيار عدد الحاويات التي تبدأ بها يمكن أن يؤثر بشكل كبير على شكلها.

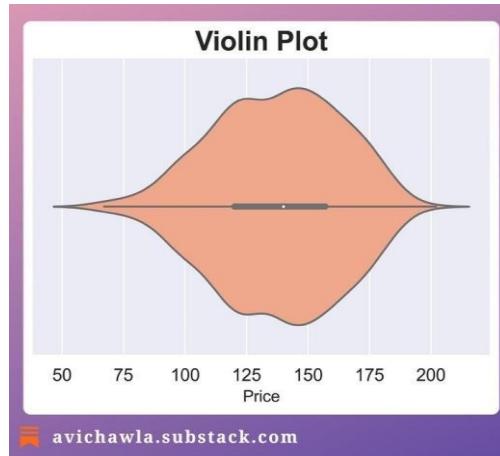
يوضح الشكل أعلاه المدرجات التكرارية التي تم الحصول عليها من نفس البيانات، ولكن عن طريق تغيير عدد الحاويات. ينقل كل مدرج تكراري قصة مختلفة، على الرغم من أن البيانات الأساسية هي نفسها.

قد يكون هذا مضللاً في بعض الأحيان وقد يؤدي بك إلى استخلاص استنتاجات خاطئة.

القصد ليس أنه لا ينبغي استخدام المدرجات التكرارية. بدلاً من ذلك، انظر إلى التوزيع الأساسي أيضاً. هنا، يمكن أن يساعد مخطط الكمان violin ومخطط KDE.

مخطط الكمان

على غرار المخططات الصندوقية box plots، تُظهر مخططات الكمان أيضًا توزيع البيانات بناءً على الرباعية quartiles. ومع ذلك، فإنه يضيف أيضًا تقدير كثافة النواة kernel density لعرض كثافة البيانات بقيم مختلفة.



يوفر هذا عرضًا أكثر تفصيلاً للتوزيع، لا سيما في المناطق ذات الكثافة العالية.

KDE مخطط

تستخدم مخططات KDE منحنياً سلساً لتمثيل توزيع البيانات، دون الحاجة إلى تجميع البيانات كما هو موضح أدناه: binning



كملاحظة مغادرة، تذكر دائمًا أنه كلما قمت بتكثيف مجموعة بيانات، فإنك تخاطر بفقدان معلومات مهمة.

وبالتالي، ضع في اعتبارك أي قيود (وافتراضات) للتصورات التي تستخدمها. أيضًا، ضع في اعتبارك استخدام طرق متعددة للتأكد من أنك ترى الصورة كاملة.

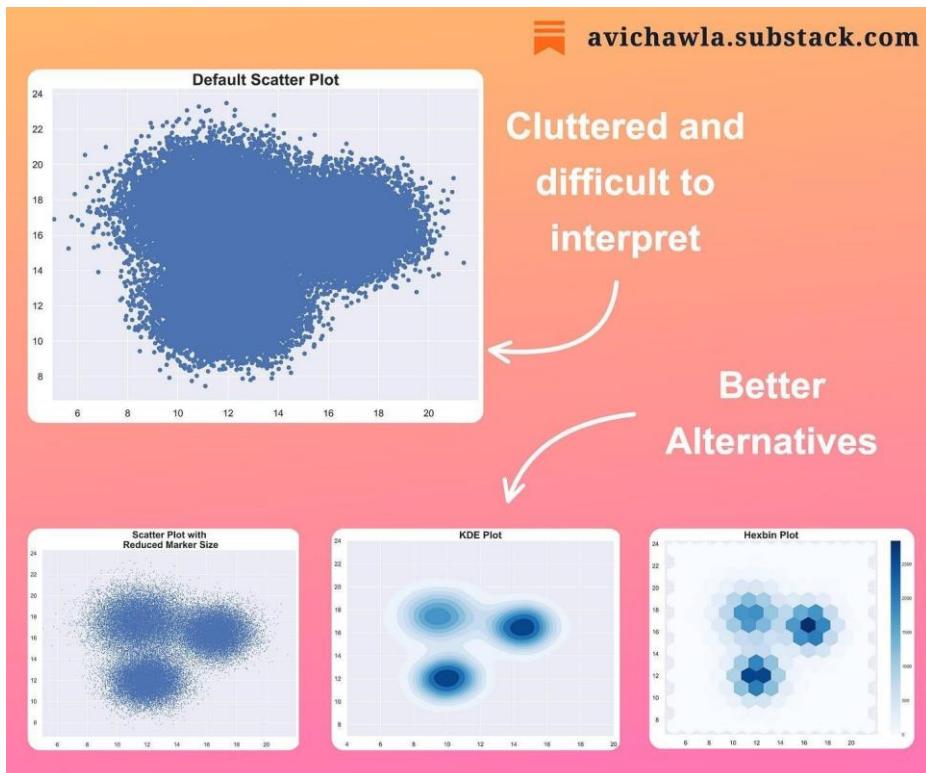
المقالة:

<https://avichawla.substack.com/p/use-histograms-with-caution-they>

الكود:

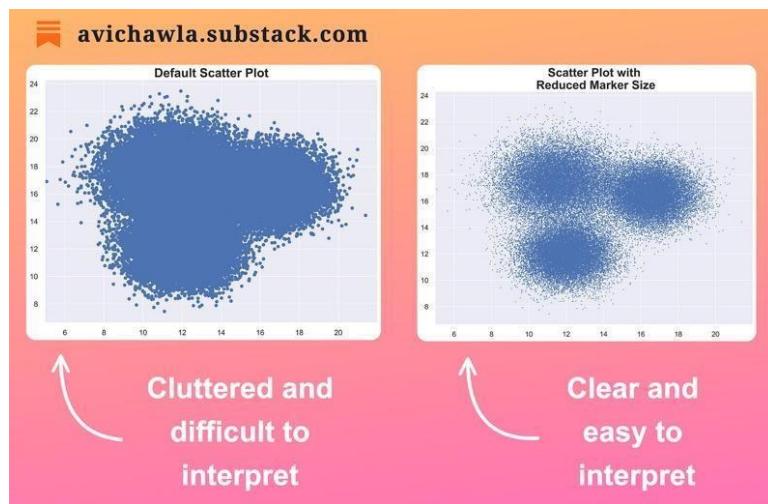
<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Misleading-Hist-Plots.ipynb>

34) ثلاثة طرق بسيطة (فورية) تجعل مخططات التشتت خالية من الفوضى Three Simple Ways To (Instantly) Make Your Scatter Plots Clutter Free

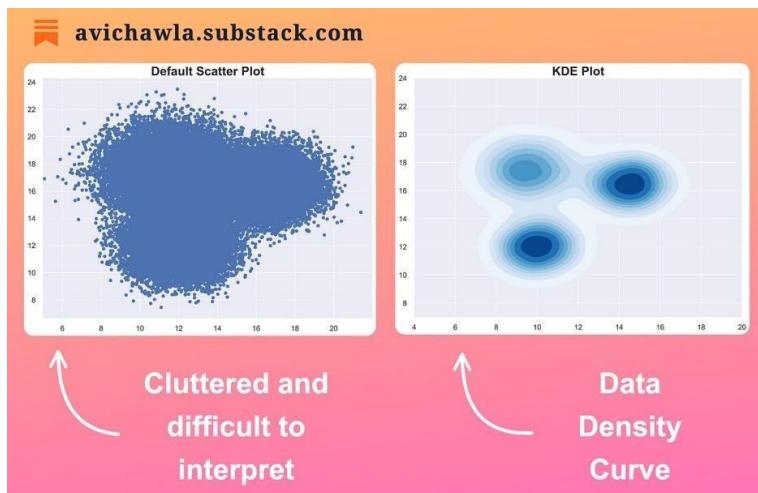


تُستخدم المخططات المبعثرة (مخططات التشتت) Scatter plots بشكل شائع في مهام تصوّر البيانات. ولكن عندما يكون لديك العديد من نقاط البيانات، فغالباً ما تكون كثيفة للغاية بحيث لا يمكن تفسيرها. فيما يلي بعض الأساليب (والبدائل) التي يمكنك استخدامها لجعل بياناتك قابلة للتفسير interpretable في مثل هذه الحالات.

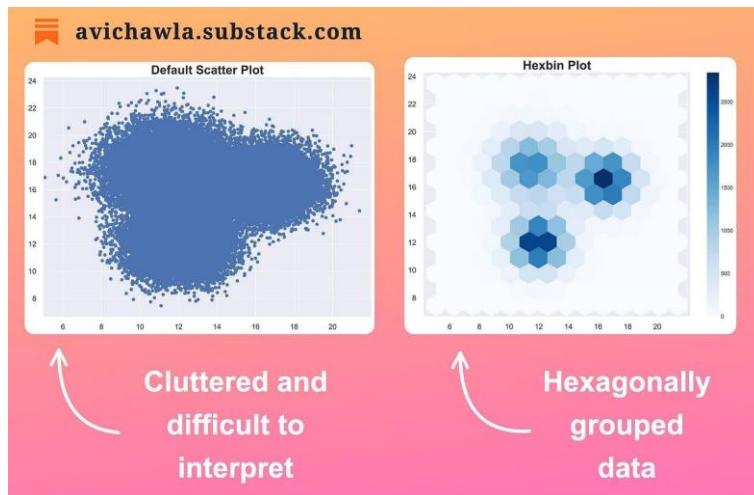
قد تكون إحدى أبسط الطرق وأكثرها فاعلية هي تقليل حجم العلامة marker. هذا، في بعض الأحيان، يمكن أن يوفر على الفور وضوحاً أفضل على المخطط الافتراضي.



بعد ذلك، كديل لمخطط التبعثر، يمكنك استخدام مخطط الكثافة density plot، الذي يصور توزيع البيانات. هذا يجعل من السهل تحديد المناطق ذات الكثافة العالية والمنخفضة، والتي قد لا تكون واضحة من مخطط التبعثر.



أخيرًا، يمكن أن يكون البديل الأفضل هو مخطط hexbin. يقوم بتجميع الرسم البياني في مناطق سداسية وتعيين كثافة اللون بناءً على عدد النقاط في تلك المنطقة.



المقالة:

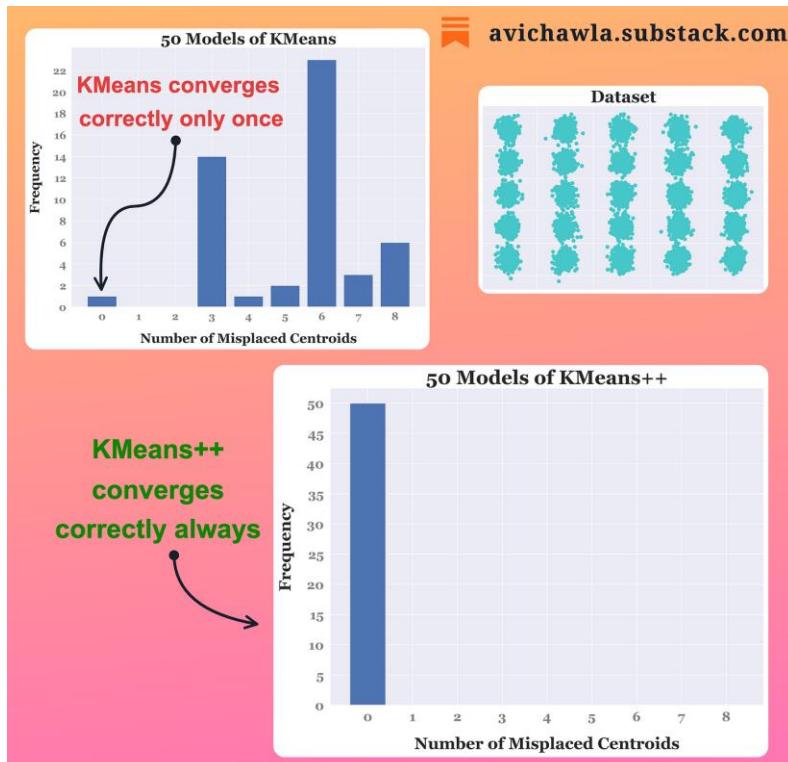
<https://avichawla.substack.com/p/three-simple-ways-to-instantly-make>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/3-Tips-For-Better-Scatter-Plots.ipynb>

(35) نقطة (عالية) مهمة يجب مراعاتها قبل استخدام KMeans

A (Highly) Important Point to Consider Before You Use KMeans Next Time



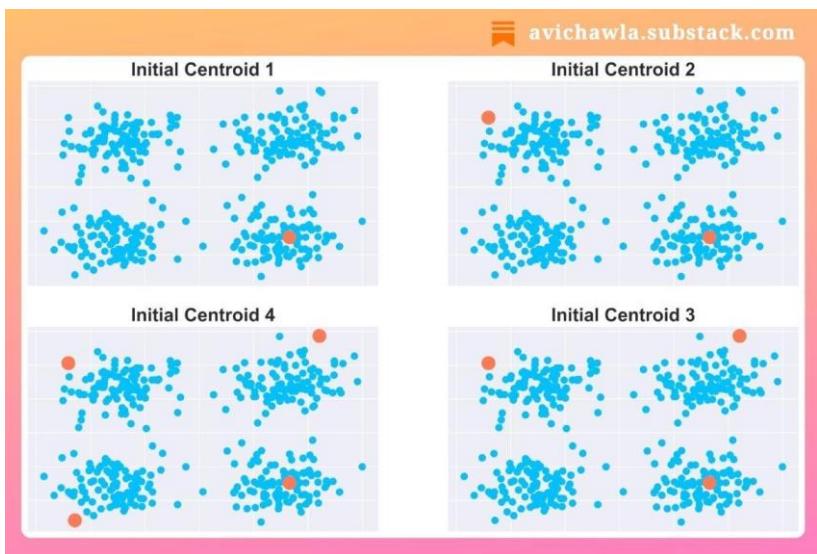
الخطوة الأكثر أهمية والتي غالباً ما يتم تجاهلها من KMeans هي تهيئة النقطة الوسطى centroid. إليك شيء يجب مراعاته قبل استخدامه في المرة القادمة. initialization.

يختار KMeans النقطة الوسطى الأولية بشكل عشوائي. نتيجة لذلك، فشل في التقارب converge في بعض الأحيان. هذا يتطلب منا تكرار التجميع clustering عدة مرات مع تهيئة مختلفة.

ومع ذلك، قد لا يضمن التجميع المتكرر أنك ستنتهي قريباً بالمجموعات الصحيحة. هذا صحيح بشكل خاص عندما يكون لديك العديد من النقطة الوسطى لتبأ.

بدلاً من ذلك، يتخذ KMeans++ نهجاً أكثر ذكاءً لتهيئة النقطة الوسطى.

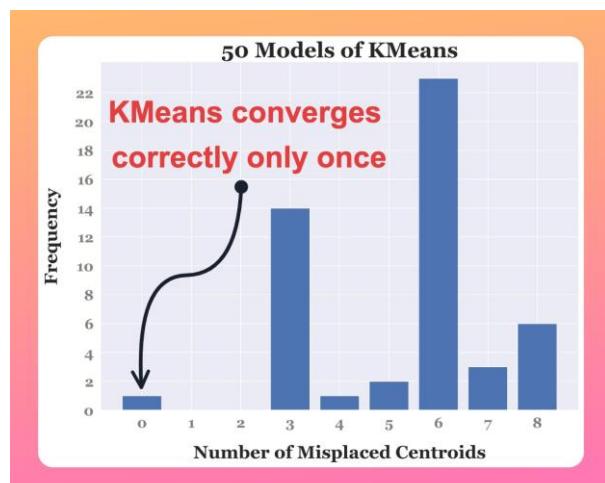
يتم اختيار النقطة الوسطى الأول عشوائياً. ولكن يتم اختيار النقطة الوسطى التالية على أساس المسافة من النقطة الوسطى الأول.



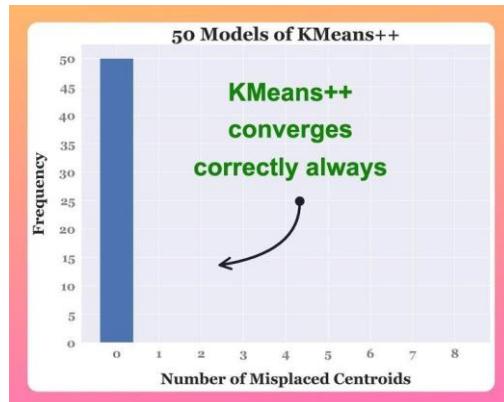
عبارة أخرى، من المرجح أن يتم تحديد النقطة بعيدة عن النقطة الوسطى الأولى كنقطة مركبة أولية. بهذه الطريقة، من المحتمل أن تقع جميع النقاط الوسطى الأولية في مجموعات مختلفة بالفعل، وقد تقارب الخوارزمية بشكل أسرع وأكثر دقة.

التأثير واضح من المخططات الشريطية bar plots الموضحة أدناه. يصورون تواتر عدد النقاط الوسطى في غير محلها التي تم الحصول عليها (تم تحليلها يدوياً) بعد تدريب 50 نموذجاً مختلفاً باستخدام KMeans++ و KMeans

في مجموعة البيانات المحددة، من بين النماذج الخمسين، أنتجت KMeans صفرًا من النقط الوسطى في غير محلها مرة واحدة، وهو معدل نجاح يبلغ 2٪ فقط.



في المقابل، لم يتحقق KMeans++ أبداً أي نقطتين الوسطى في غير محله.



لحسن الحظ، إذا كنت تستخدم sklearn، فلا داعي للقلق بشأن خطوة التهيئة. هذا لأن sklearn، بشكل افتراضي، تلجمأ إلى نهج KMeans++
و مع ذلك، إذا كان لديك تطبيق مخصص، ففكّر فيه.

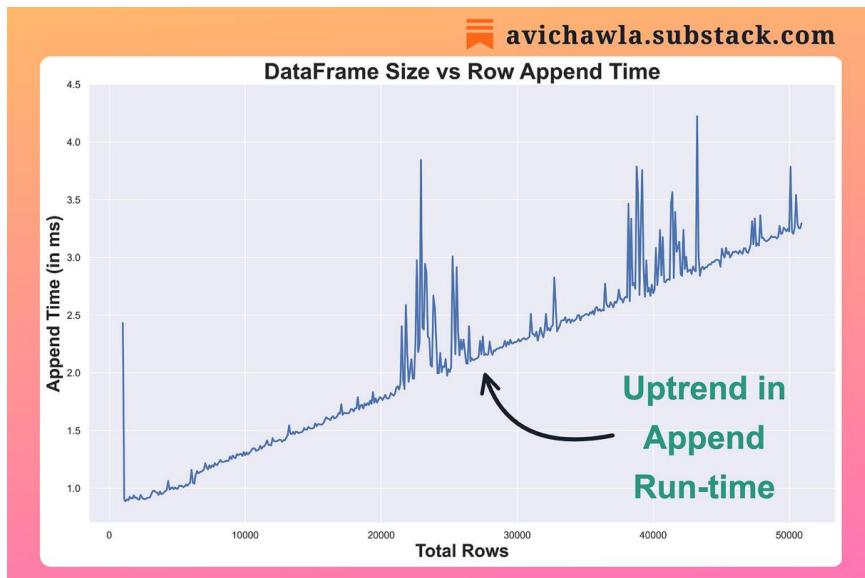
المقالة:

<https://avichawla.substack.com/p/a-highly-important-point-to-consider>

الكود:

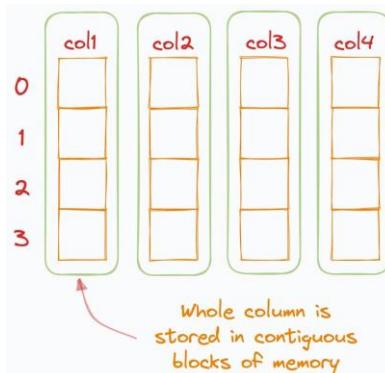
<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/50-KMeans-Models.ipynb>

Why You Should Avoid Appending Rows To A DataFrame



نظرًا لأننا نلحق append المزيد والمزيد من الصفوف rows بـ Pandas DataFrame ، يستمر وقت تشغيل الإلتحاق في الزيادة. إليكم السبب.

يعد DataFrame بنية بيانات ذات عمود رئيسي. وبالتالي، يتم تخزين العناصر المتتالية في عمود بجانب بعضها البعض في الذاكرة.



مع إضافة صفات جديدة، يُريد Pandas دائمًا الحفاظ على شكل العمود الرئيسي. ولكن أثناء إضافة صفات جديدة، قد لا تكون هناك مساحة كافية لاستيعابها مع الحفاظ أيضًا على هيكل العمود الرئيسي.

في مثل هذه الحالة، يتم نقل البيانات الموجودة إلى موقع ذاكرة جديد، حيث يجد كتلة متغيرة Pandas من الذاكرة.

وبالتالي، مع نمو الحجم، تزداد إعادة تخصيص الذاكرة تواترًا، ويستمر وقت التشغيل في الزيادة.

قد يكون سبب الارتفاعات المفاجئة في هذا الرسم البياني هو نقل عمود يأخذ ذاكرة أعلى إلى موقع جديد في هذه المرحلة، وبالتالي يستغرق وقتًا أطول لإعادة التخصيص، أو تم إزاحة العديد من الأعمدة في وقت واحد.

إذن ما الذي يمكننا فعله للتخفيف من هذا؟

تنشأ الزيادة في وقت التنفيذ فقط لأن Pandas تحاول الحفاظ على هيكل العمود الرئيسي.

وبالتالي، إذا كنت تنوی تطوير إطار بيانات (من منظور الصف row-wise) بشكل متكرر، فمن الأفضل أن تقوم أولاً بتحويل إطار البيانات إلى بنية بيانات أخرى، أو قاموس أو مصفوفة صغيرة، على سبيل المثال.

قم بتنفيذ عمليات الإلحاد هنا، وعند الانتهاء، قم بتحويله مرة أخرى إلى إطار بيانات.
ملاحظة. إضافة أعمدة جديدة ليست مشكلة. هذا لأن هذه العملية لا تتعارض مع الأعمدة الأخرى.

المقالة:

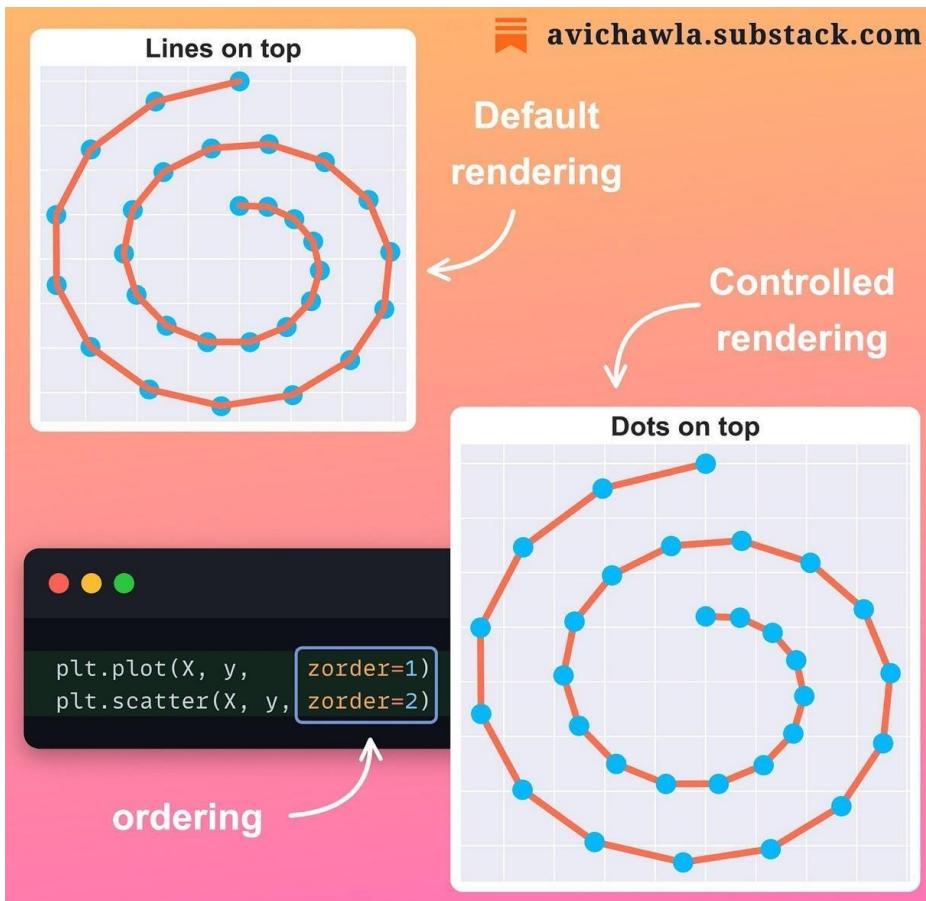
<https://avichawla.substack.com/p/why-you-should-avoid-appending-rows>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Append-Run-Time.ipynb>

(37) يحتوي Matplotlib على العديد من الأحجار الكريمة Matplotlib Has Numerous Hidden Gems. هنا واحد منها. هنا واحد منها.

Hidden Gems. Here's One of Them



تعد قابلية التخصيص customizability واحدة من أفضل إمكانيات matplotlib حتى الآن التي تم التقليل من شأنها. إليك شيء مثير للاهتمام يمكنك فعله به.

بشكل افتراضي، يعرض matplotlib أنواعاً مختلفة من العناصر elements (تسمى أيضاً artists)، مثل المخططات plots والتصنيفات legend والنوصوص texts وما إلى ذلك، بترتيب محدد.

لكن هذا الترتيب قد لا يكون مرغوبًا في جميع الحالات، خاصةً عندما تكون هناك عناصر متداخلة في المخطط، أو عندما يخفي العرض الافتراضي بعض التفاصيل المهمة.

باستخدام معلمة `zorder`، يمكنك التحكم في ترتيب العرض هذا. نتيجة لذلك، تظهر المخططات ذات قيمة `zorder` الأعلى أقرب إلى العارض ويتم رسماًها أعلى `zorder` artists بقيمة منخفضة.

أخيرًا، في العرض أعلاه، إذا حددنا `zorder=0` للمخطط الخطى `line plot`، نلاحظ أنه يسير خلف خطوط الشبكة.



يمكنك الحصول على مزيد من التفاصيل حول `zorder` هنا: [مستندة Matplotlib](#)
المقالة:

<https://avichawla.substack.com/p/matplotlib-has-numerous-hidden-gems>

ال코드:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/Matplotlib-Rendering-Order.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Matplotlib-Rendering-Order.ipynb)

A (38) شيء غير بدائي حول قواميس بايثون Counterintuitive Thing About Python Dictionaries

```
>>> my_dict = {
    1.0 : 'One (float)',
    1 : 'One (int)',
    True : 'One (bool)',
    '1' : 'One (string)'
}

>>> my_dict
{1.0 : 'One (bool)', '1' : 'One (string)'}
```

Added 4 keys

dict only has 2 keys

على الرغم من إضافة 4 مفاتيح keys مميزة إلى قاموس بايثون Python dictionary، هل يمكنك معرفة سبب احتفاظه بمفتاحين فقط؟

إليكم السبب.

في بايثون، تجد القواميس مفتاحاً يعتمد على معادلة التجزئة hash (محسوبة باستخدام hash()) ، ولكن ليس الهوية identity (محسوبة باستخدام id()).

في هذه الحالة، ليس هناك شك في أن 1.0 و 1 و True بطبيعتها أنواع بيانات مختلفة وهي أيضاً كائنات مختلفة. هذا موضح أدناه:

```
>>> id(1.0), id(1), id(True)
(153733, 127473, 493931)

>>> type(1.0), type(1), type(True)
(float, int, bool)
```

ومع ذلك، نظراً لأنها تشتراك في نفس قيمة التجزئة، فإن القاموس يعتبرها نفس المفاتيح.

```
>>> hash(1.0), hash(1), hash(True)
(1, 1, 1) ## same hash
```

لكن هل لاحظت أنه في العرض التوضيحي، المفتاح النهائي هو 1.0، بينما تتوافق القيمة مع المفتاح .True

```
>>> my_dict
{1.0: 'One (bool)', '1': 'One (string)'}
```

float key value of boolean key

هذا لأنه، في البداية، يتم إضافة 1.0 كمفتاح وقيمتها هي "One (float)". بعد ذلك، أثناء إضافة المفتاح 1، يُعرف عليه بايثون على أنه معادل لقيمة التجزئة.

وبالتالي، يتم استبدال القيمة المقابلة لـ 1.0 بـ "One (int)"، بينما يتم الاحتفاظ بالمفتاح 1.0 كما هو.

أخيراً، أثناء إضافة True، تمت مصادفة تكافؤ تجزئة آخر بمفتاح موجود 1.0. مرة أخرى، تم استبدال القيمة المقابلة لـ 1.0، والتي تم تحديتها إلى "One (int)" في الخطوة السابقة، بـ "One (bool)".

أنا متأكد من أنك ربما تكون قد خمنت الفعل سبب الاحتفاظ بمفتاح السلسلة "1".

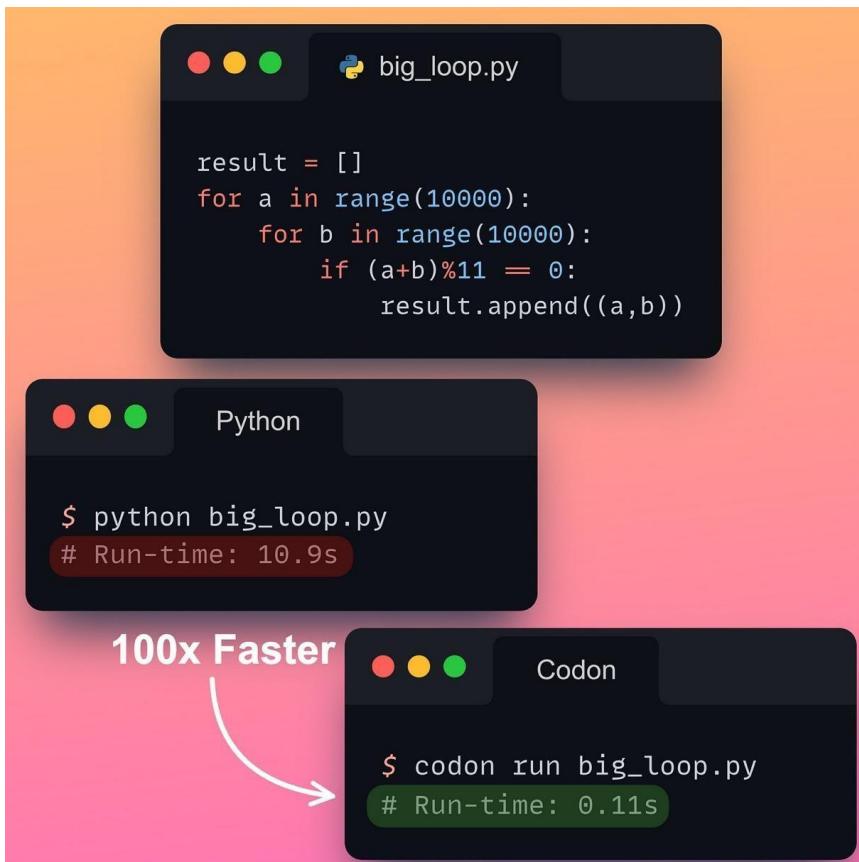
المقالة:

<https://avichawla.substack.com/p/a-counterintuitive-thing-about-python>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Counterintuitive-Dictionaries.ipynb>

(39) ربما تكون أسرع طريقة لتنفيذ كود بايثون الخاص بك Probably The Fastest Way To Execute Your Python Code



غالباً ما يشعر العديد من مبرمجي بايثون بالإحباط من وقت تشغيل بايثون. إليك كيفية جعل التعليمات البرمجية الخاصة بك سريعة للغاية عن طريق تغيير سطر واحد فقط.

Codon هو مترجم بايثون Python compiler مفتوح المصدر وعالي الأداء. على عكس كونك مترجماً، يقوم بتجميع كود بايثون الخاص بك إلى كود الآلة السريع.

وبالتالي، بعد التجميع compilation، يتم تشغيل التعليمات البرمجية الخاصة بك بسرعة كود الجهاز الأصلي. نتيجة لذلك، غالباً ما تكون عمليات التسريع النموذجية بترتيب 50 مرة أو أكثر.

وفقاً للمعهدات الرسمية، إذا كنت تعرف لغة بایثون، فأنت تعرف بالفعل 99% من Codon. توجد اختلافات دقيقة جداً بين الاثنين، يمكنك قراءتها هنا: [معهدات Codon](#)

ابحث عن المزيد من نتائج قياس الأداء بين Codon و Python أدناه:

The screenshot shows a Substack post with the title "probably-the-fastest-way-to-execute". It compares the execution times of Python and Codon for two scripts: fib.py and pi.py.

fib.py (Python):

```
def fib(N):
    """
    Function to find the
    Nth Fibonacci number.

    fib(N) = fib(N-1) + fib(N-2)
    """
    ...
```

pi.py (Python):

```
def pi_approx(n_terms):
    """
    Function to find the
    approximate value of pi.

    pi = 4*(1 - 1/3 + 1/5 - 1/7...)
    """
    ...
```

Execution Times:

Script	Language	Time	Notes
fib.py	Python	2.53s	# Time: 2.53s
fib.py	Codon	0.04s (~60x Faster)	\$ codon run fib.py # N=35 # Time: 0.04s (~60x Faster)
pi.py	Python	296s	\$ python pi.py # n_terms=10^8 # Time: 296s
pi.py	Codon	4.89s (~60x Faster)	\$ codon run pi.py # n_terms=10^8 # Time: 4.89s (~60x Faster)

المقالة:

<https://avichawla.substack.com/p/probably-the-fastest-way-to-execute>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Runtime%20Optimization/Codon-vs-Python.ipynb>

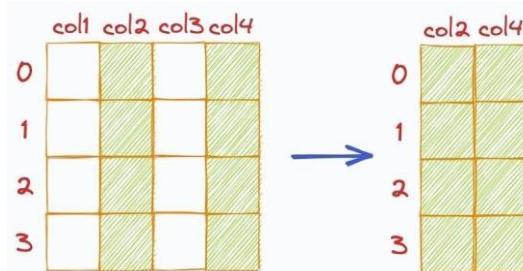
Pandas هل أنت متأكد من أنك تستخدم مصطلحات الصحيحة؟ Are You Sure You Are Using The Correct Pandas Terminologies



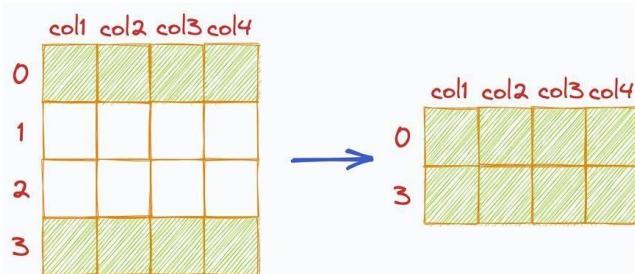
يستخدم العديد من مستخدمي Pandas المصطلحات الجزئية لإطار البيانات Dataframe subsetting بشكل غير صحيح. لذلك دعونا نقضي دقيقة لتصحيح الأمر terminologies.

SUBSETTING تعني استخراج القيمة (القيم) من إطار البيانات. يمكن القيام بذلك بأربع طرق:

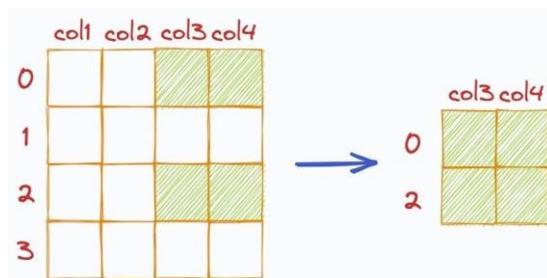
- 1) نسميها **SELECTING** عندما نستخرج واحداً أو أكثر من بناءً على موقع الفهرس index أو الاسم name. الإخراج يحتوي على بعض الأعمدة **COLUMNS** وجميع الصفوف.



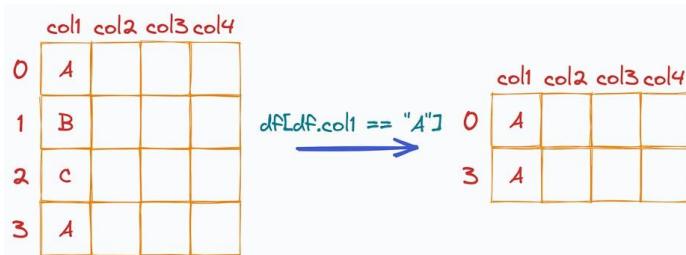
2) نسميها **SLICING** عندما نستخرج واحداً أو أكثر من بناءً على موقع الفهرس أو الاسم. يحتوي الإخراج على بعض الصفوف **ROWS** وجميع الأعمدة.



3) نسميها **INDEXING** عندما نستخرج كلاً من وبناءً على موقع الفهرس أو الاسم.



.4) نسميها **FILTERING** عندما نستخرج **ROWS** وبناءً على الشروط **conditions**



بالطبع، هناك العديد من الطرق الأخرى التي يمكنك من خلالها إجراء هذه العمليات الأربع.

إليك دليل الشامل الذي أعددته مرة واحدة: [خريطة Pandas](#). يرجى الرجوع إلى فرع "مجموعة الفرعية" لقراءة طرق subsetting المختلفة (DF

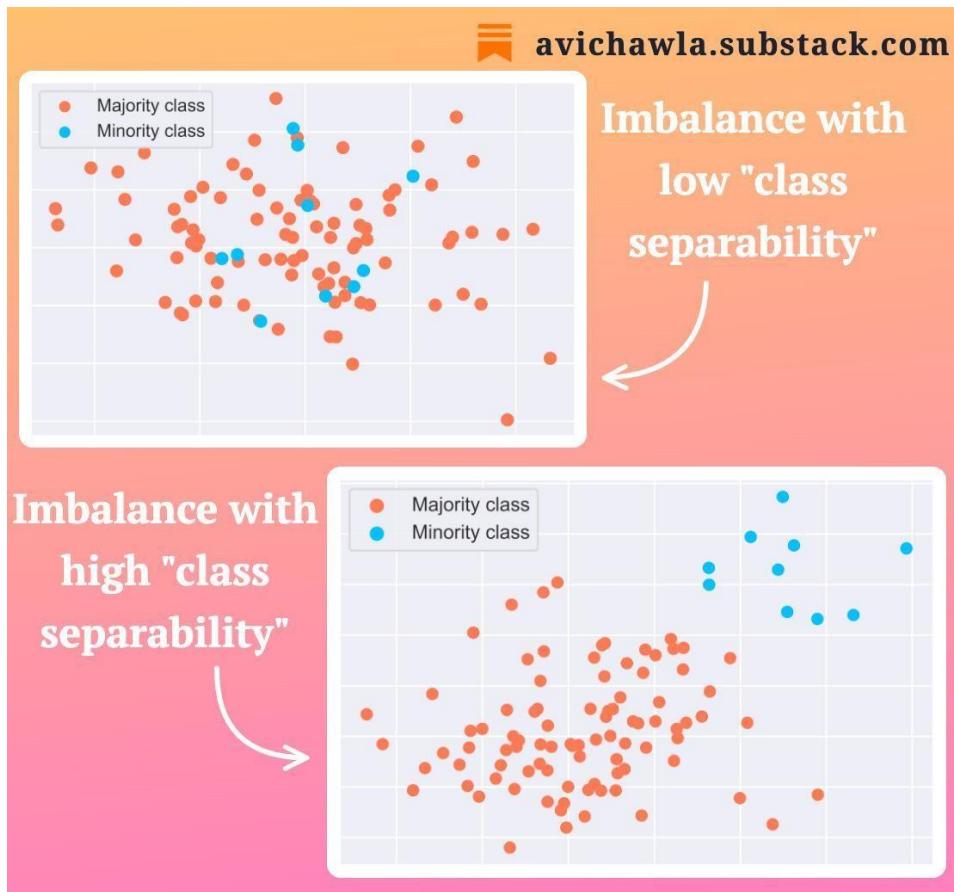
المقالة:

<https://avichawla.substack.com/p/are-you-sure-you-are-using-the-correct>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Subsetting-Terminology.ipynb>

41) هل عدم توازن الفئة دائمًا مشكلة كبيرة يجب التعامل معها؟ Is Class Imbalance Always a Big Problem to Deal With?

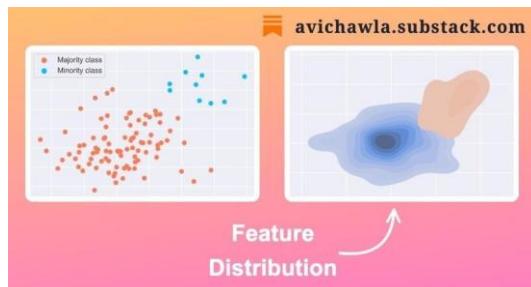


غالبًا ما تكون معالجة عدم توازن الفئة class imbalance تحديًا في التعلم الآلي. ومع ذلك، فقد لا يتسبب ذلك دائمًا في حدوث مشكلة. إليكم السبب.

أحد العوامل الرئيسية في تحديد تأثير عدم التوازن imbalance هو الفصل بين الفئات .separability

كما يوحي الاسم، فإنه يقيس الدرجة التي يمكن من خلالها التمييز بين فئتين أو أكثر أو فصلهما عن بعضهما البعض بناءً على قيم السمات feature values الخاصة بهما.

عندما تكون الفئات قابلة للفصل بدرجة كبيرة، يكون هناك القليل من التداخل بين توزيعات ميزاتها (كما هو موضح أدناه). هذا يسهل على المصنف التعرف على فئة المثال الجديد بشكل صحيح.

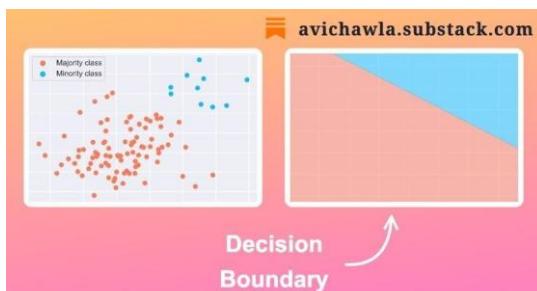


وبالتالي، على الرغم من عدم التوازن، حتى لو كانت بياناتك تتمتع بدرجة عالية من قابلية الفصل بين الفئات، فقد لا يمثل عدم التوازن مشكلة في حد ذاتها.

في الختام، ضع في اعتبارك تقدير قابلية الفصل بين الفئات قبل القفز إلى أي خطوات نمذجة معقدة.

يمكن القيام بذلك بصرياً أو عن طريق تقييم مقاييس عدم التوازن المحددة على نماذج بسيطة.

توضّح الصورة أدناه حدود القرار decision boundary التي تم تعلمها من خلال نموذج الانحدار اللوجستي logistic regression على مجموعة البيانات القابلة للفصل بين الفئات.



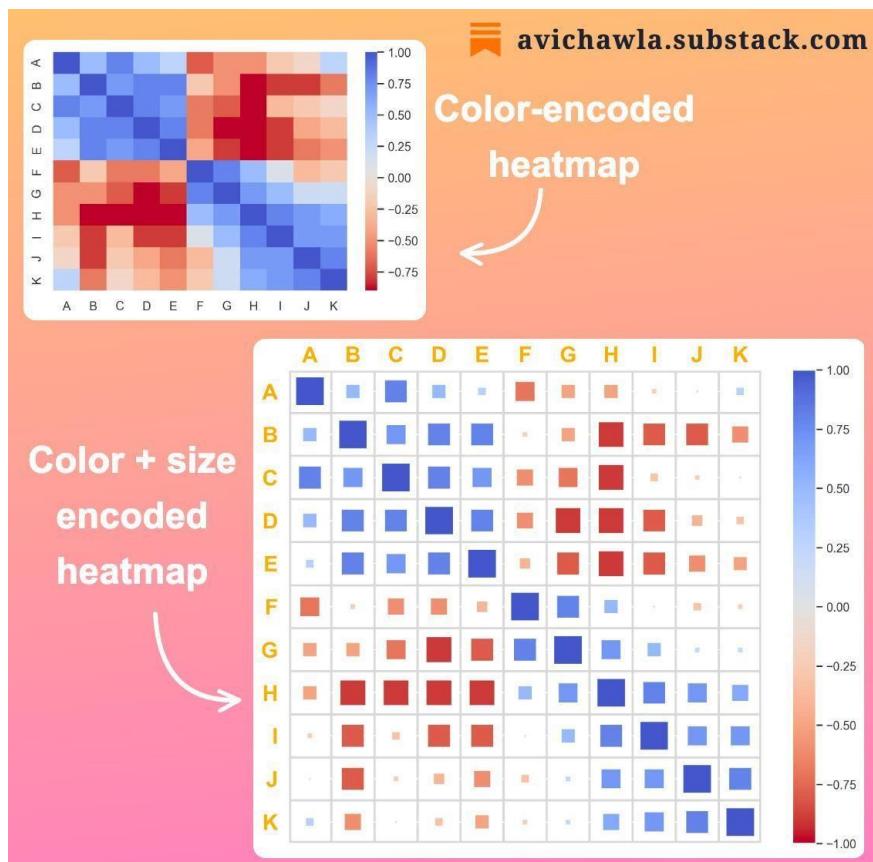
المقالة:

<https://avichawla.substack.com/p/is-class-imbalance-always-a-big-problem>

ال코드:

<https://avichawla.substack.com/p/is-class-imbalance-always-a-big-problem>

42) حيلة بسيطة تجعل الخرائط الحرارية أكثر أناقة Trick That Will Make Heatmaps More Elegant



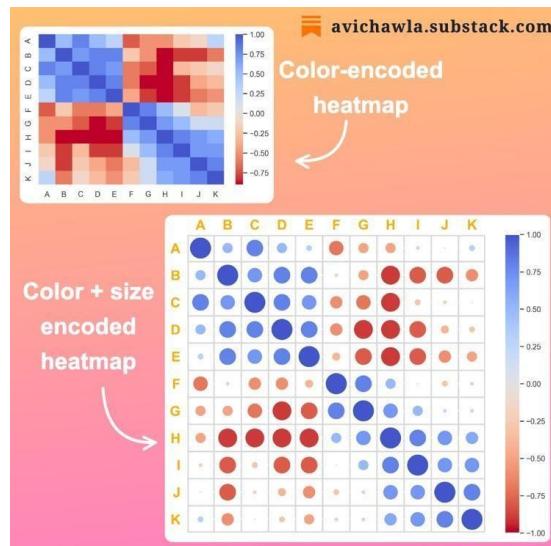
غالباً ما تجعل الخرائط الحرارية Heatmaps تحليل البيانات أسهل بكثير. ومع ذلك، يمكن إثراها بتعديل بسيط.

تمثل خريطة الحرارة التقليدية القيم باستخدام مقاييس اللون color scale. ومع ذلك، فإن تعين لون الخلية للأرقام لا يزال يمثل تحدياً.

يمكن أن يكون تضمين مكون الحجم مفيداً للغاية في مثل هذه الحالات. في جوهرها، كلما زاد الحجم، زادت القيمة المطلقة.

هذا مفيد بشكل خاص لجعل الخرائط الحرارية أكثر نظافة، حيث ستقلص العديد من القيم القريبة من الصفر على الفور.

في الواقع، يمكنك تمثيل الحجم بأي شكل آخر. أدناه، قمت بإنشاء نفس الخريطة الحرارية باستخدام دائرة بدلاً من ذلك:



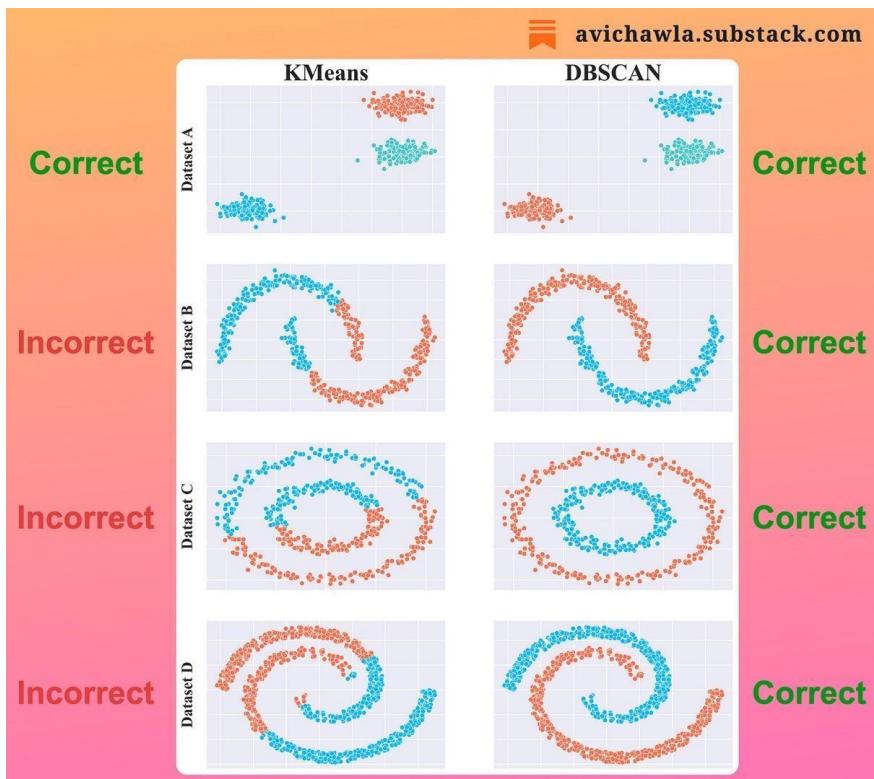
المقالة:

<https://avichawla.substack.com/p/a-simple-trick-that-will-make-heatmaps>

الكود:

<https://avichawla.substack.com/p/a-simple-trick-that-will-make-heatmaps>

(43) مقارنة مرئية بين المجاميع المحلية والقائمة على الكثافة A Visual Comparison Between Locality and Density-based Clustering



.spherical clusters KMeans على مجموعات البيانات ذات المجموعات الكروية وبالنالي، من المرجح أن ينتج عن أي تباين تجميع غير صحيح.

يمكن أن تكون خوارزميات التجميع المستندة إلى الكثافة، مثل Density-based clustering DBSCAN، بدلاً أفضلاً في مثل هذه الحالات.

يقومون بتجميع نقاط البيانات بناءً على الكثافة، مما يجعلها قوية لمجموعات البيانات ذات الأشكال والأحجام المختلفة.

توضح الصورة مقارنة KMeans مقابل DBSCAN في مجموعات بيانات متعددة.

كما هو موضح، يعمل KMeans بشكل جيد فقط عندما تحتوي مجموعة البيانات على مجموعات كروية. لكن في جميع الحالات الأخرى، فشل في إنتاج مجموعات صحيحة.

يمكنك العثور على المزيد هنا: [دليل Sklearn](#).

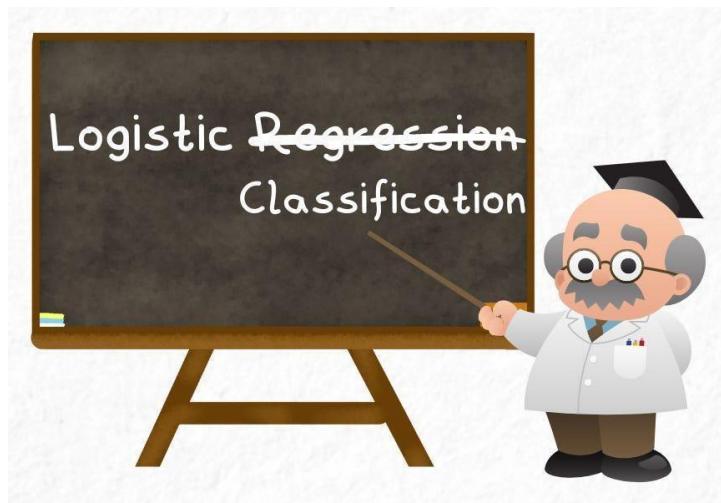
المقالة:

<https://avichawla.substack.com/p/a-visual-comparison-between-locality>

الكتاب:

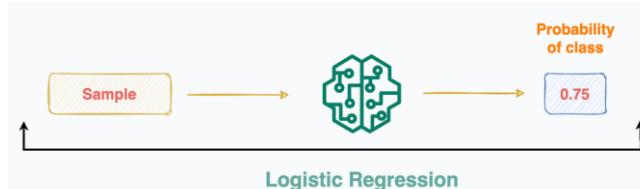
[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Machine%20Learning/KMeans-vs-DBSCAN.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/KMeans-vs-DBSCAN.ipynb)

44) لماذا لا نطلق عليه التصنيف اللوجستي بدلاً من ذلك؟ ?Why Don't We Call It Logistic Classification Instead



هل تسأعلت يوماً لماذا يسمى الانحدار اللوجستي "logistic regression" عندما نستخدمه فقط لمهام التصنيف "classification"؟ لماذا لا نسميه "تصنيف لوجستي classification" عوضاً عن ذلك؟ إليكم السبب.

يفسر معظمنا الانحدار اللوجستي على أنه خوارزمية تصنّف. ومع ذلك، فهي خوارزمية انحدار بطيئتها. هذا لأنه يتبع بنتيجة مستمرة، وهو احتمال وجود فئة class.



فقط عندما نطبق تلك العتبات thresholds ونغير تفسير ناتجها يصبح خط الأنابيب بأكمله فئة.



ومع ذلك، فمن الناحية الجوهرية، لا تقوم الخوارزمية أبداً بالتصنيف. تلتزم الخوارزمية دائمًا بالانحدار. وبدلًا من ذلك، فإن تلك الخطوة الإضافية لتطبيق عتبات الاحتمال probability thresholds هي التي تصنف العينة.

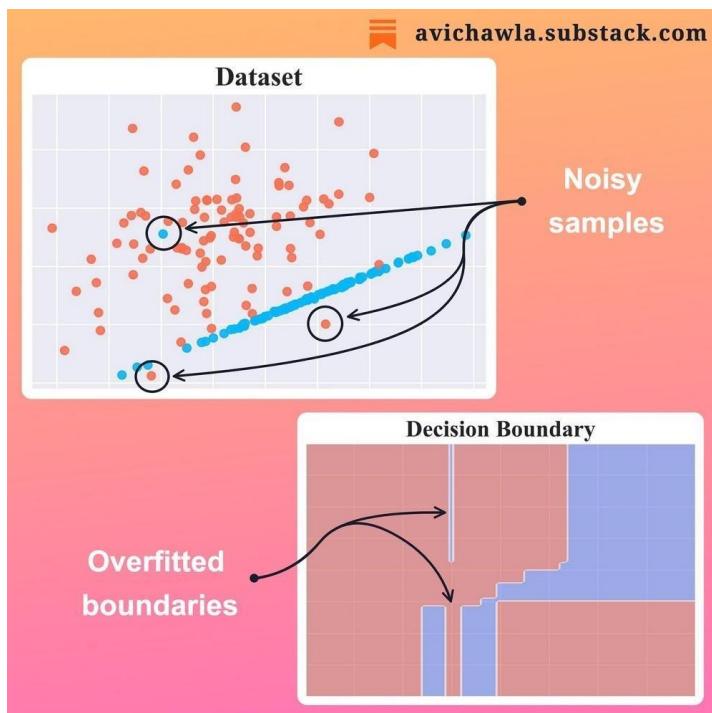
المقالة:

<https://avichawla.substack.com/p/why-dont-we-call-it-logistic-classification>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Why-logistic-reg-is-reg.ipynb>

45) شيء نموذجي حول أشجار القرار يتجاهله كثيرون في كثير من الأحيان A Typical Thing About Decision Trees Which Many Often Ignore



على الرغم من أن أشجار القرار decision trees بسيطة وبديهية، إلا أنها تحتاج دائمًا إلى مزيد من الحذر. إليك ما يجب أن تذكره دائمًا أثناء تدريبهم.

في تنفيذ sklearn، بشكل افتراضي، يُسمح لشجرة القرار بالنمو حتى تصبح جميع الأوراق نقية. يؤدي هذا إلى الضبط الزائد overfitting حيث يحاول النموذج تصنيف كل عينة في مجموعة التدريب.

هناك تقنيات مختلفة لتجنب ذلك، مثل التقليل pruning والتجميع ensembling. تأكد أيضًا من ضبط المعلمات الفائقة hyperparameters إذا كنت تستخدم تنفيذات sklearn.

كان هذا تذكيرًا لطيفًا لأن الكثير منا يميل غالباً إلى استخدام تنفيذات sklearn في تكوينها الافتراضي. من الممارسات الجيدة دائمًا معرفة ما يخفيه التنفيذ الافتراضي تحته.

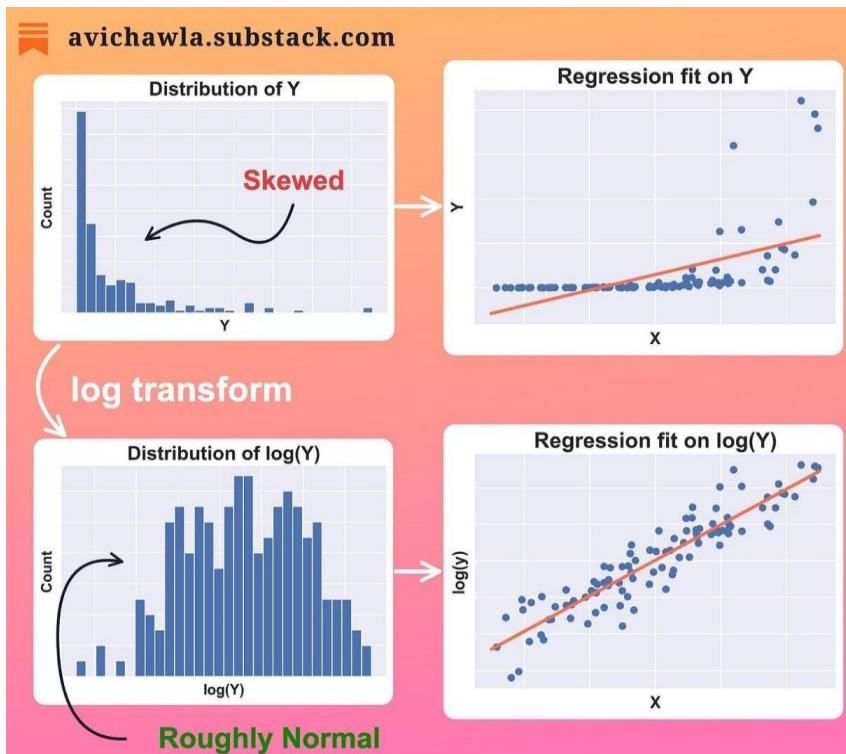
المقالة:

<https://avichawla.substack.com/p/a-typical-thing-about-decision-trees>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Use-Decision-Tree-With-Caution.ipynb>

(46) تحقق دائمًا من متغير الإخراج قبل استخدام الانحدار الخطى Always Validate Your Output Variable Before Using Linear Regression



تعتمد فعالية نموذج الانحدار الخطى linear regression إلى حد كبير على مدى توافق بياناتنا مع الافتراضات الأساسية ل الخوارزمية.

يفترض الانحدار الخطى أن القيم المتبقية residuals (التنبؤ الفعلى actual-prediction) تتبع التوزيع الطبيعي normal distribution. إحدى الطرق التي قد ينتهي بها هذا الافتراض هي عندما يكون ناتجك منحرفاً skewed.

نتيجة لذلك، سوف ينتج انحداراً غير صحيح.

لكن الشيء الجيد هو أنه يمكن تصحيحه. إحدى الطرق الشائعة لجعل الناتج متماثلاً قبل إجراء نموذج هو تطبيق تحويل السجل .log transform

يزيل الانحراف عن طريق توزيع البيانات بالتساوي، مما يجعلها تبدو طبيعية إلى حد ما. شيء واحد يجب ملاحظته هو أنه إذا كان الناتج يحتوي على قيم سالبة، فإن تحويل السجل سيؤدي إلى حدوث خطأ. في مثل هذه الحالات، يمكن للمرء تطبيق تحويل الترجمة translation transformation أو لـ على الإخراج، متبعاً بالسجل.

المقالة:

<https://avichawla.substack.com/p/always-validate-your-output-variable>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Machine%20Learning/Linear-Regression-Check-
Output.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Linear-Regression-Check-Output.ipynb)

A) حقيقة غير بدائية حول دوال بایثون

Counterintuitive Fact About Python Functions

```

# Define a function
>>> def my_func(): pass

# 1) Verify the type of function object
>>> type(my_func)
<class 'function'>

# 2) Add new attributes to function object
>>> my_func.my_attr = 'new_attribute'
>>> my_func.my_attr
'new_attribute'

# 3) Pass as an argument to other functions
>>> def new_func(f): pass
>>> new_func(my_func)

# 4) Access instance-level attributes/methods
>>> my_func.__name__
'my_func'
>>> my_func.__dict__
{'my_attr': 'new_attribute'}

```

كل شيء في بایثون هو كائن object تم إنشاء مثيل له من فئة class ما. يتضمن هذا أيضًا الدوال functions، لكن قبول هذه الحقيقة غالباً ما يكون مخالفًا للحدس في البداية. فيما يلي بعض الطرق للتحقق من أن دوال بایثون هي بالفعل كائنات.

ينشأ الاحتكاك friction عادةً بسبب معرفة الماء بلغات البرمجة الأخرى مثل C++ و Java، والتي تعمل بشكل مختلف جدًا.

ومع ذلك، فإن لغة بایثون هي لغة برمجة موجهة للકائنات object-oriented programming (OOP). أنت تستخدم دائمًا OOP، ربما دون أن تدرك ذلك.

المقالة:

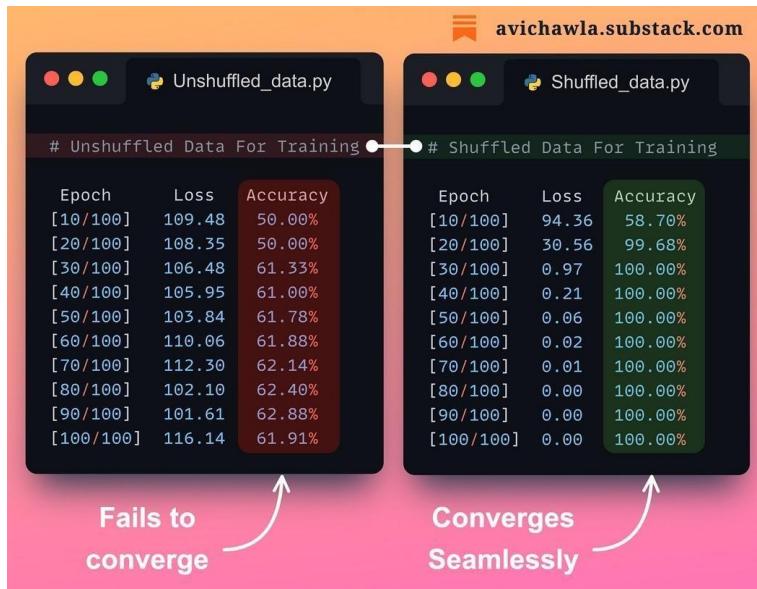
<https://avichawla.substack.com/p/a-counterintuitive-fact-about-python>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Functions-Are-Objects.ipynb>

(48) لماذا من المهم خلط مجموعة البيانات عشوائياً قبل تدريب نموذج التعلم الآلي

Why Is It Important To Shuffle Your Dataset Before Training An ML Model



قد تفشل نماذج التعلم الآلي في التقارب converge لأسباب عديدة. هنا واحد منهم غالباً ما يغفل عنه كثير من الناس.

إذا تم ترتيب بياناتك حسب التصنيفات، فقد يؤثر ذلك سلباً على تقارب النموذج ودقتة. هذا خطأ يمكن أن يمر عادة دون أن يلاحظه أحد.

في العرض أعلاه، قمت بتدريب شبكتين عصبيتين على نفس البيانات. كلتا الشبكتين لها نفس الأوزان الأولية initial weights ومعدل التعلم learning rate والإعدادات الأخرى.

ومع ذلك، في إحداها، تم ترتيب البيانات حسب التسميات (التصنيفات) labels، بينما في أخرى، تم حذفها عشوائياً.

كما هو موضح، فشل النموذج الذي يتلقى مجموعة بيانات مرتبة حسب التسمية label-ordered في التقارب. ومع ذلك، فإن عرض مجموعة البيانات يسمح للشبكة بالتعلم من عينة بيانات أكثر تمثيلاً في كل دفعه batch. هذا يؤدي إلى تعميم وأداء أفضل.

بشكل عام، يُعد اختيار مجموعة البيانات قبل التدريب ممارسة جيدة. هذا يمنع النموذج من تحديد أي أنماط تسمية محددة غير موجودة حتى الآن.

في الواقع، يوصى أيضًا بتغيير البيانات الخاصة بالدفعات في كل فترة epoch.

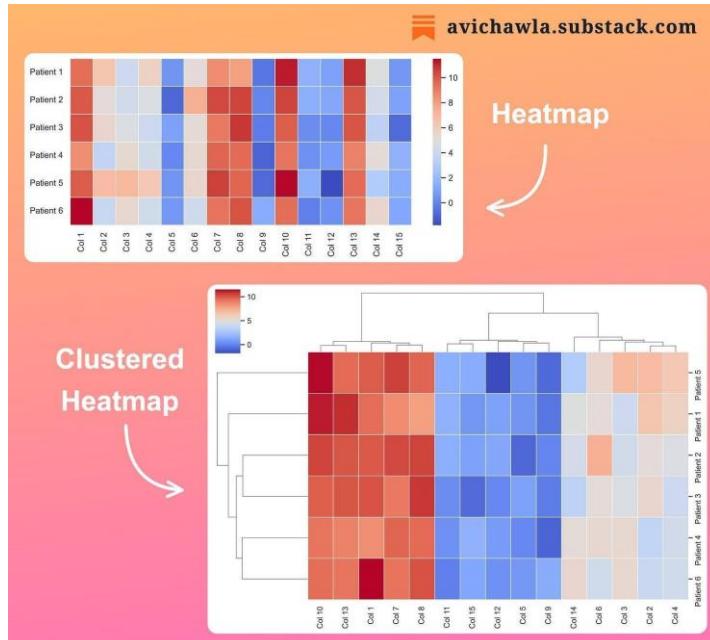
المقالة:

<https://avichawla.substack.com/p/why-is-it-important-to-shuffle-your-data>

الكتاب:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Why-Shuffle-Data.ipynb>

49) قيود الخريطة الحرارية التي تبطئ تحليل بياناتك Limitations Of Heatmap That Are Slowing Down Your Data Analysis



غالباً ما تجعل الخرائط الحرارية Heatmaps تحليل البيانات أسهل بكثير. ومع ذلك، لديهم بعض القيود.

لا تجمع خريطة الحرارة التقليدية الصنوف rows (والميزات features). بدلاً من ذلك، فإن اتجاهه هو نفس الإدخال. هذا يجعل من الصعب تحديد التشابه بين الصنوف (والميزات) بصرياً.

يمكن أن تكون الخرائط الحرارية المجمعة Clustered heatmaps خياراً أفضل في مثل هذه الحالات. يقوم بتجميع الصنوف والميزات معًا لمساعدتك على فهم البيانات بشكل أفضل.

يمكن أن تكون مفيدة بشكل خاص عند التعامل مع مجموعات البيانات الكبيرة. في حين أن خريطة الحرارة التقليدية ستكون شاقة بصرياً للنظر إليها.

ومع ذلك، فإن المجموعات الموجودة في خريطة الحرارة المجمعة تجعل من السهل تصور أوجه التشابه وتحديد الصنوف (والميزات) التي تتوافق مع بعضها البعض.

لإنشاء خريطة حرارية مجمعة، يمكنك استخدام طريقة `sns.clustermap()` من Seaborn. مزيد من المعلومات هنا: [مستنادات Seaborn](#).

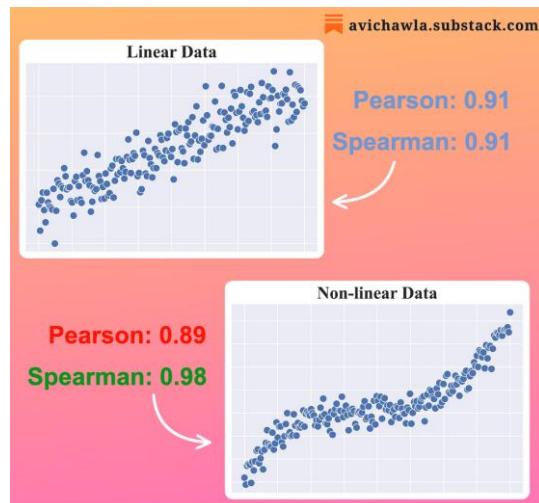
المقالة:

<https://avichawla.substack.com/p/the-limitations-of-heatmap-that-are>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Enrich-Heatmaps.ipynb>

(50) قيود ارتباط بيرسون الذي يتتجاهلها كثيرون في كثير من الأحيان The Limitation Of Pearson Correlation Which Many Often Ignore



يشيع استخدام ارتباط بيرسون Pearson correlation لتحديد الارتباط بين متغيرين مستمررين continuous variables. لكن كثيراً ما يتتجاهل الكثيرون افتراضه.

يقيس ارتباط بيرسون بشكل أساسى علاقة LINEAR بين متغيرين. نتيجة لذلك، حتى إذا كان هناك متغيرين لهما علاقة غير خطية ولكن رتيبة monotonic، فإن بيرسون يعاقب على ذلك.

أحد البدائل الرائعة هو ارتباط سبيرمان Spearman correlation. يقوم في المقام الأول بتقييم الرتبة monotonicity بين متغيرين، قد يكونان خطياً أو غير خطياً.

علاوة على ذلك، فإن ارتباط سبيرمان مفيد أيضاً في المواقف التي يتم فيها تصنيف بياناتك أو ترتيبها.

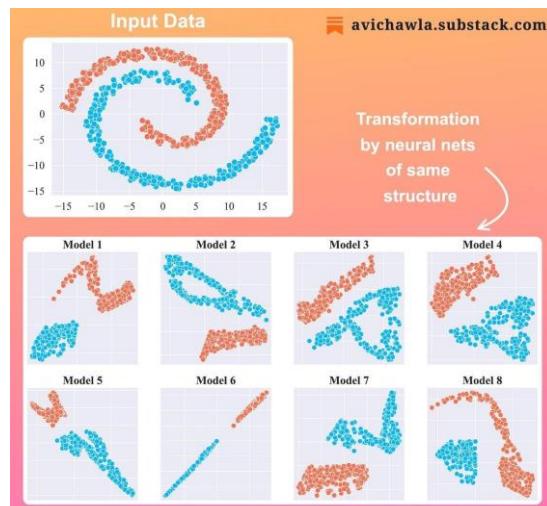
المقالة:

<https://avichawla.substack.com/p/the-limitation-of-pearson-correlation>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Statistics/Pearson-vs-Spearman.ipynb>

Why (51) لماذا يُنصح عادةً بوضع بذور للمولدات العشوائية؟ Are We Typically Advised to Set Seeds for Random Generators



من وقت لآخر، ننصحنا بتعيين البذور لأرقام عشوائية random numbers قبل تدريب نموذج التعلم الآلي. إليكم السبب.

يتم تهيئة وزن النموذج بشكل عشوائي. وبالتالي، فإن أي تجربة متكررة لا تولد أبداً نفس مجموعة الأرقام. هذا يمكن أن يعيق استنساخ النموذج الخاص بك.

كما هو موضح أعلاه، يتم تحويل بيانات الإدخال نفسها بعدة طرق بواسطة شبكات عصبية مختلفة من نفس البنية.

وبالتالي، قبل تدريب أي نموذج، تأكد دائمًا من إعداد البذور بحيث يمكن تكرار تجربتك لاحقًا.

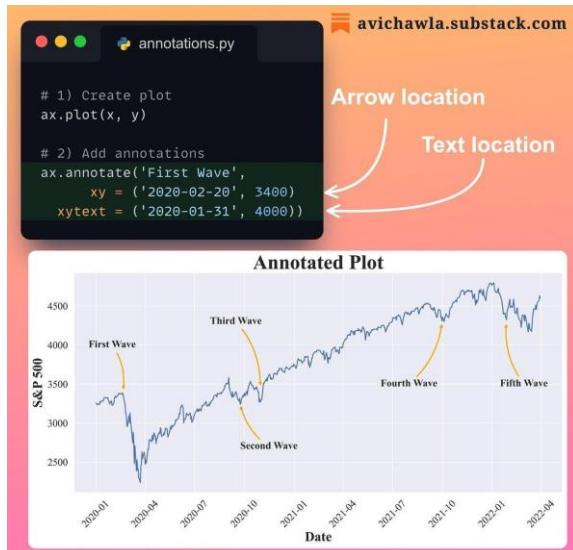
المقالة:

<https://avichawla.substack.com/p/why-are-we-typically-advised-to-set>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Why-Set-Seeds.ipynb>

52) تقنية تم الاستخفاف بها لتحسين تصورات البيانات An Underrated Technique To Improve Your Data Visualizations



في بعض الأحيان، قد يتطلب منك التأكد من أن مخططك ينقل الرسالة الصحيحة لتقديم سياق إضافي. ومع ذلك، فإن زيادة المخططات الإضافية قد تؤدي إلى تشويش تصورك بالكامل.

تمثل إحدى الطرق الرائعة لتقديم معلومات إضافية في إضافة تعليقات توضيحية نصية `text` إلى `annotations` المخطط.

في `matplotlib`، يمكنك استخدام `annotate()`. يضيف نصوصاً توضيحية إلى مخططك، مما يتيح لك توجيه انتباه المشاهد إلى مناطق محددة والمساعدة في فهمه.

يمكنك العثور على مزيد من المعلومات هنا: [مستندات Matplotlib](#)

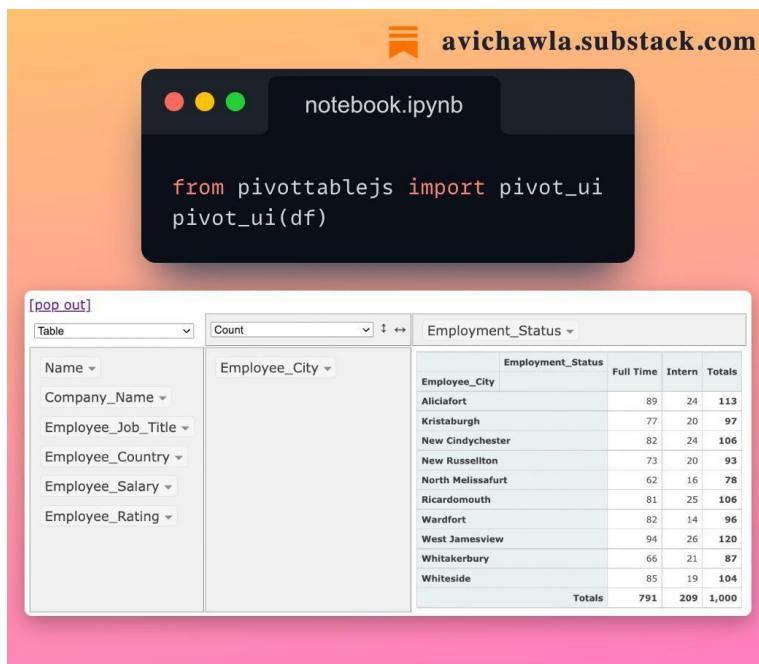
المقالة:

<https://avichawla.substack.com/p/an-underrated-technique-to-improve>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Text-annotations.ipynb>

(53) أداة بدون كود لإنشاء المخططات والجداول المحورية في A No-Code Tool to Create Charts and Pivot Tables in Jupyter



فيما يلي طريقة سريعة وسهلة لإنشاء جداول محورية charts وبيانات مجمعة pivot tables دون كتابة أي تعليمات برمجية group data.

PivotTableJS هي أداة سحب وإفلات لإنشاء جداول محورية ومحططات تفاعلية في Jupyter. علاوة على ذلك، يمكنك أيضًا زيادة الجداول المحورية باستخدام الخرائط الحرارية heatmaps لتحليل محسّن.

يمكنك العثور على مزيد من المعلومات هنا: [PivotTableJS](#)

شاهد نسخة فيديو من هذا المنشور لفهم أفضل: [الفيديو](#).

If (54) إذا لم تكن قادرًا على برمجة نهج موجه، فجرّب هذا.
You Are Not Able to Code a Vectorized Approach, .Try This

Method	Time (ms ± ms per loop)	Speed Rating
1) iterrows()	2.63 s ± 7.55 ms per loop	Slowest
2) apply()	923 ms ± 6 ms per loop	Slow
3) itertuples()	87.3 ms ± 486 µs per loop	Fast
4) to_numpy()	32.9 ms ± 240 µs per loop	Fastest

على الرغم من أنه لا ينبغي لنا أبدًا التكرار عبر إطار بيانات ونفضل الكود المتجه vectorized code، ماذا لو لم نتمكن من التوصل إلى حل متجه vectorized solution؟

في منشورى السابق حول سبب تكلفة تكرار إطار البيانات، طرح شخص ما سؤالاً حقيقياً جداً. سألوا: "دعنا نقول فقط أنك مجبر على التكرار. ما هي أفضل طريقة للقيام بذلك؟"

أولاً، افهم أن السبب الرئيسي وراء بطء التكرار يرجع إلى الطريقة التي يتم بها تخزين إطار البيانات في الذاكرة. (إذا كنت ترغب في تلخيص هذا، فاقرأ منشورى السابق [هنا](#)).

نظرًا لكونه هيكل بيانات ذات عمود رئيسي، فإن استرداد صفوفه يتطلب الوصول إلى كتل غير متغيرة من الذاكرة. هذا يزيد من وقت التشغيل بشكل كبير.

ومع ذلك، إذا كنت ترغب في إجراء عمليات تعتمد على الصحف فقط، فإن الحل السريع هو تحويل إطار البيانات إلى مصفوفة NumPy.

أسرع هنا لأنَّه ، افتراضيًّا، يخزن البيانات بطريقة رئيسية. وبالتالي، يتم استرداد صفوفها عن طريق الوصول إلى كتل متغيرة من الذاكرة، مما يجعلها فعالة في تكرار إطار البيانات.

ومع ذلك، لاحظ أن أفضل طريقة هي كتابة التعليمات البرمجية الموجهة دائمًا. استخدم منهج Pandas-to-NumPy vectorized code .

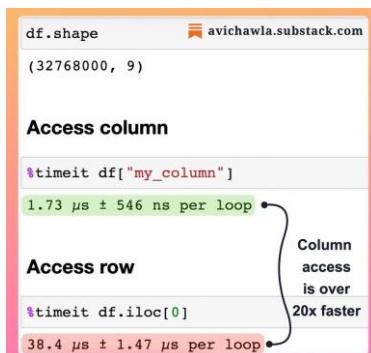
المقالة:

<https://avichawla.substack.com/p/if-you-are-not-able-to-code-a-vectorized>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Best-Way-to-Iterate-DataFrame.ipynb>

55) لماذا يُنصح عادةً بعدم التكرار مطلقاً عبر إطار بيانات؟ Why Are We Typically Advised To Never Iterate Over ?A DataFrame



من وقت لآخر، يُنصح بتجنب التكرار iterating على Pandas DataFrame. لكن ما هو السبب الدقيق وراء ذلك؟ دعني أشرح.

يعد DataFrame بُنية بيانات ذات عمود رئيسي. وبالتالي، يتم تخزين العناصر الممتالية في عمود بجانب بعضها البعض في الذاكرة.

نظرًا لأن المعالجات فعالة مع كتل متغيرة من الذاكرة، فإن استرداد عمود يكون أسرع بكثير من صفح واحد.

ولكن أثناء التكرار، حيث يتم استرداد كل صفح عن طريق الوصول إلى كتل غير متغيرة من الذاكرة، يزداد وقت التشغيل بشكل كبير.

في الصورة أعلاه، كان استرداد أكثر من 32 مليون عنصر من العمود أسرع 20 مرة من جلب تسعه عناصر مخزنة في صفح واحد فقط.

المقالة:

<https://avichawla.substack.com/p/why-are-we-typically-advised-to-never>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Run-time%20Optimization/Never-Iterate-Over-Pandas.ipynb>

56) قد يكون التلاعب بالكائنات القابلة للتغيير في لغة بايثون Manipulating Mutable Objects In Python Can Get Confusing At Times

```

Method1.py
1 # 1) Define list
2 >>> a = [1,2,3]
3
4 # 2) Assign b to a
5 >>> b = a
6
7 # 3) Modify a
8 >>> a = a + [4,5]
9
10 # 4) Print a
11 >>> a
12 [1, 2, 3, 4, 5] # Modified
13
14 # 5) Print b
15 >>> b
16 [1, 2, 3] # Unchanged

Method2.py
1 # 1) Define list
2 >>> a = [1,2,3]
3
4 # 2) Assign b to a
5 >>> b = a
6
7 # 3) Modify a
8 >>> a += [4,5]
9
10 # 4) Print a
11 >>> a
12 [1, 2, 3, 4, 5] # Modified
13
14 # 5) Print b
15 >>> b
16 [1, 2, 3, 4, 5] # Modified

```

هل تعلم أنه مع الكائنات القابلة للتغيير mutable objects، تعمل “`a +=`” and “`a = a +`” بشكل مختلف في بايثون؟ إليكم السبب.

لتفكير في قائمة list، على سبيل المثال.

عندما نستخدم عامل التشغيل =، تنشئ بايثون كائناً جديداً في الذاكرة وتخصصه للمتغير.

وبالتالي، لا تزال جميع المتغيرات الأخرى تشير إلى موقع الذاكرة السابق، والذي لم يتم تحديده مطلقاً. هذا موضح في `Method1.py` أعلاه.

ولكن مع عامل التشغيل =+، يتم فرض التغييرات في مكانها. هذا يعني أن بايثون لا تنشئ كائناً جديداً ويتم تحديث موقع الذاكرة نفسه.

وبالتالي، يمكن رؤية التغييرات من خلال جميع المتغيرات الأخرى التي تشير إلى نفس الموقع. هذا موضح في `Method2.py` أعلاه.

يمكننا أيضًا التحقق من ذلك من خلال مقارنة `id` التعين المسبق pre-assignment والتعيين اللاحق post-assignment.

```

avichawla.substack.com

Method1.py
1 # 1) Check ID
2 >>> id(a), id(b)
3 (12345, 12345)
4
5 # 2) Modify a
6 >>> a = a + [4,5]
7
8 # 3) Check ID
9 >>> id(a), id(b)
10 (98765, 12345)

Method2.py
1 # 1) Check ID
2 >>> id(a), id(b)
3 (12345, 12345)
4
5 # 2) Modify a
6 >>> a += [4,5]
7
8 # 3) Check ID
9 >>> id(a), id(b)
10 (12345, 12345)

```

باستخدام "`a = a +`"، يتم تغيير `id`، مما يشير إلى أن Python قد أنشأت كائناً جديداً. ومع ذلك، مع "`a +=`"، تظل `id` كما هي. يشير هذا إلى أنه تم تحديث نفس موقع الذاكرة.

المقالة:

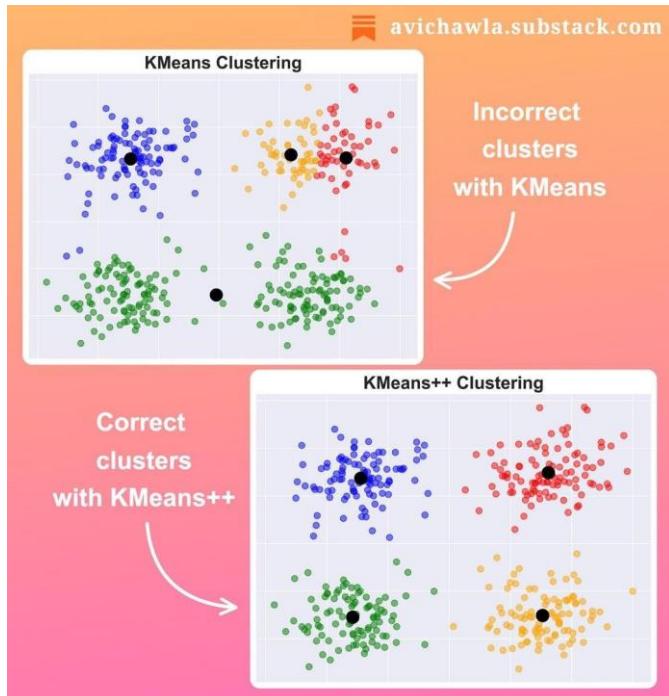
<https://avichawla.substack.com/p/manipulating-mutable-objects-in-python>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Modifying-Mutable-Object.ipynb>

(57) يمكن لهذا التعديل الصغير أن يعزز بشكل كبير من وقت تشغيل KMeans

This Small Tweak Can Significantly Boost The Run-time of KMeans



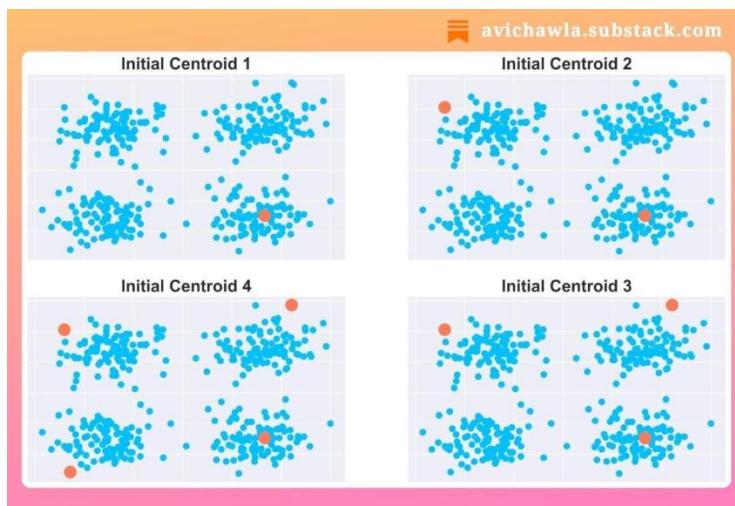
KMeans عبارة عن خوارزمية تجميعية شائعة clustering algorithm ولكنها عالية التشغيل. إليك كيف يمكن لتعديل صغير تحسين وقت تشغيله بشكل ملحوظ.

يختر KMeans النقط الوسطى centroids الأولية بشكل عشوائي. نتيجة لذلك، فشل في التقارب converge في بعض الأحيان. هنا يتطلب منا تكرار التجميع عدة مرات مع تهيئة مختلفة.

بدلاً من ذلك، يتخذ KMeans++ نهجًا أكثر ذكاءً لتهيئة النقط الوسطى. يتم اختيار النقطة الوسطى الأولى عشوائياً. ولكن يتم اختيار النقطة الوسطى التالية على أساس المسافة من النقطة الوسطى الأولى.

عبارة أخرى، من المرجح أن يتم تحديد النقطة البعيدة عن النقطة الوسطى الأولى كنقطة مركزية أولية. بهذه الطريقة، من المحتمل أن تقع جميع النقط الوسطى الأولية في مجموعات مختلفة بالفعل وقد تقارب الخوارزمية بشكل أسرع.

يوضح الرسم التوضيحي أدناه تهيئة النقطة الوسطى لـ KMeans++



المقالة:

<https://avichawla.substack.com/p/this-small-tweak-can-significantly>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Machine%20Learning/Boost-KMeans-Run-time.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Boost-KMeans-Run-time.ipynb)

(58) معظم مبرمجي بايثون لا يعرفون هذا عن البرمجة كائنية Most Python Programmers Don't Know This About Python OOP

```

class Point2D:
    def __new__(cls, x, y):
        if isinstance(x, int) and isinstance(y, int):
            # Allocate memory and return a new object
            # only when the if-condition is True
            print("Creating Object!")
            return super().__new__(cls) # Return new object
        else:
            raise TypeError("x and y must be integers")

    def __init__(self, x, y):
        self.x = x
        self.y = y
        print("Object Initialized!")

>>> p1 = Point2D(1,2)
"Creating Object!"      # from __new__() method
"Object Initialized!" # from __init__() method

>>> p2 = Point2D(1.5, 2.5)
TypeError: x and y must be integers

```

يسيء معظم مبرمجي بايثون فهم طريقة `()__init__()`. يعتقدون أنه يخلق كائناً جديداً. لكن ذلك غير صحيح. عندما نقوم بإنشاء كائن، فإن طريقة `()__init__()` ليست هي التي تخصص الذاكرة له. كما يوحي الاسم، تقوم `()__init__()` بتعيين قيمة لسمات الكائن فقط.

بدلاً من ذلك، تستدعي بايثون الطريقة `()__new__()` أولاً لإنشاء كائن جديد وتخصيص الذاكرة له. ولكن ما مدى فائدة ذلك، قد تتساءل؟ هناك العديد من الأسباب.

على سبيل المثال، من خلال تنفيذ طريقة `()__new__()`، يمكنك تطبيق فحوصات البيانات `checks.data`. هذا يضمن أن برنامجك يخصص الذاكرة فقط عند استيفاء شروط معينة.

تتضمن حالات الاستخدام الشائعة الأخرى تحديد الفئات الفردية singleton classes (الفئات التي تحتوي على كائن واحد فقط)، وإنشاء فئات فرعية من الفئات غير القابلة للتغيير immutable classes مثل المجموعات، وما إلى ذلك.

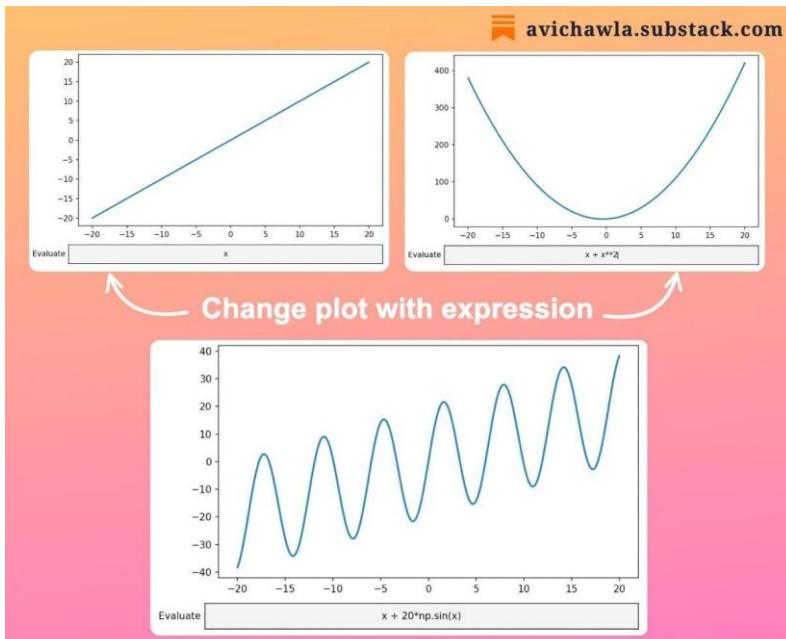
المقالة:

<https://avichawla.substack.com/p/most-python-programmers-dont-know-b55>

الكود:

https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/__init__-and__new__.ipynb

59) من قال إن Matplotlib لا يمكنه إنشاء مخططات تفاعلية؟ Who Said Matplotlib Cannot Create Interactive Plots ?Interactive Plots



يرجى مشاهدة نسخة فيديو من هذا المنشور لفهم أفضل: [رابط الفيديو](#).

في معظم الحالات، يتم استخدام Matplotlib لإنشاء مخططات ثابتة. لكن قلة قليلة من الناس يعرفون أنه يمكنه إنشاء مخططات تفاعلية أيضاً. إليك الطريقة.

بشكل افتراضي، يستخدم Matplotlib الوضع **inline** ، والذي يعرض المخططات الثابتة. ومع ذلك، باستخدام الأمر السحري **%matplotlib widget** ، يمكنك تمكين الواجهة الخلفية التفاعلية لمخططات Matplotlib.

علاوة على ذلك، تحتوي الوحدة النمطية **widgets** على العديد من عناصر واجهة المستخدم المفيدة. يمكنك دمجها مع المخططات الخاصة بك لجعلها أكثر أناقة.

اعثر على دليل مفصل هنا: [أدوات Matplotlib](#)

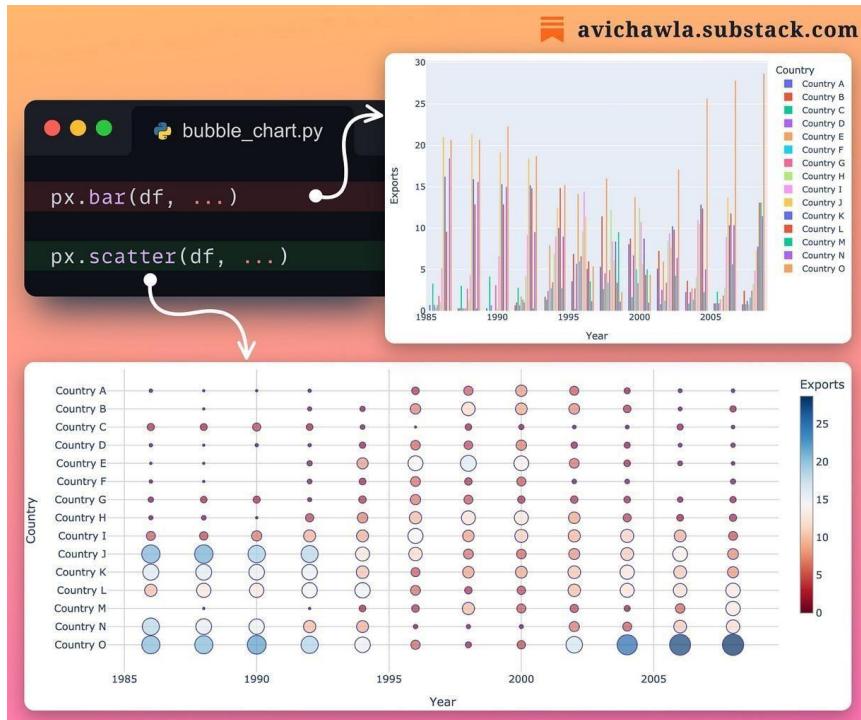
المقالة:

<https://avichawla.substack.com/p/who-said-matplotlib-cannot-create>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Interactive-Matplotlib.ipynb>

60) لا تقم بإنشاء مخططات شريطية فوضوية. بدلاً من ذلك، Don't Create Messy Bar Plots. جرب المخططات الفقاعية! Instead, Try Bubble Charts



غالباً ما تصبح المخططات الشريطية Bar plots غير مفهومة وفوضوية عندما يكون لدينا العديد من الفئات للرسم.

يمكن أن يكون المخطط الفقاعي bubble chart خياراً أفضل في مثل هذه الحالات. إنها مثل مخططات التشتت scatter plots ولكن مع محور فئوي categorical واحد وأخر مستمر continuous.

بالمقارنة مع المخطط الشريطي، فهي أقل تشوشاً ولديها فهم أفضل.

بالطبع، يعتمد اختيار المخطط في نهاية المطاف على طبيعة البيانات والأفكار المحددة التي ترغب في نقلها.

ما هو المخطط الذي تفضله عادةً في مثل هذه المواقف؟

المقالة:

<https://avichawla.substack.com/p/dont-create-messy-bar-plots-instead>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/Bubble-Charts.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Bubble-Charts.ipynb)

You Can إضافة قائمة كمفتاح قاموس (فنياً)! !Add a List As a Dictionary's Key (Technically)

```

avichawla.substack.com

>>> my_dict = {} ## dict
>>> my_list = [1,2,3] ## list
>>> my_dict[my_list] = True
TypeError: unhashable type: 'list'

hashable list
                ↗
                ↘
unhashable list

```



```

## Inherit list class and implement __hash__ func
class MyList(list):
    def __hash__(self):
        return 0

>>> my_list = MyList([1,2,3])
>>> my_dict[my_list] = True
>>> print(my_dict)
{[1, 2, 3]: True}

```

تثير بايثون خطأً عندما نضيف قائمة list كمفتاح قاموس dictionary's key. لكن هل تعرف السبب الفني وراء ذلك؟ ها أنت ذا.

أولاً، أفهم أن كل شيء في بايثون هو كائن تم إنشاء مثيل له من فئة ما. عندما نضيف كائناً كمفتاح قاموس، تستدعي بايثون الدالة **hash** لفئة ذلك الكائن.

بينما تطبق فئات int و str و tuple و frozenset و hash، فإنها مفقودة من فئة القائمة. هذا هو السبب في أننا لا نستطيع إضافة قائمة كمفتاح القاموس.

وبالتالي، من الناحية الفنية، إذا قمنا بتوسيع فئة القائمة وإضافة هذه الطريقة، يمكن إضافة قائمة كمفتاح القاموس.

على الرغم من أن هذا يجعل القائمة قابلة للتجزئة hashable، إلا أنه لا يوصى بها لأنه قد يؤدي إلى سلوك غير متوقع في التعليمات البرمجية الخاصة بك.

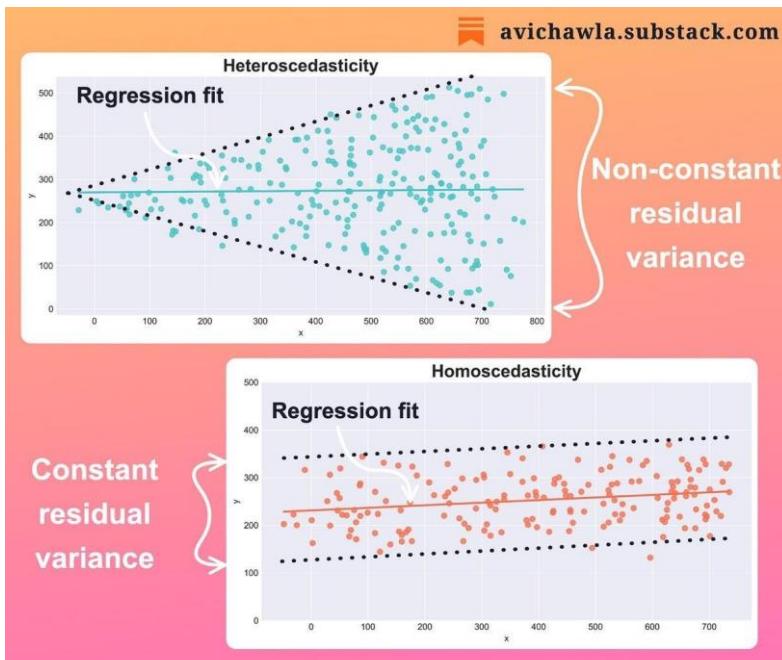
المقالة:

<https://avichawla.substack.com/p/you-can-add-a-list-as-a-dictionaries>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Python-List-As-Dict-Key.ipynb>

62) غالباً ما يهمل معظم مستخدمي التعلم الآلي هذا أثناء استخدام الانحدار الخطى Most ML Folks Often Neglect This While Using Linear Regression



يتم تحديد فعالية نموذج الانحدار الخطى linear regression بمدى توافق البيانات مع الافتراضات الأساسية للخوارزمية.

من الافتراضات المهمة للغاية، والتي غالباً ما يتم تجاهلها للانحدار الخطى، التجانس homoscedasticity.

تعتبر مجموعة البيانات متتجانسة homoscedastic إذا ظل تباين القيم المتبقية (= المتوقع الفعلى) كما هو عبر نطاق الإدخال.

في المقابل، تكون مجموعة البيانات غير متتجانسة heteroscedastic إذا كانت البقايا لها تباين غير ثابت.

تعتبر Homoscedasticity في غاية الأهمية للانحدار الخطى. هذا لأنه يضمن أن معاملات الانحدار لدينا موثوقة. علاوة على ذلك، يمكننا أن نثق في أن التنبؤات ستبقى دائمًا في نفس فترة الثقة confidence interval.

المقالة:

<https://avichawla.substack.com/p/most-ml-folks-often-neglect-this>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Machine%20Learning/Linear-Regression-
Assumption.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Linear-Regression-Assumption.ipynb)

Hidden 35 (63) مكتبة بايثون مخفية تعتبر جواهر مطلقة

Python Libraries That Are Absolute Gems



لقد راجعت أكثر من 1000 مكتبة بايثون واكتشفت هذه الجواهير الخفية التي لم أكن أعرف بوجودها من قبل.

إليك بعضًا منها ستجعلك تقع في حب بايثون وتعدد استخداماتها (حتى أكثر).

اقرأ هذه القائمة الكاملة هنا:

<https://avichawla.substack.com/p/35-gem-py-libs>

المقالة:

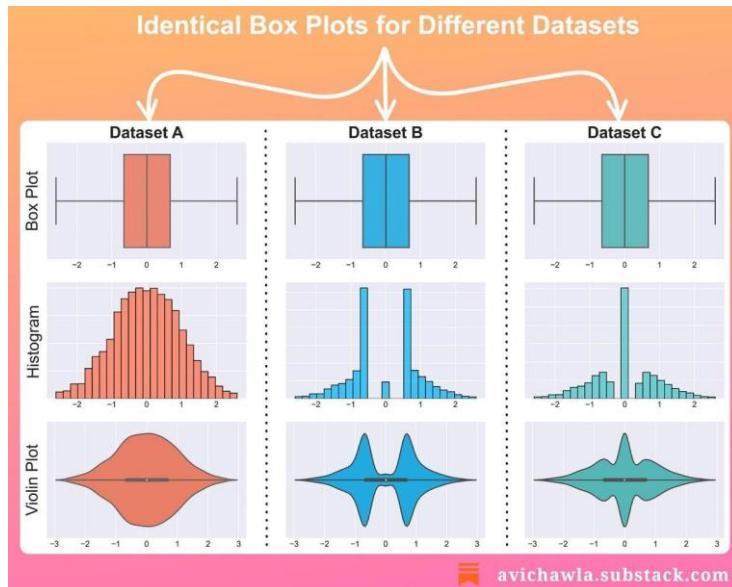
<https://avichawla.substack.com/p/35-gem-py-libs>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Documents/35-cool-libs.ipynb>

64) استخدم المخططات الصندوقية بحذر! قد تكون مضللة.

Use Box Plots With Caution! They May Be Misleading



المخططات الصندوقية Box plots شائعة جدًا في تحليل البيانات. لكنها يمكن أن تكون مضللة في بعض الأحيان. إليكم السبب.

المخطط الصندوقي هو تمثيل رسومي لخمسة أرقام فقط – دقة، وربع أول، ومتوسط، وربع ثالث، وأقصى حد.

وبالتالي، فإن مجموعتي بيانات مختلفتين لهما قيمة خمساوية متشابهة ستنتجان مخططات صندوقية متطابقة. هنا، في بعض الأحيان، يمكن أن يكون مضللاً ويمكن للمرء أن يستخلص استنتاجات خاطئة.

القصد ليس أنه لا ينبغي استخدام المخططات الصندوقية. بدلاً من ذلك، انظر إلى التوزيع الأساسي أيضًا. هنا، يمكن أن تساعد المدرجات التكرارية histograms ومخططات الكمان violin plots.

أخيرًا، تذكر دائمًا أنه عندما تقوم بتكييف مجموعة البيانات، فإنك لا ترى الصورة كاملة. أنت تفقد المعلومات الأساسية.

المقالة:

<https://avichawla.substack.com/p/use-box-plots-with-caution-they-may>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/Box-Plots-Are-Misleading.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Box-Plots-Are-Misleading.ipynb)

65) تقنية تم الاستخفاف بها لإنشاء مخططات بيانات أفضل An Underrated Technique To Create Better Data Plots



أثناء إنشاء التصورات visualizations، غالباً ما تكون هناك أجزاء معينة ذات أهمية خاصة. ومع ذلك، قد لا تكون واضحة للمشاهد على الفور.

سيضمن راوي البيانات الجيد دائمًا أن المخطط يوجه انتباه المشاهد إلى هذه المجالات الرئيسية. تمثل إحدى الطرق الرائعة في تكبير مناطق محددة ذات أهمية في المخطط. هذا يضمن أن مخططنا تنقل بالفعل ما نعتزم تصويره.

في matplotlib، يمكنك القيام بذلك باستخدام `indicate_inset_zoom()`. يضيف مربع مؤشر يمكن تكبيره لتحسين الاتصال.

يمكنك العثور على مزيد من المعلومات هنا: [مستندات Matplotlib](#)

المقالة:

<https://avichawla.substack.com/p/an-underrated-technique-to-create>

ال kod:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Better-Storytelling.ipynb>

٦٦) اضافة إطار بيانات Pandas الذي ينتظرها كل عالم بيانات The Pandas DataFrame Extension Every Data Scientist Has Been Waiting For



شاهد نسخة فيديو من هذا المنشور لفهم أفضل: [رابط الفيديو](#).

PyGWalker هو بدائل مفتوح المصدر لـ Tableau يحول إطار بيانات Pandas إلى واجهة مستخدم على غرار لوحة لاستكشاف البيانات.

يوفر واجهة مستخدم تشبه Tableau في Jupyter، مما يسمح لك بتحليل البيانات بشكل أسرع وبدون كود.

يمكنك العثور على مزيد من المعلومات هنا: [PyGWalker](#)

٦٧) عزّز Shell مع بایثون باستخدام Xonsh Shell With Python Using Xonsh



الشل التقليدية Traditional shell لها قيود على مستخدمي بایثون. في وقت واحد، يمكن للمستخدمين إما تشغيل أوامر shell أو استخدام IPython.

نتيجة لذلك، يتبع على المرء أن يفتح عدة ترمinal terminals أو التبديل ذهابًا وإيابًا بينهما في نفس الجهاز.

بدلاً من ذلك، جرب Xonsh. فهو يجمع بين سهولة الشل التقليدي وقوة بایثون. وبالتالي، يمكنك استخدام بناء جملة بایثون وكذلك تشغيل أوامر shell في نفس الشل.

يمكنك العثور على مزيد من المعلومات هنا: [Xonsh](#).

المقالة:

<https://avichawla.substack.com/p/supercharge-shell-with-python-using>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/Supercharge-Shell.ipynb>

68) لا يعرف معظم مستخدمي سطر الأوامر هذه الحيلة الرائعة Most Command-line Users Don't Know This Cool Trick About Using Terminals Don't Know This Cool Trick About Using Terminals



شاهد نسخة فيديو من هذا المنشور لفهم أفضل: [رابط الفيديو](#).

بعد تشغيل أمر command (أو سكريبت script، وما إلى ذلك)، يفتح معظم مستخدمي سطر الأوامر terminal جديد لتشغيل أوامر أخرى. لكن هذا ليس مطلوباً أبداً. إليك الطريقة.

عندما نقوم بتشغيل برنامج من سطر الأوامر، فإنه يعمل بشكل افتراضي في المقدمة. هذا يعني أنه لا يمكنك استخدام terminal حتى يكتمل البرنامج.

ومع ذلك، إذا قمت بإضافة "&" في نهاية الأمر، فسيتم تشغيل البرنامج في الخلفية وتحرير terminal على الفور.

بهذه الطريقة، يمكنك استخدام نفس terminal لتشغيل أمر آخر. لإعادة البرنامج إلى المقدمة، استخدم الأمر "fg".

المقالة:

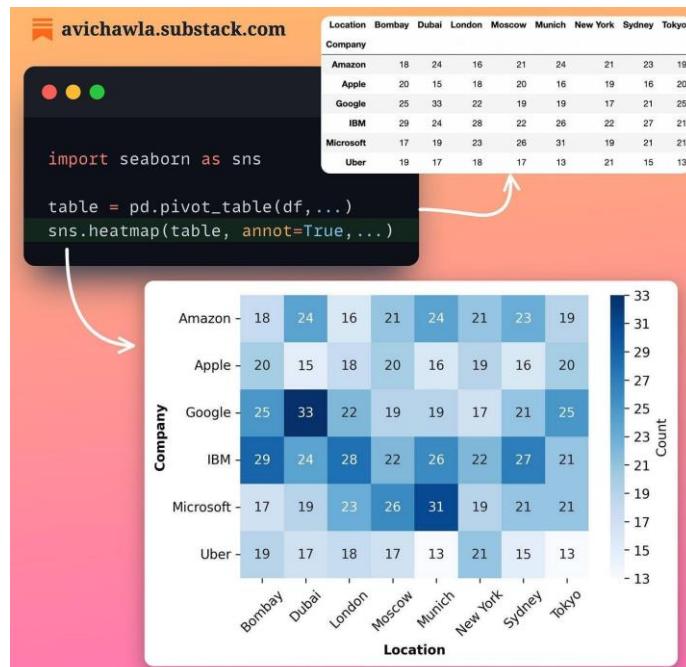
<https://avichawla.substack.com/p/most-command-line-users-dont-know>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Terminal/Free-Terminal.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Terminal/Free-Terminal.ipynb)

69) حيلة بسيطة لتحقيق أقصى استفادة من الجداول المحوりة في pandas

A Simple Trick to Make The Most Pandas Out of Pivot Tables in Pandas



تعد الجداول المحو리ّة شائعة جدًا لاستكشاف البيانات data exploration. ومع ذلك، فإن تحليل النتائج أمر شاق وصعب. علاوة على ذلك، قد يغوت المرء بعض الأفكار المهمة حول البيانات.

بدلاً من ذلك، قم بإثارة الجداول المحوّرة بخرائط الحرارة heatmaps. تسهل ترميزات الألوان color encodings تحليل البيانات وتحديد الأنماط.

المقالة:

<https://avichawla.substack.com/p/a-simple-trick-to-make-the-most-out>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Enrich-Pivot-Table.ipynb>

(70) لماذا لا تقدم بايثون تغليف برمجة كائنية التوجه حقيقي

Why Python Does Not Offer True OOP Encapsulation

```

class MyClass:
    def __init__(self):
        self.public_attr = "I'm public"      # 0 underscores
        self._protected_attr = "I'm protected" # 1 underscore
        self.__private_attr = "I'm private"    # 2 underscores

my_obj = MyClass()
>>> my_obj.public_attr
'I'm public'
>>> my_obj._protected_attr
'I'm protected'
>>> my_obj._MyClass__private_attr
'I'm private'

```

Public member accessible
Protected member accessible
Private member accessible with name mangling

يعد استخدام أدوات تعديل الوصول `public` والمحمية `protected` (العامة `public` والمحمية `protected`) أو `private` أساسياً للتغليف `encapsulation` في OOP. ومع ذلك، فشلت بايثون بطريقة ماضي تقديم تغليف حقيقي.

بالتعريف، يمكن الوصول إلى العضو العام في كل مكان. لا يمكن الوصول إلى عضو خاص إلا داخل الفئة الأساسية. يمكن الوصول إلى العضو المحمي داخل الفئة الأساسية والفئة (الفئات) الفرعية.

لكن مع بايثون، لا توجد مثل هذه الإجراءات الصارمة.

وبالتالي، فإن الأعضاء المحميين يتصرفون تماماً مثل الأعضاء العامين. علاوة على ذلك، يمكن الوصول إلى الأعضاء الخاصين خارج الفئة باستخدام تشويه الأسماء `.name mangling`.

كمبرمج، تذكر أن التغليف في بايثون يعتمد بشكل أساسي على الاصطلاحات conventions. وبالتالي، تقع على عاتق المبرمج مسؤولية متابعتها.

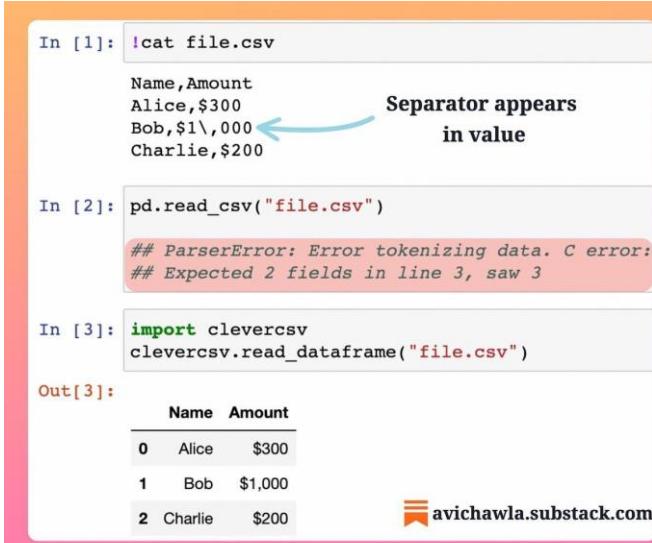
المقالة:

<https://avichawla.substack.com/p/why-python-does-not-offer-true-oop>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Python-Access-Modifiers.ipynb>

71) لا تقلق أبداً بشأن تحليل الأخطاء مرة أخرى أثناء قراءة ملف Pandas Never Worry About Parsing CSV Errors Again While Reading CSV with Pandas



```
In [1]: !cat file.csv
Name,Amount
Alice,$300
Bob,$1\,000
Charlie,$200
```

Separator appears
in value

```
In [2]: pd.read_csv("file.csv")
## ParserError: Error tokenizing data. C error:
## Expected 2 fields in line 3, saw 3
```

```
In [3]: import clevercsv
clevercsv.read_dataframe("file.csv")
```

	Name	Amount
0	Alice	\$300
1	Bob	\$1,000
2	Charlie	\$200

 avichawla.substack.com

Pandas ليست ذكية (حتى الآن) لقراءة ملفات CSV الفوضوية.

تفترض طريقة `read_csv` أن مصدر البيانات بتنسيق جدول tabular format قياسي. وبالتالي، فإن أي مخالفة في البيانات تؤدي إلى أخطاء في التحليل، والتي قد تتطلب تدخلاً يدوياً.

بدلاً من ذلك، جرب CleverCSV. يكتشف تنسيق ملفات CSV ويسهل تحميلها، مما يوفر لك الكثير من الوقت.

يمكنك العثور على مزيد من المعلومات هنا: [CleverCSV](#).

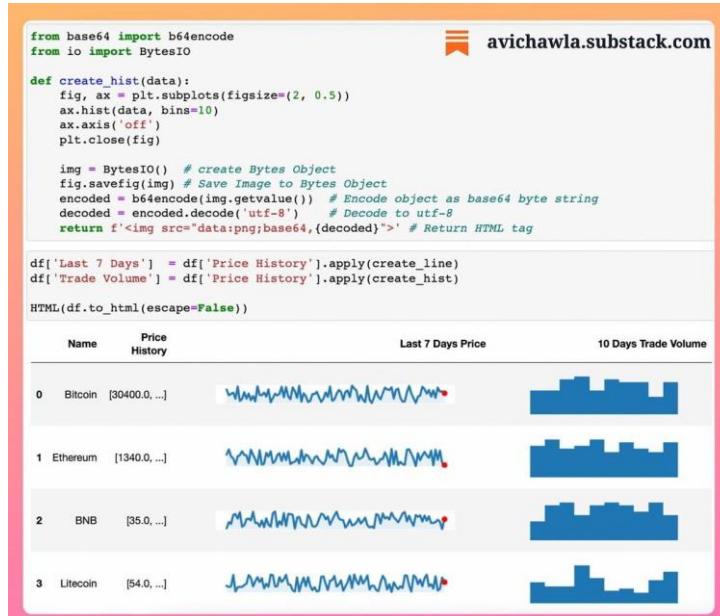
المقالة:

<https://avichawla.substack.com/p/never-worry-about-parsing-errors>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Messy-CSV-read.ipynb>

(72) طريقة مثيرة للاهتمام وغير معروفة لإنشاء مخططات Pandas An Interesting and Lesser-Known Way To Create Plots Using Pandas



عندما تقوم بطباعة / عرض إطار بيانات DataFrame في Jupyter، يتم تقديمها باستخدام HTML و CSS. يتيح لنا ذلك تنسيق الإخراج تماماً مثل أي صفحة ويب أخرى.

في مقتطف الكود أعلاه، نقوم أولاً بإنشاء مخطط كما نفعل عادةً. بعد ذلك، نعيد علامة HTML بمصدرها كمخطط. أخيراً، نقدم إطار البيانات بتنسيق HTML.

المقالة:

<https://avichawla.substack.com/p/inline-plots-pandas>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Inline-DF-Plots.ipynb>

Most For-Loop Programmers Don't Know This About Python For-loops



غالبًا عندما نستخدم حلقة for-loop في بايثون، فإننا لا نميل إلى تعديل متغير الحلقة داخل الحلقة. يأتي الدافع عادةً من التعرف على لغات البرمجة الأخرى مثل Java و C++.

لكن حلقات for لا ت عمل بهذه الطريقة في بايثون. تعديل متغير الحلقة ليس له تأثير على التكرار. هذا لأنّه، قبل كل تكرار، تقوم بايثون بفك حزم العنصر التالي الذي يوفره قابل للتكرار ((range(5)). وتخصيصه لمتغير الحلقة (num).

وبالتالي، يتم استبدال أي تغييرات في متغير الحلقة بالقيمة الجديدة القادمة من المتغير القابل للتكرار.

المقالة:

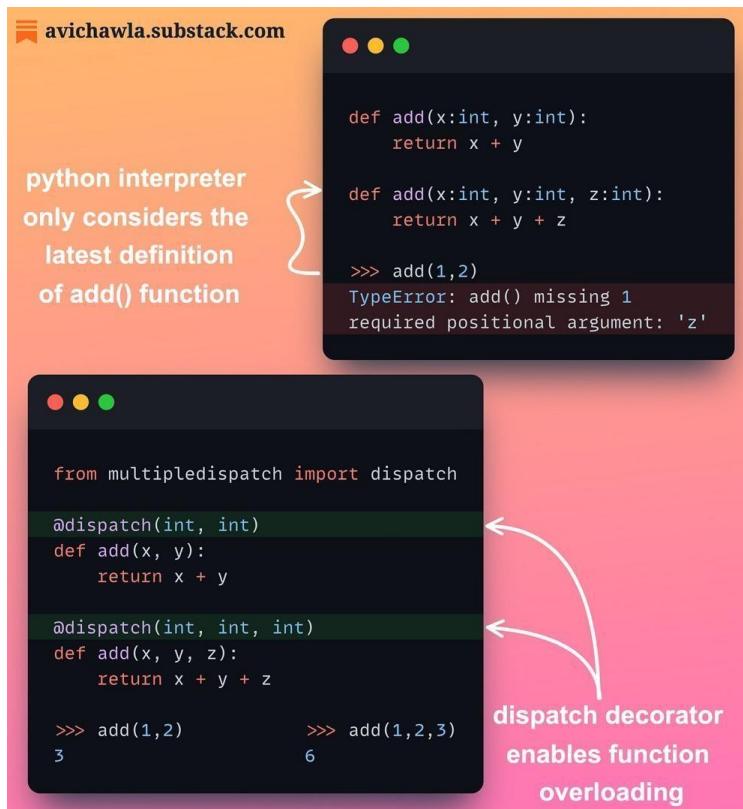
<https://avichawla.substack.com/p/most-python-programmers-dont-know>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Lesser-Known-For-Loop-Tip-Python.ipynb>

How To 74 كيفية تفعيل التحميل الزائد للدوال في بايثون

Enable Function Overloading In Python



لا يوجد دعم أصلي لبايثون للتحميل الزائد على الدوال function overloading. ومع ذلك، هناك حل سريع لها.

يعد التحميل الزائد للدالة (وجود دوال متعددة بنفس الاسم ولكن عدد / نوع مختلف من المعلمات) أحد الأفكار الأساسية وراء تعدد الأشكال polymorphism في البرمجة كائنية التوجه OOP.

ولكن إذا كان لديك العديد من الدوال بنفس الاسم، فإن بايثون لا تأخذ في الاعتبار سوى أحدث تعريف. هذا يقييد كتابة كود متعدد الأشكال.

على الرغم من هذا القيد، يسمح لك مصمم الإرسال بالاستفادة من التحميل الزائد للدالة.

يمكنك العثور على مزيد من المعلومات هنا: [Multipledispatch](#)

المقالة:

<https://avichawla.substack.com/p/how-to-enable-function-overloading>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Function-Overloading.ipynb>

75) قم بإنشاء تلميحات مفيدة أثناء كتابة كود Pandas الخاص بك Generate Helpful Hints As You Write Your Pandas Code

```
In [4]: import dovpanda
In [5]: iter_df = df.iterrows()
In [6]: df['new_col'] = df.apply(apply_func)
In [7]: merged_df = pd.concat((df, df))
```

In [4]: import dovpanda

In [5]: iter_df = df.iterrows()

In [6]: df['new_col'] = df.apply(apply_func)

In [7]: merged_df = pd.concat((df, df))

عند معالجة إطار البيانات `dataframe`, في بعض الأحيان، قد يستخدم المرء طرفة غير محسنة. علاوة على ذلك، يمكن أن تمر الأخطاء التي يتم إدخالها في البيانات بسهولة دون أن يلاحظها أحد.

للحصول على تلميحات hints وتوجيهات directions حول بياناتك / كودك، جرب `Dovpanda`. إنه يعمل كرفيق لكود Pandas الخاص بك. نتيجة لذلك، فإنه يقدم اقتراحات / تحذيرات حول خطوات معالجة البيانات الخاصة بك.

ملاحظة: عندما تقوم باستيراد `Dovpanda`, من المحتمل أن تحصل على خطأ. تجاهله واستمر في استخدام `Pandas`. ستستمر في تلقى اقتراحات من `Dovpanda`.

يمكنك العثور على مزيد من المعلومات هنا: [Dovpandas](#)

المقالة:

<https://avichawla.substack.com/p/generate-helpful-hints-as-you-write>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Pandas/Generate-Pandas-Hints.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Generate-Pandas-Hints.ipynb)

76) طرق تسريع NumPy 25x With Bottleneck

```

import bottleneck as bn
import numpy as np

arr = np.random.random((1000,))

```

NumPy Method	Bottleneck Method	Run-time (μs)	Speedup (approx.)
<code>np.sum(arr)</code>	<code>bn.nansum(arr)</code>	870	25x Faster
<code>np.mean(arr)</code>	<code>bn.nanmean(arr)</code>	477	22x Faster
<code>np.std(arr)</code>	<code>bn.nanstd(arr)</code>	687	4x Faster
<code>np.median(arr)</code>	<code>bn.nanmedian(arr)</code>	1.58 ms	4x Faster
<code>np.max(arr)</code>	<code>bn.nanmax(arr)</code>	1.26 ms	3x Faster

تم بالفعل تحسين طرق NumPy بشكل كبير للأداء. نعم، إليك كيفية زيادة تسريعها.

يوفر Bottleneck مجموعة من التطبيقات المحسنة لطرق NumPy. يعتبر Bottleneck فعالاً بشكل خاص للمصفوفات ذات قيم NaN حيث يمكن أن يصل تعزيز الأداء إلى 100–120 مرة.

يمكنك العثور على مزيد من المعلومات هنا: [Bottleneck](#).

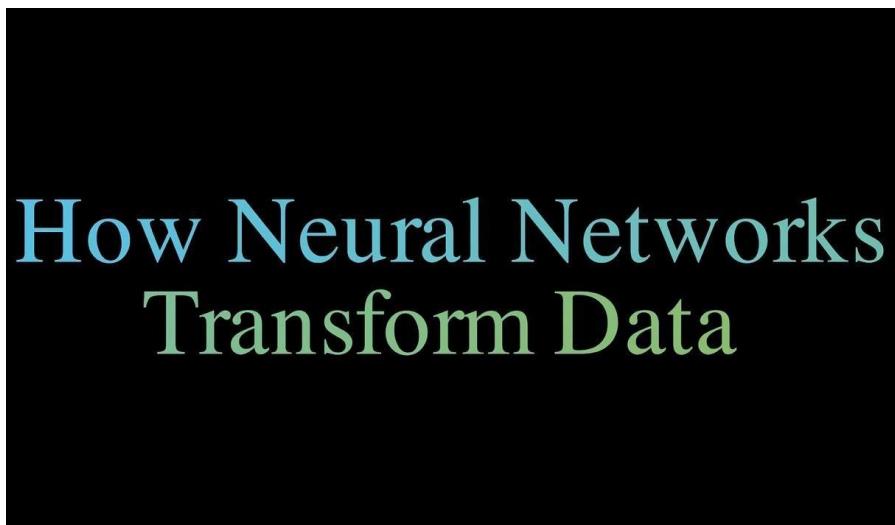
المقالة:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Speedup-NumPy-with-Bottleneck>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/NumPy/Speedup-NumPy-with-Bottleneck.ipynb>

77) تصور تحول البيانات لشبكة عصبية Visualizing The Data Transformation of a Neural Network



إذا كنت تعاني لفهم كيف تتعلم الشبكة العصبية neural network البيانات المعقدة غير الخطية، فقد صنعت انيميشن سيساعدك بالتأكيد.

يرجى مشاهدة الفيديو [هنا](#): الرسوم المتحركة للشبكة العصبية.

بالنسبة للبيانات غير القابلة للفصل خطياً linearly inseparable data، تتلخص المهمة في إسقاط البيانات على مساحة حيث تصبح قابلة للفصل خطياً.

الآن، إما يمكنك القيام بذلك يدوياً عن طريق إضافة الميزات ذات الصلة التي ستتحول بياناتك إلى نموذج خططي قابل للفصل. ضع في اعتبارك الدوائر متعددة المركز على سبيل المثال. تمرير مربع إحداثيات (x,y) كميزة ستؤدي هذه المهمة.

لكن في معظم الحالات، يكون التحول transformation غير معروف أو معقداً. وبالتالي، تعتبر دوال التنشيط غير الخطية non-linear activation functions هي أفضل رهان، ويُسمح للشبكة العصبية باكتشاف هذا "التحويل غير الخططي إلى الخططي" بمفردها.

كما هو موضح في الانيميشن، إذا قمنا بتعديل الشبكة العصبية عن طريق إضافة طبقة ثنائية الأبعاد قبل الإخراج مباشرة، وتصور هذا التحول، فإننا نرى أن الشبكة العصبية قد تعلمت فصل البيانات خطياً. نضيف طبقة ثنائية الأبعاد لأنه من السهل تصوّرها.

يمكن بسهولة تصنيف هذه البيانات القابلة للفصل خطياً حسب الطبقة الأخيرة. بعبارة أخرى، تكون الطبقة الأخيرة مماثلة لنموذج الانحدار اللوجستي logistic regression الذي يُعطي مدخلات خطية قابلة للفصل.

المقالة:

<https://avichawla.substack.com/p/visualizing-the-data-transformation>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Animations/Neural-Network-Visualization.ipynb>

78) لا تقم أبداً بإعادة بناء كودك يدوياً مرة أخرى. بدلاً من ذلك، Never Refactor Your Code !Sourcery استخدم !Manually Again. Instead, Use Sourcery



كود إعادة البناء Refactoring هو خطة مهمة في تطوير خط الأنابيب. ومع ذلك، فإن إعادة هيكلة اليدوية تستغرق وقتاً إضافياً للاختبار حيث قد يتسبب الماء عن غير قصد في حدوث أخطاء.

بدلاً من ذلك، استخدم Sourcery. إنها أداة إعادة هيكلة آلية تجعل التعليمات البرمجية الخاصة بك أنيقة وموجزة وبأيونية في لمح البصر.

باستخدام Sourcery، يمكنك إعادة بناء الكود من سطر الأوامر، كمكون إضافي لـ IDE في VS Code، وPyCharm، والتنفيذ المسبق، وما إلى ذلك.

العثور على مزيد من المعلومات هنا: [Sourcery](#).

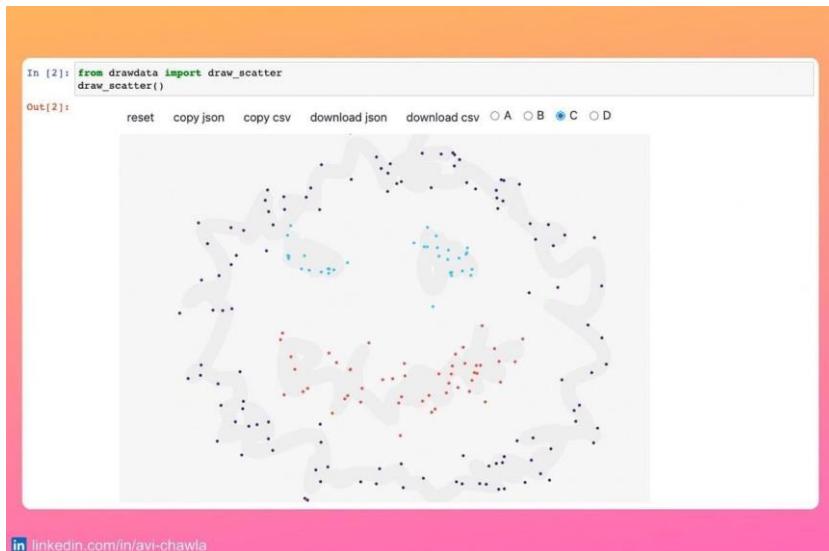
المقالة:

<https://avichawla.substack.com/p/never-refactor-your-code-manually>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Terminal/Code-refactoring.ipynb>

Draw The Data التي تبحث عنها في ثوانٍ 79 You Are Looking For In Seconds



يرجى مشاهدة نسخة فيديو من هذا المنشور لفهم أفضل: [رابط الفيديو](#).

في كثير من الأحيان، عندما تريد بيانات ذات شكل معين، فإن إنشائها برمجياً يمكن أن يكون مهمة شاقة وتستغرق وقتاً طويلاً.

بدلاً من ذلك، استخدم `drawdata`. يتيح لك ذلك رسم أي مجموعة بيانات ثنائية الأبعاد في نوتبوك وتصديرها. إلى جانب مخطط التبعثر `scatter plot`، يمكنه أيضاً إنشاء رسم بياني `histogram` ورسم خطبي `line plot`.

يمكنك العثور على مزيد من المعلومات هنا: [Drawdata](#)

المقالة:

<https://avichawla.substack.com/p/draw-the-data-you-are-looking-for>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/Draw-Data.ipynb>

٨٠) نمط مخططات Matplotlib لجعلها أكثر جاذبية Matplotlib Plots To Make Them More Attractive



يحتوي Matplotlib تقريباً على 50 نمطاً styles مختلفاً لتخصيص مظهر المخطط.

لتغيير نمط المخطط، حدد نمطاً من `plt.style.available` وأنشئ المخطط كما كنت تفعل في الأصل.

تعرف على مزيد من المعلومات حول التصميم هنا: [المستندات](#).

المقالة:

<https://avichawla.substack.com/p/style-matplotlib-plots-to-make-them>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Change-Matplotlib-Style.ipynb>

81) تسريع إدخال / إخراج باركيه من Pandas بمقدار خمس مرات Speed-up Parquet I/O of Pandas by 5x



غالباً ما يتم تخزين إطارات البيانات في ملفات باركيه parquet files وقراءتها باستخدام طريقة Pandas `.read_parquet()`

بدلاً من استخدام Pandas ، الذي يعتمد على نواة واحدة، استخدم fastparquet. إنها توفر تسريعاً هائلاً للإدخال / الإخراج على ملفات باركيه باستخدام المعالجة المتوازية parallel processing.

يمكنك العثور على مزيد من المعلومات هنا: [المستندات](#).

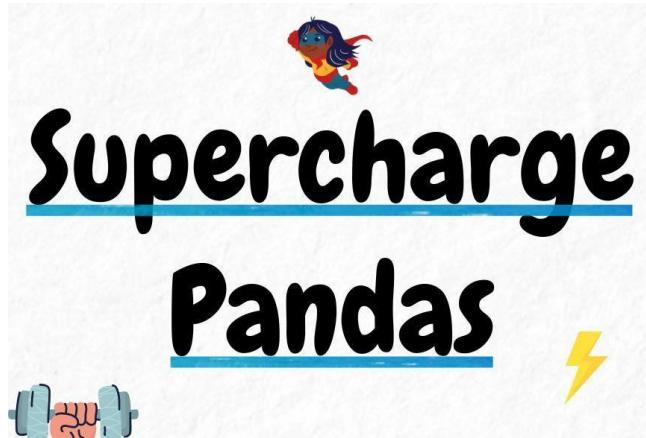
المقالة:

<https://avichawla.substack.com/p/speed-up-parquet-io-of-pandas-by>

: الكود

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Runtime%20Optimization/Parquet-Optimize.ipynb>

82) أداة مفتوحة المصدر لتعزيز سير عمل Pandas الخاص Open-Source Tools to Supercharge Your 40 بك Pandas Workflow



تتلقي Pandas أكثر من 3 ملايين تحميل يومياً. لكن 99% من مستخدميها لا يستخدمنها بكامل طاقتها. لقد اكتشفت هذه الأحجار الكريمة مفتوحة المصدر التي ستعزز بشكل كبير أعمال Pandas الخاصة بك، وما أن تبدأ في استخدامها.

اقرأ هذه القائمة هنا:

<https://avichawla.substack.com/p/37-open-source-tools-to-supercharge-pandas>

المقالة:

<https://avichawla.substack.com/p/37-open-source-tools-to-supercharge-pandas>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Documents/40-pandas-libs.ipynb>

٨٣) توقف عن استخدام طريقة الوصف في Pandas. بدلاً من ذلك، استخدم .Skimpy .Method in Pandas. Instead, use Skimpy

```
from skimpy import skim
skim(df)
```

Data Summary		Data Types		skimpy summary		Categories	
dataframe	Values	Column Type	Count			Categorical Variables	
Number of rows	1000	float64	3			class	
Number of columns	10	category	2			location	
		datetime64	2				
		int64	1				
		bool	1				
		string	1				

number

column_name	NA	NA %	mean	sd	p0	p25	p75	p100	hist
length	0	0	0.5	0.36	1.6e-05	0.13	0.86	1	
width	0	0	2	1.9	0.0021	0.6	3	14	
depth	0	0	10	3.2	2	8	12	20	
rnd	120	12	-0.02	1	-2.8	-0.74	0.66	3.7	

category

column_name	NA	NA %	ordered	unique
class	0	0	0	2
location	1	0.1	False	5

datetime

column_name	NA	NA %	first	last	frequency
date	0	0	2018-01-31	2101-04-30	M
date_no_freq	3	0.3	1992-01-05	2023-03-04	None

string

column_name	NA	NA %	words per row	total words
text	6	0.6	5.8	5800

bool

column_name	true	true rate	hist
booly_col	520	0.52	

End

in linkedin.com/in/avi-chawla

. تعزيز طريقة الوصف .Pandas describe method

Skimpy هي أداة خفيفة لتلخيص إطارات بيانات Pandas. في سطر واحد من التعليمات البرمجية، يُنشئ ملخصاً إحصائياً أكثر ثراءً من طريقة .describe().

علاوة على ذلك، يتم تجميع الملخص حسب أنواع البيانات لتحليل فعال. يمكنك استخدام Skimpy من سطر الأوامر أيضاً.

يمكنك العثور على مزيد من المعلومات هنا: [المستندات](#).

المقالة:

<https://avichawla.substack.com/p/stop-using-the-describe-method-in-85e>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Pandas/Supercharged-Describe-2.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Supercharged-Describe-2.ipynb)

84) الطريقة الصحيحة لطرح تحدیثات المكتبة في بايثون Right Way to Roll Out Library Updates in Python

The diagram illustrates the right way to roll out library updates in Python. It shows two code snippets: `my_library.py` and `project.py`.

`my_library.py` contains:

```
from deprecated import deprecated
@deprecated(reason="old_function will be \
    deprecated in the next \
    release. Use new_function.")
def old_function():
    ...
```

A red arrow labeled "Add decorator" points from the `@deprecated` annotation in `my_library.py` to the `DeprecationWarning` message in `project.py`.

`project.py` contains:

```
old_value = old_function()
DeprecationWarning: Call to deprecated function
old_function. (old_function will be deprecated
in the next release. Use new_function.)
```

A red arrow labeled "Prints warning" points from the `DeprecationWarning` message in `project.py` back to the `@deprecated` annotation in `my_library.py`.

Below the code snippets, there is a LinkedIn link: linkedin.com/in/avi-chawla

أثناء تطوير مكتبة library، قد يقرر المؤلفون إزالة بعض الدوال /functions /methods الأساليب classes. لكن طرح التحديث على الفور دون أي تحذير مسبق ليس ممارسة جيدة.

هذا لأن العديد من المستخدمين ربما لا يزالون يستخدمون الطرق القديمة وقد يحتاجون إلى وقت لتحديث التعليمات البرمجية الخاصة بهم.

باستخدام **deprecated**، يمكن للمرء أن ينقل تحذيرًا للمستخدمين حول التحديث. هذا يسمح لهم بتحديث التعليمات البرمجية الخاصة بهم قبل أن تصبح قديمة.

يمكنك العثور على مزيد من المعلومات هنا: [GitHub](#)

المقالة:

<https://avichawla.substack.com/p/the-right-way-to-roll-out-library>

: الكود

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Deprecated-Warning.ipynb>

(85) أسطر بسيطة واحدة لمعاينة شجرة القرار باستخدام Simple One-Liners to Preview a Decision Tree Sklearn Using Sklearn

```

my_tree = DecisionTreeClassifier()
my_tree.fit(X, y)

from sklearn.tree import plot_tree, export_text

```

Method 1

```

plot_tree(my_tree, feature_names=features,
          class_names=classes, filled=True)

```

```

print(export_text(my_tree, feature_names=features))

```

Method 2

```

|--- petal_width <= 0.80
|   |--- class: setosa
|--- petal_width >  0.80
|   |--- petal_width <= 1.75
|   |   |--- class: versicolor
|   |   |--- petal_width >  1.75
|   |   |--- class: virginica

```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

إذا كنت ترغب في معاينة شجرة القرار decision tree، فإن sklearn يوفر طريقتين بسيطتين للقيام بذلك.

يُنشئ تمثيلاً رسومياً لشجرة القرار .**plot_tree**.

يُنشئ تقريراً نصياً يوضح قواعد شجرة القرار .**export_text**.

يستخدم هذا عادةً لفهم القواعد التي تعلمتها شجرة القرار واكتساب فهم أفضل لسلوك نموذج شجرة القرار.

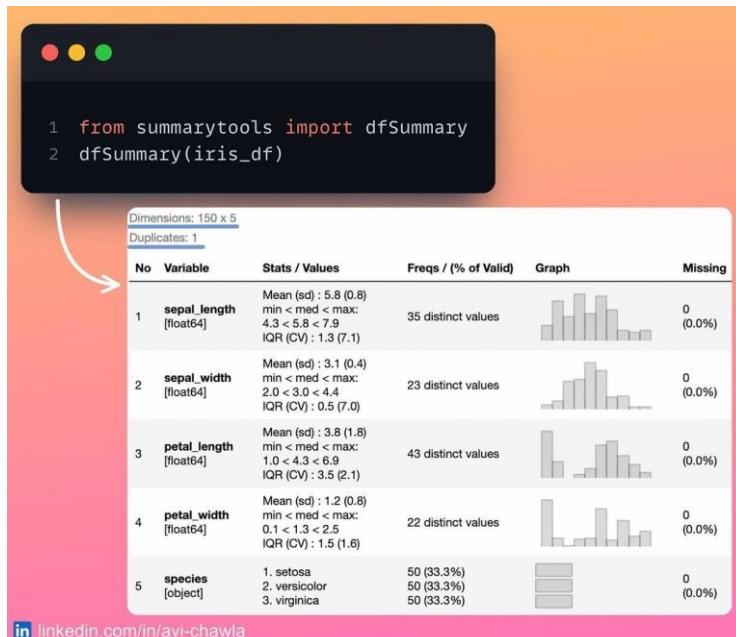
المقالة:

<https://avichawla.substack.com/p/simple-one-liners-to-preview-a-decision>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Preview-Decision-Tree.ipynb>

٨٦) توقف عن استخدام طريقة الوصف في Pandas. بدلاً من ذلك، استخدم طريقة الوصف في Summarytools.



هي أداة بسيطة لتحليل البيانات الاستكشافية EDA تعطي ملخصاً أكثر ثراءً من طريقة `describe()`. في سطر واحد من التعليمات البرمجية، يقوم بإنشاء ملخص بيانات معياري وشامل.

يتضمن الملخص إحصائيات العمود والتكرار ومخطط التوزيع والإحصائيات المفقودة.

يمكنك العثور على مزيد من المعلومات هنا: [أدوات الملخص](#).

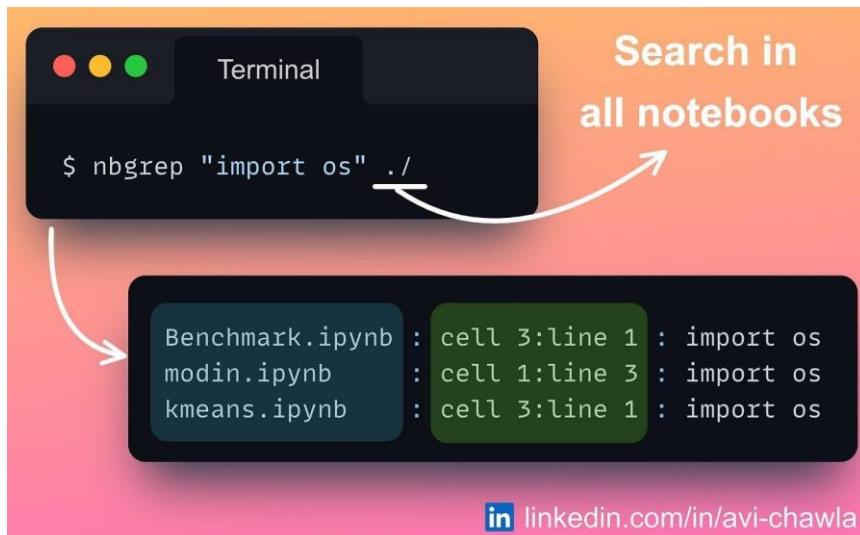
المقالة:

<https://avichawla.substack.com/p/stop-using-the-describe-method-in>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Supercharge-Describe.ipynb>

87) لا تبحث أبداً في نوتبوك Jupyter يدوياً مرة أخرى للعثور على الكود الخاص بك Never Search Jupyter Notebooks Manually Again To Find Your Code



هل سبق لك أن عانيت لتنكر دفتر Jupyter المحدد الذي كتبت فيه بعض الأكواد؟ إليك حيلة سريعة لتوفير الكثير من العمل اليدوي والوقت.

توفر **nbcommands** مجموعة من الأوامر للتفاعل مع Jupyter من الجهاز. على سبيل المثال، يمكنك البحث عن التعليمات البرمجية ومعاينة بعض الخلايا ودمج النوتبوك وغير ذلك الكثير.

يمكنك العثور على مزيد من المعلومات هنا: [GitHub](#).

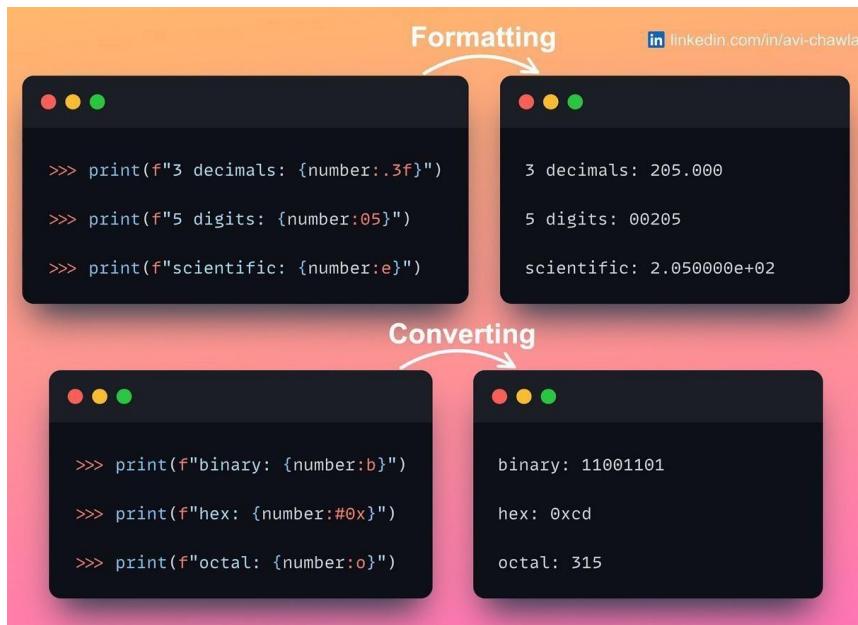
المقالة:

<https://avichawla.substack.com/p/never-search-jupyter-notebooks-manually>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/Search-Code-in-Jupyter.ipynb>

F-strings Are Much More Versatile Than You Think



فيما يلي 6 طرق أقل شهرة لتنسيق / تحويل رقم باستخدام سلاسل f (f-string). ما هو الاختلاف المفضل لديك؟

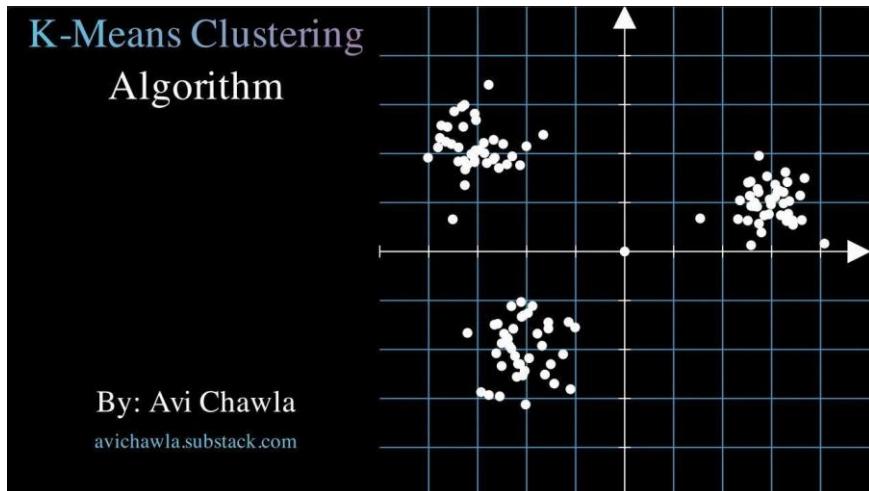
المقالة:

<https://avichawla.substack.com/p/f-strings-are-much-more-versatile>

ال코드:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/f-strings-hack.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/f-strings-hack.ipynb)

Is (89) هل هذا هو أفضل دليل متحرك لـ KMeans على الإطلاق؟ ?This The Best Animated Guide To KMeans Ever



هل واجهت صعوبة في فهم KMeans؟ كيف يعمل، كيف يتم تعيين نقاط البيانات data points إلى النقطة الوسطى centroids، أو كيف تتحرك النقطة الوسطى؟
إذا كانت الإجابة بنعم، دعني أساعدك.

لقد قمت بإنشاء انيميشن جميل باستخدام Manim لمساعدتك في بناء فهم حدسي للخوارزمية.
الرجاء العثور على هذا الفيديو هنا: [رابط الفيديو](#).

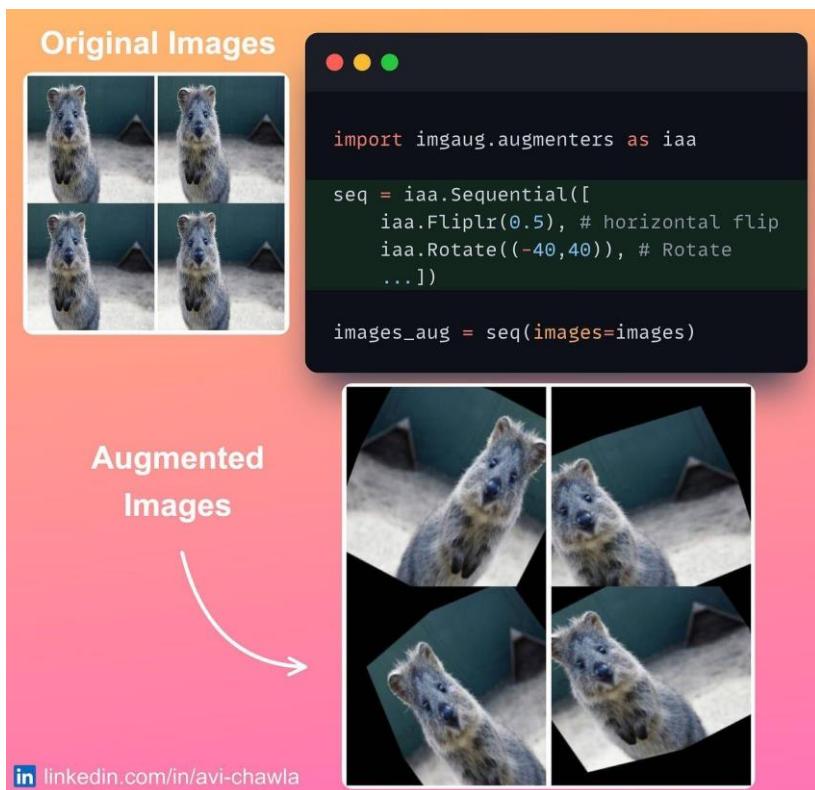
المقالة:

<https://avichawla.substack.com/p/is-this-the-best-animated-guide-to>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Machine%20Learning/KMeans-Visualization.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/KMeans-Visualization.ipynb)

٨٩) تقنية فعالة ولكن تم الاستخفاف بها لتحسين أداء النموذج An Effective Yet Underrated Technique To Improve Model Performance



نماذج التعلم الآلي القوية مدفوعة ببيانات تدريب متنوعة. إليك أسلوب بسيط ولكنه فعال للغاية يمكن أن يساعدك في إنشاء مجموعة بيانات متنوعة وزيادة أداء النموذج.

طريقة واحدة لزيادة تنوع البيانات data diversity هي استخدام زيادة البيانات data augmentation.

الفكرة هي إنشاء عينات جديدة عن طريق تحويل العينات المتوفرة. هذا يمكن أن يمنع الضبط الرائد overfitting، وتحسين الأداء improve performance، وبناء نماذج قوية build robust models.

بالنسبة للصور، يمكنك استخدام imgaug. يوفر مجموعة متنوعة من تقنيات زيادة البيانات مثل التقليل flipping والتدوير rotating والقياس scaling وإضافة الضوضاء noise إلى الصور وغير ذلك الكثير.

البحث عن مزيد من المعلومات: [Imgaug](#)

المقالة:

<https://avichawla.substack.com/p/an-effective-yet-underrated-technique>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Machine%20Learning/Data-Augmentation.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Data-Augmentation.ipynb)

٩٠) إنشاء مخططات البيانات مباشرةً من الترمinal Data Plots Right From The Terminal

```
>>> from bashplotlib.histogram import plot_hist
>>> np_arr = np.random.normal(size=1000)
>>> plot_hist(np_arr, bincount=50)

54|          o
51|      oo oo
48|      ooo ooo o
45|      ooo ooo o
43|      oooo ooo o
40|      ooooooooooo oo
37|      oo ooooooooooooooo
34|      oo oooooooooooooooo
31|      ooooooooooooooooooo
29|      ooooooooooooooooooo
26|      oooooooooooooooooooo
23|      oooooooooooooooooooo
20|      oooooooooooooooooooo
17|      oooooooooooooooooooo o
15|      oooooooooooooooooooo o
12|      oooooooooooooooooooo o
9|      oooooooooooooooooooo
6|      oo ooooooooooooooooo o
3|      o   o oooooooooooooooo o
1|  o o oooooooooooooooooooo o
```

 [linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

يمكن أن يصبح تصوّر البيانات Visualizing data أمراً صعباً عندما لا يكون لديك وصول إلى واجهة المستخدم الرسومية. ولكن إليك ما يمكن أن يساعد.

يُوفِر `Bashplotlib` طريقة سريعة وسهلة لإنشاء المخططات الأساسية مباشرةً من الترمinal `terminal`.
نظرًا لكونك بايثون خالصًا، يمكنك تثبيته بسرعة في أي مكان باستخدام `pip` وتصور بياناتك.

يمكنك العثور على مزيد من المعلومات هنا: [Bashplotlib](#).

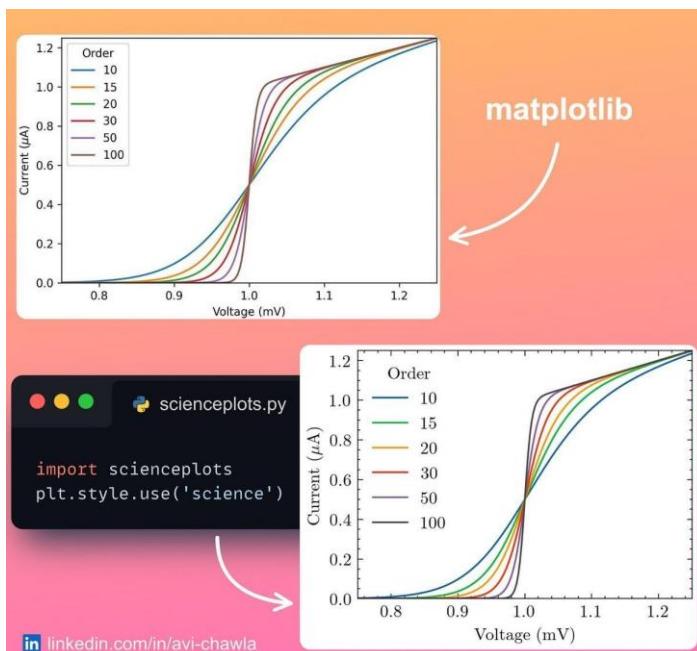
المقالة:

<https://avichawla.substack.com/p/create-data-plots-right-from-the>

الكتاب:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Terminal/Plotting-in-Terminal.ipynb>

91) اجعل مخططات Matplotlib أكثر احترافية Make Your Matplotlib Plots More Professional



تعد مخططات matplotlib الافتراضية أساسية جدًا في النمط `style`, وبالتالي، قد لا تكون الخيار المناسب دائمًا. إليك كيف يمكنك جعلها جذابة.

لإنشاء مخططات جذابة ذات مظهر احترافي للعرض التقديمية أو التقارير أو الأوراق العلمية، جرب **المخططات العلمية** .

تؤدي إضافة سطرين فقط من التعليمات البرمجية إلى تغيير مظهر المخطط تماماً. يمكنك العثور على المزيد من المعلومات هنا: [GitHub](#)

المقالة:

<https://avichawla.substack.com/p/make-your-matplotlib-plots-more-professional>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Professional-Science-Plots.ipynb>

Hidden 37 مكتبة بايثون المخفية هي جواهر مطلقة (92)

Python Libraries That Are Absolute Gems



لقد راجعت أكثر من 1000 مكتبة بايثون واكتشفت هذه الجوادر الخفية التي لم أكن أعرف بوجودها من قبل.

إليك بعضًا منها ستجعلك تقع في حب بايثون وتعدد استخداماتها (حتى أكثر).

اقرأ هذه القائمة هنا: <https://avichawla.substack.com/p/gem-libraries>

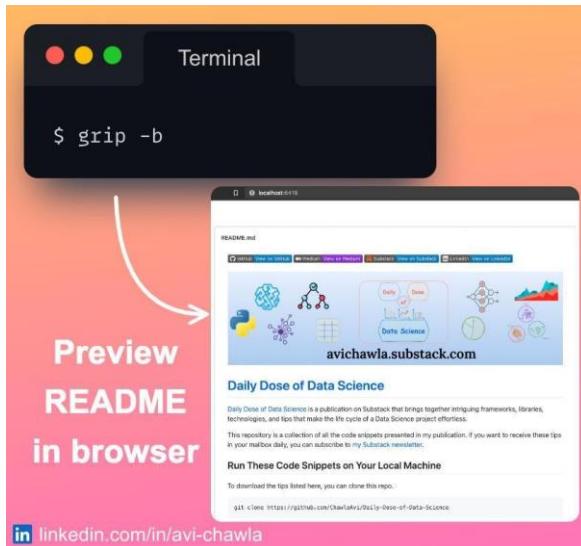
المقالة:

<https://avichawla.substack.com/p/gem-libraries>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Documents/37-cool-libs.ipynb>

93) قم بمعاينة ملف README الخاص بك محليًا بأسلوب Preview Your README File Locally In GitHub GitHub Style



يرجى مشاهدة نسخة فيديو لفهم أفضل: [رابط الفيديو](#).

هل سبق لك أن أردت معاينة ملف README قبل إلزامه بـGitHub؟ هنا كيفية القيام بذلك.

هي أداة سطر أوامر تتيح لك عرض ملف README كما سيظهر على GitHub. هذا مفيد للغاية حيث قد يرغب المرء أحيانًا في معاينة الملف قبل دفعه إلى GitHub.

والأكثر من ذلك، أن تحرير README يظهر على الفور في المتصفح دون أي تحديث للصفحة.

اقرأ المزيد: [Grip](#)

المقالة:

<https://avichawla.substack.com/p/preview-your-readme-file-locally>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/README-preview-local.ipynb>

94) تقوم NumPy و Pandas بإرجاع قيم مختلفة للانحراف المعياري. لماذا؟ Pandas and NumPy Return Different ?Values for Standard Deviation. Why

```

import numpy as np
import pandas as pd

X = np.arange(20)
df = pd.DataFrame(X)

print(f"NumPy : {np.std(X)}")
print(f"Pandas: {df.std()}")

```

std-dev using
Pandas and
NumPy

different
output

NumPy : 5.766
Pandas: 5.916

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

تفترض Pandas أن البيانات هي عينة من السكان وأن النتيجة التي تم الحصول عليها يمكن أن تكون متحيزة biased تجاه العينة.

وبالتالي، لتوليد تقدير غير متحيز unbiased estimate، فإنه يستخدم $(1 - \frac{1}{n})$ كعامل قسمة بدلاً من n . في الإحصاء، يُعرف هذا أيضًا باسم تصحيح بيسيل Bessel's correction.

NumPy ، ومع ذلك ، لا يقوم بأي تصحيح من هذا القبيل. يمكنك العثور على مزيد من المعلومات هنا: [تصحيح بيسيل](#).

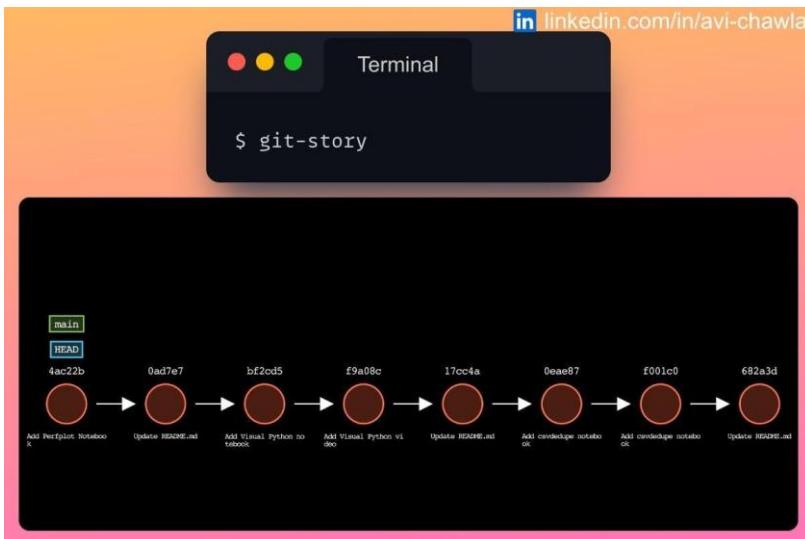
المقالة:

<https://avichawla.substack.com/p/pandas-and-numpy-return-different>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Statistics/Std-dev-Pandas-NumPy.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Statistics/Std-dev-Pandas-NumPy.ipynb)

95) تصوّر تاريخ Git Repo مع الانيميشن الجميل Visualize Commit History of Git Repo With Beautiful Animations



مع نمو حجم مشروعك، قد يكون من الصعب فهم شجرة Git.

Git-story هي أداة سطر أوامر لإنشاء رسوم متحركة أنيقة لمستودع git الخاص بك.

يقوم بإنشاء مقطع فيديو يصور الالتزامات commits والفرع branches والدمج merges و HEAD يقوم بإنشاء مقطع فيديو يصور الالتزامات commits والفرع branches والدمج merges و HEAD و غير ذلك الكثير. ابحث عن مزيد من المعلومات في التعليقات.

لـ جـ مشـاهـدة نـسـخـة فـيـديـو مـنـ هـذـا المـنشـور هـنـا: فـيـديـو.

اقرأ المزيد: Git-story

ä||ö||

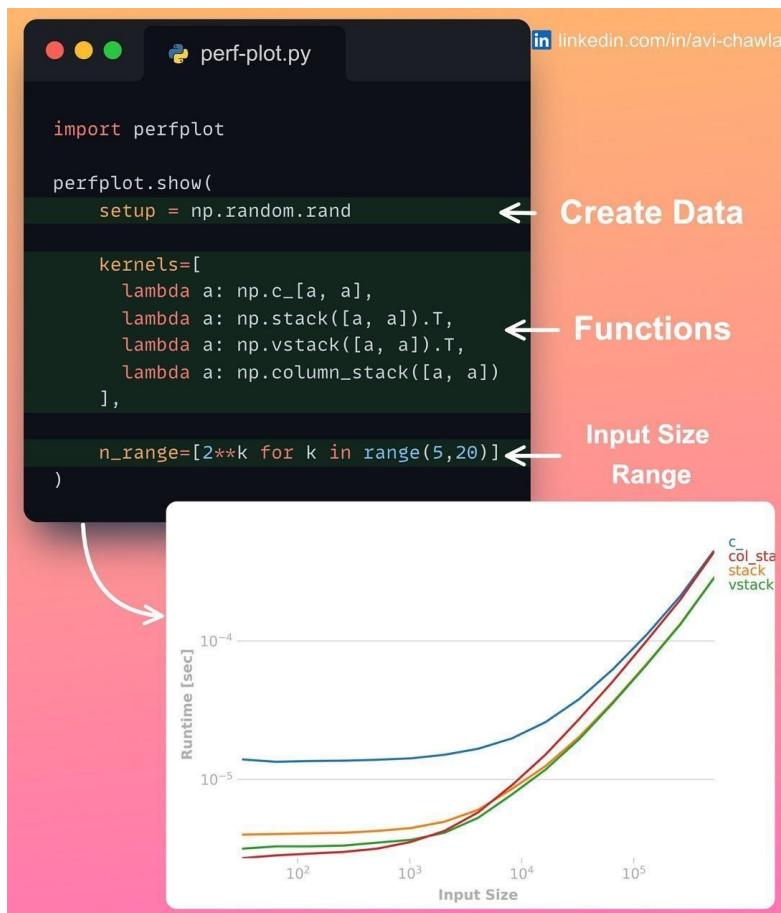
<https://avichawla.substack.com/p/visualize-commit-history-of-git-repo>

الكتاب

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Terminal/Git-Story.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Terminal/Git-Story.ipynb)

قم بقياس وقت التنفيذ وتصوره ومقارنته Perfplot: Measure, Visualize and Compare بسهولة

Run-time With Ease



إليك طريقة أنيقة لقياس وقت تنفيذ دوال بايثون المختلفة.

Perfplot هي أداة مصممة لإجراء مقارنات سريعة في وقت التنفيذ run-time للعديد من الدوال / الخوارزميات.

إنه يوسع حزمة وقت بايثون ويسمح لك بتصور وقت التنفيذ بسرعة بطريقة واضحة وغنية بالمعلومات.

العثور على مزيد من المعلومات: [Perfplot](#)

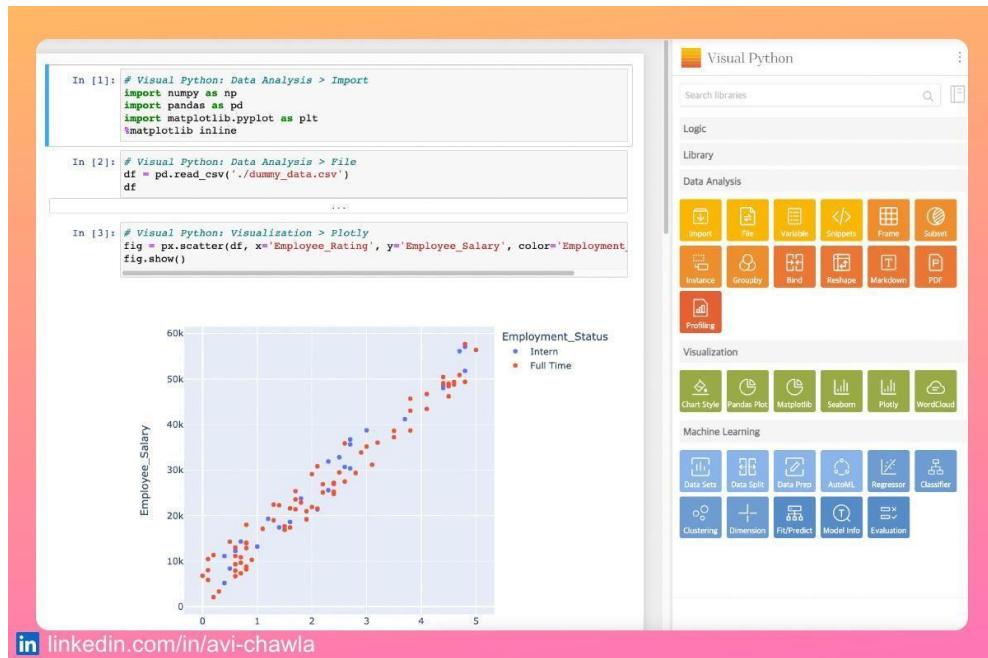
المقالة:

<https://avichawla.substack.com/p/perfplot-measure-visualize-and-compare>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/Perfplot.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Perfplot.ipynb)

يمكن أن توفر لك أداة واجهة المستخدم الرسومية هذه 97 ساعات من العمل اليدوي This GUI Tool Can Possibly Save You Hours Of Manual Work



يرجى مشاهدة نسخة فيديو من هذا المنشور لفهم أفضل: [الرابط](#).

هذه بالفعل واحدة من أروع أدوات علم البيانات المستندة إلى نوتبوك Jupyter وأكثرها فائدة.

Visual Python هو منشئ كود بايثون قائم على واجهة المستخدم الرسومية. باستخدام هذا، يمكنك بسهولة التخلص من كتابة التعليمات البرمجية للعديد من المهام المتكررة. يتضمن ذلك استيراد المكتبات وعمليات الإدخال / الإخراج وعمليات Pandas والتحطيط وما إلى ذلك.

علاوة على ذلك، بنقرة اثنين من الأزرار، يمكنك استيراد الكود للعديد من الأدوات المساعدة المستندة إلى التعلم الآلي. يغطي هذا نماذج sklearn ومقاييس التقييم ودوال تقسيم البيانات وغير ذلك الكثير.

اقرأ المزيد: [Visual Python](#).

المقالة:

<https://avichawla.substack.com/p>this-gui-tool-can-possibly-save-you>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/Visual-Python.ipynb>

98) كيف يمكنك تحديد التكرارات الضبابية في بيانات تحتوي على مليون سجل؟ How Would You Identify Fuzzy Duplicates In A Data With Million Records

	First_Name	Last_Name	Address	Phone
0	Daniel	Lopez	719 Greene St. East Rhonda	9371184929
1	Daniel	Nan	719 Green Street East Rhoda	93711-84929
2	Alan	Martin	982 Carol Harbors Apart.	7481919235
3	Alan Martin	Nan	982 Carol Aparments	748-191-9235
4	Philip	Owens	2578 Banks Ford	869-6922x9581
5	Shannon	White	USCGC Molina	(150)082-7982
6	Julia	Anderson	09162 Mason Mnts.	698-1590x3236
7	Juliya	Anderrson	9162 Mason Street Mountain	69815903236

Data with
fuzzy
duplicates

```
$ csvdedupe input.csv \
--field_names First_Name Last_Name Address Phone \
--output_file output.csv
```

Marked
Duplicates

	Cluster ID	First_Name	Last_Name	Address	Phone
0	0	Daniel	Lopez	719 Greene St. East Rhonda	9371184929
1	0	Daniel	nan	719 Green Street East Rhoda	93711-84929
2	1	Alan	Martin	982 Carol Harbors Apart.	7481919235
3	1	Alan Martin	nan	982 Carol Aparments	748-191-9235
4	2	Philip	Owens	2578 Banks Ford	869-6922x9581
5	3	Shannon	White	USCGC Molina	(150)082-7982
6	4	Julia	Anderson	09162 Mason Mnts.	698-1590x3236
7	4	Juliya	Anderrson	9162 Mason Street Mountain	69815903236

تخيل أن لديك أكثر من مليون سجل مع تكرارات ضبابية fuzzy duplicates. كيف يمكنك تحديد التكرارات المحتملة؟

النهج الساذج لمقارنة كل زوج من السجلات غير ممكن في مثل هذه الحالات. هذا أكثر من 10^{12} مقارنات (n^2). بافتراض سرعة 10000 مقارنة في الثانية، سيستغرق الأمر 3 سنوات تقريباً حتى تكمل.

تعمل أداة csvdedupe على حل هذه المشكلة عن طريق تقليل المقارنات بذكاء. على سبيل المثال، لا معنى لمقارنة اسم "Daniel" بـ "Philip" أو "Shannon" بـ "Julia". هم مضمونون ليكونوا سجلات مميزة.

وبالتالي، فإنه يقوم بتجميع البيانات في مجموعات أصغر بناءً على القواعد. يمكن أن تمثل إحدى القواعد في تجميع جميع السجلات بنفس الأحرف الثلاثة الأولى في الاسم.

بهذه الطريقة، فإنه يقلل بشكل كبير من عدد المقارنات بدقة كبيرة. اقرأ المزيد: [.csvdedupe](#)

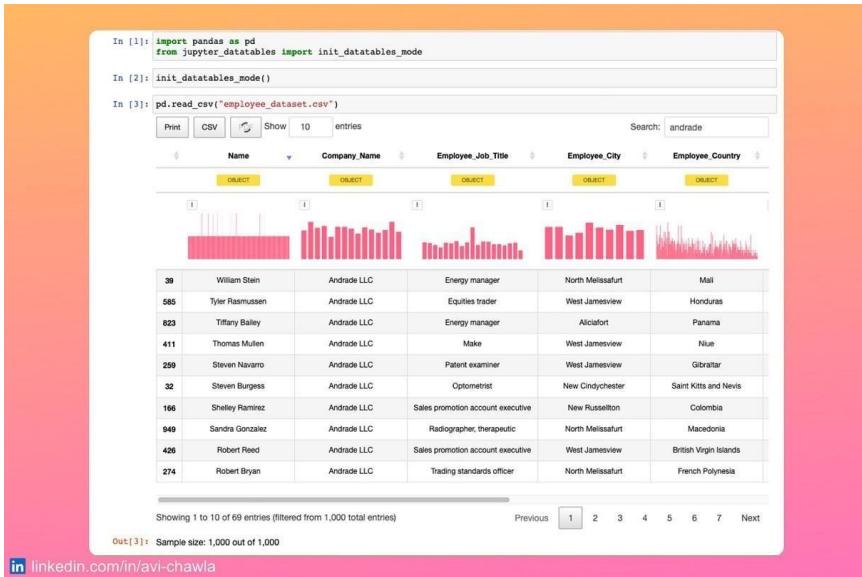
المقالة:

<https://avichawla.substack.com/p/how-would-you-identify-fuzzy-duplicates>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Terminal/Data-Deduplication.ipynb>

٩٩) أوقف معاينة إطارات البيانات الخام. بدلاً من ذلك، استخدم Stop Previewing Raw DataFrames. .DataTables Instead, Use DataTables



The screenshot shows a Jupyter Notebook interface. In the code cell (In [3]), the following Python code is run:

```
In [1]: import pandas as pd
from jupyter_datatables import init_datatables_mode

In [2]: init_datatables_mode()

In [3]: pd.read_csv('employee_dataset.csv')
```

The output cell (Out[3]) displays a preview of a DataFrame named 'employee_dataset.csv'. The DataFrame has columns: Name, Company_Name, Employee_Job_Title, Employee_City, and Employee_Country. The preview shows 10 entries from a total of 69. The data includes:

	Name	Company_Name	Employee_Job_Title	Employee_City	Employee_Country
39	William Stein	Andrade LLC	Energy manager	North Melissafurt	Mali
565	Tyler Rasmussen	Andrade LLC	Equities trader	West Jamesview	Honduras
823	Tiffany Bailey	Andrade LLC	Energy manager	Aliciafort	Panama
411	Thomas Mullen	Andrade LLC	Make	West Jamesview	Niue
259	Steven Navarro	Andrade LLC	Patent examiner	West Jamesview	Gibraltar
32	Steven Burgess	Andrade LLC	Optometrist	New Cindychester	Saint Kitts and Nevis
166	Shelley Ramirez	Andrade LLC	Sales promotion account executive	New Russellton	Colombia
949	Sandra Gonzalez	Andrade LLC	Radiographer, therapeutic	North Melissafurt	Macedonia
426	Robert Reed	Andrade LLC	Sales promotion account executive	West Jamesview	British Virgin Islands
274	Robert Bryan	Andrade LLC	Trading standards officer	North Melissafurt	French Polynesia

Below the preview, there is a search bar with the value 'andrade' and a navigation bar with buttons for 'Previous' and 'Next'.

بعد تحميل أي إطار بيانات في Jupyter، تقوم بمعاينته. لكنها بالكاد تخبر أي شيء عن البيانات. على المرء أن يتعمق أكثر من خلال تحليله، والذي يتضمن كوداً بسيطاً ولكنه متكرر. بدلاً من ذلك، استخدم [Jupyter-DataTables](#).

إنه يعزز المعاينة الافتراضية لإطارات البيانات مع العديد من العمليات المشتركة. يتضمن ذلك الفرز والترشيح والتتصدير وتحطيط توزيع الأعمدة وطباعة أنواع البيانات وتقسيم الصفحات.

يرجى عرض نسخة الفيديو هنا لفهم أفضل: [رابط المنشور](#).

المقالة:

<https://avichawla.substack.com/p/stop-previewing-raw-dataframes-instead>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/Jupyter-DataTables.ipynb>

A) سطر واحد سيجعل كود بايثون الخاص بك أسرع (100) Single Line That Will Make Your Python Code Faster

func_without_numba.py:

```
def func_without_numba():
    result = []
    for a in range(10000):
        for b in range(10000):
            if (a+b)%11 == 0:
                result.append((a,b))

func_without_numba()
# Run-time: 8.34 sec
```

func_with_numba.py:

```
from numba import njit

@njit
def func_with_numba():
    # same code

func_with_numba()
# Run-time: 0.25 sec
```

~33x Faster

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

إذا كنت محبطاً من وقت تشغيل بايثون، فإليك كيف يمكن لسطر واحد أن يجعل التعليمات البرمجية الخاصة بك سريعة للغاية.

Numba هو مترجم في الوقت المناسب (JIT) just-in-time لبايثون. هذا يعني أنه يأخذ كود بايثون موجود لديك وينشئ كوداً سريعاً للآلية (في وقت التنفيذ).

وبالتالي، التجميع اللاحق post compilation، يتم تشغيل التعليمات البرمجية الخاصة بك بسرعة كود الجهاز الأصلي. يعمل Numba بشكل أفضل على الكود الذي يستخدم مصفوفات ودوال NumPy والحلقات.

البدء: دليل [Numba](#)

المقالة:

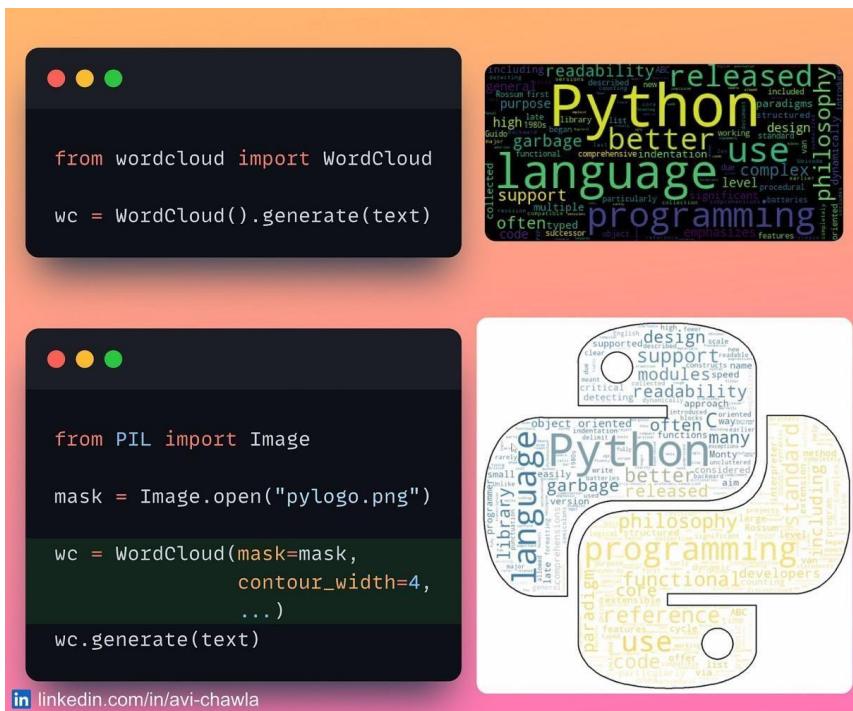
<https://avichawla.substack.com/p/faster-python-with-one-line>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Run-time%20Optimization/Numba-Faster-Python.ipynb>

101) قم بتجميل سحابة الكلمات في بايثون

Clouds In Python



[in linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

إذا كنت تستخدم سحابة الكلمات word clouds كثيراً، فالليك طريقة سريعة لجعلها أجمل.

في بايثون، يمكنك بسهولة تغيير شكل ولون سحابة الكلمات. من خلال توفير صورة قناع mask image، ستأخذ سحابة العالم الناتجة شكلها وتبدو أكثر روعة.

المقالة:

<https://avichawla.substack.com/p/pretty-word-clouds-in-python>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Pretty-Word-Cloud.ipynb>

102) كيفية ترميز الميزات الفئوية مع العديد من الأصناف؟

How to Encode Categorical Features With Many Categories



غالباً ما نقوم بترميز الأعمدة الفئوية categorical columns بترميز واحد ساخن one-hot encoding. لكن مصفوفة الميزات sparse feature matrix تصبح مت坦رة تصبح مت坦رة ولا يمكن التحكم فيها مع العديد من الفئات.

توفر مكتبة مشفرات الفئات category-encoders مجموعة من المشفرات على وجه التحديد للمتغيرات الفئوية categorical variables. هذا يجعل من الصعب تجربة تقنيات الترميز المختلفة. على سبيل المثال، استخدمت المشفر الثنائي binary encoder أعلاه لتمثيل عمود فئوي بتنسيق ثنائي.

اقرأ المزيد: [الوثيق](#).

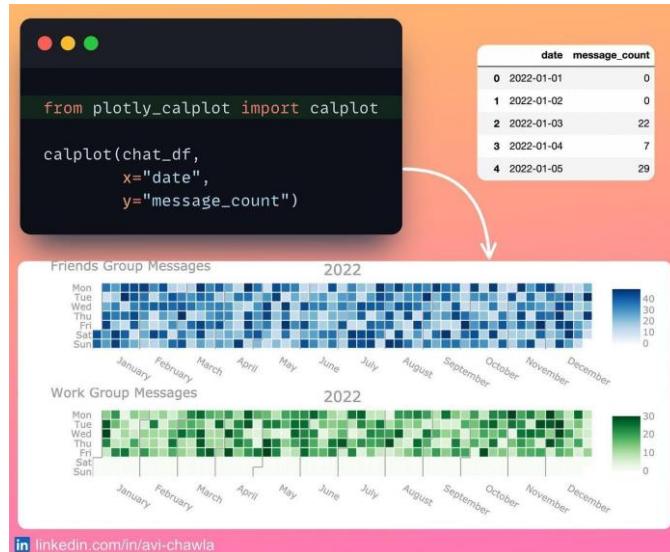
المقالة:

<https://avichawla.substack.com/p/how-to-encode-categorical-features>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Categorical-Encoding.ipynb>

103) خريطة التقويم كبدائل أغنى للمخطط الخطى Map As A Richer Alternative to Line Plot



هل رأيت واحدة من تلك الخرائط الحرارية للتقويم calendar heat maps؟ إليك كيفية إنشاء سطر من سطرين من كود بايثون.

توفر خريطة التقويم calendar map طريقة أنيقة لتصور البيانات اليومية. في بعض الأحيان، يكونون أفضل في تصوير الموسمية الأسبوعية / الشهرية في البيانات بدلًا من المخططات الخطية line plots. على سبيل المثال، تخيل إنشاء مخطط خطى لـ "رسائل مجموعة العمل Work Group Messages" أعلاه.

لإنشاء واحدة، يمكنك استخدام "plotly_calplot". يجب أن يكون إدخاله إطار بيانات DataFrame يمثل الصف القيمة المقابلة للتاريخ.

اقرأ المزيد: [Plotly Calplot](#)

المقالة:

<https://avichawla.substack.com/p/calendar-map-as-a-richer-alternative>

الكلود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Calendar-Map.ipynb>

104 أدوات آلية من تحليل البيانات الاستكشافية ستتوفر
Automated EDA 10 (الشاق) ساعات من العمل Tools That Will Save You Hours Of (Tedious) Work

10 Automated EDA

Tools That Will

Save You Hours

Of (Tedious) Work

تظل معظم الخطوات في مهمة تحليل البيانات كما هي عبر المشاريع. ومع ذلك، فإن البحث في البيانات يدوياً أمر شاق ويستغرق وقتاً طويلاً، مما يعيق الإنتاجية.

فيما يلي 10 أدوات من تحليل البيانات الاستكشافية EDA تعمل على أتمتها هذه الخطوات المتكررة وتحرير بياناتك في ثوانٍ. [رابط المنشور](#).

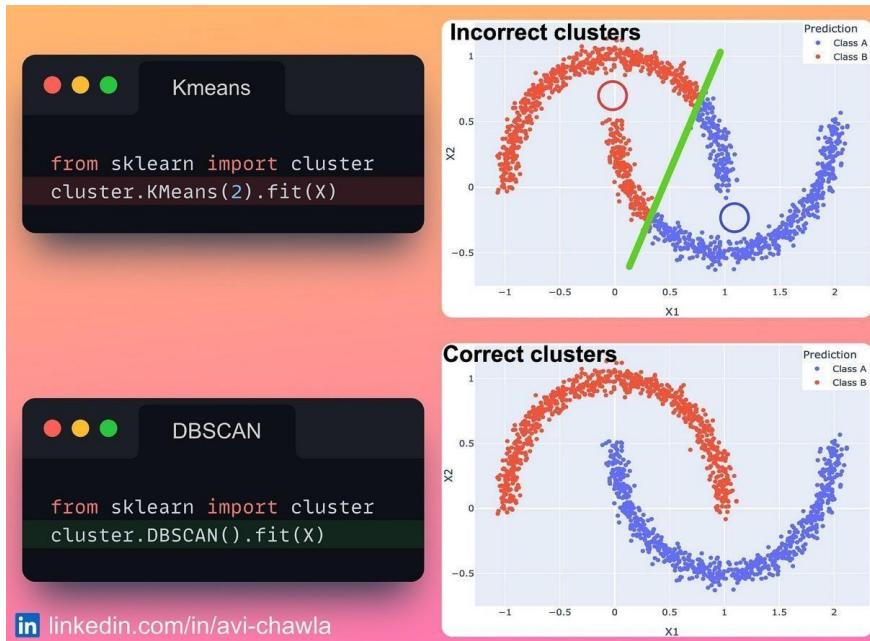
المقالة:

<https://avichawla.substack.com/p/10-automated-eda-tools-that-will>

الكود:

<https://avichawla.substack.com/p/10-automated-eda-tools-that-will>

105) لماذا قد لا تكون KMeans خوارزمية التجميع المناسبة Why KMeans May Not Be The Apt Clustering Algorithm Always



KMeans هي خوارزمية تجميع clustering شائعة. ومع ذلك، فإن قيودها يجعلها غير قابلة للتطبيق في كثير من الحالات.

على سبيل المثال، يقوم KMeans بتجميع النقاط استناداً إلى المنطقة المحلية من النقط الوسطى centroids. وبالتالي، يمكن أن تنشئ مجموعات خاطئة عندما تحتوي نقاط البيانات على أشكال عشوائية.

من بين العديد من البدائل الممكنة DBSCAN، وهي خوارزمية تجميع تعتمد على الكثافة-density based clustering algorithm. وبالتالي، يمكنه تحديد مجموعات clusters من الشكل والحجم التعسفيين.

هذا يجعلها قوية للبيانات ذات المجموعات غير الكروية والكثافات المتفاوتة.

يمكنك العثور على المزيد هنا: دليل [Sklearn](#).

المقالة:

<https://avichawla.substack.com/p/why-kmeans-may-not-be-the-apt-clustering>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/DBSCAN-vs-KMeans.ipynb>

106) ربما لم يكن تحويل LaTeX إلى Python بهذه البساطة Converting Python To LaTeX Has Possibly Never Been So Simple

```

import latexify
import math

@latexify.function Add decorator
def roots(a, b, c):
    return (-b + math.sqrt(b**2 - 4*a*c)) / (2*a)

roots
roots(a, b, c) = 
$$\frac{-b + \sqrt{b^2 - 4ac}}{2a}$$


@latexify.function
def fib(n):
    if n<2:
        return 1
    else:
        return fib(n-1) + fib(n-2)

fib
fib(n) = 
$$\begin{cases} 1, & \text{if } n < 2 \\ fib(n-1) + fib(n-2), & \text{otherwise} \end{cases}$$


```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

إذا كنت تري عرض كود بايثون وإخراجه كـ LaTeX، فجرّب `latexify_py`. باستخدام هذا، يمكنك طباعة كود بايثون كتعبير LaTeX وجعل كودك أكثر قابلية للتفسير interpretable .
علاوة على ذلك، يمكنه أيضًا إنشاء كود LaTeX لكود بايثون. هذا يوفر الكثير من الوقت والجهد في كتابة التعبيرات يدوياً في LaTeX.

يمكنك العثور على مزيد من المعلومات هنا: [المستودع](#).

المقالة:

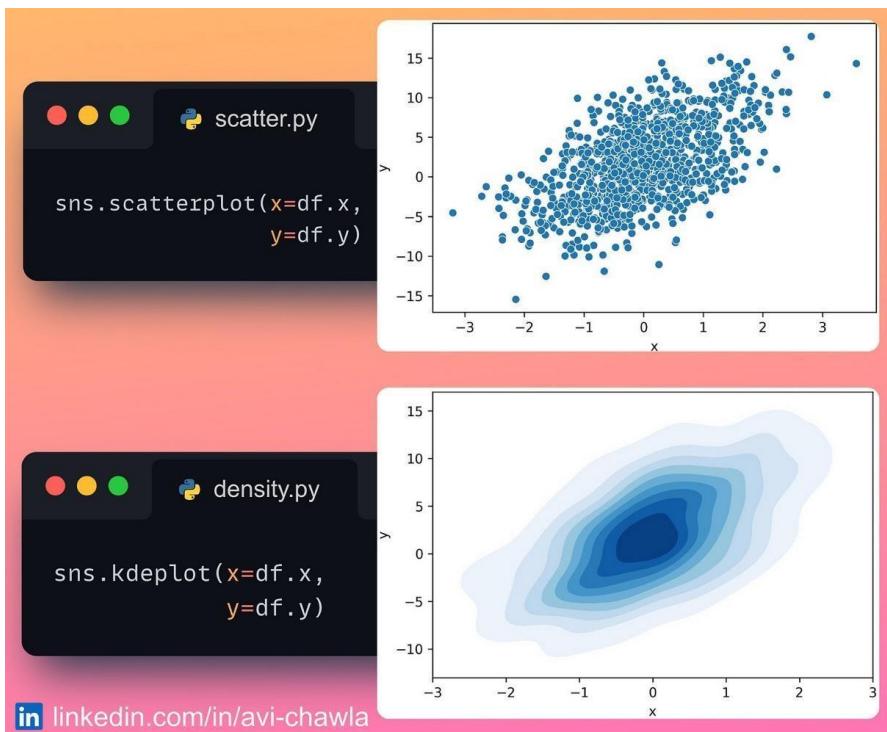
<https://avichawla.substack.com/p/converting-python-to-latex-has-possibly>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/Python-To-LaTeX.ipynb>

107) مخطط الكثافة كبديل أغنى لمخطط التشتت

Plot As A Richer Alternative to Scatter Plot



المخططات المبعثرة (التشتت) مفيدة للغاية لتصور مجموعتين من المتغيرات العددية. ولكن عندما يكون لديك، على سبيل المثال، آلاف نقاط البيانات، يمكن أن تصبح المخططات المبعثرة كثيفة للغاية بحيث لا يمكن تفسيرها.

يمكن أن يكون مخطط الكثافة density plot خياراً جيداً في مثل هذه الحالات. يصور توزيع النقاط باستخدام الألوان (أو المنسوب contours). هذا يجعل من السهل تحديد المناطق ذات الكثافة العالية والمنخفضة.

علاوة على ذلك، يمكن أن تكشف بسهولة عن مجموعات من نقاط البيانات التي قد لا تكون واضحة في مخطط التشتت.

اقرأ المزيد: [المستندات](#).

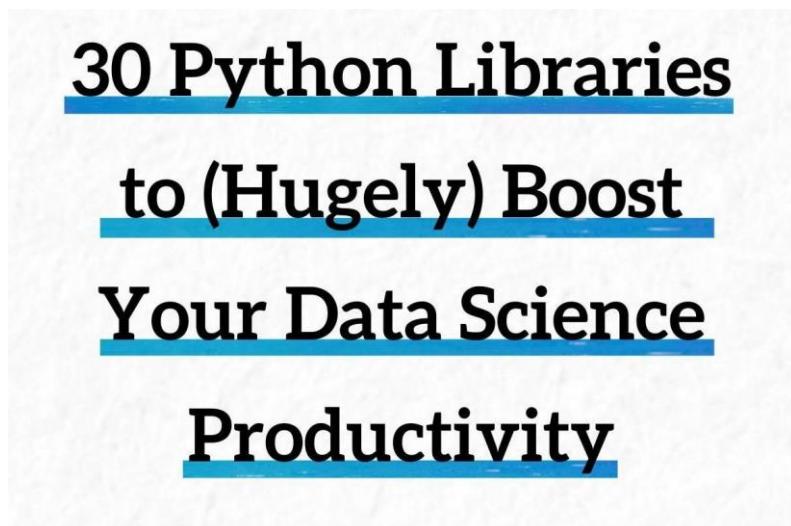
المقالة:

<https://avichawla.substack.com/p/density-plot-as-a-richer-alternative>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/Density-Plot.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Density-Plot.ipynb)

108) 30 مكتبة بايثون (بشكل كبير) لتعزيز إنتاجية علم البيانات الخاصة بك 30 Python Libraries to (Hugely) Boost Your Data Science Productivity



فيما يلي مجموعة من 30 مكتبة أساسية لعلم البيانات مفتوحة المصدر. لكل منها حالة استخدام خاصة به وإمكانات هائلة لزيادة مهاراتك في علوم البيانات. [الرابط](#).

المقالة:

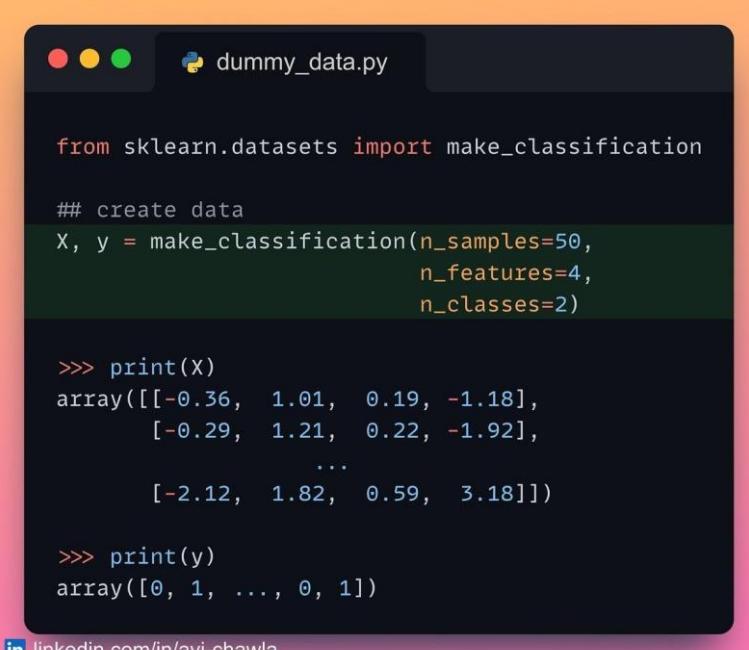
<https://avichawla.substack.com/p/30-python-libraries-to-hugely-boost>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Documents/30%20Python%20Libraries%20to%20\(Hugely\)%20Boost%20Your%20Data%20Science%20Productivity.pdf](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Documents/30%20Python%20Libraries%20to%20(Hugely)%20Boost%20Your%20Data%20Science%20Productivity.pdf)

Sklearn واحد لـ Sklearn لتوليد البيانات التركيبية (109)

One-liner to Generate Synthetic Data



```

from sklearn.datasets import make_classification

## create data
X, y = make_classification(n_samples=50,
                           n_features=4,
                           n_classes=2)

>>> print(X)
array([[-0.36,  1.01,  0.19, -1.18],
       [-0.29,  1.21,  0.22, -1.92],
       ...
       [-2.12,  1.82,  0.59,  3.18]])

>>> print(y)
array([0, 1, ..., 0, 1])

```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

في كثير من الأحيان لاختبار / بناء خط أنابيب البيانات، قد تحتاج إلى بعض البيانات الوهمية `dummy`.data

باستخدام Sklearn، يمكنك بسهولة إنشاء مجموعة بيانات وهمية لمهام الانحدار `regression` والتصنيف `classification` والتجميع `clustering`.

مزيد من المعلومات هنا: [وثائق Sklearn](#)

المقالة:

<https://avichawla.substack.com/p/sklearn-one-line-to-generate-synthetic>

ال코드:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Sklearn/Sklearn-Synthetic-Data.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Sklearn/Sklearn-Synthetic-Data.ipynb)

11) قم بتسمية بياناتك بنقرة زر The Click Of A Button



```
In [3]: from ipyannotate import annotate
from ipyannotate.buttons import ValueButton as Button

In [5]: annotation = annotate(images_data,
                           buttons=[Button('Dog'), # label buttons list
                                     Button('Cat'))]
annotation

Dog Cat
====

In [6]: labels = [task.value for task in annotation.tasks] # get labels
labels

Out[6]: ['Cat', 'Dog', 'Dog', 'Cat']
```

in linkedin.com/in/avi-chawla

في كثير من الأحيان مع البيانات غير المسممة (غير المصنفة) unlabeled data، قد يضطر المرء إلى قضاء بعض الوقت في التعليق `annotating` عليها / تصنيفها `.labeling`.

للقيام بذلك بسرعة في نوتبوك jupyter، استخدم **ipyannotate**. باستخدام هذا، يمكنك إضافة تعليق توضيحي لبياناتك بمجرد النقر فوق الزر المقابل.

اقرأ المزيد: [.ipyannotate](#)

شاهد نسخة فيديو من هذا المنشور على LinkedIn: رابط النشر.

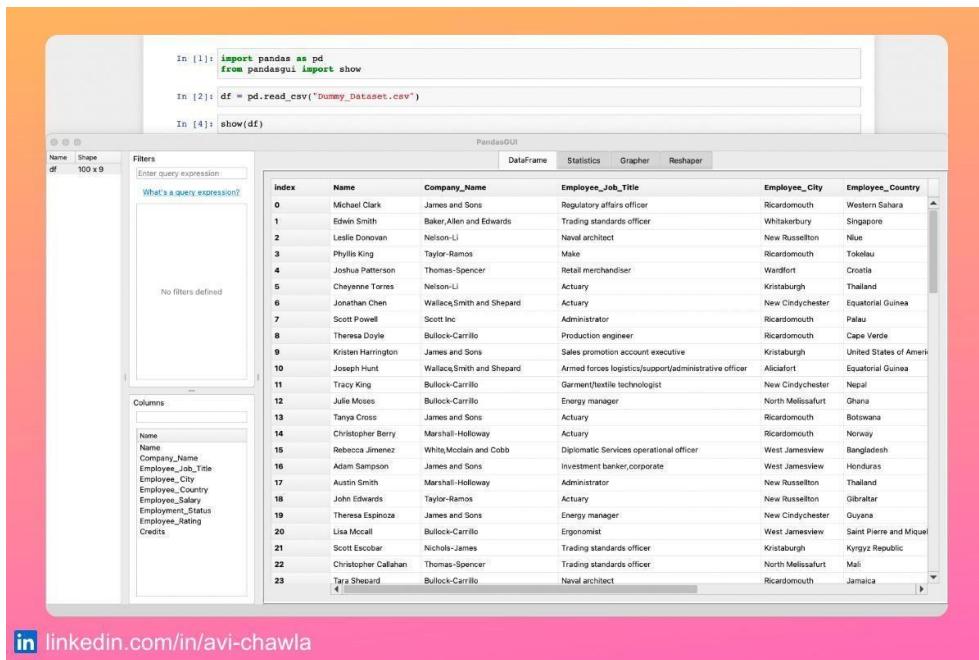
المقالة:

<https://avichawla.substack.com/p/label-your-data-with-the-click-of>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/annotate-data.ipynb>

111) تحليل إطار بيانات Pandas بدون كود DataFrame Without Code



[in linkedin.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

إذا كنت ترغب في تحليل إطار البيانات الخاص بك في تطبيق قائم على واجهة المستخدم الرسومية-GUI، فجرب Pandas based application GUI. يوفر واجهة مستخدم رسومية أنيقة لعرض مجموعات البيانات المجدولة وترسيحها وفرزها ووصفها، وما إلى ذلك.

علاوة على ذلك، باستخدام وظيفة السحب والإفلات البديهية، يمكنك بسهولة إنشاء مجموعة متنوعة من المخططات وتصديرها ككود.

شاهد نسخة فيديو من هذا المنشور على [LinkedIn](#)

المقالة:

<https://avichawla.substack.com/p/analyze-a-pandas-dataframe-without>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandas-GUI.ipynb>

112) سطر واحد بايثون لإنشاء مخططات مرسومة يدوياً Python One-Liner To Create Sketchy Hand-drawn Plots



تشتهر xkcd Comic بأسلوبها غير الرسمي والفكاهي، فضلاً عن اشكالها اللاصقة والرسومات البسيطة.

يعد إنشاء مثل هذه المخططات المرسومة يدوياً الجذابة بصرياً أمراً بسيطاً جدًا باستخدام matplotlib. فقط ضع مسافة بادئة للكود في سياق plt.xkcd() لعرضها بأسلوب فكاهي.

هل لاحظ أن هذا النمط يستخدم فقط لتحسين جماليات المخطط من خلال تأثيرات مرسومة باليد. ومع ذلك، لا يوصى باستخدامه في العروض التقديمية الرسمية والمنشورات hand-drawn effects وما إلى ذلك.

اقرأ المزيد: [الوثائق](#).

المقالة:

<https://avichawla.substack.com/p/python-one-liner-to-create-sketchy>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/xkcd-style-plots.ipynb>

70 Pandas (113 x Faster Pandas By Changing 70 Lines of Code)

The image shows two Jupyter notebook environments side-by-side. The left notebook, titled 'Pandas.py', contains Python code for reading a large dataset and concatenating it. The right notebook, titled 'Modin.py', contains similar code but uses the Modin library instead of Pandas. Arrows point from the 'Import Statement' in the Pandas code to the 'Import Statement' in the Modin code, and from the '7 GB Dataset' text to the Modin code's performance metrics.

```

Pandas.py:
import pandas as pd

data = "file.csv" ## 2M Rows

df = pd.read_csv(data)
## 3.6 sec

pd.concat([df for _ in range(20)])
## 7.1 sec

7 GB
Dataset

```

```

Modin.py:
import modin.pandas as pd

data = "file.csv" ## 2M Rows

df = pd.read_csv(data)
## 1.3 sec (2.75x Faster)

pd.concat([df for _ in range(20)])
## 0.1 sec (70x Faster)

Modify Import Statement

```

linkedin.com/in/avi-chawla

من الصعب العمل على مجموعات البيانات الكبيرة في Pandas. هنا، في بعض الأحيان، يتطلب الكثير من التحسين ويمكن أن يصبح مملاً مع نمو مجموعة البيانات بشكل أكبر.

بدلاً من ذلك، جرب Modin. تقدم تحسينات فورية دون بذل جهد إضافي. قم بتغيير بيان الاستيراد واستخدمه مثل Pandas API، مع تسريع مهم.

إقرأ المزيد: دليل [Modin](#)

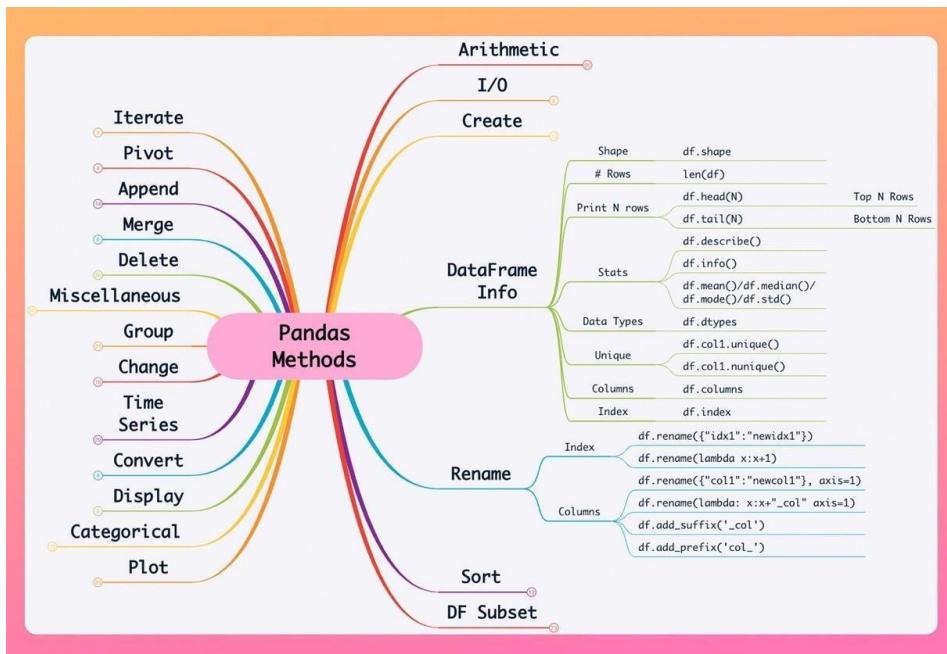
المقالة:

<https://avichawla.substack.com/p/70x-faster-pandas-by-changing-just>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Memory%20Optimization/70x-Faster-Pandas.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Memory%20Optimization/70x-Faster-Pandas.ipynb)

An Interactive Guide To Master Pandas In One Go



إليك خريطة ذهنية توضح طرق Pandas في صفحة واحدة. كم واحدة تعرف:

- تحميل / حفظ Load/Save
- عامل تصفية Filter
- معلومات إطار بيانات DataFrame info
- دمج Merge
- مخطط Time-series
- السلسلة الزمنية Plot
- وغيرها الكثير في خريطة واحدة.

ابحث عن الرسم التخطيطي الكامل هنا: [خريطة ذهنية للباندا](#)

115) اجعل التدوين النقطي أكثر قوة في بايثون Notation More Powerful in Python

```

class Square:
    def __init__(self, length):
        self._side = length

    @property
    def side(self):
        return self._side

    @side.setter
    def side(self, length):
        if length<0:
            raise ValueError("Side cannot be negative")
        else:
            self._side = length

```

Raises errors during assignment

```

>>> s = Square(10)
>>> s.side # Getter
10

>>> s.side = -2 # Setter (with dot)
ValueError: Side cannot be negative

```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

يوفر التدوين النقطي Dot notation طريقة بسيطة وأنيقة للوصول إلى سمات مثيل وتعديلها. ومع ذلك، فمن الممارسات البرمجية الجيدة استخدام طريقة getter و setter لمثل هذه الأغراض. هذا لأنّه يوفر مزيداً من التحكم في كيفية الوصول إلى السمات / تغييرها.

للاستفادة من كليهما في بايثون، استخدم **@property** decorator. نتيجة لذلك، يمكنك استخدام التدوين النقطي ولا يزال لديك تحكم صريح في كيفية الوصول / تعين السمات.

المقالة:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/Powerful-Dot-Notation.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Powerful-Dot-Notation.ipynb)

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/Powerful-Dot-Notation.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Powerful-Dot-Notation.ipynb)

The Coolest Jupyter Notebook Hack (116) أروع اختراق لـ Jupyter Notebook Hack

The screenshot shows a Jupyter Notebook interface with the following code and outputs:

```

In [1]: import numpy as np
In [2]: np.array([1,2,3])
Out[2]: array([1, 2, 3])

1 In [3]: _2
Out[3]: array([1, 2, 3])

2 In [4]: Out[2]
Out[4]: array([1, 2, 3])

3 In [5]: _oh[2]
Out[5]: array([1, 2, 3])

```

Three arrows labeled 1, 2, and 3 point from the right side of the screen to the corresponding code cells (In [3], In [4], and In [5]).

linkedin.com/in/avi-chawla

هل نسيت يوماً تخصيص النتائج لمتغير في Jupyter؟ بدلاً من إعادة حساب النتيجة عن طريق إعادة تشغيل الخلية، إليك ثلاث طرق لاسترداد الإخراج.

1) استخدم الشرطة السفلية متبوعة بفهرس خلية الإخراج.

.key 2/3) استخدم قاموس **Out** أو **_oh** وحدد فهرس خلية الإخراج كمفتاح .key

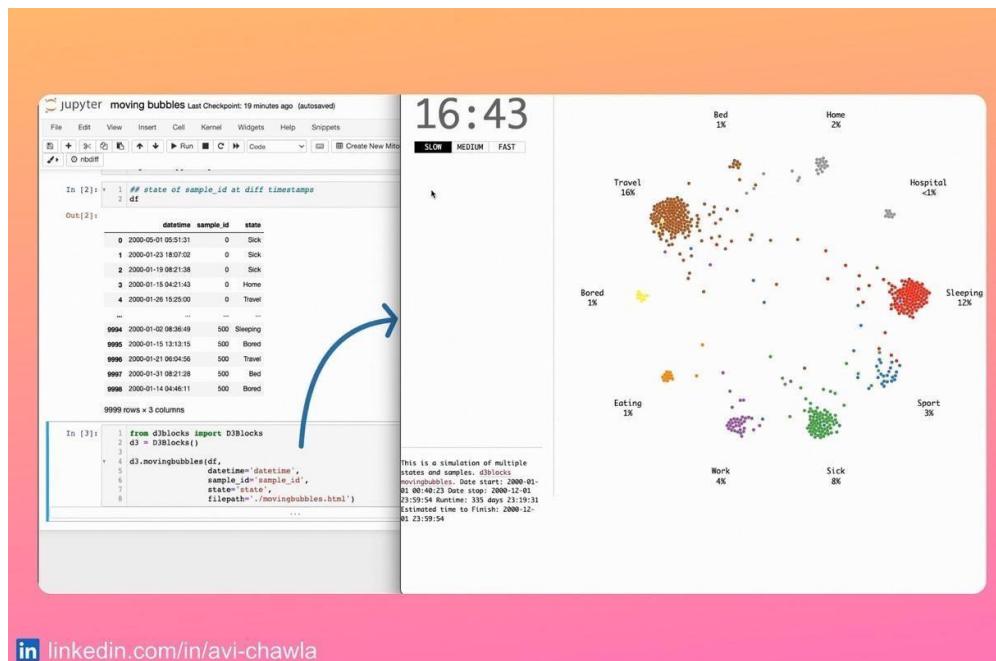
المقالة:

<https://avichawla.substack.com/p/the-coolest-jupyter-notebook-hack>

الكلود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/Retrieve-Output-Three-Ways.ipynb>

117) قم بإنشاء مخطط الفقاعي المتحرك في بايثون Create a Moving Bubbles Chart in Python



[in linkedin.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

هل رأيت واحدة من تلك الرسوم البيانية للنقط المتحركة moving points charts؟ إليك كيفية إنشاء واحد في بايثون في ثلاثة أسطر فقط من التعليمات البرمجية.

يُعد المخطط الفقاعي المتحرك Moving Bubbles chart طريقة أنيقة لتصوير تحركات الكيانات عبر الوقت. باستخدام هذا، يمكننا بسهولة تحديد متى تظهر المجموعات في بياناتنا وفي أي حالة (حالات).

لإنشاء واحدة، يمكنك استخدام "d3blocks". يجب أن يكون إدخالها إطار بيانات. يمثل الصف حالة عينة في طابع زمني معين.

المقالة:

<https://avichawla.substack.com/p/create-a-moving-bubbles-chart-in>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Moving-Bubbles.ipynb>

استخدام Scikit-Learn API على نماذج Scikit-Learn API :Skorch (118)

Skorch: Use Scikit-learn API on PyTorch PyTorch Models

Define Pytorch model

```
class MyModel(nn.Module):
    def __init__(self):
        ## Define Network

    def forward(self, x):
        ## Forward Pass
```

Use Scikit-learn API on model

```
from skorch import NeuralNetClassifier

model = NeuralNetClassifier(
    MyModel,
    lr=0.1,
    criterion=nn.MSELoss
)

model.fit(X, y)
preds = model.predict(X)
```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

skorch هي مكتبة عالية المستوى لـ PyTorch توفر تواافق Scikit-Learn الكامل. بمعنى آخر، فهو يجمع بين قوة PyTorch وأناقة sklearn.

وبالتالي، يمكنك تدريب نماذج Pytorch بطريقة تشبه Scikit-Learn، باستخدام دوال مثل الملازمة `fit`، `predict`، `score`، `result`، إلخ.

باستخدام skorch، يمكنك أيضًا وضع نموذج PyTorch في خط أنابيب sklearn وغير ذلك الكثير. بشكل عام، تهدف إلى أن تكون مرنة مثل PyTorch بينما تتمتع بواجهة نظيفة مثل .sklearn.

اقرأ المزيد: [الوثيق](#).

المقالة:

<https://avichawla.substack.com/p/skorch-use-scikit-learn-api-on-pytorch>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Machine%20Learning/Skorch.ipynb>

٪90) تقليل استخدام الذاكرة لإطار بيانات Pandas بنسبة ٪90 Reduce Memory Usage Of A Pandas DataFrame By 90%

The screenshot shows a Jupyter Notebook interface. On the left, Python code is displayed:

```

## df.shape: (10^7, 2)
>>> df.A.dtype
dtype('int64')
## Range: [-2^63, 2^63-1]
>>> df.A.min(), df.A.max()
(1, 100)
>>> df.A.memory_usage()
76.3 MB

```

A table on the right shows the DataFrame structure:

	A	B
0	38	46
1	28	58
2	47	82
3	88	87
4	13	78

Annotations on the left side of the code explain the current state:

- Supported range larger than required (points to the range annotation)
- Convert to smaller datatype (points to the conversion code)

On the right, the result of the conversion is shown:

```

df["A"] = df.A.astype(np.int8)
## Range: [-128, 127]
>>> df.A.memory_usage()
9.5 MB # (~90% Lower)

```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

بشكل افتراضي، تقوم Pandas دائمًا بتعيين أعلى نوع بيانات ذاكرة لأعمدتها. على سبيل المثال، يحصل العمود ذو القيمة الصحيحة دائمًا على نوع البيانات int64، بغض النظر عن نطاق.

لتقليل استخدام الذاكرة، قم بتمثيلها باستخدام نوع بيانات محسن، وهو ما يكفي لتوسيع نطاق العقيم في أعمدتك.

اقرأ هذه [المدونة](#) لمزيد من المعلومات. تفاصيل العديد من التقنيات لتحسين استخدام الذاكرة من إطار بيانات Pandas.

المقالة:

<https://avichawla.substack.com/p/reduce-memory-usage-of-a-pandas-dataframe>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Memory%20Optimization/Pandas-Optimize-Datatype.ipynb>

An (120) طريقة أنيقة لاداء مهام إيقاف التشغيل في بايثون Elegant Way To Perform Shutdown Tasks in Python

The code in 'my_file.py' is:

```
import atexit
@atexit.register
def final_function():
    print("COMPLETED EXECUTION!")

for i in range(5):
    print(f"num = {i}")
```

The terminal output shows:

```
$ python my_file.py
num = 0
num = 1
num = 2
num = 3
num = 4
COMPLETED EXECUTION!
```

Add decorator to method

The decorator invokes the function at the end

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

غالبًا في نهاية تنفيذ البرنامج، تقوم بتنفيذ بعض المهام الأساسية مثل حفظ الكائنات وطباعة السجلات وما إلى ذلك.

لاستدعاء طريقة مباشرة قبل إغلاق المترجم الفوري، قم بتزويدها باستخدام المصمم **.@atexit.register** decorator

الشيء الجيد هو أنه يعمل حتى إذا تم إنهاء البرنامج بشكل غير متوقع. وبالتالي، يمكنك استخدام هذه الطريقة لحفظ حالة البرنامج أو طباعة أي تفاصيل ضرورية قبل توقيفه.

اقرأ المزيد: [التوثيق](#).

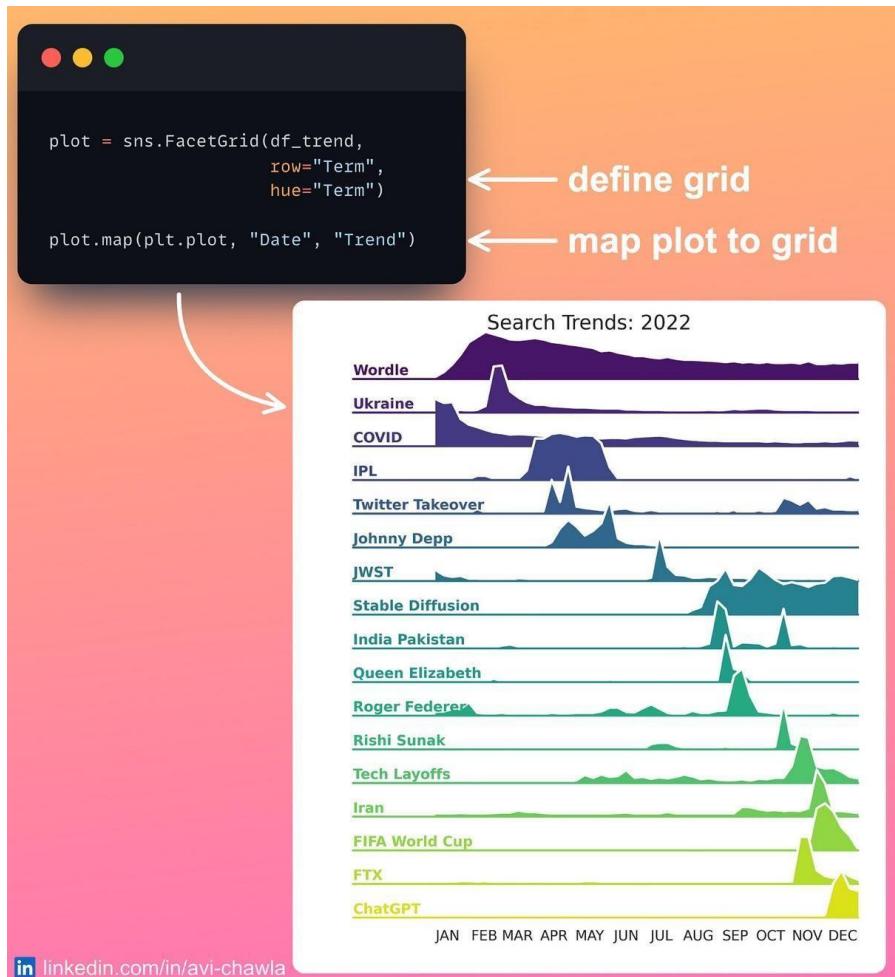
المقالة:

<https://avichawla.substack.com/p/an-elegant-way-to-perform-shutdown>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/atexit-register-decorator.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/atexit-register-decorator.ipynb)

121) تصور اتجاهات بحث Google لعام 2022 باستخدام بايثون Visualizing Google Search Trends of 2022 using Python



إذا كانت بياناتك تحتوي على العديد من المجموعات، فإن تصور توزيعها معًا يمكن أن يؤدي إلى إنشاء مخططات مزدحمة. هذا يجعل من الصعب تصور الأنماط الأساسية.

بدلاً من ذلك، ضع في اعتبارك تخطيط التوزيع عبر المجموعات الفردية باستخدام FacetGrid. يتبع لك هذا مقارنة توزيعات مجموعات متعددة جنباً إلى جنب ومعرفة كيف تختلف.

كما هو موضح أعلاه، يتيح لنا FacetGrid أن نرى بوضوح كيف اتجهت مصطلحات البحث المختلفة عبر عام 2022.

ملاحظة. لقد استخدمت مستودع اتجاهات البحث العام لجلب بيانات الاتجاه data.trend.

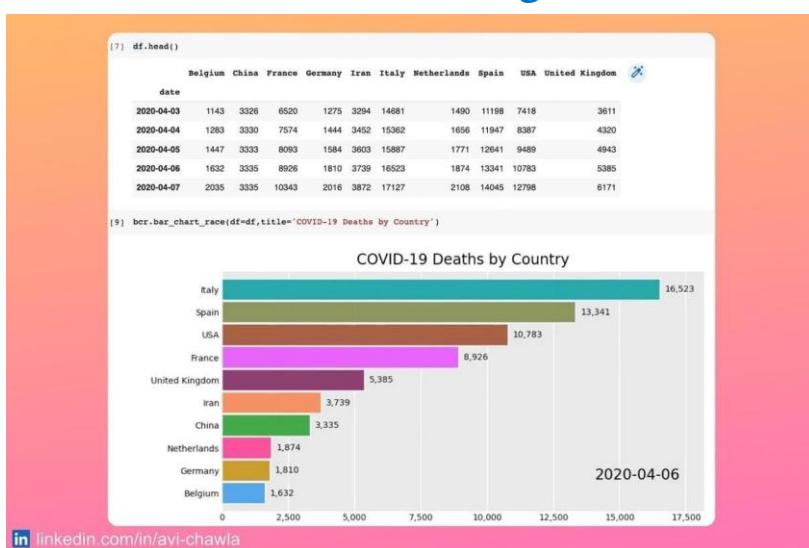
المقالة:

<https://avichawla.substack.com/p/visualizing-google-search-trends>

الكلود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/FacetGrid.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/FacetGrid.ipynb)

122) قم بإنشاء مخطط شريط السباق في بايثون Create A Racing Bar Chart In Python



هل رأيت أحد مخططات شريط السباق racing bar charts هذه؟ إليك كيفية إنشاء واحد في بايثون في سطرين فقط من التعليمات البرمجية.

عادةً ما يتم استخدام مخطط شريط السباق لتوضيح تقدم القيم المتعددة بمرور الوقت.

لإنشاء واحدة، يمكنك استخدام مكتبة "**bar-chart-race**".

يجب أن يكون مدخلاته عبارة عن إطار بيانات Pandas حيث يمثل كل صف طابعًا زمنياً واحداً. يحتوي العمود على القيم المقابلة لفئة معينة.

اقرأ المزيد: [الوثيق](#).

المقالة:

<https://avichawla.substack.com/p/create-a-racing-bar-chart-in-python>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Race-Bar-Chart.ipynb>

Speed-up NumPy 5 مرات مع Pandas Apply (123) تسريع Pandas Apply 5x with NumPy

```
Pandas Apply
```

```
def assign_class(num):
    if num<10:
        return "Class A"
    if num<50:
        return "Class B"
    return "Class C"

df.A.apply(assign_class)
## 1.02 s ± 20.5 ms per loop
```

	A	B	C	D
0	19	80	39	36
1	20	97	47	9
2	3	63	16	69
3	68	20	58	37
4	63	71	51	32

10⁷ rows


```
NumPy Select
```

```
condlist = [ df["A"]<10 , df["A"]<50 ]
resultlist = [ "Class A" , "Class B" ]

np.select(condlist, resultlist, "Class C")
## 0.20 s ± 7.14 ms per loop
```

~5x Faster

Default

[in](https://linkedin.com/in/avi-chawla) linkedin.com/in/avi-chawla

أثناء إنشاء أعمدة شرطية conditional columns في Pandas، نميل إلى استخدام طريقة `apply()` التي تستغرق وقتاً طوالاً تقريرياً.

ومع ذلك، فإن `apply()` في Pandas ليست سوى حلقة متآلة. نتيجة لذلك، فإنه يخطئ الهدف الكامل من التوجيه .vectorization.

بدلاً من ذلك، يجب عليك استخدام طريقة `np.select()` لإنشاء أعمدة شرطية. إنها تقوم بنفس الوظيفة ولكنها سريعة للغاية.

يتم تمرير الشروط والنتائج المقابلة لها كأول وسيطين. الوسيطة الأخيرة هي النتيجة الافتراضية.

اقرأ المزيد هنا: مستندات [NumPy](#)

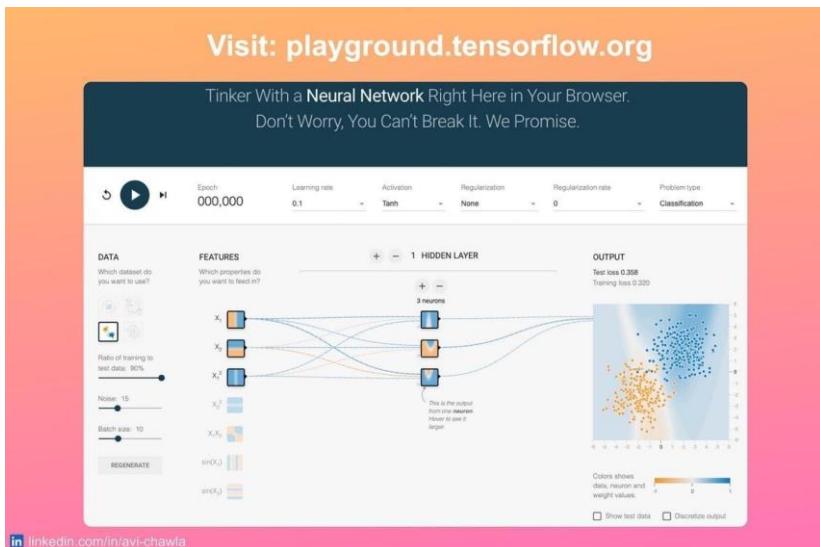
المقالة:

<https://avichawla.substack.com/p/speed-up-pandas-apply-5x-with-numpy>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Pandas/Pandas-Apply-With-Numpy-Select.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandas-Apply-With-Numpy-Select.ipynb)

124) أداة على الإنترنت بدون كود لاستكشاف وفهم الشبكات العصبية A No-Code Online Tool To Explore and Understand Neural Networks



يمكن أن تكون الشبكات العصبية Neural networks مخيفة للمبتدئين. أيضًا، لا يوفر التجربة برمجيًا ما يكفي من الفهم الحدسي عنها.

بدلاً من ذلك، جرب TensorFlow Playground. تسمح لك واجهة المستخدم الأنيقة ببناء الشبكات العصبية وتدربيها وتصورها دون أي كود.

ببعض نقرات، يمكن للمرء أن يرى كيف تعمل الشبكات العصبية وكيف تؤثر المعلمات الفائقة hyperparameters المختلفة على أدائها. هذا يجعلها مفيدة بشكل خاص للمبتدئين.

جرب هنا: [TensorFlow Playground](#)

المقالة:

<https://avichawla.substack.com/p/a-no-code-online-tool-to-explore>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/Understand-Neural-Net.ipynb>

What Are Class Methods and When To Use Them

The diagram illustrates the definition and usage of class methods. It shows two code snippets in a terminal-like interface.

Top Snippet:

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

    @classmethod
    def from_square(cls, size):
        return Rectangle(size, size)
```

A callout arrow labeled "Define classmethod" points to the `@classmethod` decorator in the first snippet.

Bottom Snippet:

```
create object
using classmethod →
rect = Rectangle.from_square(5)

print(rect.width) # Output: 5
print(rect.height) # Output: 5
```

A callout arrow labeled "create object using classmethod" points to the line `rect = Rectangle.from_square(5)` in the second snippet.

linkedin.com/in/avi-chawla

طرق الفئة Class methods، كما يوحى الاسم، مرتبطة بالفئة (الكلاس) وليس بمثيلاتها instances. إنها مفيدة بشكل خاص ل توفير واجهة بديلة لإنشاء المثلثات.

علاوة على ذلك، يمكن استخدامها أيضاً لتحديد دوال المرافق المرتبطة بالفئة بدلاً من مثيلاتها.

على سبيل المثال، يمكن للمرء تحديد طريقة فئة تُرجع قائمة بجميع مثيلاتها. استخدام آخر يمكن أن يكون لحساب إحصائية على مستوى الفئة بناءً على الحالات.

لتحديد طريقة الفئة في بايثون، استخدم `@classmethod` decorator. نتيجة لذلك، يمكن استدعاء هذه الطريقة مباشرة باستخدام اسم الفئة.

المقالة:

<https://avichawla.substack.com/p/what-are-class-methods-and-when-to>

الكلود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/class-methods.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/class-methods.ipynb)

Make Sklearn KMeans أسرع 20 مرة (126)

KMeans 20x times faster

The image shows two terminal windows side-by-side. The left window, titled 'sklearn.py', contains Python code for training a KMeans model with 8 clusters on a dataset 'x_train' (shape: 500000, 1024). The command is 'kmeans = KMeans(8).fit(x_train)' and it includes a note '# Training Time: 162s'. The right window, titled 'faiss.py', contains similar code using the 'faiss' library: 'import faiss', 'kmeans = faiss.Kmeans(d=1024, k=8)', and 'kmeans.train(x_train)'. It also includes '# Training Time: 7.8s'. A pink arrow points from the 'faiss.py' window to the 'sklearn.py' window, with the text '~20x Faster' written above the arrow. Below the windows, there is a LinkedIn profile link: linkedin.com/in/avi-chawla.

```
from sklearn.cluster import KMeans
kmeans = KMeans(8).fit(x_train)
# Training Time: 162s
```

```
import faiss
kmeans = faiss.Kmeans(d=1024, k=8)
kmeans.train(x_train)
# Training Time: 7.8s
```

تُستخدم خوارزمية KMeans بشكل شائع لتجميع البيانات غير المسممة unlabeled data. ولكن مع مجموعات البيانات الكبيرة، يستغرق scikit-Learn الكثير من الوقت للتدريب والتنبؤ.

لتسريع برنامج KMeans، استخدم Faiss من Facebook AI Research. يوفر بحثاً وتجميعاً أسرع في أقرب الجيران.

يستخدم Faiss "الفهرس المقلوب Inverted Index"، وهي بنية بيانات محسنة لتخزين وفهرسة نقاط البيانات. وهذا يجعل إجراء التجميع فعالاً للغاية.

بالإضافة إلى ذلك، يوفر Faiss موازاة ودعم GPU، مما يحسن أداء خوارزميات التجميع الخاصة به.

اقرأ المزيد: [GitHub](#)

المقالة:

<https://avichawla.substack.com/p/speed-up-numpy-20x-with-numexpr>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Run-time%20Optimization/Faiss-vs-Sklearn-KMeans.ipynb>

Speed-up Numexpr 20 مرة مع NumPy (127) تسريع NumPy 20x with Numexpr

```

import numpy as np
import numexpr as ne

a = np.random.random(10**7)
b = np.random.random(10**7)

%timeit np.cos(a) + np.sin(b)
142 ms ± 257 µs per loop

%timeit ne.evaluate("cos(a) + sin(b)")
32.5 ms ± 229 µs per loop ~5x Faster

```

in [linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

تقدم Numpy بالفعل عمليات موجهة محسنة وسريعة. ومع ذلك، فهو لا يدعم التوازي parallelism. يوفر هذا مجالاً إضافياً لتحسين وقت تشغيل NumPy.

للقىام بذلك، استخدم Numexpr. يتيح لك تسريع العمليات الحسابية العددية من خلال التجميع للقيام بذلك، استخدم Numexpr. يتيح لك تسريع العمليات الحسابية العددية من خلال التجميع متعدد الخيوط multi-threading والتجميع في الوقت المناسب compilation just-in-time. عادة، من المتوقع اعتماداً على مدى تعقيد التعبير، يمكن أن تتراوح عمليات التسريع من 0.95x و 20x. عادة، من المتوقع أن يكون 2x-5x.

اقرأ المزيد: [التوثيق](#).

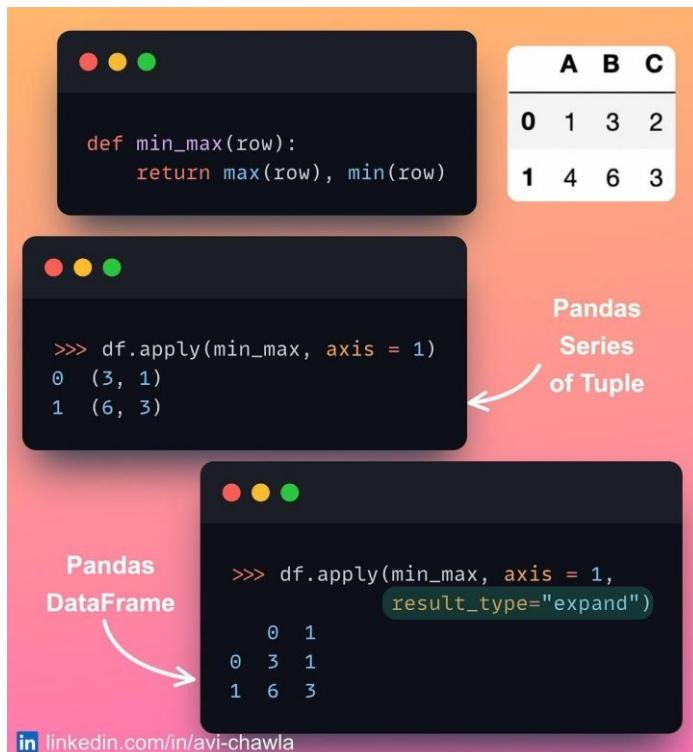
المقالة:

<https://avichawla.substack.com/p/speed-up-numpy-20x-with-numexpr>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/NumPy/Speed-up-NumPy.ipynb>

A Lesser- Known Feature of Apply Method In Pandas



بعد تطبيق طريقة على إطار بيانات DataFrame، غالباً ما نرجع قيمًا متعددة على هيئة صفات tuple. يتطلب هذا خطوات إضافية لعرضه مرة أخرى كأعمدة منفصلة.

بدلاً من ذلك، باستخدام الوسيطة `result_type`، يمكنك التحكم في الشكل ونوع الإخراج. حسب الرغبة، يمكن أن يكون الإخراج إما إطار بيانات أو سلسلة Series.

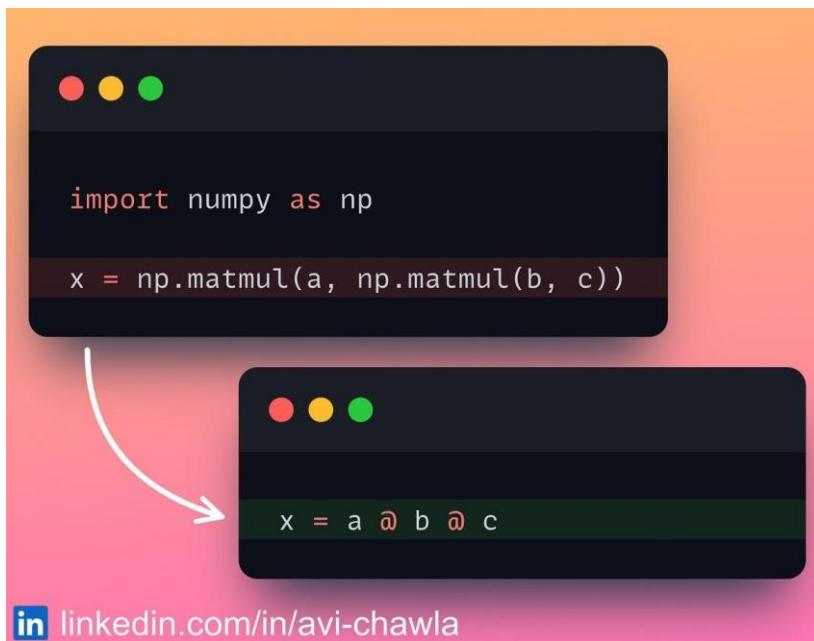
المقالة:

<https://avichawla.substack.com/p/a-lesser-known-feature-of-apply-method>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Apply-DataFrame-Output.ipynb>

An Elegant Way (129) طريقة أنيقة لأداء ضرب المصفوفة To Perform Matrix Multiplication



[in linkedin.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

يعد ضرب المصفوفة Matrix multiplication عملية شائعة في التعلم الآلي. ومع ذلك، فإن تسلسل عمليات الضرب المتكررة باستخدام الدالة **matmul** يجعل الكود مشوشًا وغير قابل للقراءة. إذا كنت تستخدم NumPy، فيمكنك بدلاً من ذلك استخدام عامل التشغيل `@` للقيام بنفس الشيء.

المقالة:

<https://avichawla.substack.com/p/an-elegant-way-to-perform-matrix>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/NumPy/Matrix-Multiplication-Operator.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/NumPy/Matrix-Multiplication-Operator.ipynb)

Create Dataclass إطار بيانات من Pandas (129) قم بإنشاء إطار بيانات من Pandas DataFrame from Dataclass

```

from dataclasses import dataclass

@dataclass
class Point:
    x_loc:int
    y_loc:int

```

define
dataclass

```

points = [Point(5, 5),
          Point(1, 4),
          Point(2, 3)]

pd.DataFrame(points)
"""
      x_loc  y_loc
0      5      5
1      1      4
2      2      3
"""

```

list of
dataclass
objects

linkedin.com/in/avi-chawla

غالباً ما يتم إنشاء إطار بيانات Pandas من قائمة بايثون list، وقاموس dictionary، من خلال قراءة الملفات، الخ. ومع ذلك، هل تعلم أنه يمكنك أيضاً إنشاء إطار بيانات DataFrame من فئة ?Dataclass

توضح الصورة كيف يمكنك إنشاء إطار بيانات من قائمة كائنات dataclass.

المقالة:

<https://avichawla.substack.com/p/pandas-df-from-dataclass>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Pandas/Pandas-df-From-Dataclass.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandas-df-From-Dataclass.ipynb)

130) إخفاء السمات أثناء طباعة كائن من Dataclass Attributes While Printing A Dataclass Object

The image shows two screenshots of a terminal window. The top screenshot displays the following code:

```
from dataclasses import dataclass

@dataclass
class Student:
    name:str
    key:str

Jane = Student("Jane", "27HD")

print(Jane)
Student(name='Jane', key='27HD')
```

A red box highlights the last line of the code. A white arrow points from this line to the text "Prints all attributes" located to the right of the terminal window.

The bottom screenshot displays the following code, which uses the `field(repr=False)` parameter to hide the `key` attribute from the print output:

```
from dataclasses import dataclass, field

@dataclass
class Student:
    name:str
    key:str = field(repr=False)

Jane = Student("Jane", "27HD")

print(Jane)
Student(name='Jane')
```

A pink box highlights the line `key:str = field(repr=False)`. A white arrow points from this line to the text "Attribute hidden in print" located to the left of the terminal window.

linkedin.com/in/avi-chawla

بشكل افتراضي، تقوم `dataclass` بطباعة جميع سمات الكائن الذي تم الإعلان عنه أثناء التهيئة.

ولكن إذا كنت تريد إخفاء بعض السمات المحددة، فقم بالإعلان عن `repr = False` في حقلها، كما هو موضح أعلاه.

المقالة:

<https://avichawla.substack.com/p/hide-dataclass-attributes>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/hide-dataclass-attributes.ipynb>

131) القائمة: الصد :: المجموعة: المجموعة؟

set.py

```
my_set = {1, 2, 3}
my_dict = {my_set: "A set"}
## TypeError: unhashable type: 'set'
```

A set cannot be added as a key

frozensest.py

```
## frozensest
my_set = frozenset({1, 2, 3})
my_dict = {my_set: "A frozen set"}
my_dict[my_set]
"A frozen set"
```

Use frozensest

linkedin.com/in/avi-chawla

تطلب القواميس Dictionaries في بايثون أن تكون مفاتيحها غير قابلة للتغيير immutable. نتيجة لذلك، لا يمكن استخدام المجموعة set كمفاتيح لأنها قابلة للتغيير mutable.

ومع ذلك، إذا كنت تريد استخدام مجموعة، ففكري في إعلانها على أنها .frozensest.

إنها مجموعة غير قابلة للتغيير، مما يعني أنه لا يمكن تغيير عناصرها بعد إنشائها. لذلك، يمكن استخدامها بأمان كمفتاح للقاموس.

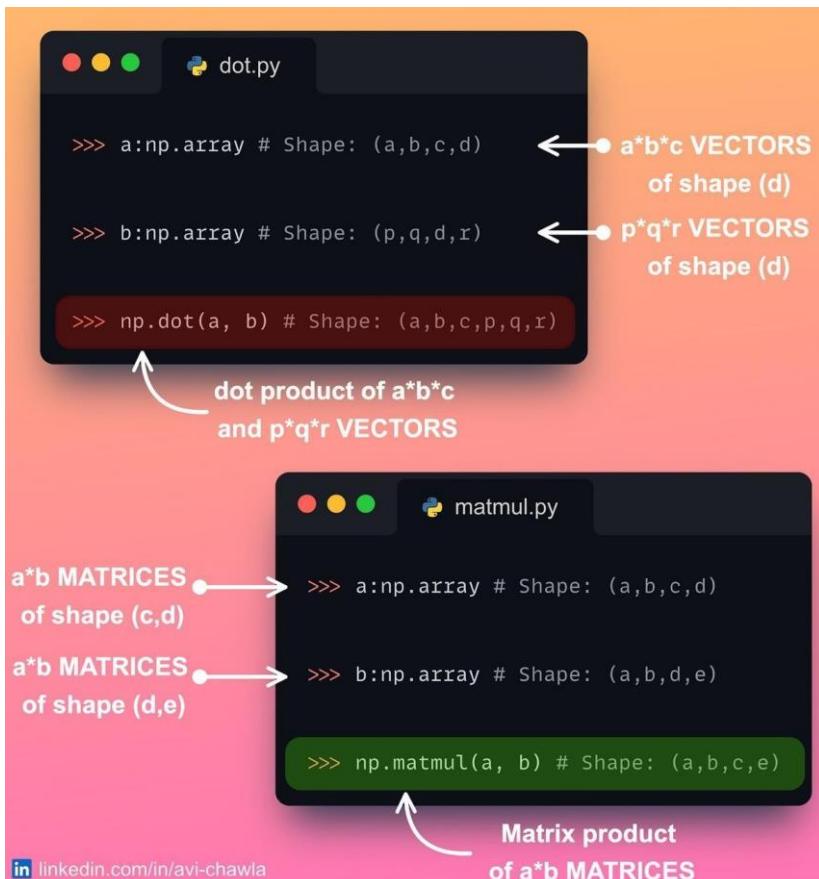
المقالة:

<https://avichawla.substack.com/p/frozensest>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/frozensest.ipynb>

Difference NumPy في Matmul و Dot (132) الفرق بين Matmul و Dot في NumPy



تنتج الطرق (`np.dot`) و (`np.matmul`) نفس الإخراج للمصفوفات ثنائية الأبعاد 2D (و أحادية الأبعاد 1D). هذا يجعل الكثيرين يعتقدون أنهم متماثلون ويمكن استخدامهم بالتبادل، لكن هذا ليس صحيحاً.

طريقة (`np.dot`) تدور حول المتجهات الفردية individual vectors (أو المصفوفات 1D). وبالتالي، فإنه يحسب حاصل الضرب النقطي dot product لجميع أزواج المتجهات في المدخلين.

الطريقة (`np.matmul`، كما يوحى الاسم)، مخصصة للمصفوفات matrices. وبالتالي، فإنه يحسب ضرب المصفوفة matrix multiplication للمصفوفات المقابلة في المدخلين.

المقالة:

<https://avichawla.substack.com/p/dot-and-matmul-difference>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/NumPy/Dot-and-Matmul.ipynb>

133) قم بتشغيل SQL في Jupyter لتحليل إطار بيانات Pandas Run SQL in Jupyter To Analyze A Pandas DataFrame



يوفر Pandas بالفعل مجموعة واسعة من الدوال لتحليل البيانات المجدولة tabular data. ومع ذلك، قد تكون هناك مواقف يشعر فيها المرء بالراحة عند استخدام SQL على بايثون.

باستخدام DuckDB، يمكنك تحليل Pandas DataFrame باستخدام بناء جملة SQL في Jupyter. دون أي فرق وقت تشغيل مهم.

اقرأ الدليل هنا للبدء: [المستندات](#).

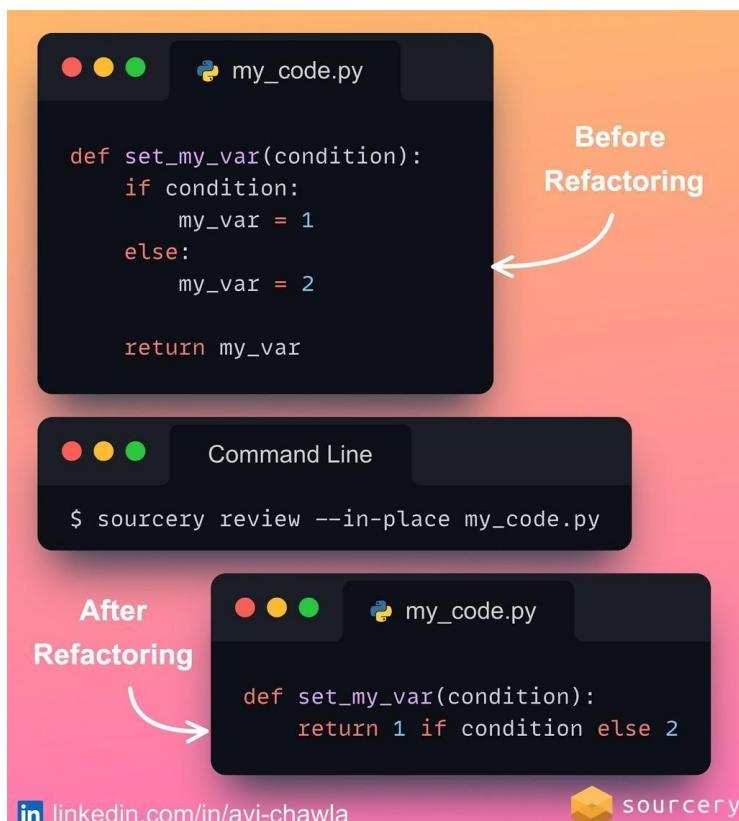
المقالة:

<https://avichawla.substack.com/p/sql-in-jupyter>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/SQL-in-Jupyter.ipynb>

134) إعادة هيكلة الكود الآلي مع المصادر Automated Code Refactoring With Sourcery



تعد إعادة بناء قاعدة الكودات Refactoring codebase مهمة كبيرة ولكنها تستغرق وقتاً طويلاً. علاوة على ذلك، في بعض الأحيان، قد يرتكب المرء أخطاء أثناء إعادة البناء دون علمه.

يستغرق هذا وقتاً إضافياً للاختبار ويصبح مملاً مع المزيد من إعادة البناء، خاصةً عندما تكون قاعدة التعليمات البرمجية كبيرة.

بدلاً من اتباع هذا النهج، استخدم [Sourcery](#). إنها أداة إعادة هيكلة آلية تجعل التعليمات البرمجية الخاصة بك أنيقة وموجزة وببايثونية في لمح البصر.

باستخدام Sourcery، يمكنك إعادة بناء الكود بعدة طرق. على سبيل المثال، يمكنك إعادة صياغة السكريبتات من خلال سطر الأوامر، كمكون إضافي لـ IDE في PyCharm و VS Code، إلخ.

اقرأ مدونتي الكاملة على Sourcery هنا: [Medium](#)

المقالة:

<https://avichawla.substack.com/p/automated-code-refactoring-with-sourcery>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Terminal/Sourcery.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Terminal/Sourcery.ipynb)

إضافة سمات إلى تهيئة عملية نشر كائن : Post_init(135)

Add Attributes To A Dataclass :Post_init Dataclass Object Post Initialization



```

from dataclasses import dataclass

@dataclass
class StudentMarks:
    student_id:str
    marks:float

    def __post_init__(self):
        if self.marks>30:
            self.grade = "Pass"
        else:
            self.grade = "Fail"

Peter = StudentMarks("B20", 43)

print(Peter.grade) # Pass

```

linkedin.com/in/avi-chawla

بعد تهيئة كائن فئة، غالباً ما نقوم بإنشاء سمات مشتقة من المتغيرات الموجودة.

للقيام بذلك في فئات البيانات dataclasses، يمكنك استخدام طريقة `__post_init__`. كما يوحى الاسم، يتم استدعاء هذه الطريقة بعد طريقة `__init__`.

يكون هذا مفيدةً إذا كنت بحاجة إلى إجراء عمليات إعداد إضافية على مثيل dataclass الخاص بك.

المقالة:

<https://avichawla.substack.com/p/dataclass-post-init>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/dataclass-post-init.ipynb>

Simplify Your Functions (136) تبسيط دوالك بدوال جزئية With Partial Functions

```

def quadratic(x, a, b, c):
    return a*(x**2) + b*x + c

```

```

from functools import partial
quadratic_c1 = partial(quadratic, c=1)

quadratic_c1(x=1, a=4, b=5)
# Output: 10

```

Fix some parameters

عندما تأخذ دالتك function العديد من الوسيطات arguments، قد يكون من الجيد تبسيطها باستخدام دوال جزئية partial functions.

يتتيحون لك إنشاء إصدار جديد من الدالة مع بعض الوسيطات التي تم إدخالها على قيم محددة.

يمكن أن يكون هذا مفيداً في تبسيط التعليمات البرمجية الخاصة بك وجعلها أكثر سهولة للقراءة وإيجازاً. علاوة على ذلك، يساعدك أيضاً على تجنب تكرار نفسك أثناء استدعاء الدوال.

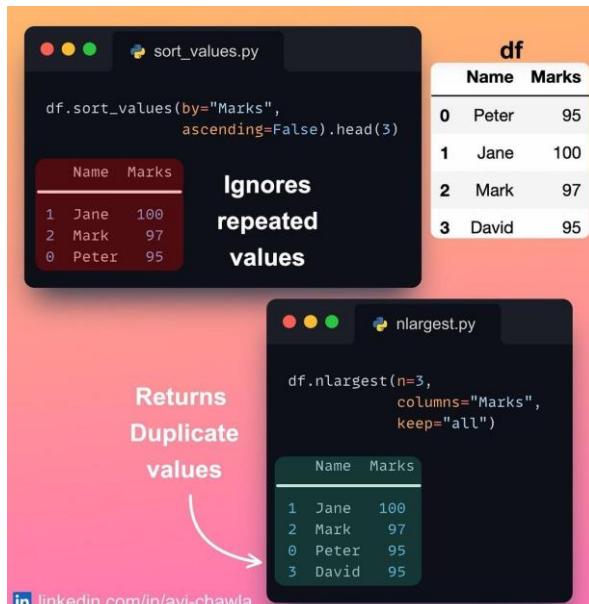
المقالة:

<https://avichawla.substack.com/p/partial-functions>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/partial-functions.ipynb>

Pandas head() عندما لا يجب استخدام طريقة head() في Pandas



غالباً ما يسترد المربع `k` من الصنوف من Pandas DataFrame المصنفة باستخدام طريقة `.head()` ومع ذلك، هناك عيب في هذا النهج.

إذا كانت بياناتك تحتوي على قيم مكررة، فلن تأخذ `head()` في الاعتبار ذلك وستعيد فقط الصنوف الأولى.

إذا كنت تريد مراعاة القيم المتكررة، فاستخدم `nsmallest` أو `nlargest` بدلاً من ذلك. هنا، يمكنك تحديد السلوك المطلوب للقيم المكررة باستخدام المعلمة `keep`.

[المقالة:](#)

<https://avichawla.substack.com/p/nsmallest-and-nlargest>

[ال코드:](#)

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/nsmallest-and-nlargest.ipynb>

DotMap: بديل أفضل لقاموس بايثون A:DotMap (138)

Better Alternative to Python Dictionary

```
from dotmap import DotMap

students = DotMap()

students.john.id = "12A"
students.john.english = 45

students.mary.id = "12B"
students.mary.english = 49

students.dave.id = "12C"
students.dave.english = 34
```

Add using dot notation

Pretty Print

```
people pprint()

{'dave': {'english': 34, 'id': '12C'},
 'john': {'english': 45, 'id': '12A'},
 'mary': {'english': 49, 'id': '12B'}}
```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

قواميس بايثون رائعة، لكن لها العديد من القيود.

من الصعب إنشاء بيانات هرمية ديناميكية. كما أنها لا توفر التدوين النقطي dot notation المعتمد على نطاق واسع لقيم الوصول.

بدلاً من ذلك، استخدم DotMap. يتصرف مثل قاموس بايثون ولكنه يعالج أيضًا القيود المذكورة أعلاه.

ما هو أكثر من ذلك، أنه يحتوي أيضًا على طريقة طباعة جميلة مضمونة لعرضه قاموس / JSON لتصحيح خطأ الكائنات الكبيرة.

اقرأ المزيد: [GitHub](#)

المقالة:

<https://avichawla.substack.com/p/dotmap-a-better-alternative-to-dict>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/DotMap-Better-Dictionary.ipynb>

139) منع الاستيرادات السيئة مع `__all__` في بايثون in Python `__all__ Wild Imports With`

`my_functions.py`

```
__all__ = ["func1", "func2"]

def func1():
    return "Function 1"

def func2():
    return "Function 2"

def func3():
    return "Function 3"
```

`Specify __all__`

`module.py`

```
from my_functions import *

>>> func1()
"Function 1"

>>> func2()
"Function 2"

>>> func3()
NameError: name 'func3' is not defined
```

[in linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

تعتبر عمليات الاستيراد (`from module import *`) ممارسة برمجة سيئة. ومع ذلك، إليك كيفية منعه إذا قام شخص ما بذلك بشكل غير مسؤول أثناء استخدام التعليمات البرمجية الخاصة بك.

في الوحدة النمطية الخاصة بك، يمكنك تحديد الدوال / الفئات / المتغيرات القابلة للاستيراد في `__all__`. نتيجة لذلك، عندما يقوم شخص ما بعملية استيراد سيئة، فإن بايثون ستنسخ الرموز المحددة هنا فقط.

يمكن أن يكون هذا مفيداً أيضاً في نقل الرموز الموجودة في الوحدة النمطية الخاصة بك إلى أن تكون خاصة.

المقالة:

<https://avichawla.substack.com/p/prevent-wild-imports-with-all>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Save-Wild-imports.ipynb>

140) ثلات نصائح أقل شهرة لقراءة ملف CSV باستخدام Pandas



فيما يلي ثلات نصائح مفيدة للغاية ولكنها أقل شهرة لقراءة ملف CSV مع Pandas

1. إذا كنت ت يريد قراءة الصحفة القليلة الأولى فقط من الملف، فحدد المعلمة **nrows**
1. لتحميل بعض الأعمدة المحددة، حدد المعلمة **usecols**
2. إذا كنت ت يريد تخطي بعض الصحفة أثناء القراءة، فقم بتمرير المعلمة **skiprows**

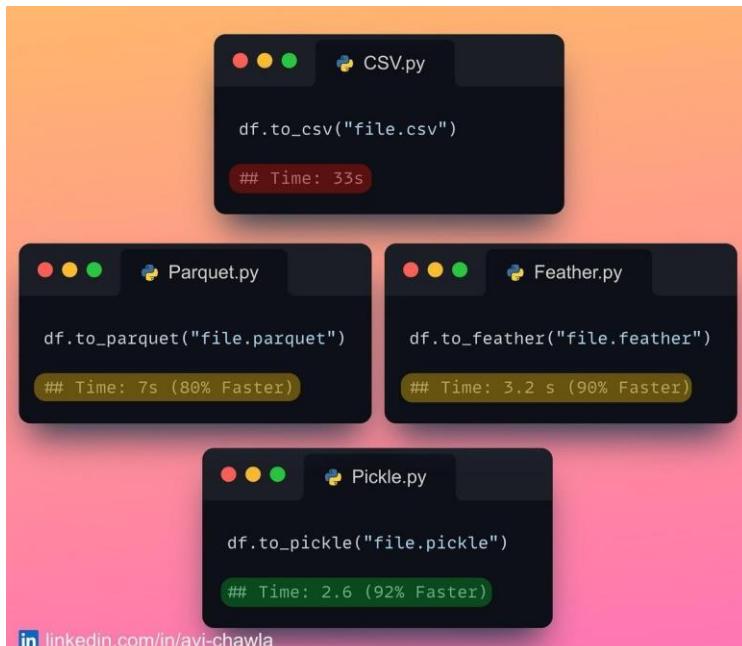
المقالة:

<https://avichawla.substack.com/p/three-lesser-known-tips-for-reading>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/3-Read-CSV-Features.ipynb>

141) أفضل تنسيق ملف لتخزين إطار بيانات Pandas



في الصورة أعلاه، يمكنك العثور على مقارنة وقت التنفيذ لتخزين Pandas DataFrame بتنسيقات ملفات مختلفة.

على الرغم من أن تنسيق CSV هو تنسيق معتمد على نطاق واسع، إلا أنه أبطأ تنسيق في هذه القائمة. وبالتالي، يجب تجنب ملفات CSV إلا إذا كنت ترغب في فتح البيانات خارج بايثون (في Excel، على سبيل المثال).

اقرأ المزيد في مدونتي: [Medium](#)

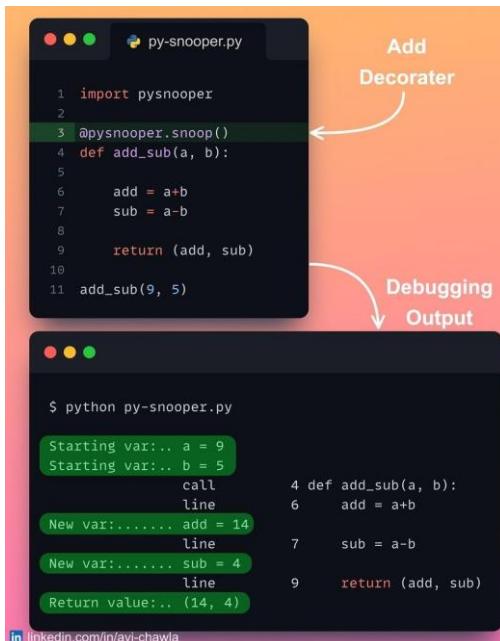
المقالة:

<https://avichawla.substack.com/p/the-best-file-format-to-store-a-pandas>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/CSV-Feather-Parquet-Pickle.ipynb>

PySnooper (142) تصحيح الأخطاء أصبح سهلاً مع PySnooper



بدلاً من استخدام العديد من عبارات print لتصحيح أخطاء كود بايثون الخاص بك، جرب PySnooper

باستخدام سطر واحد فقط من التعليمات البرمجية، يمكنك بسهولة تتبع المتغيرات في كل خطوة من خطوات تنفيذ التعليمات البرمجية الخاصة بك.

اقرأ المزيد: [المستودع](#).

المقالة:

<https://avichawla.substack.com/p/debugging-made-easy-with-pysnooper>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Debugging/pysnooper-debugging.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Debugging/pysnooper-debugging.ipynb)

143) الميزة الأقل شهرة لطريقة الدمج في Pandas Known Feature of the Merge Method in Pandas



أثناء دمج DataFrames في Pandas، يمكن أن يكون تبع مصدر كل صفت في الإخراج مفيداً للغاية.

يمكنك القيام بذلك باستخدام الوسيطة `indicator` للطريقة `merge()`. ونتيجة لذلك، يقوم بزيادة عمود إضافي في الإخراج المدمج، والذي يخبر مصدر كل صفت.

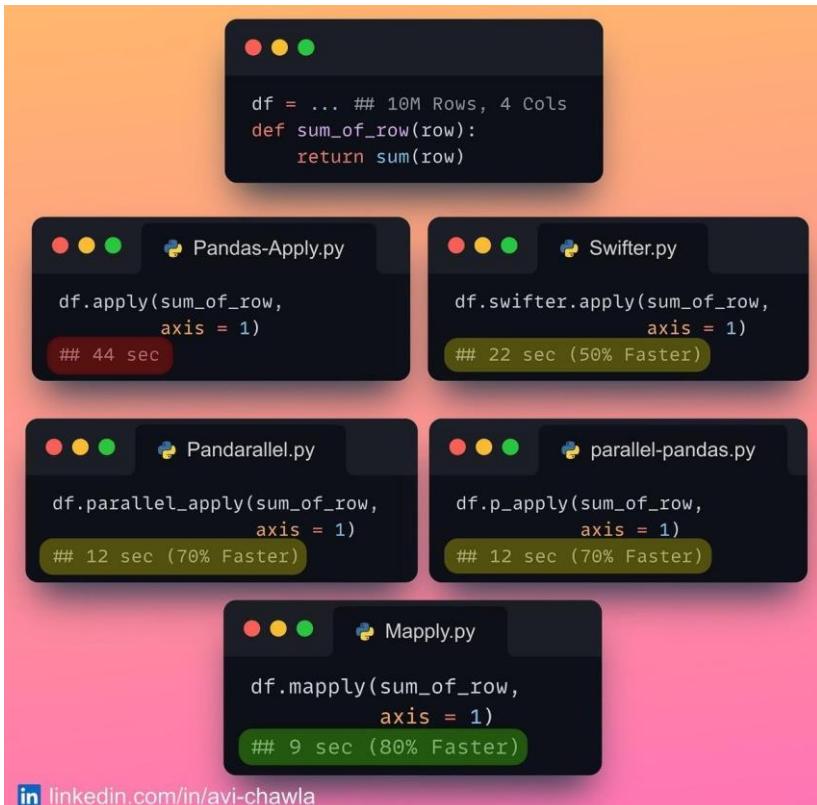
المقالة:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandas-Merge-Indicator.ipynb>

الكلود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandas-Merge-Indicator.ipynb>

The Best Pandas Apply() Way to Use Apply() in Pandas



تُظهر الصورة أعلاه مقارنة وقت التشغيل للمكتبات مفتوحة المصدر الشائعة التي توفر دعم موازٍ لـ Pandas.

يمكنك العثور على روابط هذه المكتبات هنا. أيضًا، إذا كنت تعرف أي مكتبات أخرى مماثلة مبنية على Pandas، فقم بنشرها في التعليقات أو الرد على هذا البريد الإلكتروني.

المقالة:

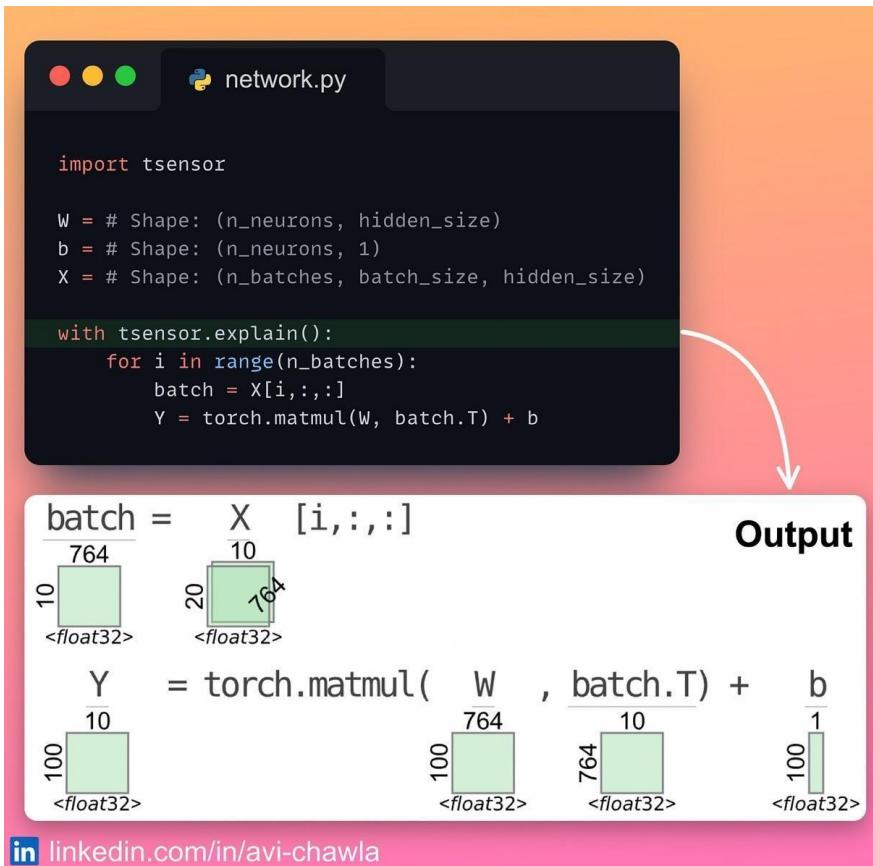
<https://avichawla.substack.com/p/the-best-way-to-use-apply-in-pandas>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Best-Pandas-Apply.ipynb>

١٤٥) أصبح تصحيح أخطاء شبكة التعلم العميق أمرًا سهلاً

Deep Learning Network Debugging Made Easy



يمكن أن تكون محاذاة شكل الموترات tensors (أو المتجهات vectors / المصفوفات matrices) في الشبكة أمرًا صعبًا في بعض الأحيان.

مع نمو الشبكة، من الشائع فقدان مسار الأبعاد في تعبير معقد.

بدلاً من طباعة أشكال الموتر بشكل صريح لتصحيح الأخطاء، استخدم **TensorSensor**. يولد تصورًا أنهياً لكل بيان يتم تنفيذه داخل الكتلة الخاصة به. هذا يجعل تتبع الأبعاد سهلاً وسريعاً.

في حالة وجود أخطاء، فإنه يعزز رسائل الخطأ الافتراضية بمزيد من التفاصيل المفيدة. يؤدي هذا إلى زيادة سرعة عملية التصحيح.

اقرأ المزيد: [الترشيق](#).

المقالة:

<https://avichawla.substack.com/p/deep-learning-network-debugging-made>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Sklearn/Debug-Network.ipynb>

Lovely-NumPy! NumPy مصفوفات لا تطبع 146 بدلاً من ذلك. - Don't Print NumPy Arrays! Use Lovely-.NumPy Instead

```
>>> array = np.random.rand(...)
>>> array
tensor([[0.59, 0.03, ..., 0.44, 0.41],
       [0.60, 0.72, ..., 0.92, 0.61],
       ...,
       [0.57, 0.98, ..., 0.01, 0.91],
       [0.00, 0.53, ..., 0.54, 0.54]])
```

Only numbers

Summary of Array

```
from lovely_numpy import lo

>>> array = np.random.rand(...)
>>> lo(array)
array[10, 20] n=200
x∈[0.0, 1.0] μ=0.51 σ=0.3
```

```
>>> array = np.zeros(...)
>>> lo(array)
array[10] all_zeros
```

```
>>> array = # With NaN and Inf
>>> lo(array)
array[10, 20] n=200
x∈[0.0, 1.0] μ=0.51 σ=0.3 +Inf! NaN!
```

lovely_numpy.py

غالباً ما نطبع المصفوفات الخام غير الدقيقة أثناء التصحيح debugging. لكن هذا النهج ليس مفيداً جدًا. هذا لأن الطباعة لا تنقل الكثير من المعلومات حول البيانات التي تحتفظ بها، خاصة عندما تكون المصفوفة كبيرة.

بدلاً من ذلك، استخدم **lovely-numumpy**. بدلاً من عرض المصفوفات الأولية، يقوم بطباعة ملخص للمصفوفة. وهذا يشمل الشكل والتوزيع والمتوسط والانحراف المعياري وما إلى ذلك.

كما يوضح أيضًا ما إذا كانت المصفوفة العددية تحتوي على قيم NaN وInf، وما إذا كانت تحتوي على أصفار، وغير ذلك الكثير.

ملاحظة. إذا كنت تعمل مع الموترات tensors، فيمكنك استخدام **lovely-tensors**

اقرأ المزيد: [التوثيق](#).

المقالة:

<https://avichawla.substack.com/p/dont-print-numpy-arrays-use-lovely>

ال코드:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/NumPy/Lovely-Numpy.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/NumPy/Lovely-Numpy.ipynb)

Performance 3.10 وبايثون 3.11 أداء مقارنة (147)

Comparison of Python 3.11 and Python 3.10



تم إصدار Python 3.11 مؤخراً، ووفقاً للإصدار الرسمي، من المتوقع أن تكون أسرع بنسبة 10–60٪ من Python 3.10.

أجريت بعض تجارب قياس الأداء الأساسية للتحقق من تعزيز الأداء. في الواقع، Python 3.11 أسرع بكثير.

على الرغم من أنه قد يميل الماء إلى الترقية في أسرع وقت ممكن، إلا أن هناك بعض الأشياء التي يجب أن تعرفها. اقرأ المزيد [هنا](#).

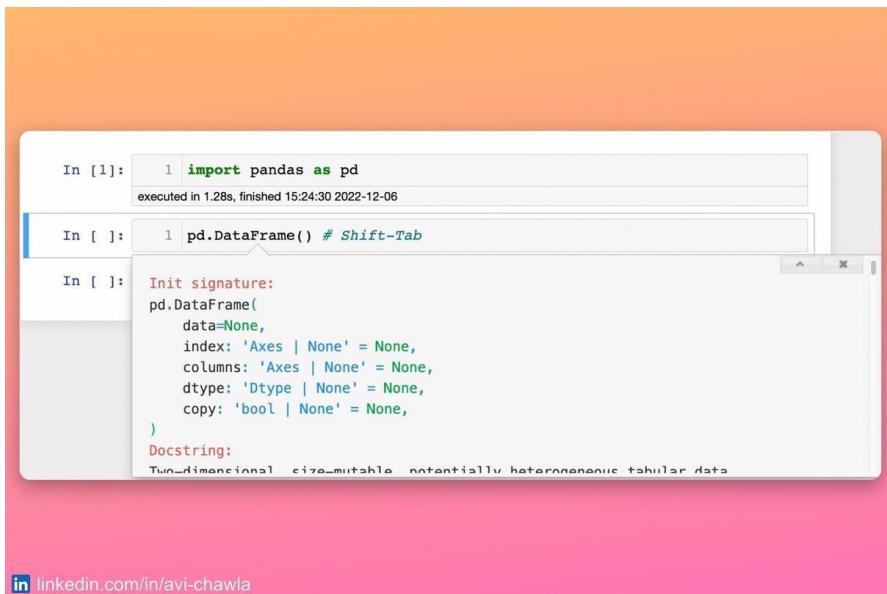
المقالة:

<https://avichawla.substack.com/p/performance-comparison-of-python>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Python-3.11-Benchmark.ipynb>

148) عرض الوثائق في Jupyter Notebook Documentation in Jupyter Notebook



أثناء العمل في Jupyter، من الشائع أن ننسى معلمات الدالة وزيارة المستندات الرسمية (أو Stackoverflow). ومع ذلك، يمكنك عرض الوثائق في التوبوك نفسه.

الضغط على **Shift-Tab** يفتح لوحة التوثيق. هذا مفيد للغاية ويوفر الوقت حيث لا يتعدى على المرء فتح المستندات الرسمية في كل مرة.

تعمل هذه الميزة أيضًا مع دوالك المخصصة.

قم بمشاهدة نسخة فيديو من هذا المنشور على [LinkedIn](#) : [رابط المنشور](#).

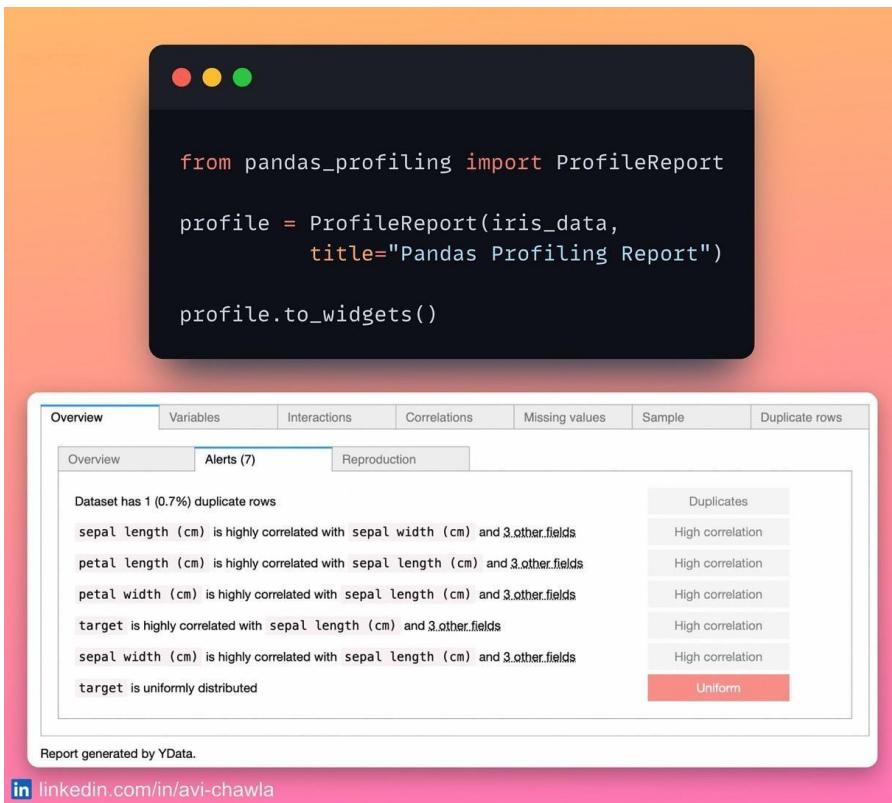
المقالة:

<https://avichawla.substack.com/p/view-documentation-in-jupyter-notebook>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/View-Documentation.ipynb>

A No-code Tool (149) أداة بدون كود لفهم بياناتك بسرعة To Understand Your Data Quickly



غالباً ما تكون الخطوات الأولية لأي مهمة نموذجية من تحليل البيانات الاستكشافية EDA هي نفسها. ومع ذلك، عبر المشاريع، نميل إلى كتابة نفس الكود لتنفيذ هذه المهام. هذا يتكرر ويستغرق وقتاً طويلاً. بدلاً من ذلك، استخدم **pandas-profiling**. يقوم تلقائياً بإنشاء تقرير موحد لفهم البيانات في أي وقت من الأوقات. واجهة المستخدم البديهية يجعل هذا سهل وسريع.

يتضمن التقرير بعد البيانات وإحصائيات القيمة المفقودة وأنواع بيانات العمود. علاوة على ذلك، فإنه يوضح أيضاً توزيع البيانات والتفاعل والارتباط بين المتغيرات وما إلى ذلك.

أخيراً، يتضمن التقرير أيضاً التنبؤات، والتي يمكن أن تكون مفيدة للغاية أثناء التحليل / النمذجة.

اقرأ المزيد: [الوثيق](#).

المقالة:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandas-Data-Report.ipynb>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandas-Data-Report.ipynb>

Why 256 is 257 هو 256 ولكن 257 ليس 256 ? ?256 But 257 is not 257

```

>>> a = 256
>>> b = 256

>>> a is b
True

>>> a = 257
>>> b = 257

>>> a is b
False

>>> a, b = 257, 257

>>> a is b
True

```

linkedin.com/in/avi-chawla

قد تكون مقارنة كائنات بايثون صعبة في بعض الأحيان. هل يمكنك معرفة ما يجري في مثل الكود أعلاه؟ الإجابة أدناه:

عندما نقوم بتشغيل بايثون، فإنها تُحمل مسبقاً قائمة عالمية من الأعداد الصحيحة في النطاق [256, 5]. في كل مرة يُشار فيها إلى عدد صحيح في هذا النطاق، لا تنشئ بايثون كائناً جديداً. بدلاً من ذلك، فإنه يستخدم النسخة المخبأة .cached version.

يتم ذلك لأغراض التحسين. واعتبر أن المبرمجين يستخدمون هذه الأرقام كثيراً. لذلك، من المنطقي أن تكون جاهزاً عند بدء التشغيل.

ومع ذلك، فإن الإشارة إلى أي عدد صحيح يتجاوز 256 (أو قبل 5-) سيؤدي إلى إنشاء كائن جديد في كل مرة.

في المثال الأخير، عندما يتم تعين `a` و `b` على 257 في نفس السطر، يقوم مترجم بایثون بإنشاء كائن جديد. ثم يشير إلى المتغير الثاني بنفس الكائن.

راجع هذا المنشور على [LinkedIn](#): رابط المنشور.

يجب أن تمنحك الصورة أدناه فهماً أفضل:

```

a = 256
b = 256
print(a is b) # Output: True
>>> id(a), id(b)      Same
(1284834, 1284834)

a = 257
b = 257
print(a is b) # Output: False
>>> id(a), id(b)      Different
(1848542, 1744933)
>>> print(a==b) # Output: True

a, b = 257, 257
print(a is b) # Output: True
>>> id(a), id(b)      Same
(23829574, 23829574)

```

[linkedIn.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

المقالة:

<https://avichawla.substack.com/p/why-256-is-256-but-257-is-not-257>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Comparing-Integer-Objects.ipynb>

Make a Class Behave Like a Function

class object behaves like function

```

class Quadratic:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c

    def __call__(self, x):
        return (self.a * x**2) +
               (self.b * x) +
               self.c
  
```

define __call__ method

```

f = Quadratic(1, 2, 3)

print(f(1)) # Output: 6
print(f(2)) # Output: 11

print(callable(f)) # Output: True
  
```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

إذا كنت تريid جعل كائن فئة class object قابلاً للاستدعاء callable، أي التصرف كدالة function. فيمكنك القيام بذلك عن طريق تعريف طريقة __call__.

تتيح لك هذه الطريقة تحديد سلوك الكائن عند استدعائه كدالة.

يمكن أن يكون لهذا العديد من المزايا. على سبيل المثال، يسمح لنا بتنفيذ الأشياء التي يمكن استخدامها بطريقة مرتنة وبديهية. علاوة على ذلك، فإن بناء الجملة المألوف لاستدعاء الدالة، في بعض الأحيان، يمكن أن يجعل كودك أكثر قابلية للقراءة.

أخيراً، يسمح لك باستخدام كائن فئة في السياقات حيث يتوقع الاستدعاء. استخدام فئة كdecorator، على سبيل المثال.

المقالة:

<https://avichawla.substack.com/p/make-a-class-object-behave-like-a>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Make-Class-Callable.ipynb>

Lesser-known Pickle ميزة أقل شهرة لملفات (152) feature of Pickle Files

```

dump.py
import pickle
a, b, c = 1, 2, 3
with open("data.pkl", "wb") as f:
    pickle.dump(a, f)
    pickle.dump(b, f)
    pickle.dump(c, f)

← Store 3 Variables

```



```

load.py
import pickle
with open("data.pkl", "rb") as f:
    a = pickle.load(f)
    b = pickle.load(f)
print(f"{{a = }} {{b = }}")
## a = 1 b = 2

Load Only First 2 → Variables

```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

تستخدم Pickles على نطاق واسع لخزن كائنات البيانات على القرص. لكن الناس غالباً ما يرثون شيئاً واحداً فقط في Pickle. علاوة على ذلك، يخلق المرء Pickles متعددة لتخزين كائنات متعددة. ومع ذلك، هل تعلم أنه يمكنك تخزين أي عدد تريده من العناصر داخل Pickle واحد؟ علاوة على ذلك، عند إعادة التحميل، ليس من الضروري تحميل جميع الكائنات.

فقط تأكد من تفريغ الكائنات داخل نفس مدير السياق (باستخدام **with**).

بالطبع، أحد الحلول هو تخزين الكائنات معافي مجموعة. ولكن أثناء إعادة التحميل، سيتم تحميل المجموعة بالكامل. قد لا يكون هذا مرغوباً في بعض الحالات.

المقالة:

<https://avichawla.substack.com/p/lesser-known-feature-of-pickle-files>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Pickle-Multiple-Dump.ipynb>

153) المخطط النقطي: بديل محتمل للمخطط الشريطي

Dot Plot: A Potential Alternative to Bar Plot



[in linkedin.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

تعد المخططات الشريطية Bar plots مفيدة للغاية لتصور المتغيرات الفئوية categorical variables مقابل قيمة مستمرة continuous value. ولكن عندما يكون لديك العديد من الفئات لتصویرها، فقد تصبح كثيفة للغاية بحيث لا يمكن تفسيرها.

في مخطط شريطي به العديد من الأشرطة، غالباً ما لا نولي اهتماماً لأطوال الأشرطة الفردية. بدلاً من ذلك، نعتبر في الغالب نقاط النهاية الفردية التي تشير إلى القيمة الإجمالية.

يمكن أن يكون المخطط النقطي Dot plot خياراً أفضل في مثل هذه الحالات. إنها مثل مخططات التشتت scatter plots ولكن مع محور فئوي واحد وأخر مستمر.

بالمقارنة مع المخطط الشريطي، فهي أقل تشوشاً ولديها فهم أفضل. هذا صحيح بشكل خاص في الحالات التي يكون لدينا فيها العديد من الفئات و / أو عدة أعمدة فئوية لتصویرها في المخطط.

اقرأ المزيد: [التوثيق](#).

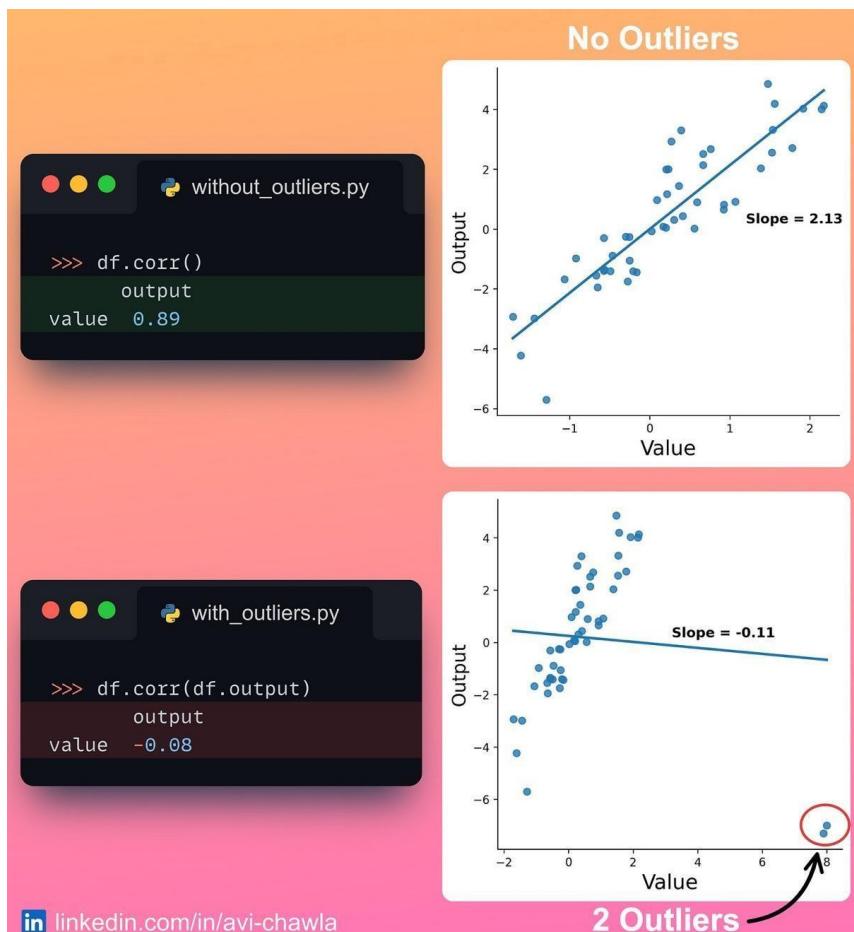
المقالة:

<https://avichawla.substack.com/p/dot-plot-a-potential-alternative>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/Dot-Plots.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Dot-Plots.ipynb)

154) لماذا يمكن أن يكون الارتباط (والإحصائيات الأخرى) مضللة . Why Correlation (and Other Statistics) Can Be Misleading



غالباً ما يستخدم الارتباط Correlation لتحديد الارتباط بين متغيرين مستمررين. لكنها تحتوي على عيب كبير غالباً ما لا يلاحظه أحد.

غالباً ما يستخلص الناس استنتاجات باستخدام مصفوفة الارتباط correlation matrix دون النظر إلى البيانات. ومع ذلك، يمكن أن تكون الإحصاءات statistics التي تم الحصول عليها مدفوعة بشدة بالقيم المتطرفة outliers.

هذا موضح في المخططات أعلاه. أدت إضافة اثنين فقط من القيم المتطرفة إلى تغيير الارتباط وخط الانحدار بشكل كبير.

وبالتالي، فإن النظر إلى البيانات وفهم خصائصها الأساسية يمكن أن ينقد من استخلاص استنتاجات خاطئة. الإحصائيات مهمة، لكنها قد تكون مضللة للغاية في بعض الأحيان.

المقالة:

<https://avichawla.substack.com/p/why-correlation-and-other-statistics>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Statistics/Statistics-Can-Mislead.ipynb>

Sidetable مع Pandas في value_counts() (155) Supercharge value_counts() Method in Pandas With Sidetable

```

import sidetable
df.freq(["City"], style=True)

```

	City	count	percent	cumulative_count	cumulative_percent
0	West Jamesview	120	12.00%	120	12.00%
1	Aliciafort	113	11.30%	233	23.30%
2	Ricardomouth	106	10.60%	339	33.90%
3	New Cindychester	106	10.60%	445	44.50%
4	Whiteside	104	10.40%	549	54.90%
5	Kristaburgh	97	9.70%	646	64.60%
6	Wardfort	96	9.60%	742	74.20%
7	New Russellton	93	9.30%	835	83.50%
8	Whitakerbury	87	8.70%	922	92.20%
9	North Melissafurt	78	7.80%	1,000	100.00%

[in](https://linkedin.com/in/avi-chawla) linkedin.com/in/avi-chawla

تُستخدم طريقة value_counts() بشكل شائع لتحليل الأعمدة الفرعية categorical columns، لكن لها العديد من القيود.

على سبيل المثال، إذا أراد المرء عرض النسبة المئوية والعدد التراكمي وما إلى ذلك، في مكان واحد، تصبح الأمور مملة بعض الشيء. هذا يتطلب المزيد من التعليمات البرمجية ويستغرق وقتاً طويلاً.

بدلاً من ذلك، استخدم sidetable.value_counts(). اعتبرها نسخة معززة من sidetable.freq(). كما هو موضح أدناه، توفر طريقة freq() ملخصاً أكثر فائدة من value_counts().

بالإضافة إلى ذلك، يمكن لـ `sidetable` تجميع أعمدة متعددة أيضًا. يمكنك أيضًا توفير نقاط حد لدمج البيانات في مجموعة واحدة. علاوة على ذلك، يمكنه طباعة إحصائيات البيانات المفقودة `missing data` وقيم الطباعة الجميلة وما إلى ذلك.

اقرأ المزيد: [GitHub](#).

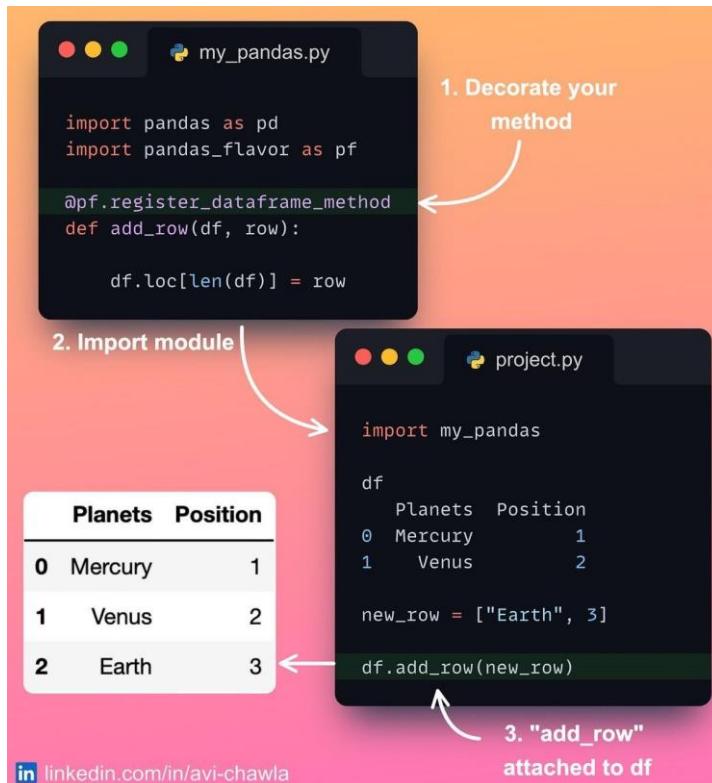
المقالة:

https://avichawla.substack.com/p/supercharge-value_counts-method-in

الكود:

https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Better-Value_Counts-Method.ipynb

156) اكتب نكهة الخاصة بك Flavor Of Pandas



إذا كنت ترغب في إرفاق دالة مخصصة بكائن Pandas DataFrame (أو سلسلة Series)، فاستخدم `.pandas-flavor`

يسمح لك decorators الخاص به بإضافة طرق مباشرة إلى كائن Pandas

هذا مفید بشكل خاص إذا كنت تقوم ببناء مشروع مفتوح المصدر يتضمن Pandas. بعد تثبيت مكتبتك، يمكن للآخرين الوصول إلى أساليب مكتبتك باستخدام كائن `dataframe`.

ملاحظة. هكذا نرى `df.parallel_apply()` ، `tqdm` من `df.progress_apply()` وغيرها الكثير. **Pandarallel**

اقرأ المزيد: [الوثيق](#).

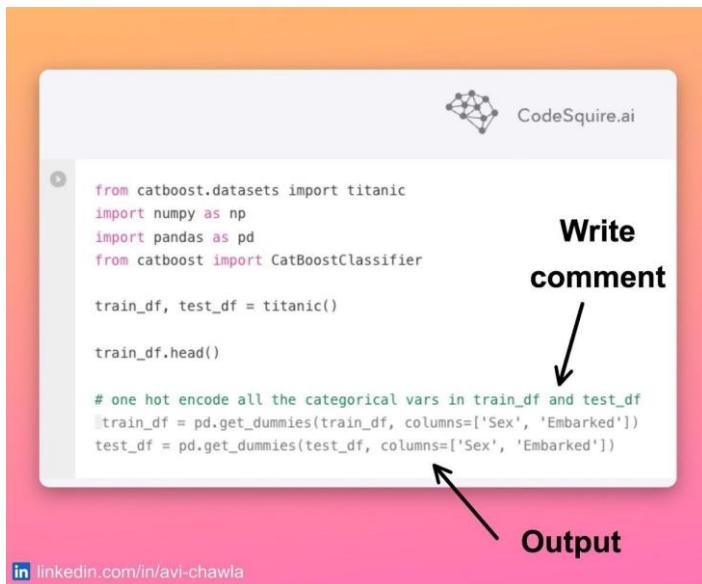
المقالة:

<https://avichawla.substack.com/p/write-your-own-flavor-of-pandas>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Customize-Pandas.ipynb>

CodeSquire: مساعد الترميز AI الذي يجب عليك استخدامه عبر GitHub Coding Assistant You Should Use Over GitHub Copilot



مساعدي البرمجة مثل GitHub Copilot شوريون حيث أنهم يقدمون العديد من المزايا. ومع ذلك، فإن فائدة Copilot محدودة لمحترفي البيانات. هذا لأنه غير متوافق مع IDEs المستندة إلى الويب (Jupyter / Colab).

علاوة على ذلك، في علم البيانات، يتم تحديد الخطوات الاستكشافية اللاحقة من خلال المخرجات السابقة. لكن Copilot لا يعتبر ذلك (وحتى خلايا العلامات markdown cells) لدفع اقتراحات الكود الخاصة به.

CodeSquire هو مساعد ترميز مذهل للذكاء الاصطناعي يعالج قيود برنامج Copilot. الشيء الجيد هو أنه تم تصميمه خصيصاً لعلماء ومهندسي ومحليي البيانات.

إلى جانب إنشاء الكود السلس، يمكنه إنشاء استعلامات SQL من النص وشرح التعليمات البرمجية. يمكنك الاستفادة من إنشاء الكود المدعوم بالذكاء الاصطناعي ببساطة عن طريق تثبيت امتداد متصفح.

اقرأ المزيد: [CodeSquire](#)

شاهد نسخة فيديو من هذا المنشور على LinkedIn : [رابط المنشور](#).

المقالة:

<https://avichawla.substack.com/p/codesquire-the-ai-coding-assistant>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Jupyter%20Tips/CodeSquire-AI-code-completion.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/CodeSquire-AI-code-completion.ipynb)

158) لا يضمن التوجيه دائمًا أداءً أفضل Not Always Guarantee Better Performance



تم اعتماد التوجيه Vectorization بشكل جيد لتحسين أداء وقت التنفيذ. باختصار، يتيح لك تشغيل البيانات على دفعات batches بدلاً من معالجة قيمة واحدة في كل مرة.

على الرغم من أن التوجيه فعال للغاية، يجب أن تعلم أنه لا يضمن دائمًا مكاسب في الأداء. علاوة على ذلك، يرتبط التوجيه أيضًا بحمل الذاكرة .memory overheads.

كما هو موضح أعلاه، يوفر الكود غير المتجه non-vectorized code أداءً أفضل من الإصدار المتجه.

ملاحظة. apply() هي أيضًا حلقة من حلقات for

قراءات إضافية: [هنا](#).

المقالة:

<https://avichawla.substack.com/p/in-defense-of-match-case-statements>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/match-case.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/match-case.ipynb)

In Defense عن حالات match-case بلغة بايثون (159) of Match-case Statements in Python

The diagram illustrates the evolution of a function from a traditional `if-else` structure to a modern `match-case` statement.

if-else.py:

```
def make_point(point):
    if isinstance(point, (tuple, list)):
        if len(point) == 2:
            x, y = point
            return Point3D(x, y, 0)

        elif len(point) == 3:
            x, y, z = point
            return Point3D(x, y, z)

    else:
        raise TypeError("Unsupported")
    else:
        raise TypeError("Unsupported")
```

Annotations for `if-else.py`:

- Check type**: Points to the `isinstance` check.
- Check length**: Points to the `len` checks.
- Explicit unpacking**: Points to the tuple assignment (`x, y` or `x, y, z`).

No type checks: A callout points to the lack of type checking in the original code.

No length checks: A callout points to the lack of length checking in the original code.

No unpacking: A callout points to the lack of unpacking in the original code.

match-case.py:

```
def make_point(point):
    match point:
        case (x, y):
            return Point3D(x, y, 0)

        case (x, y, z):
            return Point3D(x, y, z)

        case _: ## Default
            raise TypeError("Unsupported")

>>> make_point((1, 2))
Point3D(x=1, y=2, z=0)

>>> make_point([1, 2, 3])
Point3D(x=1, y=2, z=0)

>>> make_point((1, 2, 3, 4))
TypeError: Unsupported
```

linkedin.com/in/avi-chawla

صادفت مؤخرًا منشورًا على `match-case` بلغة بايثون. في الخلاصة، بدء تشغيل Python 3.10 يمكنك استخدام عبارات `if-else` لتقليد سلوك `match-case`.

اقترحت العديد من الردود على هذا المنشور أن الأنماط وسهولة القراءة أعلى. هذا مثال للدفاع عن `match-case`.

في حين أن `if-else` مقبولة تقليدياً، فإنها تأتي أيضاً مع العديد من الجوانب السلبية. على سبيل المثال، في كثير من الأحيان، يتبع على المرء كتابة سلاسل معقدة من عبارات `if-else` المتداخلة. يتضمن ذلك استدعاءات متعددة لأساليب (`len()` و `isinstance()`) وما إلى ذلك.

علاوة على ذلك، `if-else`، يتبع على المرء تدمير البيانات بشكل صريح لاستخراج القيم. هذا يجعل الكود الخاص بك غير أنيق وفوضوي.

من ناحية أخرى، تتطابق `Match-case` مع مطابقة الأنماط الهيكيلية مما يجعل هذا الأمر بسيطاً وموजزاً. في المثال أعلاه، تتعامل `Match-case` تلقائياً مع مطابقة النوع `type-matching` وفحص الطول `.variable unpacking` والتفريج المتغير `length check`.

اقرأ المزيد هنا: [مستندات بايثون](#).

المقالة:

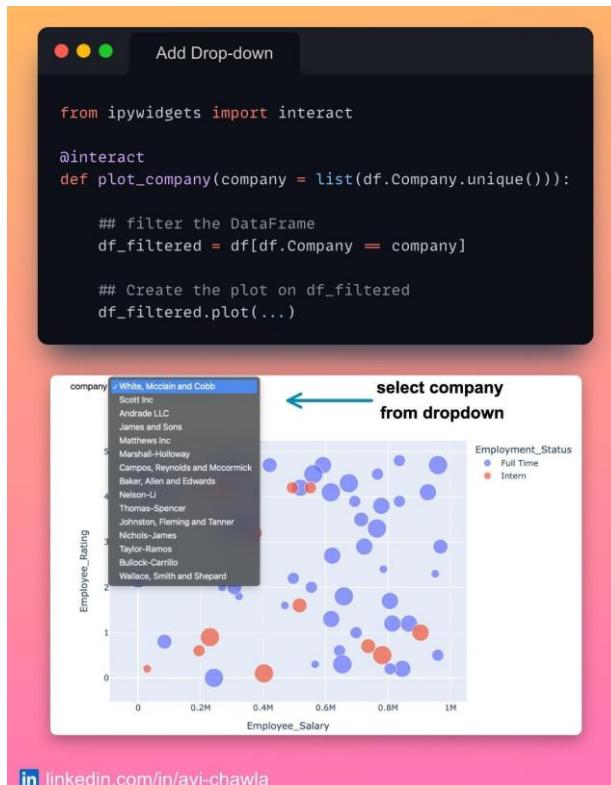
<https://avichawla.substack.com/p/using-dictionaries-in-place-of-if>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/match-case.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/match-case.ipynb)

160) إثراء النوتبوک الخاص بك مع عناصر تحكم تفاعلية

Enrich Your Notebook With Interactive Controls



أثناء استخدام Jupyter، غالباً ما نعيد تشغيل نفس الخلية بشكل متكرر بعد تغيير الإدخال قليلاً. هذا يستغرق وقتاً طويلاً ويجعل أيضاً مهام استكشاف البيانات مملة وغير منظمة.

بدلاً من ذلك، قم بالتركيز على إنشاء عناصر تحكم تفاعلية interactive controls في النوتبوک الخاص بك. يسمح لك هذا بتعديل المدخلات دون الحاجة إلى إعادة كتابة التعليمات البرمجية وإعادة تشغيلها.

في Jupyter، يمكنك القيام بذلك باستخدام الوحدة النمطية **IPywidgets**. يعد تضمين عناصر تحكم تفاعلية أمرًا بسيطًا مثل استخدام **decorator**.

ونتيجة لذلك، فإنه يوفر لك عناصر تحكم تفاعلية مثل القوائم المنسدلة dropdowns وشرائط التمرير sliders. هذا يوفر عليك الكثير من الترميز المتكرر ويجعل النوتبوک الخاص بك منظماً.

شاهد نسخة فيديو من هذا المنشور على LinkedIn : [رابط المنشور](#).

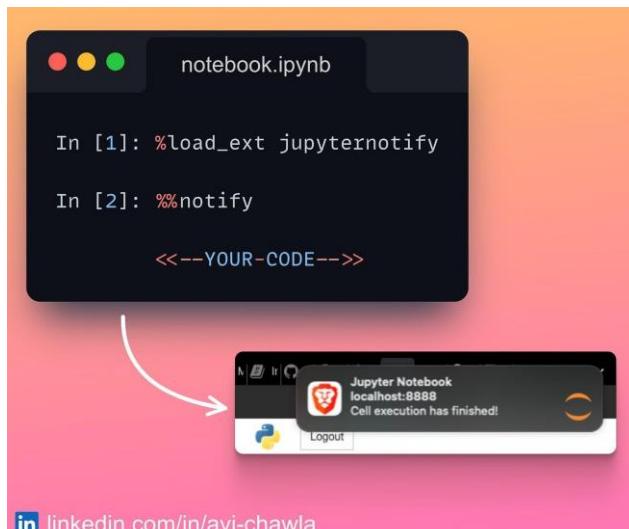
المقالة:

<https://avichawla.substack.com/p/enrich-your-notebook-with-interactive>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Interactive-Controls.ipynb>

Get Jupyter على إشعار عند تنفيذ خلية Notified When Jupyter Cell Has Executed



بعد تشغيل بعض التعليمات البرمجية في خلية Jupyter، غالباً ما ننتقل بعيداً للقيام ببعض الأعمال الأخرى في هذه الأثناء.

هنا، يتعين على المرء العودة مراراً وتكراراً إلى علامة التبويب Jupyter للتحقق مما إذا كانت الخلية قد تم تنفيذها أم لا.

لتتجنب ذلك، يمكنك استخدام الأمر السحري `%%notify`. كما يوحى الاسم، فإنه يبلغ المستخدم عند إكمال خلية Jupyter (سواء كانت ناجحة أو غير ناجحة) عبر إشعار المستعرض. يؤدي النقر فوق الإشعار إلى العودة إلى علامة التبويب Jupyter.

اقرأ المزيد: [GitHub](#)

المقالة:

<https://avichawla.substack.com/p/get-notified-when-jupyter-cell-has>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/Cell-Notification.ipynb>

162) تحليل البيانات باستخدام pandas بدون كود في Jupyter

Data Analysis Using No-Code Pandas In Jupyter

	Name	Company_Name	Employee_City	Employee_Salary	Employment_Status	Employee_Rating
99	Christopher Jones	Matthews Inc	Aliciafort	5,187.04	Full Time	1.50
96	Mitchell Hill	Baker, Allen and Edwards	Aliciafort	4,078.71	Full Time	1.40
44	Dawn Bailey	White, Mcclain and Clark	Aliciafort	11,379.39	Full Time	4.50
80	Donald Bowman	Scott Inc	Aliciafort	4,292.43	Full Time	1.30
48	Kelly Liu	Matthews Inc	Aliciafort	4,413.51	Intern	0.90
20	David Mills	Johnston, Fleming and Clark	Aliciafort	6,917.60	Intern	1.90
75	Vanessa Lamb	Taylor-Ramos	Aliciafort	8,391.54	Full Time	2.70
67	Douglas Kennedy	Andrade LLC	Aliciafort	2,815.63	Full Time	0.80
54	Jeffrey Gonzalez	Taylor-Ramos	Aliciafort	10,401.33	Full Time	4.60
37	Emily Weber	Matthews Inc	Kristaburgh	7,676.39	Intern	2.30
45	Zachary Ellison	James and Sons	Kristaburgh	7,194.54	Full Time	2.60
50	Gina Acosta	Nichols-James	Kristaburgh	7,239.98	Full Time	2.10
22	Jason Reyes	Matthews Inc	Kristaburgh	6,760.68	Full Time	2.80
53	James Wright	Nelson-Li	Kristaburgh	3,980.27	Intern	1.00

(100 rows, 6 cols)

linkedin.com/in/avi-chawla

توفر Pandas API مجموعة واسعة من الدوال لتحليل مجموعات البيانات المجدولة.

ومع ذلك، عبر المشاريع، غالباً ما نستخدم نفس الأساليب مراراً وتكراراً لتحليل بياناتنا. يتكرر هذا بسرعة ويستغرق وقتاً طويلاً.

لتجنب ذلك، استخدم Mito. إنها أداة رائعة تسمح لك بتحليل بياناتك داخل واجهة جدول بيانات في Jupyter، دون كتابة أي كود.

أروع شيء في Mito هو أن كل تعديل في جدول البيانات يولد تلقائياً كود بايثون مكافئاً. هذا يجعل من الملائم للغاية إعادة إنتاج التحليل لاحقاً.

اقرأ المزيد: [التوثيق](#).

المقالة:

<https://avichawla.substack.com/p/data-analysis-using-no-code-pandas>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Mito--No-code-Pandas.ipynb>

Using If-conditions In Place of Dictionaries

```

if-else.py
number = int(input())
if number == 1:
    func1()
elif number == 2:
    func2()
else:
    func3()

dict.py
number = int(input())
func_map = {1:func1,
            2:func2}
func_map.get(number, func3())

```

linkedin.com/in/avi-chawla

تُستخدم القواميس Dictionaries بشكل أساسي كهيكل بيانات في بايثون للحفاظ على أزواج القيمة - المفتاح .key-value pairs

ومع ذلك، هناك حالة استخدام خاصة أخرى يمكن للقواميس التعامل معها. هذا هو - إزالة شروط IF من التعليمات البرمجية الخاصة بك.

ضع في اعتبارك مقتطف الكود أعلاه. هنا، بما يتوافق مع قيمة الإدخال، نستدعي دالة محددة. تتطلب الطريقة التقليدية منك برمجة كل حالة.

ولكن باستخدام القاموس، يمكنك استرداد الدالة المقابلة مباشرة من خلال تزويدها بالمفتاح. هذا يجعل شفترتك موجزة وأنيقة.

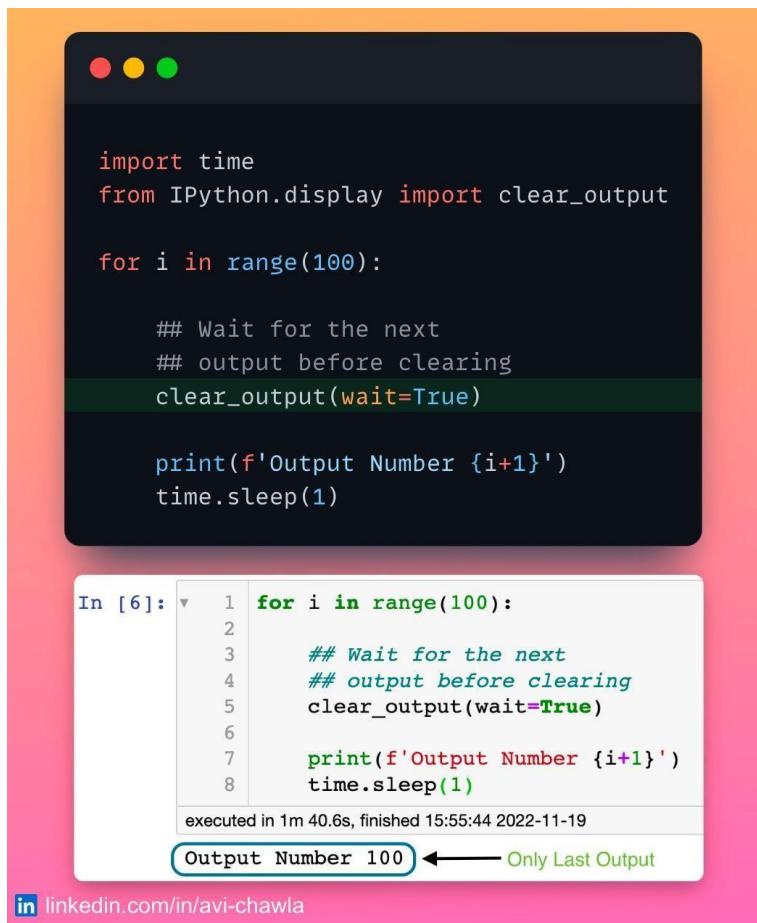
المقالة:

<https://avichawla.substack.com/p/using-dictionaries-in-place-of-if>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/dict-vs-ifelse.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/dict-vs-ifelse.ipynb)

164) مسح إخراج الخلية في نوتبوك Jupyter أثناء وقت التنفيذ Clear Cell Output In Jupyter Notebook During Run-time



```

import time
from IPython.display import clear_output

for i in range(100):

    ## Wait for the next
    ## output before clearing
    clear_output(wait=True)

    print(f'Output Number {i+1}')
    time.sleep(1)

```

In [6]:

```

1 for i in range(100):
2
3     ## Wait for the next
4     ## output before clearing
5     clear_output(wait=True)
6
7     print(f'Output Number {i+1}')
8     time.sleep(1)

```

executed in 1m 40.6s, finished 15:55:44 2022-11-19

Output Number 100 ← Only Last Output

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

أثناء استخدام Jupyter، غالباً ما نطبع العديد من التفاصيل لتبني تقدم الكود.

ومع ذلك، يصبح الأمر محبطاً عندما تراكم لوحة الإخراج مجموعة من التفاصيل، لكننا مهتمون فقط بأحدث المخرجات. علاوة على ذلك، يمكن أن يكون التمرير إلى أسفل الإخراج في كل مرة مزعجاً أيضاً.

لمسح إخراج الخلية، يمكنك استخدام طريقة `clear_output` من الحزمة `IPython`. عند الاستدعاء، ستزيل الإخراج الحالي للخلية، وبعد ذلك يمكنك طباعة أحدث التفاصيل.

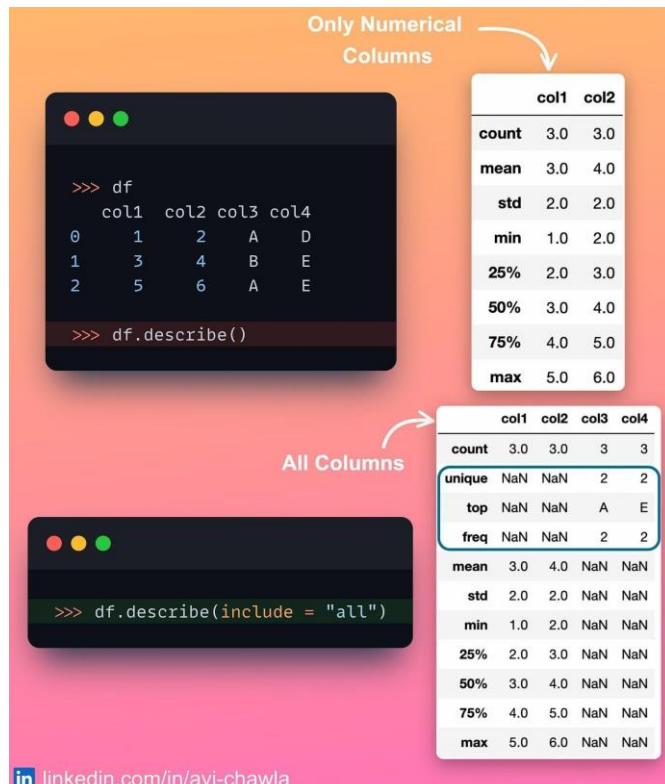
المقالة:

<https://avichawla.substack.com/p/clear-cell-output-in-jupyter-notebook>

الكود:

<https://avichawla.substack.com/p/clear-cell-output-in-jupyter-notebook>

A Hidden Pandas Feature of Describe Method In Pandas



[in](https://www.linkedin.com/in/avi-chawla) linkedin.com/in/avi-chawla

تُستخدم طريقة `describe()` في Pandas بشكل شائع لطباعة الإحصائيات الوصفية حول البيانات.

ولكن هل سبق لك أن لاحظت أن ناتجها يقتصر دائمًا على الأعمدة العددية؟

بالطبع، تفاصيل مثل المتوسط `mean`، المتوسط `median`، الانحراف المعياري `std`. `dev`، وما إلى ذلك، لا يحمل أي معنى للأعمدة غير الرقمية، لذا فإن النتائج منطقية تماماً.

ومع ذلك، يمكن أن توفر `describe()` أيضاً ملخصاً سريعاً للأعمدة غير الرقمية. يمكنك القيام بذلك بتحديد `include="all"`. نتيجة لذلك، سيزيد عدد العناصر الفريدة، العنصر العلوي مع تكراره.

اقرأ المزيد: [الوثيق](#).

المقالة:

<https://avichawla.substack.com/p/a-hidden-feature-of-describe-method>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Hidden-Feature-of-Describe.ipynb>

166) استخدم Slotted Class لتحسين كود بایثون الخاص بك Use Slotted Class To Improve Your Python Code

```
Without_slots.py
class Person:
    def __init__(self, name, age):
        self.Name = name
        self.Age = age
    person = Person('Mike', 22)
    person.name = 'Peter'
    ## No Error
```

'Name' mistakenly written as 'name' raises no error


```
With_slots.py
class Person:
    __slots__ = ['Name', 'Age']
    def __init__(self, name, age):
        self.Name = name
        self.Age = age
    person = Person('Mike', 22)
    person.name = 'Peter'
    ## AttributeError: 'Person' object has no attribute 'name'
```

defining __slots__ raises AttributeError

[in linkedin.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

إذا كنت ترغب في تحديد السمات التي يمكن أن تحملها الفئة class، ففكري في تحديدها ك slotted class.

أثناء تحديد الفئات، تسمح لك `slots` بتحديد سمات الفئة صراحةً. هذا يعني أنه لا يمكنك إضافة سمات جديدة بشكل عشوائي إلى كائن slotted class. هذا يوفر العديد من المزايا.

على سبيل المثال، تعتبر slotted classes فعالة في الذاكرة وتتوفر وصولاً أسرع إلى سمات الفئة. علاوة على ذلك، يساعدك أيضاً على تجنب الأخطاء المطبعية الشائعة. قد يكون هذا، في بعض الأحيان، خطأ مكلفاً يمكن أن يمر دون أن يلاحظه أحد.

اقرأ المزيد: [StackOverflow](#)

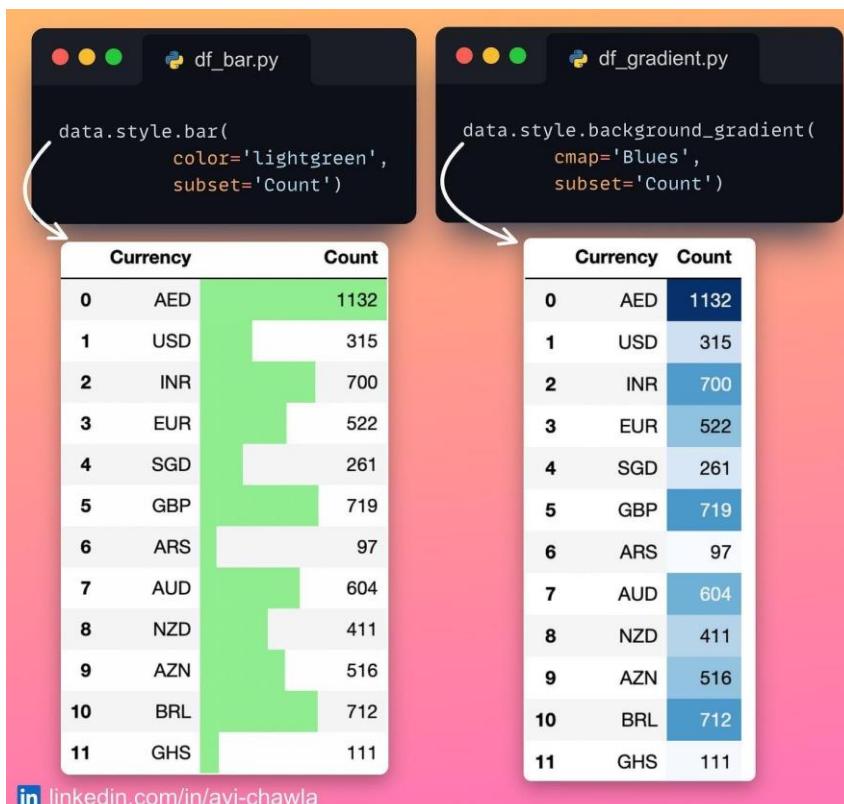
المقالة:

<https://avichawla.substack.com/p/use-slotted-class-to-improve-your>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/Slotted-Classes.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Slotted-Classes.ipynb)

167) أوقف تحليل الجداول الخام. استخدم التصميم بدلاً من ذلك! Stop Analysing Raw Tables. Use Styling Instead



Jupyter DataFrame هو DataFrame قائم على الويب. وبالتالي، عندما تقوم بطباعة / عرض في IDE Jupyter يتم تقديمها باستخدام HTML و CSS.

هذا يعني أنه يمكنك تصميم مخرجاتك بعدة طرق مختلفة.

للحفاظ بذلك، استخدم Styling API من Pandas. هنا، يمكنك إجراء العديد من التعديلات المختلفة على كائن DataFrame's styler (df.style). نتيجة لذلك، سيتم عرض DataFrame بتصميم محدد.

التصميم Styling يجعل هذه الجداول جذابة بصرياً. علاوة على ذلك، فإنه يسمح بفهم البيانات بشكل أفضل من عرض الجداول الأولية.

اقرأ المزيد هنا: [التوثيق](#).

Explore (168) استكشاف بيانات CSV مباشرةً من الترمinal CSV Data Right From The Terminal



إذا كنت ترغب في استكشاف بعض بيانات CSV بسرعة، فقد لا تحتاج دائمًا إلى تشغيل جلسة Jupyter. بدلاً من ذلك، باستخدام "csvkit"، يمكنك القيام بذلك من الترمinal نفسه. كما يوحى الاسم، فإنه يوفر مجموعة من أدوات سطر الأوامر لتسهيل مهام تحليل البيانات.

يتضمن ذلك تحويل Excel إلى CSV، وعرض أسماء الأعمدة، وإحصاءات البيانات، والاستعلام باستخدام SQL. علاوة على ذلك، يمكنك أيضًا أداء وظائف Pandas الشائعة مثل الفرز sorting والدمج merging والتقطيع slicing.

اقرأ المزيد: [الوثيق](#).

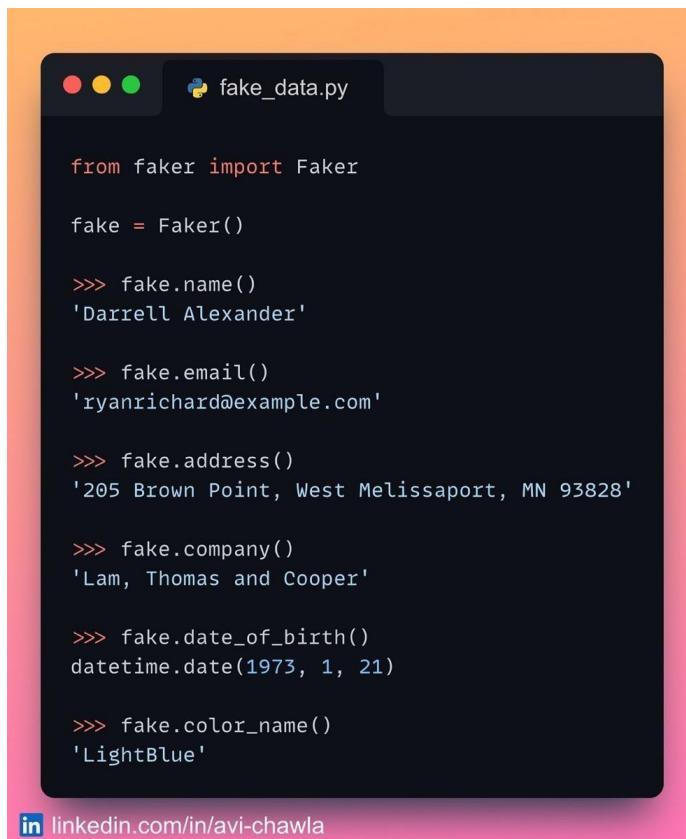
المقالة:

<https://avichawla.substack.com/p/explore-csv-data-right-from-the-terminal>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/CSV-From-Terminal.ipynb>

169) أنشئ بياناتك المزيفة في ثوانٍ Generate Your Own Fake Data In Seconds



```
from faker import Faker

fake = Faker()

>>> fake.name()
'Darrell Alexander'

>>> fake.email()
'ryanrichard@example.com'

>>> fake.address()
'205 Brown Point, West Melissaport, MN 93828'

>>> fake.company()
'Lam, Thomas and Cooper'

>>> fake.date_of_birth()
datetime.date(1973, 1, 21)

>>> fake.color_name()
'LightBlue'
```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

عادة، لتنفيذ / اختبار خط أتايب، نحتاج إلى تزويده ببعض البيانات الوهمية data dummy.

على الرغم من استخدام مكتبة "random" لبايثون، يمكن للمرء إنشاء سلاسل عشوائية واعداد حقيقية وأعداد صحيحة. ومع ذلك، نظرًا لكونه عشوائياً، فإنه لا ينتج أي بيانات ذات معنى مثل أسماء الأشخاص وأسماء المدن ورسائل البريد الإلكتروني وما إلى ذلك.

هنا، يمكن أن يستغرق البحث عنمجموعات بيانات مفتوحة المصدر وقتاً طويلاً. علاوة على ذلك، من المحتمل أن مجموعة البيانات التي تجدها لا تلبي متطلباتك بشكل جيد.

تعد الوحدة النمطية **Faker** في بايثون حلًاً مثالياً لذلك. يسمح لك Faker بإنشاء بيانات مزيفة للغاية (لكنها ذات معنى) بسرعة. علاوة على ذلك، يمكنك أيضًا إنشاء بيانات محددة لمجموعة ديمografية.

اقرأ المزيد هنا: [التوثيق](#).

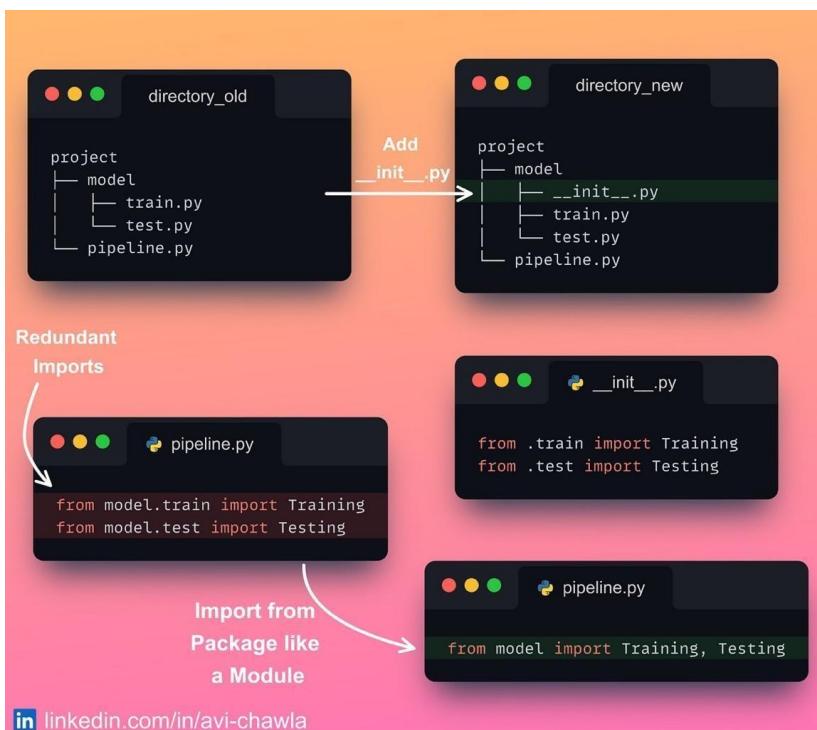
المقالة:

<https://avichawla.substack.com/p/generate-your-own-fake-data-in-seconds>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Testing/Generate-Fake-Data.ipynb>

170) قم باستيراد حزمة بایثون الخاصة بك كوحدة نمطية Import Your Python Package as a Module



وحدة بایثون هي ملف بایثون (**.py**). تسمى المجموعة المنظمة لمثل هذه الملفات حزمة بایثون **.python package**

أثناء تطوير المشاريع الكبيرة، من الممارسات الجيدة تحديد ملف داخل الحزمة.

ضع في اعتبارك أن لها فئة **train.py** و **test.py** بها فئة **Training** و **Testing**.

بدون **__init__.py** ، يتبعن على المرء أن يستوردها صراحةً من ملف بایثون محدد. نتيجة لذلك، من غير الضروري كتابة عبارات الاستيراد.

باستخدام **__init__.py** ، يمكنك تجميع ملفات بایثون في وحدة نمطية واحدة قابلة للاستيراد. بمعنى آخر، يوفر آلية للتعامل مع الحزمة بأكملها كوحدة بایثون.

هذا يحميك من كتابة عبارات استيراد مكررة ويجعل كودك أكثر نظافة في نص الاستدعاء.

اقرأ المزيد في هذه المدونة: [رابط المدونة](#).

المقالة:

<https://avichawla.substack.com/p/import-your-python-package-as-a-module>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Import-Package-As-Module.ipynb>

171) حدد الحلقات ونفذها في %%timeit

Runs In %%timeit



number of loops (1000)

number of runs (4)

```
In [1]: %%timeit -n 1000 -r 4
time.sleep(2)

## 2 s ± 800 µs per loop
## (mean ± std. dev. of 4 runs, 1000 loops each)
```

[in linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

نستخدم عادةً الأمر السحري **timeit** (أو **%%timeit**) لقياس وقت تنفيذ التعليمات البرمجية الخاصة بنا.

هنا، تحدد **timeit** عدد عمليات التشغيل اعتماداً على المدة التي يستغرقها السكريبت للتنفيذ. هذا هو السبب في أنك ترى عدداً مختلفاً من الحلقات loops (والتنفيذات runs) عبر أجزاء مختلفة من التعليمات البرمجية.

ومع ذلك، إذا كنت تريد تحديد عدد الحلقات والتنفيذات بشكل صريح، فاستخدم الخيارين **-n** و **-r**. استخدم **-n** لتحديد الحلقات و **-r** لرقم التنفيذ.

المقالة:

<https://avichawla.substack.com/p/specify-loops-and-runs-in-timeit>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Change-Loops-in-Timeit.ipynb>

172) مخططات الشلال: أفضل بديل للمخطط الخطى / Waterfall Charts: A Better Alternative to Line/Bar Plot



إذا كنت ترغب في تصور قيمة خلال فترة ما، فقد لا يكون المخطط الخطى (أو المخطط الشريطي bar-plot) دائمًا خيارًا مناسبًا.

المخطط الخطى (أو المخطط الشريطي) يصور القيم الفعلية في المخطط. وبالتالي، في بعض الأحيان، قد يكون من الصعب تقدير حجم التغييرات المتزايدة بصرياً.

بدلاً من ذلك، يمكنك استخدام مخطط الشلال waterfall chart يصور هذه الفروق المتدرجية بأناقة.

لإنشاء واحدة، يمكنك استخدام `waterfall_chart` في بايثون. هنا، يتم تمثيل قيم البداية والنهاية بواسطة الشريطين الأول والأخير. يتم أيضًا تميز التغييرات الهامشية تلقائياً بالألوان، مما يسهل تفسيرها.

اقرأ المزيد هنا: [GitHub](#).

المقالة:

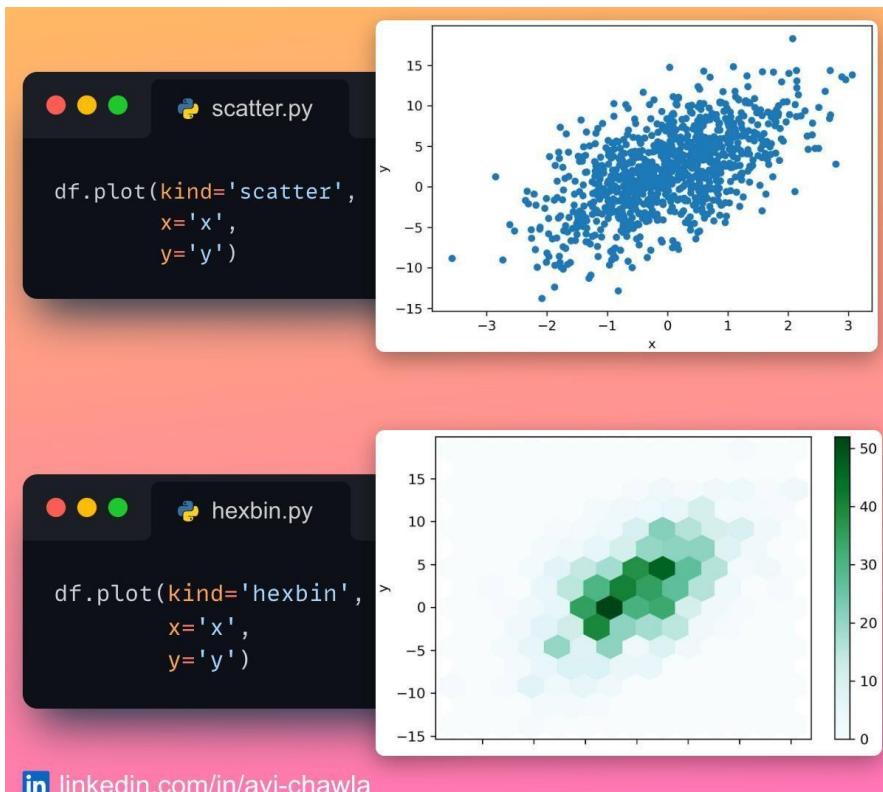
<https://avichawla.substack.com/p/waterfall-charts-a-better-alternative>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Waterfall-Charts.ipynb>

173) مخططات Hexbin كبديل أكثر ثراءً للمخططات المبعثرة

Hexbin Plots As A Richer Alternative to Scatter Plots



المخططات المبعثرة Scatter plots مفيدة للغاية لتصور مجموعتين من المتغيرات العددية. ولكن عندما يكون لديك، على سبيل المثال، آلاف نقاط البيانات، يمكن أن تصبح المخططات المبعثرة كثيفة للغاية بحيث لا يمكن تفسيرها.

يمكن أن يكون Hexbins اختياراً جيداً في مثل هذه الحالات. كما يوحى الاسم، يقومون بتجميع مساحة المخطط في مناطق سداسية hexagonal regions. يتم تعين كثافة لون لكل منطقة بناءً على طريقة التجميع المستخدمة (عدد النقاط، على سبيل المثال).

تعتبر Hexbins مفيدة بشكل خاص لفهم انتشار البيانات. غالباً ما يعتبر بدلاً أنيقاً لمخطط التشتت. علاوة على ذلك، يسهل binning تحديد مجموعات البيانات وتصوير الأنماط.

المقالة:

<https://avichawla.substack.com/p/hexbin-plots-as-a-richer-alternative>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/HexBins.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/HexBins.ipynb)

174) أصبح استيراد الوحدات النمطية سهلاً باستخدام Importing Modules Made Easy with Pyforest Pyforest



linkedin.com/in/avi-chawla

يبدأ الطلاب النموذجيون المرتبطون بالبرمجة في علم البيانات باستيراد الوحدات النمطية ذات الصلة.

ومع ذلك، عبر النوتبوك / المشاريع، فإن الوحدات التي يستوردها المرء هي نفسها في الغالب. وبالتالي، فإن مهمة استيراد جميع المكتبات الفردية متكررة نوعاً ما.

باستخدام **pyforest**، يمكنك استخدام مكتبات بايثون الشائعة دون استيرادها صراحةً. الشيء الجيد هو أنه يستورد جميع المكتبات باتفاقياتها القياسية. على سبيل المثال، يتم استيراد **pandas** مع الاسم المستعار **pd**.

مع ذلك، يجب أن تلاحظ أيضاً أنه من الممارسات الجيدة إبقاء Pyforest مقصورةً على مراحل النماذج الأولية. هذا لأنه بمجرد أن تقول، طور خط الأنابيب الخاص بك وافتحه، قد يواجه المستخدمون الآخرون بعض الصعوبات في فهمه.

ولكن إذا كنت ترغب في بعض التجارب غير الرسمية، فلماذا لا تستخدمها بدلاً من كتابة جميع عمليات الاستيراد يدوياً؟

اقرأ المزيد: [GitHub](#)

المقالة:

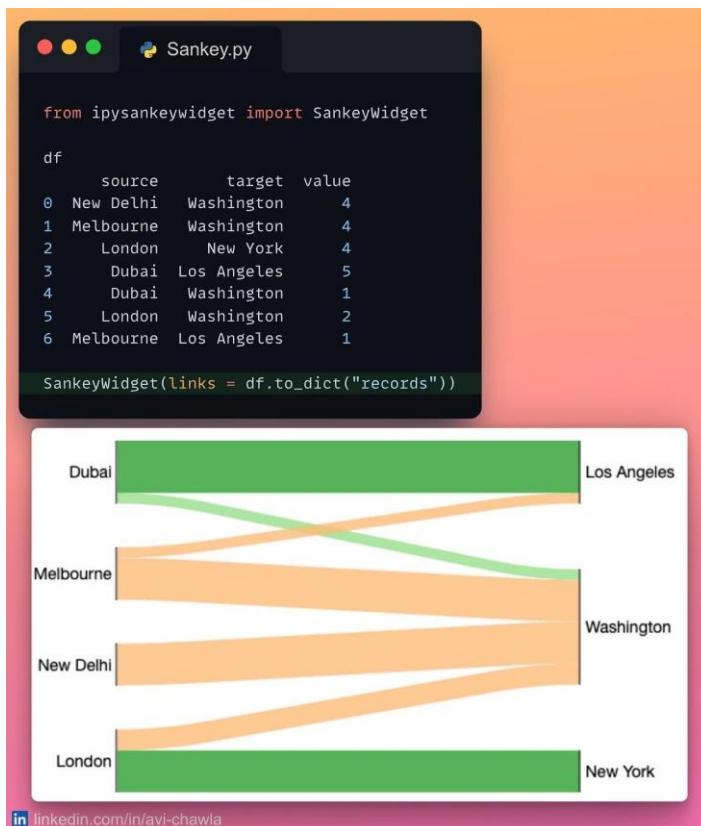
<https://avichawla.substack.com/p/importing-modules-made-easy-with>

الكلوب:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Cool%20Tools/No-Library-Importing.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/No-Library-Importing.ipynb)

175) تحليل بيانات التدفق باستخدام مخططات سانكي

Analyse Flow Data With Sankey Diagrams



يمكن تفسير العديد من مهام تحليل البيانات المجدولة tabular data analysis على أنها تدفق بين المصدر والهدف.

هنا، لا يعد التحليل اليدوي للتقارير / البيانات المجدولة لرسم الأفكار هو النهج الصحيح عادةً.

بدلاً من ذلك، تعمل المخططات الانسية Flow diagrams كبديل رائع في مثل هذه الحالات.

نظرًا لكونها جذابة بصرياً، فإنها تساعدك بشكل كبير في استخلاص رؤى مهمة من بياناتك، والتي قد تجد صعوبة في استنتاجها من خلال النظر إلى البيانات يدوياً.

على سبيل المثال، من الرسم البياني أعلاه، يمكن للمرء أن يستنتج بسرعة ما يلي:

1. تستضيف واشنطن جزءاً من جميع الأصول.

2. تستقبل نيويورك الركاب من لندن فقط.
3. غالبية الرحلات في لوس أنجلوس تأتي من دبي.
4. جميع المعالم السياحية من نيودلهي تذهب إلى واشنطن.

تخيل الآن القيام بذلك بمجرد النظر إلى البيانات المجدولة. لن يستغرق الأمر وقتاً طويلاً فحسب، بل هناك أيضاً احتمالية أن تفوتك بعض الأفكار.

لإنشاء رسم تخطيطي للانسياب، يمكنك استخدام `floWeaver`. يساعدك على تصور بيانات الانسياب باستخدام مخططات سانكي `Sankey diagrams`.

اقرأ المزيد هنا: [التوثيق](#).

المقالة:

<https://avichawla.substack.com/p/analyse-flow-data-with-sankey-diagrams>

ال코드:

<https://avichawla.substack.com/p/analyse-flow-data-with-sankey-diagrams>

176) تتبع الميزات أصبح بسيطًا في محولات Sklearn Tracking Made Simple In Sklearn Transformers

```
numpy_output.py
```

```
from sklearn.preprocessing import PolynomialFeatures

df = pd.DataFrame({'col_A': [1, 3, 5], 'col_B': [2, 4, 6]})

array([[ 1.,  1.,  2.,  1.,  2.,  4.],
       [ 1.,  3.,  4.,  9., 12., 16.],
       [ 1.,  5.,  6., 25., 30., 36.]])
```

```
pandas_output.py
```

```
from sklearn import set_config

set_config(transform_output = "pandas")

df = pd.DataFrame({'col_A': [1, 3, 5], 'col_B': [2, 4, 6]})

      1  col_A  col_B  col_A^2  col_A*col_B  col_B^2
0    1.0      1.0      2.0      1.0      2.0      4.0
1    1.0      3.0      4.0      9.0     12.0     16.0
2    1.0      5.0      6.0     25.0     30.0     36.0
```

[in linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

في الآونة الأخيرة، أعلنت scikit-Learn عن إصدار أحد أكثر التحسينات المنتظرة في الخلاصة، يمكن الآن ضمان sklearn لـPandas DataFrames لإخراج

حتى الآن، كانت محولات Sklearn's transformers مؤمنة لقبول Pandas DataFrame كمدخلات. لكنهم عادوا دائمًا إلى مصفوفة NumPy كإخراج. نتيجة لذلك، كان لابد من عرض الإخراج يدوياً مرة أخرى على Pandas DataFrame. هذا، في بعض الأحيان، جعل من الصعب تتبع وتعيين أسماء للسممات.

على سبيل المثال، ضع في اعتبارك مقتطف الكود أعلاه.

في **numpy_output.py** ، من الصعب استنتاج اسم (أو حساب) عمود من خلال النظر إلى مصفوفة NumPy

ومع ذلك، في الإصدار القادم، يمكن للمحول إرجاع Pandas DataFrame هذا يجعل أسماء ميزات التتبع بسيطة بشكل لا يصدق.

اقرأ المزيد: [صفحة الإصدار](#).

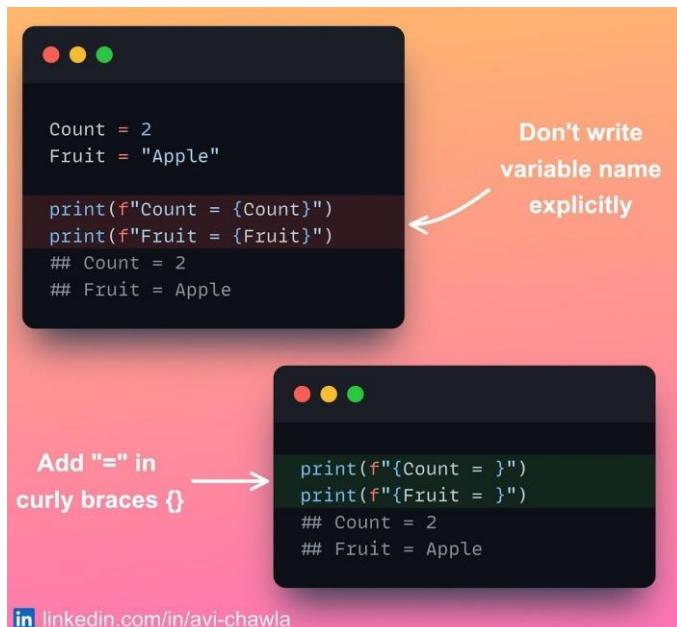
المقالة:

<https://avichawla.substack.com/p/feature-tracking-made-simple-in-sklearn>

الكلوب:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Sklearn/Sklearn%20-%20Pandas%20Output%20Demo.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Sklearn/Sklearn%20-%20Pandas%20Output%20Demo.ipynb)

177) ميزة أقل شهرة لسلالس f في بايثون Feature of f-strings in Python



[in](https://www.linkedin.com/in/avi-chawla) linkedin.com/in/avi-chawla

أثناء التصحيح debugging، غالباً ما يطبع المreu اسم المتغير بشكل صريح بقيمةه لتحسين فحص الكود.

على الرغم من عدم وجود خطأ في هذا النهج، إلا أنه يجعل بياناتك المطبوعة فوضوية وطويلة.
توفر سلاسل f-f-strings (في بايثون حالاً أنيقاً لهذا الأمر).

لطباعة اسم المتغير، يمكنك إضافة علامة يساوي (=) في الأقواس المترجة بعد المتغير. سيؤدي هذا إلى طباعة اسم المتغير مع قيمته ولكنه موجز ونظيف.

المقالة:

<https://avichawla.substack.com/p/less-known-feature-of-f-strings>

: الكود

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Debugging/f-string-print-var-name.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Debugging/f-string-print-var-name.ipynb)

Don't لقياس وقت التنفيذ (178) Use time.time() To Measure Execution Time

```
time.py
import time
start = time.time()
time.sleep(10)
end = time.time()

print(end - start)
## 10.00482
```



```
perf_counter.py
import time
start = time.perf_counter()
time.sleep(10)
end = time.perf_counter()

print(end - start)
## 10.00435
```

كثيراً ما تستخدم طريقة `time.time()` من مكتبة الوقت لقياس وقت التنفيذ `.execution time`.

ومع ذلك، فإن `(time)` ليس مخصوصاً لتوقيت الكود الخاص بك. بدلاً من ذلك، الغرض الفعلي هو معرفة الوقت الحالي. هذا، في كثير من الأحيان، يعرض للخطر دقة قياس وقت التنفيذ الدقيق.

الطريقة الصحيحة هي استخدام `(time.perf_counter())`، التي تعامل مع الوقت النسبي `.relative time`. وبالتالي، تعتبر الطريقة الأكثر دقة لتوقيت الكود الخاص بك.

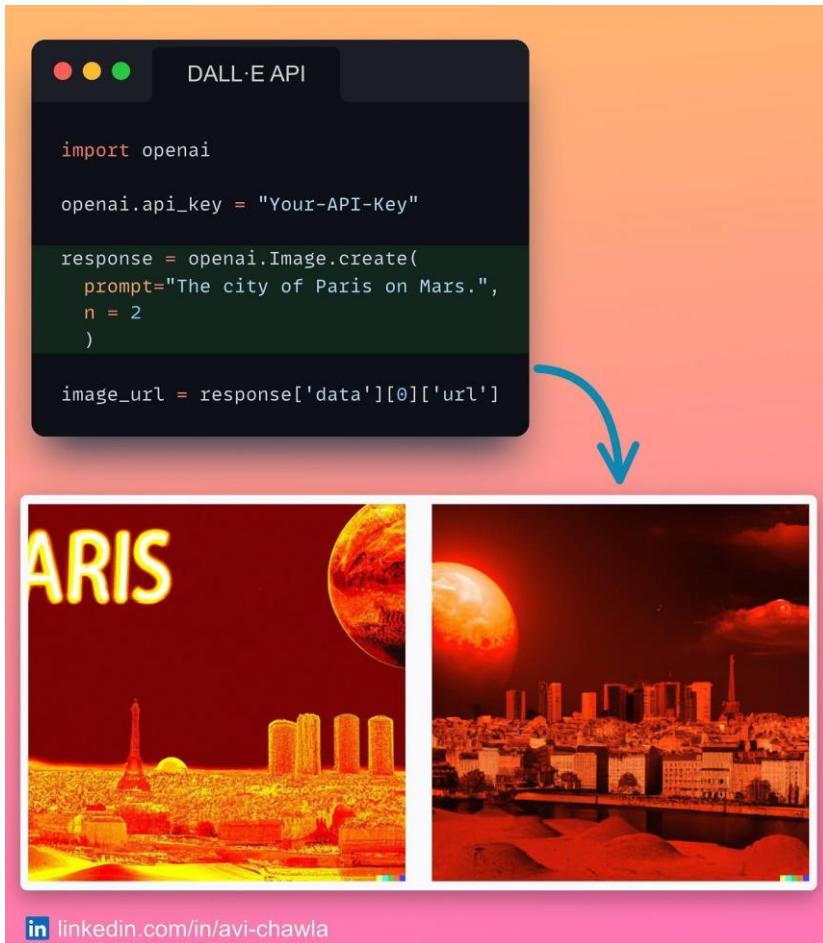
المقالة:

<https://avichawla.substack.com/p/dont-use-timetime-to-measure-execution>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Python/Dont-Use-Time.Time.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Dont-Use-Time.Time.ipynb)

179) يمكنك الآن استخدام DALL·E مع واجهة برمجة Now You Can Use DALL·E With OpenAI API OpenAI API



يمكن الآن الوصول إلى DALL·E · OpenAI API باستخدام واجهة برمجة تطبيقات OpenAI API.

أصدرت شركة OpenAI مؤخرًا إعلانًا كبيراً في الخلاصة، يمكن للمطورين الآن دمج نموذج تحويل النص إلى صورة الشهير من OpenAI DALL·E في تطبيقاتهم باستخدام OpenAI API.

لتحقيق ذلك، أولاً، حدد مفتاح API الخاص بك (تم الحصول عليه بعد التسجيل). بعد ذلك، قم بتمرير مطالبة نصية text prompt لإنشاء الصورة المقابلة.

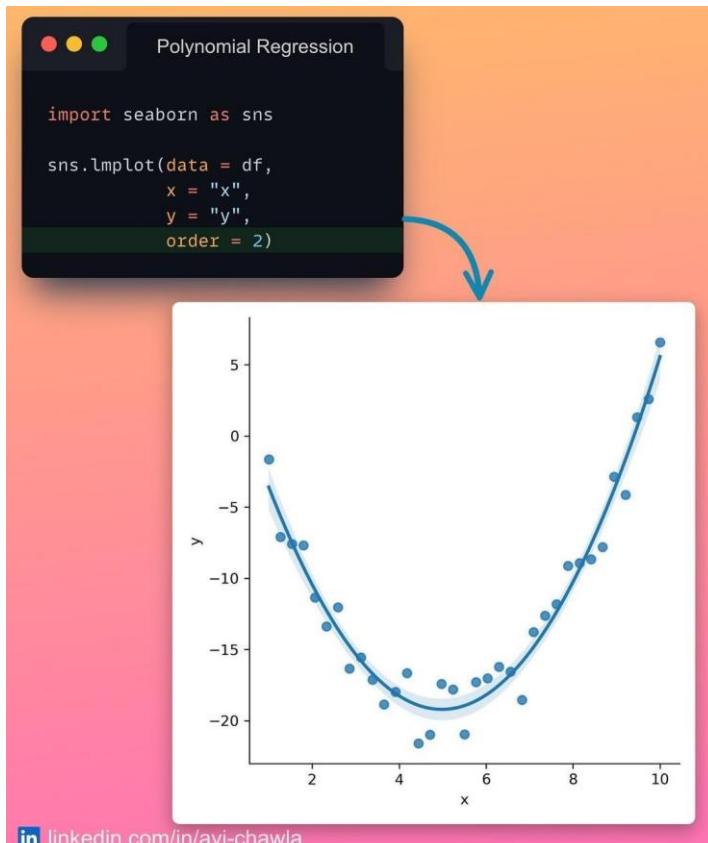
المقالة:

<https://avichawla.substack.com/p/now-you-can-use-dalle-with-openai>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/ML-AI%20News/DALL%C2%B7E-API.ipynb>

180) مخطط الانحدار الخطي متعدد الحدود أصبح سهلاً مع Polynomial Linear Regression Plot Made Easy Seaborn With Seaborn



أثناء إنشاء مخططات مبعثرة scatter plots، غالباً ما يهتم المرء بعرض الانحدار الخطي linear regression (البسيط) أو متعدد الحدود polynomial regression على نقاط البيانات.

هنا، يمكن أن يكون تدريب نموذج وتصميمه يدوياً في المخطط مهمة شاقة يجب القيام بها. بدلاً من ذلك، مع lmplot() من SeaBorn، يمكنك إضافة انحدار إلى المخطط، دون تدريب نموذج صريح.

حدد درجة كثير الحدود كمعامل "order". سيضيف Seaborn الانحدار المقابل على مخطط التشتت.

اقرأ المزيد هنا: [Seaborn Docs](#)

المقالة:

<https://avichawla.substack.com/p/polynomial-linear-regression-plot>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/Poly-LR-Plot.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Poly-LR-Plot.ipynb)

181) استرجع المخرجات المحسوبة مسبقاً في Jupyter Notebook Retrieve Previously Computed Output In Jupyter Notebook

```
In [3]: df.groupby("col1").col2.mean().reset_index()
Out[3]:
   col1  col2
0     A    4.0
1     B    3.0
2     C    5.0

In [4]: Out[3]
Out[4]:
   col1  col2
0     A    4.0
1     B    3.0
2     C    5.0
```

linkedin.com/in/avi-chawla

هذا بالفعل أحد أروع الأشياء التي تعلمتها عن Jupyter Notebooks مؤخراً.

هل سبق لك أن كنت في موقف نسيت فيه تخصيص النتائج التي تم الحصول عليها بعد إجراء بعض الحسابات لمتغير؟ إذا لم يكن هناك خيار، يتبعن على المرء أن يعيد حساب النتيجة عن غير قصد وإسنادها إلى متغير لاستخدامها مرة أخرى.

لحسن الحظ، ليس عليك القيام بذلك بعد الآن!

يوفر IPython قاموس "Out" ، والذي يمكنك استخدامه لاسترداد إخراج الخلية. كل ما عليك فعله هو تحديد رقم الخلية كمفتاح القاموس، والذي سيعيد الإخراج المقابل. أليس هذا رائعًا؟

اعرض نسخة فيديو من هذا المنشور على LinkedIn : رابط المنشور.

المقالة:

<https://avichawla.substack.com/p/retrieve-previous-computed-output>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/Retrieve-Cell-Output.ipynb>

Parallelize Swifter مع Pandas Apply() موازاة (182) "Pandas Apply()" With Swifter



مكتبة Pandas ليس لديها دعم متصل لموازاة parallelize عملياتها. وبالتالي، فهي تتلزم دائمًا بحساب أحادي النواة، حتى عندما تكون النوى الأخرى خاملة.

ترداد الأمور سوءً عندما نستخدم `apply()` في Pandas. `apply()` ليست سوى حلقة متألقة. نتيجة لذلك، لا يمكنها حتى الاستفادة من التوجيه `vectorization`.

الحل السريع للتوازي vectorization هو استخدام `swifter` بدلاً من ذلك.

يتيح لك `Swifter` تطبيق أي دالة على Pandas DataFrame بطريقة متوازية. ونتيجة لذلك، فإنه يوفر مكاسب كبيرة في الأداء مع الحفاظ على البنية القديمة. كل ما عليك فعله هو استخدام `.df.apply` بدلاً من `.df.swifter.apply`

اقرأ المزيد هنا: [.Swifter Docs](#)

المقالة:

<https://avichawla.substack.com/p/parallelize-pandas-apply-with-swifter>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Faster-Apply-With-Swifter%20.ipynb>

183) إنشاء إطار بيانات خالي من المشاكل باستخدام الحافظة

Create DataFrame Hassle-free By Using Clipboard



يعتقد العديد من مستخدمي Pandas أنه لا يمكن تحميل إطار بيانات DataFrame إلا من الفرض. مهما يكن ... هذه ليست الحقيقة.

تخيل أن المرء يريد إنشاء DataFrame من بيانات مجدولة مطبوعة على موقع ويب. هنا، من المرجح أن يتم إغرائهم بنسخ المحتويات إلى ملف CSV وقراءتها باستخدام طريقة `read_csv()`. لكن هذا ليس نهجًا مثالياً هنا.

بدلاً من ذلك، باستخدام طريقة `read_clipboard()`، يمكنك التخلص من خطوة CSV تماماً.

تتيح لك هذه الطريقة إنشاء DataFrame من البيانات المجدولة tabular data المخزنة في مخزن الحافظة clipboard buffer. وبالتالي، تحتاج فقط إلى نسخ البيانات واستدعاء الطريقة لإنشاء إطار بيانات. هذا أسلوب أنيق يوفر الكثير من الوقت.

اقرأ المزيد هنا: [Pandas Docs](#)

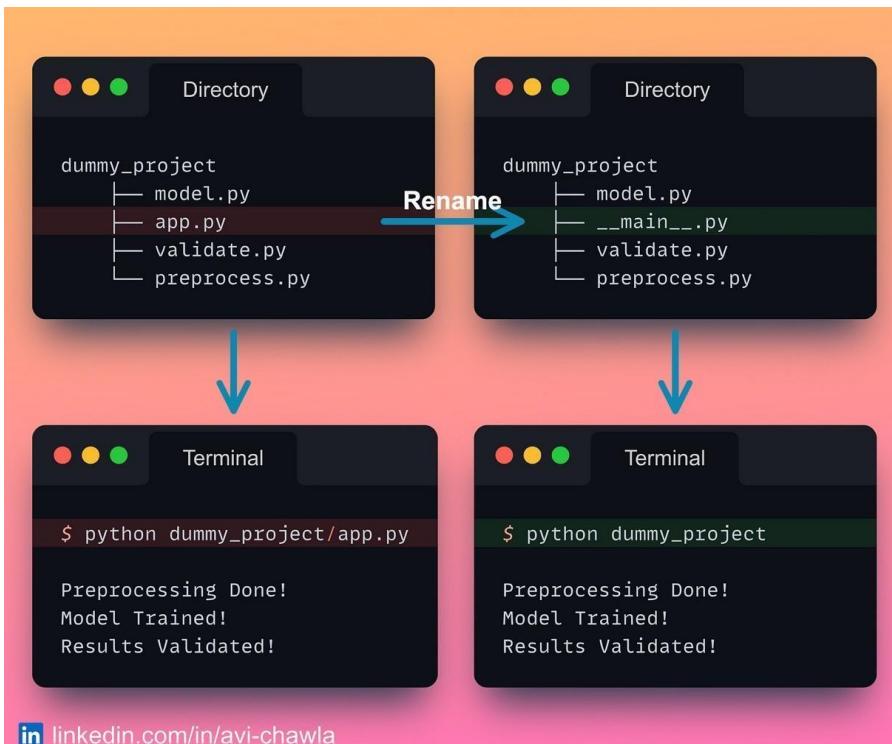
المقالة:

<https://avichawla.substack.com/p/create-dataframe-hassle-free-by-using>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Pandas/DataFrame-From-Cipboard.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/DataFrame-From-Cipboard.ipynb)

184) قم بتشغيل دليل مشروع بايثون كـسكريبت Run Python Project Directory As A Script



[in linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

يتم تنفيذ سكريبت بايثون عندما ندير ملف **py**. في المشاريع الكبيرة التي تحتوي على العديد من الملفات، غالباً ما يكون هناك مصدر (أو قاعدة) بايثون نبدأ منه ببرامجنا.

لجعل الأمور أكثر بساطة، يمكنك بدلاً من ذلك إعادة تسمية هذه القاعدة إلى **main.py**. نتيجة لذلك، يمكنك تنفيذ خط الأنابيب بأكمله عن طريق تشغيل الدليل الأصلي نفسه.

هذا موجز ويسهل أيضاً على المستخدمين الآخرين استخدام مشروعك.

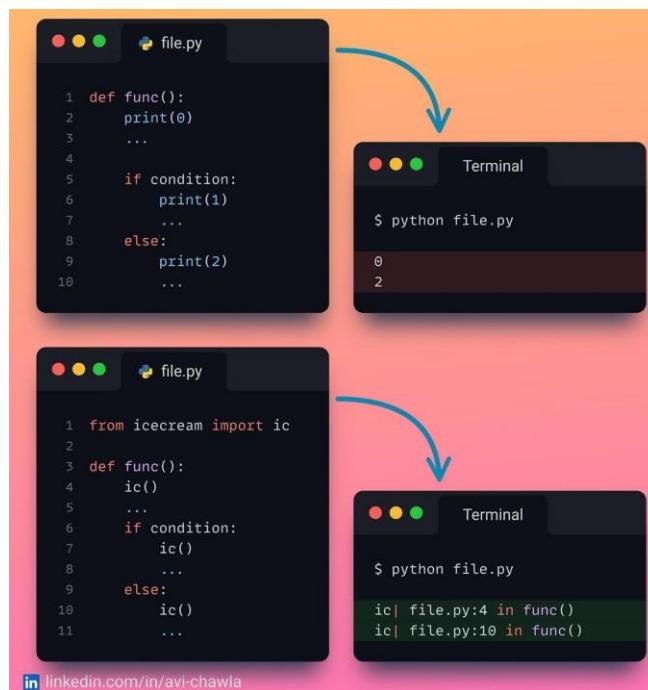
المقالة:

<https://avichawla.substack.com/p/run-python-project-directory-as-a>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Run-Directory-As-Script.ipynb>

185) فحص تدفق البرنامج باستخدام IceCream



أثناء التصحيح debugging، غالباً ما يكتب الماء العديد من عبارات `print()` لفحص تدفق البرنامج program's flow. هذا صحيح بشكل خاص عندما يكون لدينا العديد من شروط IF.

يمكن أن يكون استخدام عبارات `() ic` الفارغة من مكتبة IceCream بدلاً أفضل هنا. يقوم بإخراج العديد من التفاصيل الإضافية التي تساعدي في فحص تدفق البرنامج.

يتضمن هذا رقم السطر، واسم الدالة، واسم الملف، وما إلى ذلك.

اقرأ المزيد في مدونتي على [Medium](#).

المقالة:

<https://avichawla.substack.com/p/inspect-program-flow-with-icecream>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Debugging/Inspect-Flow.ipynb>

186) لا تقم بإنشاء أعمدة شرطية في Pandas مع Apply Don't Create Conditional Columns in Pandas with Apply

```

def assign_class(num):
    if num>0.5:
        return "Class A"
    else:
        return "Class B"

df.col1.apply(assign_class)
## 987 ms ± 47.1 ms per loop

```



```

import numpy as np

np.where(df["col1"]>0.5,
         "Class A",
         "Class B")
## 194 ms ± 23.7 ms per loop

```

If condition is True

If condition is False

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

إنشاء إعمدة شرطية في Pandas، نميل إلى استخدام طريقة `apply()` طوال الوقت تقريباً.

ومع ذلك، فإن `apply()` في Pandas ليست سوى حلقة متآلة. نتيجة لذلك، فإنه يخطئ الهدف الكامل من التوجيه .vectorization

بدلاً من ذلك، يجب عليك استخدام طريقة `np.where()` لإنشاء أعمدة شرطية. إنها تقوم بنفس الوظيفة ولكنها سريعة للغاية.

يتم تمرير الحالة كوسيدة أولى. ويتبع ذلك بالنتيجة إذا تم تقييم الشرط إلى True (الوسيدة الثانية) وFalse (الوسيدة الثالثة).

اقرأ المزيد هنا: [NumPy مستندات](#)

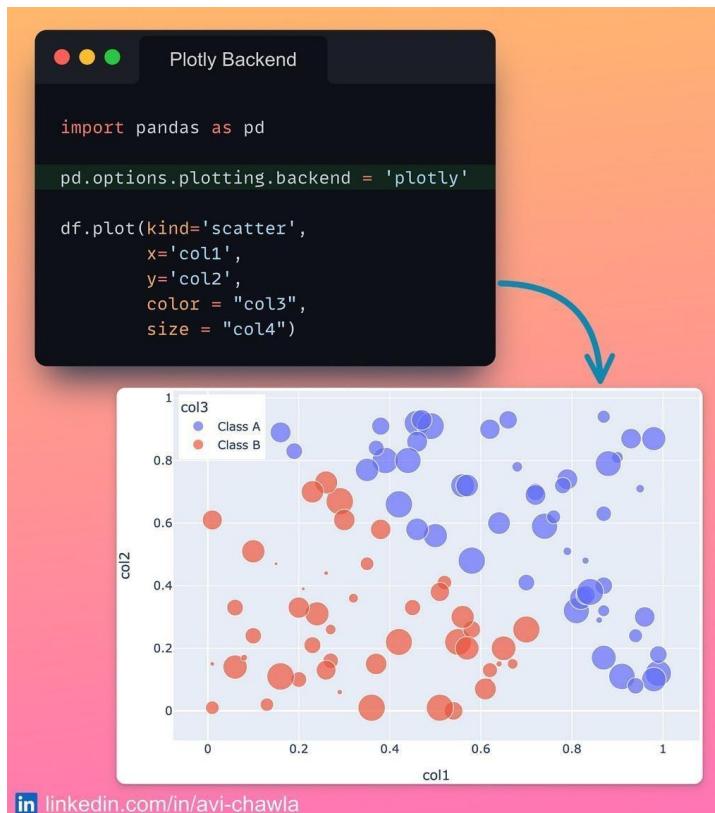
المقالة:

<https://avichawla.substack.com/p/dont-create-conditional-columns-in>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Conditional-Columns.ipynb>

Pretty Plotting With Pandas



linkedin.com/in/avi-chawla

Matplotlib هي واجهة برمجة التطبيقات الافتراضية للتخطيط في Pandas. هذا يعني أنه يمكنك إنشاء مخطط Matplotlib في Pandas، دون حتى استيرادها.

على الرغم من ذلك، كانت هذه المخططات دائمًا أساسية وليس جذابة بصريًا. غالباً ما يعتبر الرسم البياني، بمخططاته الجميلة والتفاعلية، بدلاً مناسباً. لكن التعرف على مكتبة جديدة بالكامل وبنيتها اللغوية يمكن أن يستغرق وقتاً طويلاً.

لحسن الحظ، تسمح لك Pandas بتغيير الخلفية الافتراضية للتخطيط. وبالتالي، يمكنك الاستفادة من مكتبات التصور التابعة لجهات خارجية للتخطيط باستخدام Pandas. هذا يجعل من الصعب إنشاء مخططات أجمل مع الحفاظ على بناء الجملة القديم تقريباً.

المقالة:

<https://avichawla.substack.com/p/pretty-plotting-with-pandas>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Plotly-Backend-in-Pandas.ipynb>

Build Sklearn نماذج أساسية بسهولة مع (188 Baseline Models Effortlessly With Sklearn



```
from sklearn.dummy import DummyClassifier

dummy_clf = DummyClassifier(
    strategy="most_frequent"
).fit(X, y)

>>> dummy_clf.predict(X)
array([0, 0, 0, 0, 0])

>>> dummy_clf.score(X, y)
0.6
```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

قبل تطوير نموذج تعلم آلي معقد، من المنطقي دائمًا إنشاء خط أساسي أو لاً.

يعمل خط الأساس كمعيار للنموذج الهندسي. علاوة على ذلك، فإنه يضمن أن النموذج أفضل من عمل تنبؤات عشوائية (أو ثابتة). لكن بناء خطوط أساس ب استراتيجيات مختلفة (عشوائي، ثابت، أكثر شيوعاً، وما إلى ذلك) يمكن أن يكون مملاً.

بدلاً من ذلك، فإن `DummyRegressor()` و `DummyClassifier()` في Sklearn يجعلها سهلة و مباشرة تماماً. يمكنك تحديد السلوك المحدد للخط الأساسي باستخدام المعلمة `strategy`.

اقرأ المزيد هنا: [التوثيق](#).

المقالة:

<https://avichawla.substack.com/p/build-baseline-models-effortlessly>

: الكود

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Sklearn/Baseline-Model.ipynb>

Fine- Python 3.11 تبع الأخطاء بدقة باستخدام grained Error Tracking With Python 3.11

```
$ python expt.py

Traceback (most recent call last):

  File "expt.py", line 11, in <module>
    print(function(a=2, b=0))
                  ^^^^^^^^^^^^^^

  File "expt.py", line 6, in function
    return (b / a) + (a / b)
                     ~~^~~

ZeroDivisionError: division by zero
```

linkedin.com/in/avi-chawla

تم إصدار Python 3.11 اليوم، وتم تقديم العديد من الميزات المثيرة. على سبيل المثال، تم تنفيذ العديد من تحسينات السرعة. وفقاً للإصدار الرسمي، تعد Python 3.11 في المتوسط، أسرع بنسبة 25٪ من Python 3.10. اعتناداً على عملك، يمكن أن تصل إلى 10-60٪ أسرع. واحدة من أروع الميزات هي تبع الأخطاء بدقة fine-grained error tracking. في Python 3.10 وما قبله، أظهر المترجم السطر المحدد الذي تسبب في الخطأ. هذا، في كثير من الأحيان، تسبب في الغموض أثناء التصحيح .debugging. في Python 3.11، سيشير المترجم الفوري إلى الموقع الدقيق الذي تسبب في حدوث الخطأ. سيساعد هذا المبرمجين بشكل كبير أثناء تصحيح الأخطاء.

اقرأ المزيد هنا: [الإصدار الرسمي](#).

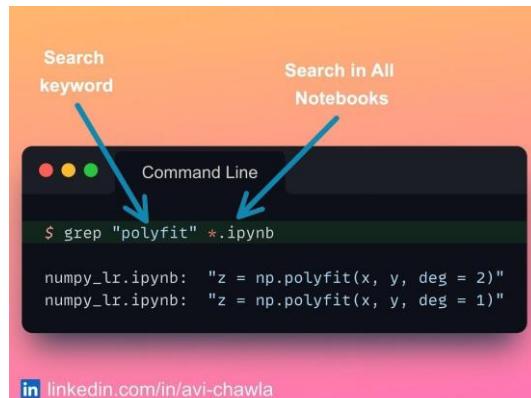
المقالة:

<https://avichawla.substack.com/p/fine-grained-error-tracking-with>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Python/Python3.11-Update.ipynb>

190) اعثر على الكود الخاص بك مختبئاً في بعض النوتبووك من Find Your Code Hiding In Some Jupyter Notebook With Ease



غالباً ما يشير المبرمجون الذين يستخدمون Jupyter إلى النوتبوك القديمة الخاصة بهم للعثور على جزء من التعليمات البرمجية.

ومع ذلك، يصبح الأمر مملاً عندما يكون لديهم العديد من الأشياء التي يبحثون عنها ولا يمكنهم تذكر النوتبوك المحدد الذي يهتمون. اسم الملف **Untitled1.ipynb** و ... و

Untitled82.ipynb ، لا يجعل الأمر أسهل.

الأمر "grep" هو حل أفضل بكثير لهذا الأمر. أعلم جداً أنه يمكنك استخدام "grep" في سطر الأوامر للبحث في النوتبوك، كما تفعل في الآخرين (txt، على سبيل المثال). هذا يوفر الكثير من العمل اليدوي والوقت.

ملاحظة. كيف تجد بعض التعليمات البرمجية المكتوبة مسبقاً في النوتبوك الخاصة بك (إن لم يكن يدوياً)؟

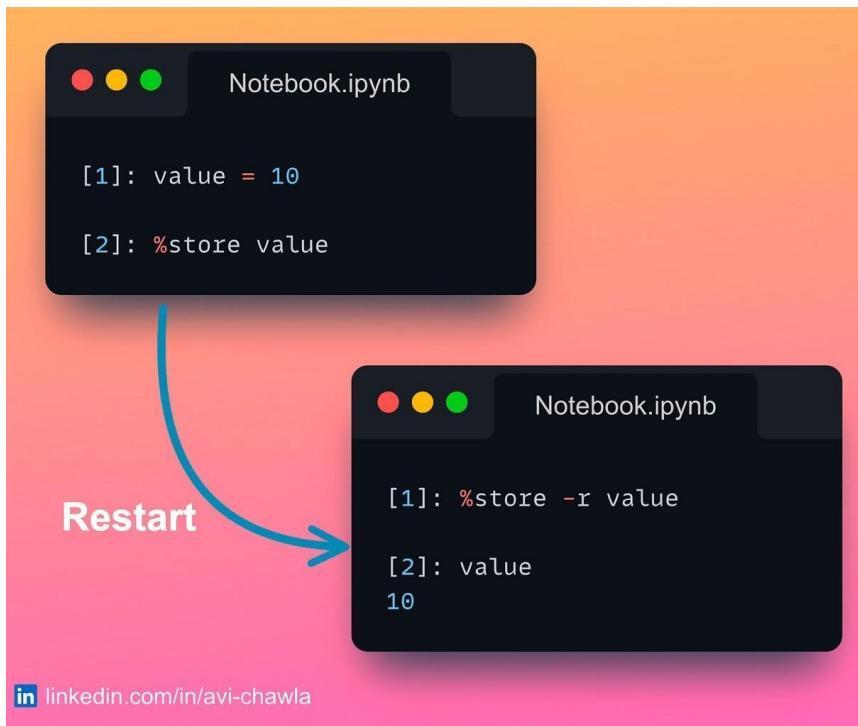
المقالة:

<https://avichawla.substack.com/p/find-your-code-hiding-in-some-jupyter>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Jupyter%20Tips/Find-Hidden-Code.ipynb>

191) أعد تشغيل الكيرنل دون فقد المتغيرات Restart the Kernel Without Losing Variables



[in](https://www.linkedin.com/in/avi-chawla) linkedin.com/in/avi-chawla

أثناء العمل في Jupyter Notebook، قد ترغب في إعادة تشغيل الكيرنل (النواة kernel) لعدة أسباب. ولكن قبل إعادة التشغيل، غالبًا ما يميل المرء إلى تفريغ كائنات البيانات على القرص لتجنب إعادة حسابها في التشغيل التالي.

يعمل الأمر السحري "store" كحل مثالي لذلك. هنا، يمكنك الحصول على قيمة محسوبة مسبقًا حتى بعد إعادة تشغيل النواة الخاصة بك. علاوة على ذلك، لن تحتاج أبدًا إلى الخوض في متاعب تفريغ الكائن على القرص.

How to Read عدّة ملفات CSV بكفاءة (192) كيف تقرأ عدّة ملفات CSV بكفاءة

Multiple CSV Files Efficiently

The image shows two Jupyter Notebook cells side-by-side. The top cell, titled 'Pandas_read.py', contains Python code to read six CSV files ('jan.csv' through 'jun.csv') into a list of DataFrames and then concatenate them. The bottom cell, titled 'Datatable_read.py', contains similar code using the 'datatable' library. A blue arrow points from the 'Run-time: 64 seconds' annotation in the Pandas cell to the 'Run-time: 36 seconds' annotation in the Datatable cell.

```

Pandas_read.py:
import pandas as pd

files = ["jan.csv", "feb.csv",
         "mar.csv", "apr.csv",
         "may.csv", "jun.csv"]
## 300 MBs each

df_list = []
for i in files:
    df_list.append(pd.read_csv(i))
data = pd.concat(df_list)

Datatable_read.py:
import datatable as dt

files = ["jan.csv", "feb.csv",
         "mar.csv", "apr.csv",
         "may.csv", "jun.csv"]
## 300 MBs each

df = dt.iread(files) ## read files
df = dt.rbind(df) ## concatenate row-wise
df = df.to_pandas() ## convert to Pandas

```

في كثير من الحالات، غالباً ما يتم تقسيم البيانات إلى ملفات CSV متعددة ونقلها إلى فريق علم البيانات / التعلم الآلي لاستخدامها.

نظرًا لأن Pandas لا يدعم الموازاة parallelization، يتبع على المرء تكرار قائمة الملفات وقراءتها واحدة تلو الأخرى لمزيد من المعالجة.

يمكن لـ "Datatable" توفير حل سريع لهذا الغرض. بدلاً من قراءتها بشكل متكرر مع Pandas، يمكنك استخدام Datatable لقراءة مجموعة من الملفات. كونه متوازيًا، فإنه يوفر تعزيزًا ملحوظًا في الأداء مقارنة بـ Pandas.

لا يقتصر اكتساب الأداء على الإدخال / الإخراج فحسب، بل يتم ملاحظته في العديد من العمليات المجدولة الأخرى أيضًا.

اقرأ المزيد هنا: [DataTable Docs](#)

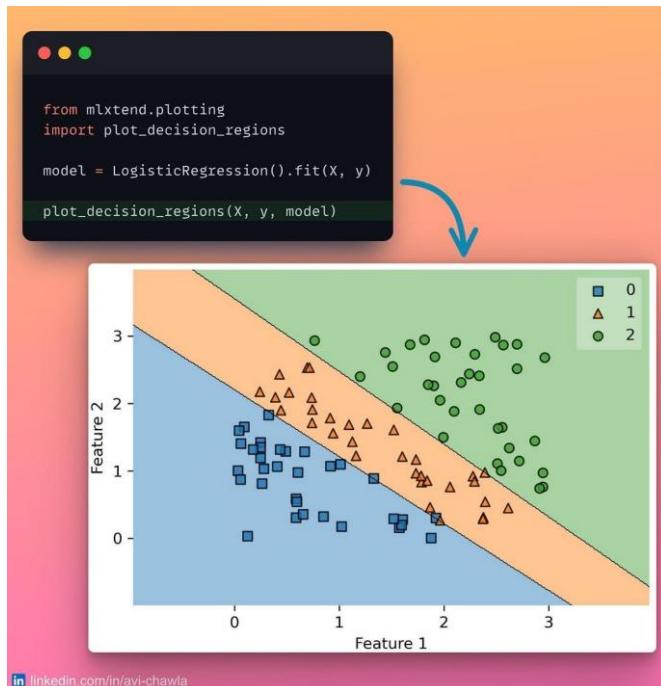
المقالة:

<https://avichawla.substack.com/p/how-to-read-multiple-csv-files-efficient>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Read-Multiple-CSVs.ipynb>

Elegantly Plot the Decision Boundary of a Classifier (193)



يمكن أن يكشف رسم حدود القرار decision boundary للمصنف classifier عن العديد من الرؤى الحاسمة حول أداءه.

هنا، غالباً ما تُعتبر المخططات المظللة بالمنطقة region-shaded plots اختياراً مناسباً لأغراض التصور. ولكن، يمكن أن يكون إنشاء واحدة بشكل صريح مستهلكاً للوقت ومعقداً للغاية.

يكشف Mlxtend ذلك إلى سطوة واحد بسيطي في بايثون. هنا، يمكنك رسم حدود قرار المصنف بسهولة، بمجرد تزويده بالموديل والبيانات.

المقالة:

<https://avichawla.substack.com/p/elegantly-plot-the-decision-boundary>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Plotting/Classifier%20Decision%20Boundary.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Classifier%20Decision%20Boundary.ipynb)

An Sklearn أنيقة لاستيراد المقاييس من 194 Elegant Way to Import Metrics From Sklearn

The image shows two Jupyter Notebook cells side-by-side. The top cell, titled 'Import all metrics individually', contains the following code:

```
from sklearn.metrics
import accuracy_score, f1_score,
precision_score, recall_score,
roc_auc_score, ...
```

The bottom cell, titled 'Get a scorer from string', contains the following code:

```
from sklearn.metrics import get_scorer
accuracy = get_scorer("accuracy")
>>> accuracy._score_func(y_true, y_pred)
0.5
```

`precision = get_scorer("precision")`

`>>> precision._score_func(y_true, y_pred)`

0.8

A blue double-headed arrow connects the two cells, indicating they are equivalent. A blue arrow points from the top cell to the bottom cell, labeled 'Get a scorer from string'. A blue arrow points from the bottom cell to the top cell, labeled 'Import all metrics individually'.

[in linkedin.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

أثناء استخدام scikit-Learn، غالباً ما يستورد الماء مقاييس metrics متعددة لتقييم نموذج. على الرغم من عدم وجود خطأ في هذه الممارسة، إلا أنها تجعل الكود غير أنيق ومشوش - مع زيادة تحميل الأسطر القليلة الأولى من الملف بالواردات.

بدلاً من استيراد المقاييس بشكل فردي، يمكنك استخدام طريقة `get_scorer()`. هنا، يمكنك تمرير اسم المقاييس كسلسلة string، وإرجاع كائن مسجل scorer object لك.

اقرأ المزيد هنا: [صفحة Scikit-Learn](#)

المقالة:

<https://avichawla.substack.com/p/an-elegant-way-to-import-metrics>

الكلود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Sklearn/Metric%20Import.ipynb>

Pandas Configure لـ إخراج إطار بيانات Sklearn (195) تكوين Sklearn To Output Pandas DataFrame

```

Scikit-learn 1.1
from sklearn.preprocessing import StandardScaler
X_train = ... ## Pandas DataFrame
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_train)
type(X_scaled) ## numpy.ndarray
Output is NumPy Array

Scikit-learn 1.2.dev
scaler = StandardScaler()
scaler.set_output(transform="pandas")
X_scaled = scaler.fit_transform(X_train)
type(X_scaled) ## pandas.core.frame.DataFrame
Output is Pandas DataFrame

```

[in linkedin.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

في الآونة الأخيرة، أعلنت Scikit-Learn عن إصدار أحد أكثر التحسينات المنتظرة. في الخلاصة، يمكن الآن ضمان sklearn لإخراج Pandas DataFrames بدلاً من مصفوفات NumPy.

حتى الآن، كانت محولات Sklearn مؤمنة لقبول Pandas DataFrame كمدخلات. لكنهم عادوا دائمًا إلى مصفوفة NumPy كإخراج. نتيجة لذلك، كان لابد من عرض الإخراج يدوياً مرة أخرى على Pandas DataFrame.

الآن، ستسمح واجهة برمجة التطبيقات **set_output** للمحولات بإخراج Pandas DataFrame بدلاً من ذلك.

سيؤدي هذا إلى جعل خطوط الأنابيب الجارية على ذلك، سيفر أكثر سلاسة. علاوة على ذلك، سيوفر طرقًا أفضل لتعيين أسماء الميزات.

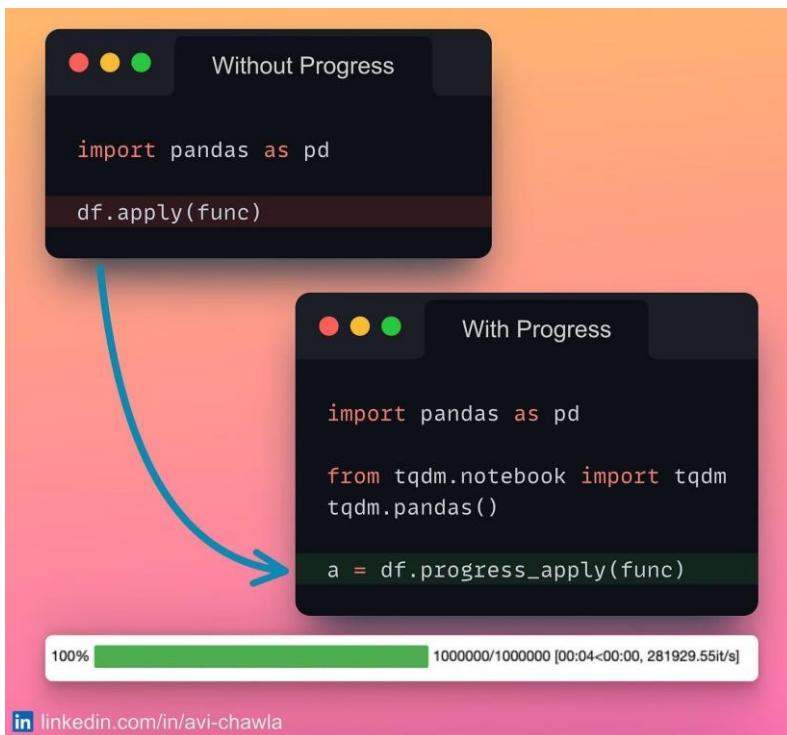
المقالة:

<https://avichawla.substack.com/p/configure-sklearn-to-output-pandas>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Sklearn%20-%20Pandas%20Output.ipynb>

Display Pandas Apply() مع شريط التقدم (196) | عرض شريط التقدم مع Apply() في Pandas



أثناء تطبيق طريقة على DataFrame باستخدام apply، لا يمكننا رؤية التقدم والوقت المتبقى المقدر.

لحل هذه المشكلة، يمكنك بدلاً من ذلك استخدام progress_apply() من tqdm لعرض شريط تقدم أثناء تطبيق الطريقة.

اقرأ المزيد هنا: [GitHub](#).

المقالة:

<https://avichawla.substack.com/p/display-progress-bar-with-apply-in>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Apply%20-%20Progress%20Bar.ipynb>

197) تعديل دالة أثناء وقت التنفيذ During Run-time

The screenshot shows two code cells in a Jupyter Notebook. The first cell contains:

```
from time import sleep
from reloading import reloading

@reloading
def func(num):
    if number % 2:
        print(f"{number} is Odd")
    else:
        pass

for i in range(100):
    func(i)
```

A red arrow labeled "Modified During Run-time" points from the first cell to the second cell. The second cell contains the same code with a modified "else" block:

```
from time import sleep
from reloading import reloading

@reloading
def func(num):
    if number % 2:
        print(f"{number} is Odd")
    else:
        print(f"{number} is Even")

for i in range(100):
    func(i)
```

Both cells output the results of the function calls:

1 is Odd
3 is Odd
5 is Odd
7 is Odd
9 is Odd
11 is Odd
13 is Odd
14 is Even
15 is Odd
16 is Even
17 is Odd
18 is Even
19 is Odd

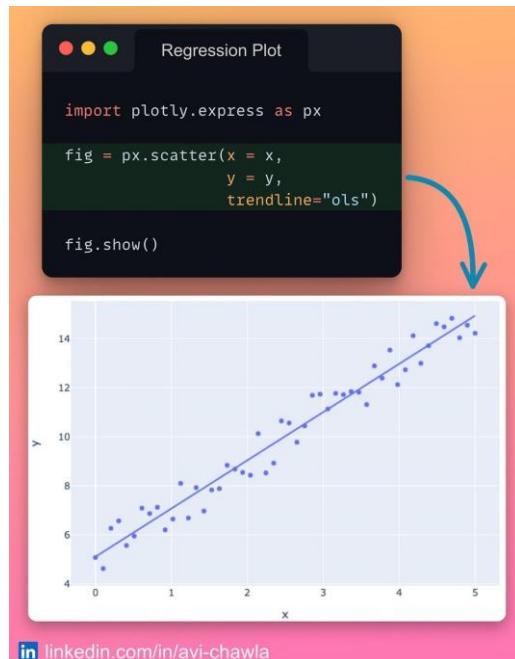
هل سبق لك أن كنت في موقف ترغب فيه في إضافة المزيد من التفاصيل إلى كود قيد التنفيذ بالفعل؟

يُلاحظ هذا عادةً في التعلم الآلي حيث غالباً ما ينسى المرء طباعة جميع تفاصيل / مقاييس التدريب الأساسية. لا يعد تنفيذ الكود بالكامل مرة أخرى، خاصةً عندما يكون قيد التنفيذ لبعض الوقت، أسلوباً مثالياً هنا.

إذا كنت ترغب في تعديل دالة أثناء التنفيذ، فقم بتزويتها بمصمم إعادة التحميل (reloading decorator). نتيجة لذلك، سيعيد بايثون تحميل الدالة من المصدر قبل كل تنفيذ.

رابط لإعادة التحميل: [GitHub](#)

Regression Plotly أصبحت سهلة مع 198 Plot Made Easy with Plotly



[in linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

أثناء إنشاء المخططات المبعثرة scatter plots، غالباً ما يهتم المرء بعرض الانحدار الخطي linear regression على نقاط البيانات.

هنا، يمكن أن يكون تدريب نموذج وتصميمه يدوياً في المخطط مهمّة شاقة يجب القيام بها.

بدلاً من ذلك، باستخدام Plotly، يمكنك إضافة خط انحدار regression line إلى المخطط، دون تدريب نموذج صريح.

اقرأ المزيد [هنا](#).

المقالة:

<https://avichawla.substack.com/p/regression-plot-made-easy-with-plotly>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Plotting/Plotly%20Regression%20Plot.ipynb>

NumPy (199) الانحدار الخطي متعدد الحدود باستخدام Polynomial Linear Regression with NumPy

The image shows two Jupyter notebook cells side-by-side. The top cell is titled 'Sklearn' and the bottom one is titled 'NumPy'. Both cells contain Python code for polynomial regression.

```

Sklearn:
## 1 Degree Polynomial
model = LinearRegression().fit(x, y)

## 2 Degree Polynomial
x = np.hstack((x, x**2))
model = LinearRegression().fit(x, y)

>>> x = 2
>>> inp = np.array([[x, x**2]])
>>> model.predict(inp)
-10.4

```

```

NumPy:
coeff = np.polyfit(x, y, deg = 2)
model = np.poly1d(coeff)

>>> inp = 2
>>> model(inp)
-10.4

```

Annotations with arrows point from specific lines of code to their descriptions:

- An arrow points from the line `x = np.hstack((x, x**2))` to the text "Create Polynomial Features".
- An arrow points from the line `coeff = np.polyfit(x, y, deg = 2)` to the text "Specify Degree".

[in linkedin.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

يعد الانحدار الخطي متعدد الحدود Polynomial linear regression مملاً حيث يتعين على المرء ترميز ميزاته بشكل صريح. يمكن أن يصبح هذا الأمر صعباً عندما يتغير على المرء أن يبني بشكل متكرر نماذج متعددة الحدود ذات درجة أعلى.

طريقة `polyfit()` من NumPy هي بديل ممتاز لذلك. هنا، يمكنك تحديد درجة كثير الحدود كمعامل. نتيجة لذلك، يقوم تلقائياً بإنشاء ميزات كثيرة الحدود المقابلة.

الجانب السلبي هو أنه لا يمكنك إضافة ميزات مخصصة مثل المثلثية /trigonometric/ اللوغاريتمية logarithmic. بمعنى آخر، أنت مقيد بسمات كثيرة الحدود فقط. ولكن إذا لم يكن هذا هو مطلبك، فيمكن أن تكون طريقة `polyfit()` من NumPy طريقة أفضل.

اقرأ المزيد من [هذا](#).

المقالة:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Sklearn/LinearReg%20NumPy%20vs%20Sklearn.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Sklearn/LinearReg%20NumPy%20vs%20Sklearn.ipynb)

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Sklearn/LinearReg%20NumPy%20vs%20Sklearn.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Sklearn/LinearReg%20NumPy%20vs%20Sklearn.ipynb)

(200) قم بتعديل نوع بيانات الأعمدة المتعددة مرة واحدة Alter the Datatype of Multiple Columns at Once

```

>>> df
   col1  col2  col3 col4
0     1      7     4    A
1     3      9     6    B
2     6      2     5    A

df["col1"] = df.col1.astype(np.int32)
df["col2"] = df.col2.astype(np.int16)
df["col3"] = df.col3.astype(np.float16)

```

Multiple Calls


```

df = df.astype({
    "col1":np.int32,
    "col2":np.int16,
    "col3":np.float16
})

```

Single Call

[in linkedin.com/in/avi-chawla](https://linkedin.com/in/avi-chawla)

تتمثل إحدى الطرق الشائعة لتعديل نوع بيانات الأعمدة المتعددة في استدعاء طريقة `astype()` في كل عمود.

على الرغم من أن الطريقة تعمل كما هو متوقع، إلا أنها تتطلب استدعاءات دوال متعددة والمزيد من التعليمات البرمجية. قد يكون هذا صعباً بشكل خاص عندما ت يريد تعديل نوع البيانات للعديد من الأعمدة.

طريقة أفضل، يمكنك تكثيف كل التحويلات في استدعاء دالة واحدة. يتم تحقيق ذلك عن طريق تمرير قاموس لتعيين نوع العمود إلى نوع البيانات، كما هو موضح أدناه.

المقالة:

<https://avichawla.substack.com/p/alter-the-datatype-of-multiple-columns>

ال코드:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Pandas/Datatype-Alter.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Datatype-Alter.ipynb)

Pandas) نوع البيانات لمعالجة أعمدة القيم المفقودة في Datatype For Handling Missing Valued Columns in Pandas

The screenshot shows two code snippets in a Jupyter Notebook environment.

Top Snippet (NaN column):

```
>>> len(df.col1)
## Total entries: 1,000,000

>>> len(df[df.col1.isna()])
## NaN entries: 700,000 (70%)
```

Bottom Snippet (Sparse Datatype):

```
df.col1.memory_usage()
## Memory usage before conversion: 7.6 MB

df["col1"] = df.col1.astype("Sparse[float32]")

df.col1.memory_usage()
## Memory usage after conversion: 2.0 MB
```

LinkedIn profile link: [in/avi-chawla](https://linkedin.com/in/avi-chawla)

إذا كانت بياناتك تحتوي على أعمدة ذات قيمة NaN ، فإن Pandas توفر نوع بيانات datatype محدداً لتمثيلها – يسمى نوع البيانات المتفرقة .Sparse datatype

يكون هذا مفيداً بشكل خاص عند العمل مع مشاريع كبيرة تعتمد على البيانات مع العديد من القيم المفقودة .missing values

يقارن مقتطف الكود استخدام الذاكرة لأنواع البيانات float والمترافق في Pandas .

المقالة:

<https://avichawla.substack.com/p/datatype-for-handling-missing-valued>

الكلود:

<https://avichawla.substack.com/p/datatype-for-handling-missing-valued>

قم بموازنة Pandas مع Pandarallel (202) with Pandarallel

The screenshot shows two code cells in a Jupyter Notebook. The top cell contains:

```
from pandarallel import pandarallel
pandarallel.initialize()

def add_row(row):
    return sum(row)

df = ... ## 10M Rows, 2 Columns
```

The bottom cell contains a performance comparison:

```
df.apply(add_row, axis = 1)
## 53 secs
```

```
df.parallel_apply(add_row, axis = 1)
## 11 secs
```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

عمليات Pandas لا تدعم الموازنة parallelization. نتيجة لذلك، فإنه يتلزم بحساب أحدى النواة، حتى عند توفر النوى الأخرى. وهذا يجعلها غير ملائمة وصعبة، لا سيما في مجموعات البيانات الكبيرة.

يسمح لك "Pandarallel" بموازنة عملياته مع نوى متعددة لوحدة المعالجة المركزية CPU – عن طريق تغيير سطر واحد فقط من التعليمات البرمجية. تتضمن الطرق المدعومة apply() و applymap() و groupby() و rolling() و gmap().

اقرأ المزيد: [GitHub](#)

المقالة:

<https://avichawla.substack.com/p/parallelize-pandas-with-pandarallel>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Pandarallel.ipynb>

Why you should not dump DataFrames to a CSV (203) لماذا لا يجب خزن إطارات البيانات في ملف CSV



يُستخدم تنسيق ملف CSV على نطاق واسع لحفظ إطارات بيانات Pandas. لكن هل تدرك قيودها؟ على سبيل المثال لا الحصر:

1. لا يخزن ملف CSV معلومات نوع البيانات. وبالتالي، إذا قمت بتعديل نوع بيانات العمود (الأعمدة)، وحفظته في ملف CSV، ثم التحميل مرة أخرى، فلن يقوم Pandas بإرجاع نفس أنواع البيانات.
2. لا يتم تحسين حفظ DataFrame إلى تنسيق CSV مثل التنسيقات الأخرى المدعومة بواسطة Pandas. وتشمل هذه Pickle، Parquet، إلخ.

بالطبع، إذا كنت بحاجة إلى عرض بياناتك خارج بايثون (Excel، على سبيل المثال)، فأنت ملزماً باستخدام ملف CSV. ولكن إذا لم يكن الأمر كذلك، ففضل تنسيقات الملفات الأخرى.

قراءة إضافية: [لماذا توقفت عن خزن إطارات البيانات في ملف CSV ولماذا يجب عليك ذلك أيضاً](#).

المقالة:

<https://avichawla.substack.com/p/why-you-should-not-dump-dataframes>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Dont-Dump-To-CSV.ipynb>

Save Memory with Moladat بايثون (204) حفظ الذاكرة مع مولدات بايثون Python Generators

The image shows two terminal windows side-by-side. The left window is titled 'List.py' and the right is titled 'Generator.py'. Both windows run the same code, which imports sys, defines a large list 'my_list' or generator 'my_gen', calculates its size using getsizeof(), and then calculates the sum of its elements. The output shows that the list uses significantly more memory (89095160 bytes vs 112 bytes) and takes longer to calculate the sum (49999995000000 vs 0).

```

List.py
1 from sys import getsizeof
2
3 my_list = [i for i in range(10**7)]
4 ## use [] to create a list
5
6 >>> getsizeof(my_list)
7 ## 89095160 bytes
8
9 >>> sum(my_list)
10 ## 49999995000000
11
12 >>> sum(my_list)
13 ## 49999995000000

Generator.py
1 from sys import getsizeof
2
3 my_gen = (i for i in range(10**7))
4 ## use () to create a generator
5
6 >>> getsizeof(my_gen)
7 ## 112 bytes
8
9 >>> sum(my_gen)
10 ## 49999995000000
11
12 >>> sum(my_gen)
13 ## 0

```

إذا كنت تستخدم متكررات ثابتة static iterables كبيرة في بايثون، فقد لا تكون القائمة list هي الخيار الأمثل، خاصة في التطبيقات المقيدة بالذاكرة.

قائمة تخزن المجموعة بأكملها في الذاكرة. ومع ذلك، يقوم المولد generator بحساب وتحميل عنصر واحد في وقت واحد فقط عندما يكون ذلك مطلوباً. هذا يوفر كلاً من الذاكرة ووقت إنشاء الكائن.

بالطبع، هناك بعض القيود على المولدات أيضاً. على سبيل المثال، لا يمكنك استخدام عمليات القائمة list الشائعة مثل () append() والتقطيع slicing وما إلى ذلك.

علاوة على ذلك، في كل مرة تريده فيها إعادة استخدام عنصر، يجب إعادة إنشائه (انظر Generator.py السطر 12).

المقالة:

<https://avichawla.substack.com/p/save-memory-with-python-generators>

ال코드:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Memory%20Optimization/Generators.ipynb>

205) لا تستخدم print() لتصحيح التعليمات البرمجية .Don't use print() to debug your code.

```

Print
def func(arr, n):
    print("arr =", arr, "n =", n)

func([1,2,3], 2)
## arr = [1, 2, 3] n = 2

Icecream
from icecream import ic

def func(arr, n):
    ic(arr, n)

func([1,2,3], 2)
## ic| arr: [1, 2, 3], n: 2

```

linkedin.com/in/avi-chawla

التصحيح Debugging باستخدام جمل الطباعة print statements هو نهج فوضوي وغير آنيق. من المربك تعين الإخراج إلى بيان التصحيح المقابل له. علاوة على ذلك، يتطلب تنسيقاً يدوياً إضافياً لفهم الإخراج.

مكتبة "icecream" في بايثون هي بديل ممتاز لذلك. فهو يجعل تصحيح الأخطاء سهلاً وقابلًا للقراءة، مع الحد الأدنى من التعليمات البرمجية. تتضمن الميزات تعبيرات الطباعة، وأسماء المتغيرات، وأسماء الدوال، وأرقام الأسطر، وأسماء الأسماء، وغيرها الكثير.

ملاحظة. مقتطف الكود يعطي فقط شرحاً موجزاً. ومع ذلك، فإن الدوال الفعلية أكثر قوة وأناقة مقارنة بالتصحيح باستخدام print().

المزيد عن [icecream هنا](#).

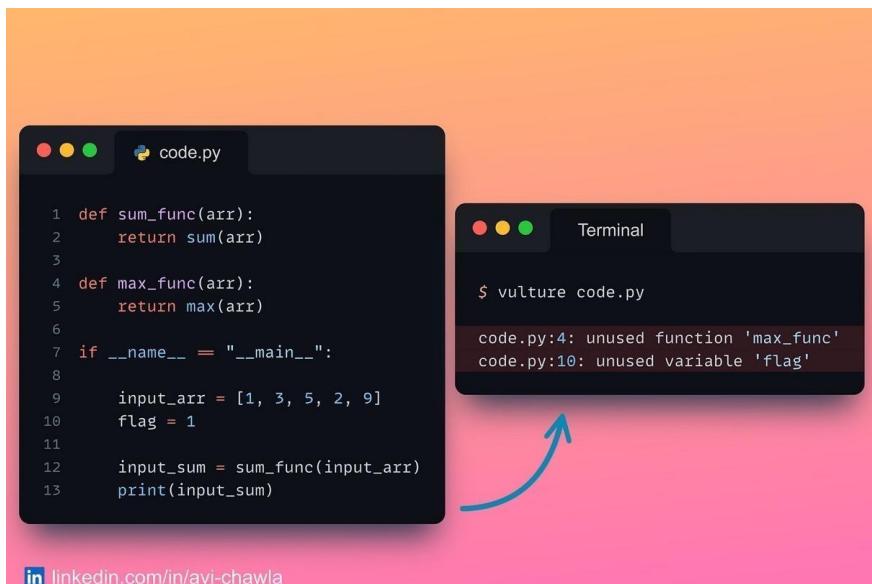
المقالة:

<https://avichawla.substack.com/p/dont-use-print-to-debug-your-code>

ال코드:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Debugging/Icecream.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Debugging/Icecream.ipynb)

Find (206) ابحث عن كود بايثون غير المستخدم بسهولة Unused Python Code With Ease



linkedin.com/in/avi-chawla

مع زيادة حجم قاعدة الشفرة الخاصة بك، يزداد عدد مثيلات التعليمات البرمجية غير المستخدمة. هذا يمكن قراءتها وإيجازها.

باستخدام وحدة "vulture" في بايثون، يمكنك تحديد كود ميت dead code (غير مستخدم) في خط الأنابيب الخاص بك، كما هو موضح في مقتطف الكود.

المقالة:

<https://avichawla.substack.com/p/find-unused-python-code-with-ease>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/Dead-Code.ipynb>

Define the Correct DataType for Categorical Columns

The image shows two adjacent Jupyter Notebook cells. The left cell, titled 'Categorical Col.', contains the following Python code:

```
import pandas as pd
len(df.Gender)
## 1500
df.Gender.unique()
## ["Male", "Female"]
```

The right cell, titled 'Reduce Memory Usage', contains the following code to convert the 'Gender' column to a categorical type:

```
import pandas as pd
df.Gender.memory_usage(), df.Gender.dtype
## 90.5 KB, object
df["Gender"] = df.Gender.astype("category")
df.Gender.memory_usage(), df.Gender.dtype
## 1.8 KB, CategoricalDtype
```

A small LinkedIn link ([in/avi-chawla](https://linkedin.com/in/avi-chawla)) is visible at the bottom left of the screenshot.

إذا كانت بياناتك تحتوي على أعمدة فئوية categorical columns، فلا يجب أن تمثلها كنوع بيانات .int / string

بدلاً من ذلك، يوفر Pandas نوع بيانات محسناً على وجه التحديد للأعمدة الفئوية. يكون هذا مفيداً بشكل خاص عند العمل مع مشاريع كبيرة تعتمد على البيانات.

يقارن مقتطف الكود استخدام الذاكرة للسلسلة النصية string وأنواع البيانات الفئوية في Pandas.

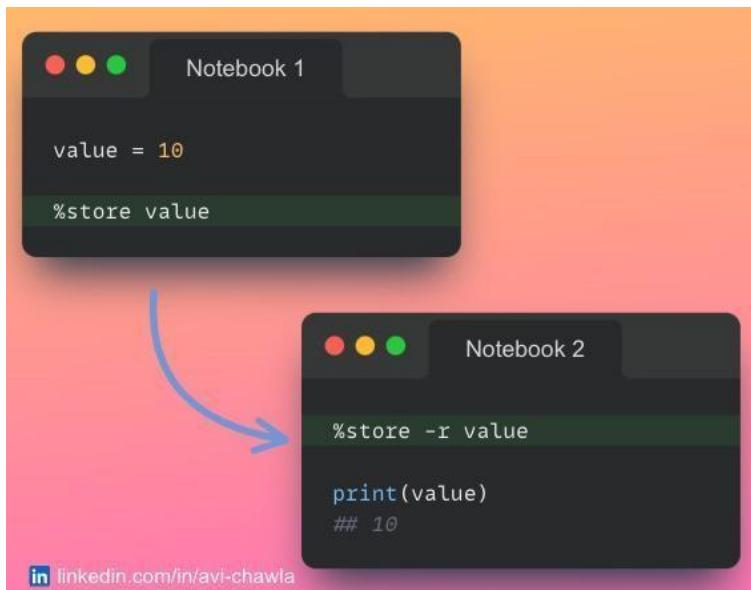
المقالة:

<https://avichawla.substack.com/p/define-the-correct-datatype-for-categorical>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Categorical-Datatype.ipynb>

208) نقل المتغيرات بين Jupyter Notebooks Variables Between Jupyter Notebooks



أثناء العمل مع العديد من Jupyter Notebooks، قد تحتاج إلى مشاركة الكائنات بينها.

باستخدام الأمر السحري "store"، يمكنك نقل المتغيرات عبر Notebooks دون تخزينها على القرص.

ملاحظة. يمكنك أيضًا إعادة تشغيل النواة (الكيرنل) واسترداد متغير قديم باستخدام ".store".

المقالة:

<https://avichawla.substack.com/p/transfer-variables-between-jupyter>

ال코드:

<https://avichawla.substack.com/p/transfer-variables-between-jupyter>

Why Pandas CSV مع 209 You Should Not Read CSVs with Pandas



تلزوم Pandas بحساب أحادي النواة، مما يجعل عملياتها غير فعالة للغاية للغاية، لا سيما في مجموعات البيانات الكبيرة.

تعد مكتبة "datatable" في بايثون بديلاً ممتازاً بواجهة برمجة تطبيقات API تشبه Pandas. دعم معالجة البيانات متعدد الخيوط يجعله أسرع من Pandas.

يوضح مقتطف الكود مقارنة وقت التشغيل لـ"إنشاء DataFrame" من ملف CSV باستخدام Datatable و Pandas.

المقالة:

<https://avichawla.substack.com/p/why-you-should-not-read-csvs-with>

الكود:

[https://github.com/ChawlaAvi/Daily-Dose-of-Data-
Science/blob/main/Pandas/Fast-CSV-Read.ipynb](https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Pandas/Fast-CSV-Read.ipynb)

Modify Python Code During Run-Time

```

from time import sleep
from reloading import reloading

for number in reloading(range(100)):

    if number % 2:
        print(f"{number} is Odd")
    else:
        pass

```

Modified During Run-time

```

from time import sleep
from reloading import reloading

for number in reloading(range(100)):

    if number % 2:
        print(f"{number} is Odd")
    else:
        print(f"{number} is Even")

```

[linkedin.com/in/avi-chawla](https://www.linkedin.com/in/avi-chawla)

هل سبق لك أن كنت في موقف أردت فيه إضافة المزيد من التفاصيل إلى كود قيد التنفيذ بالفعل (طباعة المزيد من التفاصيل في حلقة for، على سبيل المثال)؟

لا يعد تنفيذ الكود بالكامل مرة أخرى، خاصةً عندما يكون قيد التنفيذ لبعض الوقت، هو الأسلوب المثالي هنا.

باستخدام مكتبة "reloading" في بايثون، يمكنك إضافة المزيد من التفاصيل إلى التعليمات البرمجية قيد التنفيذ دون فقد أي تقدم موجود.

المقالة:

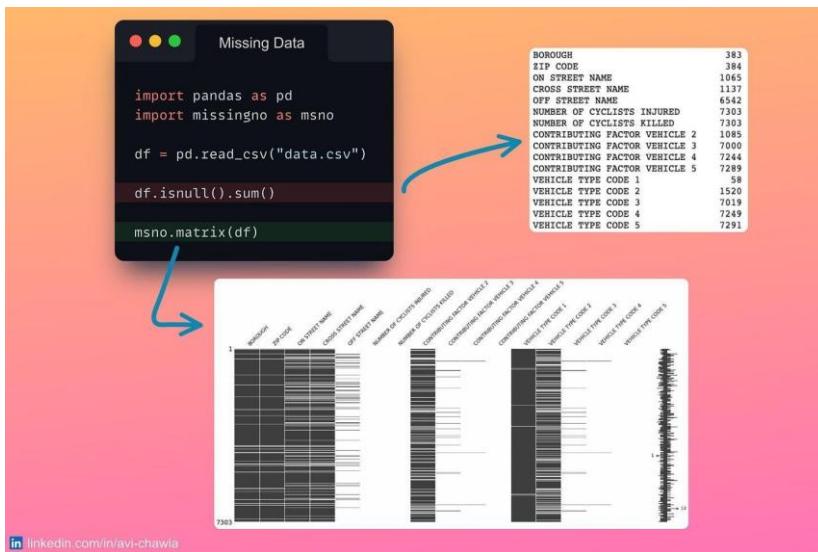
<https://avichawla.substack.com/p/modify-python-code-during-run-time>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Cool%20Tools/Modify%20Code%20During%20Run-time.ipynb>

210) التعامل مع البيانات المفقودة مع Missingno

Handle Missing Data With Missingno



إذا كنت ترغب في تحليل القيم المفقودة missing values في مجموعة البيانات الخاصة بك، فقد لا يكون Pandas خياراً مناسباً.

تحفي أساليب Pandas العديد من التفاصيل المهمة حول القيم المفقودة. يتضمن ذلك موقعها ودورياتها correlation والارتباط periodicity location وما إلى ذلك.

تعد مكتبة "missingno" في بايثون مصدرًا ممتازًا لاستكشاف البيانات المفقودة. يقوم بإنشاء تصورات إعلامية لتحسين تحليل البيانات.

يوضح مقتطف الكود تحليل البيانات المفقودة باستخدام Pandas و Missingno .

المقالة:

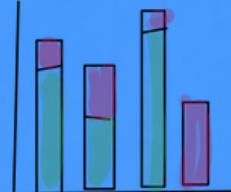
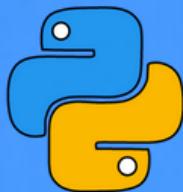
<https://avichawla.substack.com/p/handle-missing-data-with-missingno>

الكود:

<https://github.com/ChawlaAvi/Daily-Dose-of-Data-Science/blob/main/Missing%20Data/Missingno.ipynb>

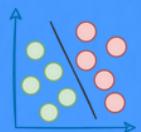
DATA SCIENCE

200+ Python & Data Science Tips



Translated into Arabic by

Dr. Alaa Taima



Daily Dose of
Data Science



avichawla.substack.com