DATA SCIENCE NASHVILLE

New Tricks for Old Data Scientists: Scikit-learn

Jason King and Charlie Apigian

Charlie's Demo

Jason's Demo

Food and Drinks Sponsored by the Belmont Data Collaborative

BELMONT DATA COLLABORATIVE

Data Science *Nashville* - General Information

Meetings

- Every 3rd Monday (nominally)
  - Upcoming talks on 3/20 (Mon), **4/19 (Wed)**, 5/15 (Mon), 6/19 (Mon), 7/17 (Mon), …
- Format include light food and drinks, conversation, and presentation.
- Who can attend? Anyone interested in Data Science.
- Who can do a presentation? Anyone that has a good topic - technology and infrastructure, how data science affects our community, tips and tricks, etc.

Announcements

- New to the group? Introduce yourself.
- Events in the area
- Job opportunities

Outline for *Presentation*

- Speaker introductions
- Motivations for this talk
- An overview and brief history of Scikit-learn
- Our respective paths to Python and Scikit-learn
- Demos and code walkthroughs
  - Histogram-based Gradient Boosting Method (HGBM)
  - Quality of life improvements
  - Online Non-negative Matrix Factorization (NMF)
  - Online One Class Support Vector Machines (SVM)
- Takeaways
- Questions

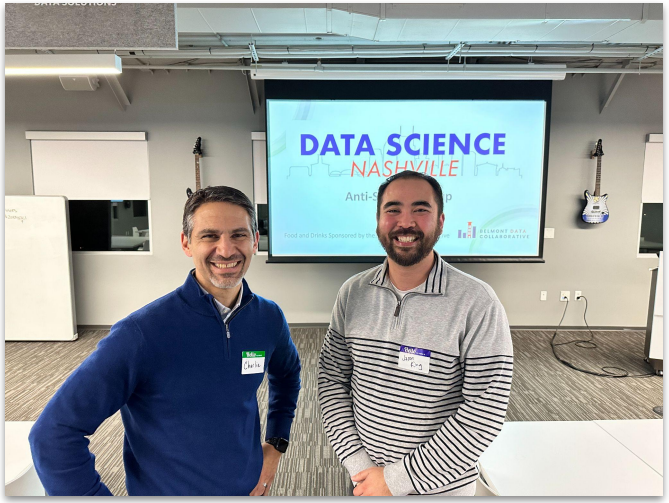
Your *Presenters* - Jason King and Charlie Apigian

Jason King - XSOLIS

- Senior Director of Data Science - Xsolis
  - Healthcare Artificial Intelligence
  - Administrative workflows (not clinical-facing!)
- 2022 NTC Data Scientist of the Year

Charlie Apigian - Belmont University

- Executive Director - Belmont Data Collaborative
  - Data for All and Data for Good!
- Co-Host - Data 4 All Podcast - data4all.io
- 2020 NTC Data Scientist of the Year, 2018 and 2023 NTC Community Leader of the Year



Motivation

- Why Scikit-learn?
  - Accessibility
    - Small footprint (around 10MB)
    - Excellent documentation
    - Popular (lots of examples)
  - Robustness
    - Stable codebase and API
    - Scaleable (for training and deployment)
- (Re-)Learning to Learn
  - Our field moves fast
  - Codebases and best practices change
  - It is easy to get stuck in a rut
  - New tools change your way of tackling problems

Hugging Face

At Hugging Face we're using NLP and probabilistic models to generate conversational Artificial intelligences that are fun to chat with. Despite using deep neural nets for a few of our NLP tasks, scikit-learn is still the bread-and-butter of our daily machine learning routine. The ease of use and predictability of the interface, as well as the straightforward mathematical explanations that are here when you need them, is the killer feature. We use a variety of scikit-learn models in production and they are also operationally very pleasant to work with.

Julien Chaumond, Chief Technology Officer

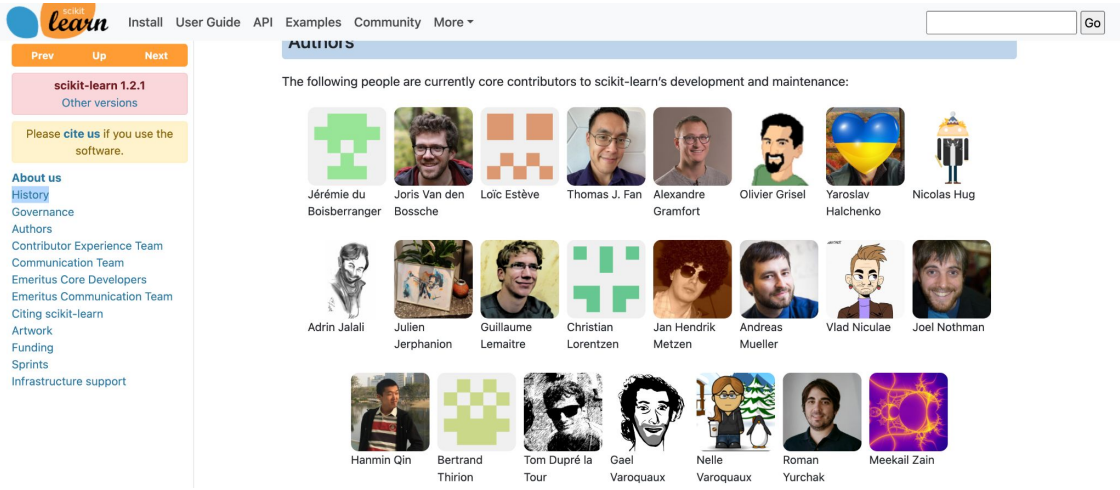
Spotify

Scikit-learn provides a toolbox with solid implementations of a bunch of state-of-the-art models and makes it easy to plug them into existing applications. We've been using it quite a lot for music recommendations at Spotify and I think it's the most well-designed ML package I've seen so far.

Erik Bernhardsson, Engineering Manager Music Discovery & Machine Learning, Spotify

History of *Scikit-learn*

- Started in 2007 as a Google Summer of Code project by David Cournapeau.
  - First public release, February the 1st 2010.
  - Since then, it follows a 3-month cycle, and has a thriving international community has been leading the development.



- https://scikit-learn.org/stable/

Scikit-learn *Release History*

Release History

Release notes for all scikit-learn releases are linked in this page.

Tip: Subscribe to scikit-learn releases on libraries.io to be notified when new versions are released.

- Version 1.2.1
- Version 1.2.0
- Version 1.1.3
- Version 1.1.2
- Version 1.1.1
- Version 1.1.0
- Version 1.0.2
- Version 1.0.1
- Version 1.0.0
- Version 0.24.2
- Version 0.24.1
- Version 0.24.0
- Version 0.23.2
- Version 0.23.1
- Version 0.23.0
- Version 0.22.2.post1
- Version 0.22.1
- Version 0.22.0
- Version 0.21.3
- Version 0.21.2
- Version 0.21.1
- Version 0.21.0
- Version 0.20.4
- Version 0.20.3
- Version 0.20.2
- Version 0.20.1
- Version 0.20.0
- Version 0.19.2
- Version 0.19.1
- Version 0.19
- Version 0.18.2
- Version 0.18.1
- Version 0.18
- Version 0.17.1 April 2016
- Version 0.17
- Version 0.16.1
- Version 0.16 March 2015
- Version 0.15.2
- Version 0.15.1
- Version 0.15
- Version 0.14
- Version 0.13.1
- Version 0.13
- Version 0.12.1
- Version 0.12
- Version 0.11
- Version 0.10
- Version 0.9
- Version 0.8
- Version 0.7
- Version 0.6
- Version 0.5
- Version 0.4
- Earlier versions

Release History *Since 1.0*

- Scikit-learn 1.2.X **(1.2.1 released January 2023)**
  - Pandas output with set_output API
  - Interaction constraints in Histogram-based Gradient Boosting Trees
  - New and enhanced displays
  - Faster parser in fetch_openml
  - Experimental Array API support in LinearDiscriminantAnalysis
  - Improved efficiency of many estimators
- Scikit-learn 1.1.X **(1.1.3 released October 2022)**
  - Quantile loss in ensemble.HistGradientBoostingRegressor
  - get_feature_names_out Available in all Transformers
  - Grouping infrequent categories in OneHotEncoder
  - Performance improvements
  - MiniBatchNMF: an online version of NMF
  - BisectingKMeans: divide and cluster
- Scikit-learn 1.0.X **(1.0.2 released November 2021)**
  - Keyword and positional arguments
  - Spline Transformers
  - Quantile Regressor
  - Feature Names Support
  - A more flexible plotting API
  - Online One-Class SVM
  - Histogram-based Gradient Boosting Models are now stable
  - New documentation improvements

Charlie's *Python and Scikit-learn Journey*

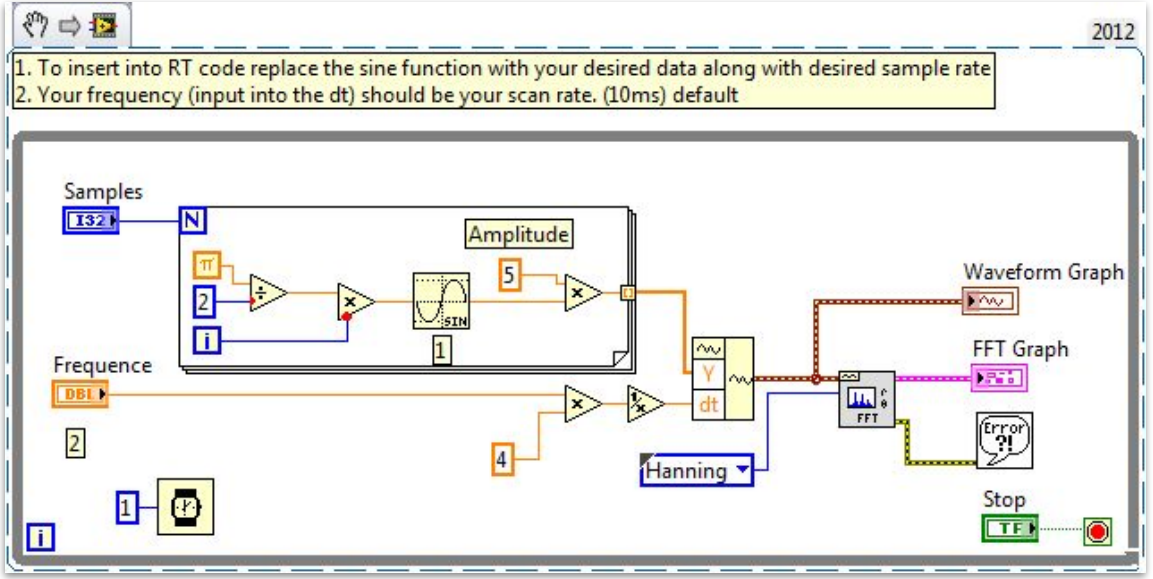
- Started to convert from R (novice from PhD. days) to Python in 2015
- Started teaching graduate course in ML and data cleansing (2016) - taught at least 3 times per year
- Started the MTSU Data Science Institute in 2018
- Started the Belmont Data Collaborative at Belmont University in 2021
- Started using Scikit-learn in 2015 with version 0.16
    - Since then there has been 36 revisions from 0.16 to 1.2.1

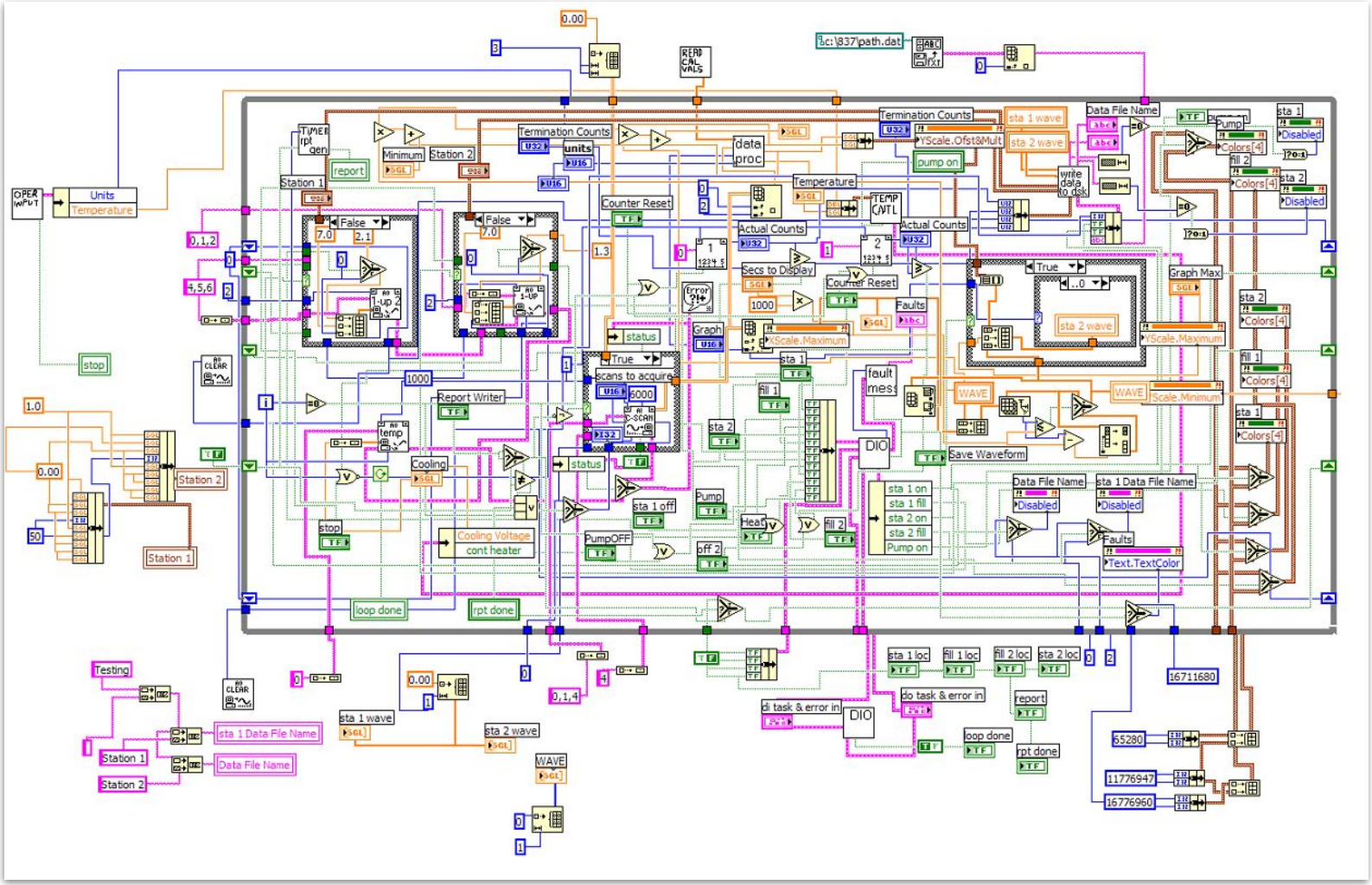
Charlie's *Python and Scikit-learn Journey*

- How did I get started?
  - Every known MOOC
  - Conferences and workshops
  - Projects - Projects - Projects
  - No better way to learn than to teach or present
- Three ways of learning
  - Classroom
  - Online
  - Experience
- Intent of code
  - To be simple to read and explain
  - Meant for teaching

Jason's *Python and Scikit-learn Journey*

- Physicist-turned-Data-Scientist
- Started my career with LabVIEW and MATLAB
- Slowly transitioned to Python around ~2010




OPER INPUT
Units
Temperature
stop
1.0
0.00
50
Station 2
Station 1
0,1,2
4,5,6
TIMER rpt gen
report
Minimum
Station 2
Station 1
False
7.0
2.1
False
7.0
1-UP
CLEAR
1000
Report Writer
temp
Cooling
stop
Cooling Voltage
cont heater
loop done
rpt done
Termination Counts
units
Counter Reset
1.3
status
True
scans to acquire
6000
status
sta 1 off
PumpOFF
off 2
data proc
Temperature
Actual Counts
Secs to Display
1000
Counter Reset
Graph
YScale.Maximum
sta 1
fill 1
sta 2
Pump
Heat
fill 2
DIO
sta 1 on
sta 1 fill
sta 2 on
sta 2 fill
Pump on
Termination Counts
YScale.Ofst&Mult
pump on
write data to disk
Actual Counts
Faults
True
sta 2 wave
fault mess
WAVE
Save Waveform
Data File Name
sta 1 Data File Name
Disabled
Faults
Text.TextColor
Graph Max
YScale.Maximum
YScale.Minimum
Pump
Colors[4]
fill 2
Colors[4]
sta 2
Disabled
sta 1
Disabled
fill 1
Colors[4]
sta 2
Colors[4]
sta 1
Colors[4]
c:\837\path.dat
READ CAL VALS
Testing
sta 1 Data File Name
Station 1
Station 2
Data File Name
sta 1 wave
sta 2 wave
WAVE
0,1,4
di task & error in
DIO
sta 1 loc
fill 1 loc
fill 2 loc
sta 2 loc
do task & error in
loop done
report
rpt done
16711680
65280
11776947
16776960

Spyder (Python 3.9)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\jkkph\Downloads

C:\Users\jkkph\Downloads\Trap_sim_V4.py

gen_fgs_cnr.py | Trap_sim_V4.py

```python
# -*- coding: utf-8 -*-
"""
Created on Thu Feb 07 11:29:22 2013
Trap_sim.py: Simulate trapping of a single nanoparticle with all the delays
from experimental setup.
V2: Runs the experiment for 30s. Calulates RMS.
V3: Allows for reduced loop response on individual axes.
V4: Adds error into measurements. Fixed loop handling.
@author: jking1
"""

import numpy as np
import pylab as plt

D = 12.22 ## Microns^2/s
dt = 1E-4 ## 100 microseconds
mean = np.sqrt(2*D*dt)

t1 = 333
t2 = 330
phase = 150 ## This is the time delay between camera output and LabVIEW loop
n = 300000
meas_pos = np.zeros(shape = (n,3))
true_pos = np.zeros(shape = (n,3))
r = np.zeros(n)
t = np.zeros(shape = n)
j = 0; k = phase;
store = [np.array([0,0,0]),np.array([0,0,0])]; V = []
scale = np.array([0.5,0.5,0.5])
for i in range(n-1):
    t[i+1] = t[i]+dt
    step = np.random.normal(scale = mean, size = 3)
    true_pos[i+1] = true_pos[i] + step

    if j == t1:
        meas_pos[i+1] = true_pos[i+1]
        ## Add measurement error (minimum precision)
        err = np.array([np.random.normal(scale=0.024), np.random.normal(scale=0.017), np.random.normal(scale=0.172)])
        store.append(meas_pos[i+1] + err)
        j = 0
    else:
        meas_pos[i+1] = meas_pos[i]
        j += 1

    if k == t2:
        true_pos[i+1] = true_pos[i+1] - scale*store[-2]
        V.append(t[i+1])
        k = 0
    else:
        k += 1

    r[i+1] = np.sqrt(np.sum(np.power(true_pos[i+1],2)))

RMS = np.sqrt(np.sum(np.power(r,2))/len(r))
print(RMS)
V = np.array(V)
fig = plt.figure('Position', (6,6))
ax1 = fig.add_subplot(311)
ax1.plot(t, true_pos[:,0], color = 'blue')
ax1.plot(t, meas_pos[:,0], color = 'green')
ax1.vlines(V, -10, 10, color = 'red', linestyle = 'dashed')
ax1.grid(True)
ax1.axis([0,10000*dt,-5,5])
ax2 = fig.add_subplot(312)
ax2.plot(t, true_pos[:,1], color = 'blue')
ax2.plot(t, meas_pos[:,1], color = 'green')
ax2.vlines(V, -10, 10, color = 'red', linestyle = 'dashed')
ax2.grid(True)
ax2.axis([0,10000*dt,-5,5])
ax3 = fig.add_subplot(313)
ax3.plot(t, true_pos[:,2], color = 'blue')
ax3.plot(t, meas_pos[:,2], color = 'green')
ax3.vlines(V, -10, 10, color = 'red', linestyle = 'dashed')
ax3.grid(True)
ax3.axis([0,10000*dt,-5,5])
plt.show()
```

Name	Type	Size	Value
ax1	axes._subplots.AxesSubplot	1	AxesSubplot object of matplotlib.axes._subplots module
ax2	axes._subplots.AxesSubplot	1	AxesSubplot object of matplotlib.axes._subplots module
ax3	axes._subplots.AxesSubplot	1	AxesSubplot object of matplotlib.axes._subplots module
D	float	1	12.22
dt	float	1	0.0001
err	Array of float64	(3,)	[-5.07552569e-02 6.94400188e-05 1.11100163e-01]
fig	figure.Figure	1	Figure object of matplotlib.figure module
i	int	1	299998
j	int	1	67
k	int	1	263
mean	float64	1	0.04943682837723311
meas_pos	Array of float64	(300000, 3)	[[0. 0. 0.] [0. 0. 0. . ...
n	int	1	300000
phase	int	1	150
r	Array of float64	(300000,)	[0. 0.07296769 0.19100325 ... 2.69113949 2.72078226 2.81065431 ...
scale	Array of float64	(3,)	[0.5 0.5 0.5]
step	Array of float64	(3,)	[0.00521995 -0.04084161 -0.108324]
store	list	900	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
[illegible]	[illegible]	[illegible]	[illegible]

Help | Variable Explorer | Plots | Files

Console 1/A

```
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/jkkph/Downloads/Trap_sim_V4.py', wdir='C:/Users/jkkph/Downloads')
  File "C:\Users\jkkph\Downloads\Trap_sim_V4.py", line 55
    print RMS
          ^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(RMS)?


In [2]: runfile('C:/Users/jkkph/Downloads/Trap_sim_V4.py', wdir='C:/Users/jkkph/Downloads')
2.3870840150851573
```

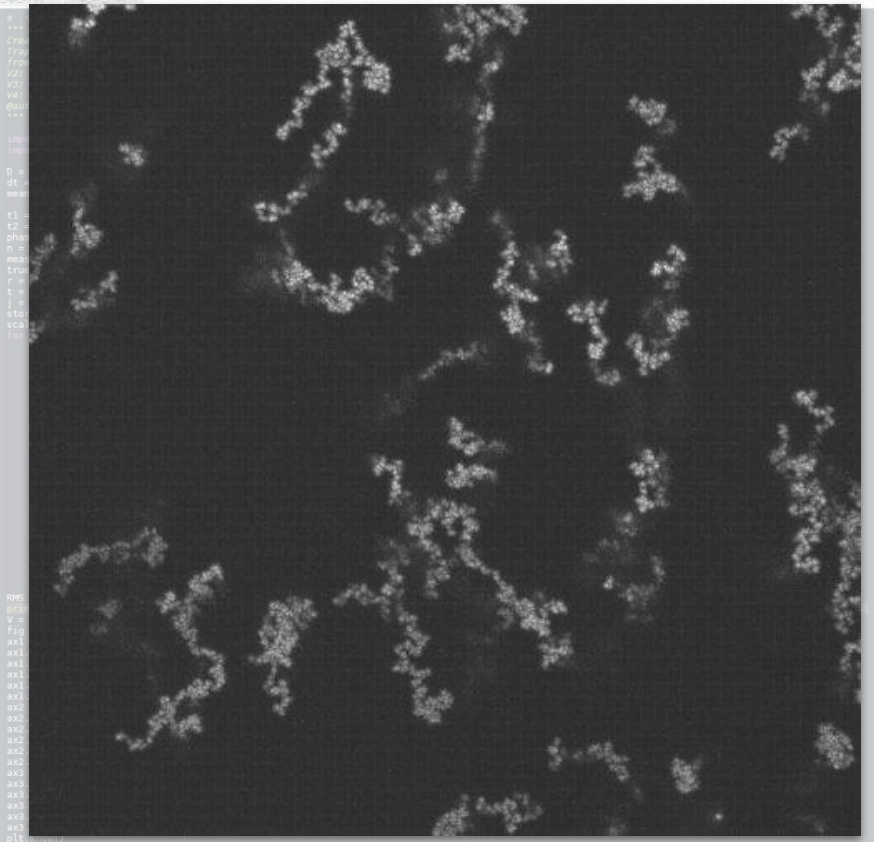
Warning

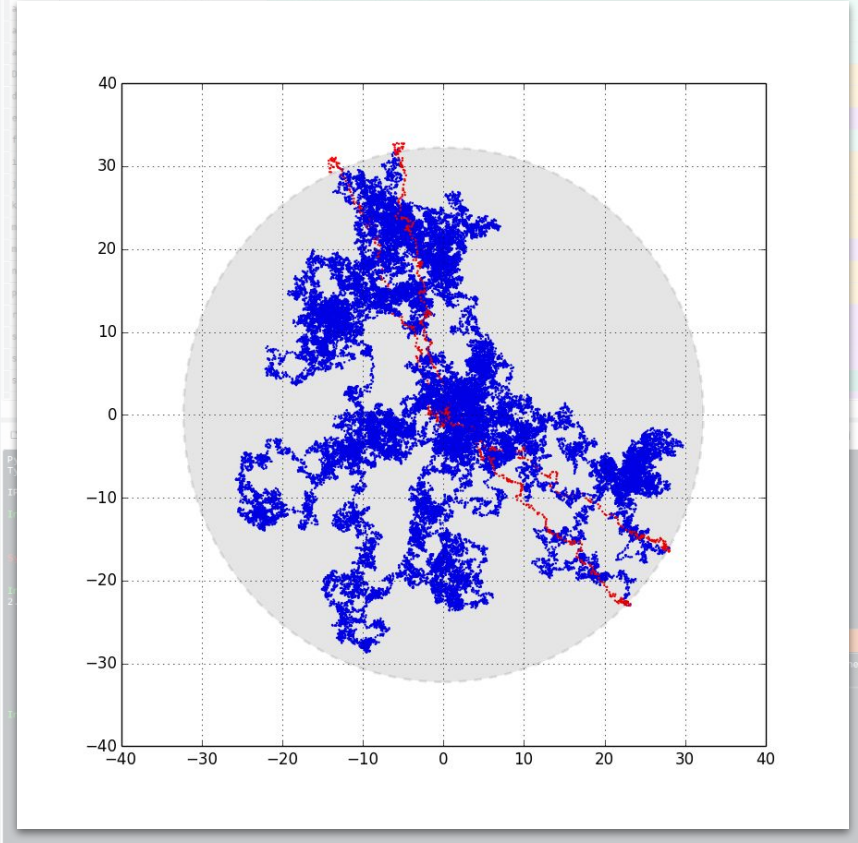
Figures now render in the Plots pane by default. To make them also appear inline in the Console, uncheck "Mute Inline Plotting" under the Plots pane options menu.

```
In [3]:
```

IPython Console | History

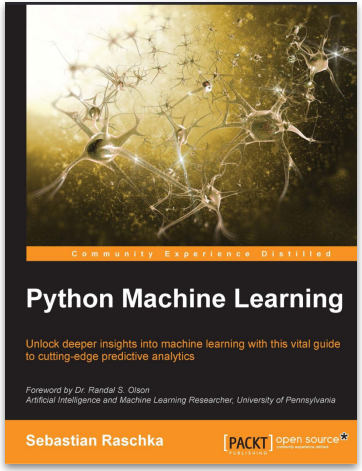
LSP Python: ready | conda: base (Python 3.9.13) | Line 55, Col 11 | UTF-8 | CRLF | RW | Mem 36%

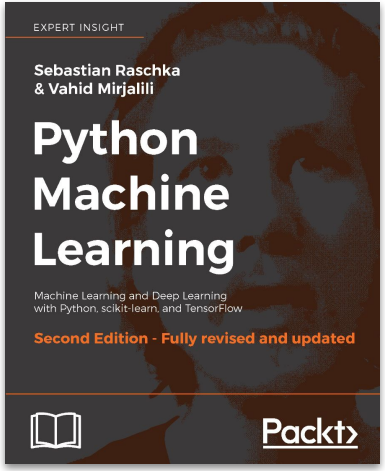


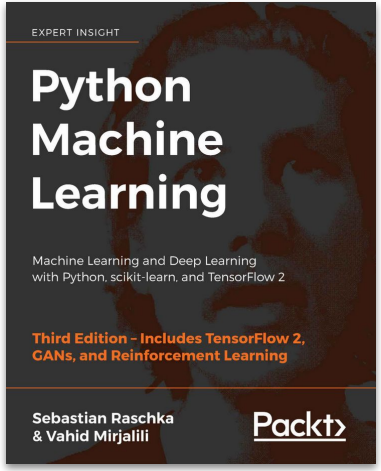

40
30
20
10
0
−10
−20
−30
−40
−40
−30
−20
−10
0
10
20
30
40

Jason's *Python and Scikit-learn Journey*

- Physicist-turned-Data-Scientist
- Started my career with LabVIEW and MATLAB
- Slowly transitioned to Python around ~2010
- Started getting into data science around 2015 (self-study)
- Landed on Scikit-learn when I started with Xsolis in 2016







Demo *Histogram-based Gradient Boosting Models*

sklearn.ensemble.HistGradientBoostingClassifier ¶

class sklearn.ensemble.**HistGradientBoostingClassifier**(*loss='log_loss', *, learning_rate=0.1, max_iter=100, max_leaf_nodes=31, max_depth=None, min_samples_leaf=20, l2_regularization=0.0, max_bins=255, categorical_features=None, monotonic_cst=None, interaction_cst=None, warm_start=False, early_stopping='auto', scoring='loss', validation_fraction=0.1, n_iter_no_change=10, tol=1e-07, verbose=0, random_state=None, class_weight=None*) [source]

Histogram-based Gradient Boosting Classification Tree.

This estimator is much faster than `GradientBoostingClassifier` for big datasets (n_samples >= 10 000).

This estimator has native support for missing values (NaNs). During training, the tree grower learns at each split point whether samples with missing values should go to the left or right child, based on the potential gain. When predicting, samples with missing values are assigned to the left or right child consequently. If no missing values were encountered for a given feature during training, then samples with missing values are mapped to whichever child has the most samples.

This implementation is inspired by LightGBM.

Read more in the User Guide.

New in version 0.21.

Parameters: **loss : *{'log_loss', 'auto', 'binary_crossentropy', 'categorical_crossentropy'}, default='log_loss'***
The loss function to use in the boosting process.

For binary classification problems, 'log_loss' is also known as logistic loss, binomial deviance or binary crossentropy. Internally, the model fits one tree per boosting iteration and uses the logistic sigmoid function

- Boosting refers to a class of ensemble learning algorithms that add tree models to an ensemble sequentially.
- Each tree model added to the ensemble attempts to correct the prediction errors made by the tree models already present in the ensemble.
- Can be sped up significantly by ***reducing*** the number of values for continuous input features.
- This can be achieved by ***binning values*** into a fixed number of buckets.

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html

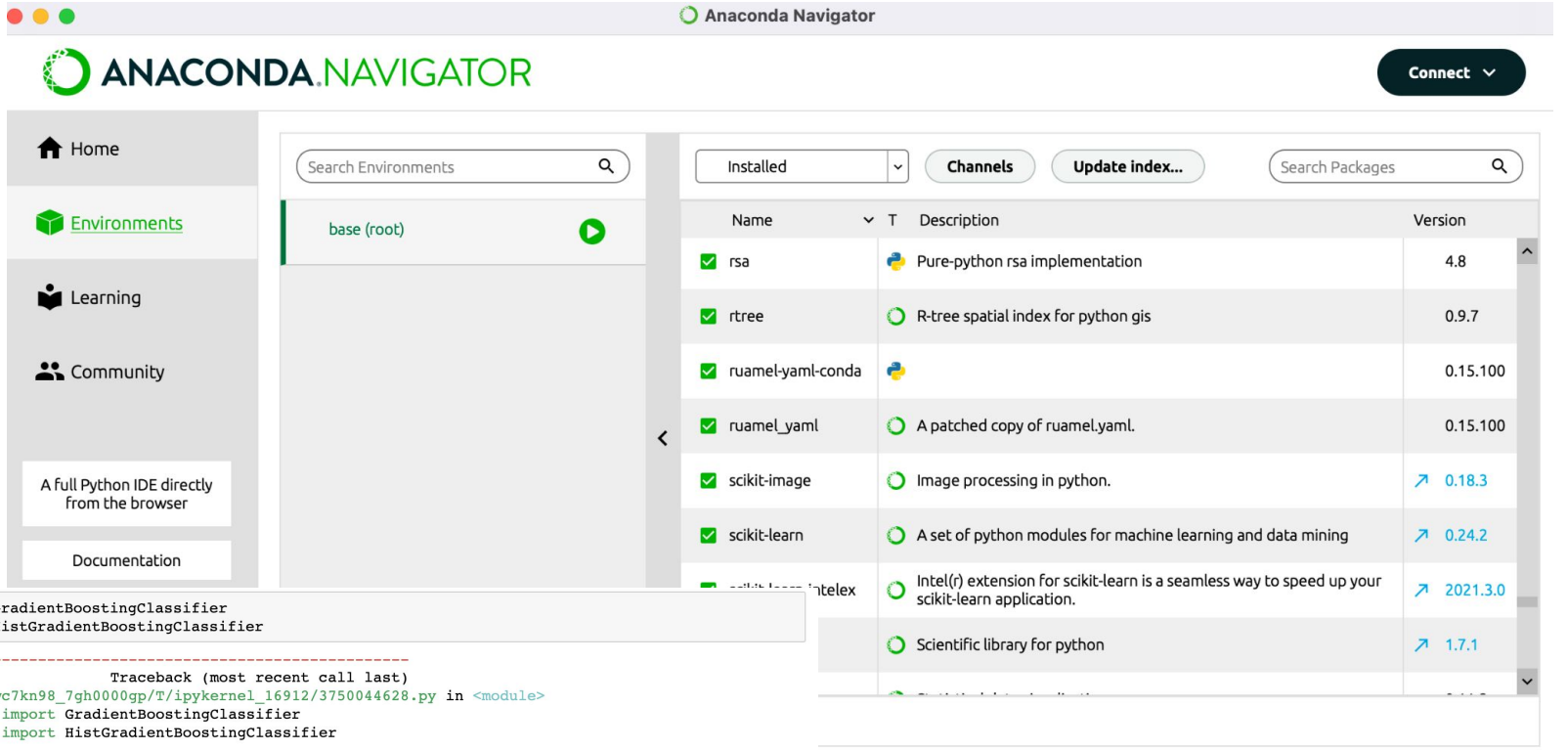
Demo *Histogram-based Gradient Boosting Models*

- Charlie had an example that he was using in a class with Statewide Hypertension data - grouped at the U.S. Census tract.
- Target variable: BP% high in tract (top 20%)
- Features: SDOH
- Techniques: Classification modeling - previously used:
  - Logistic Regression (probabilistic)
  - Decision Tree (Tree-based)
  - Random Forest (Ensemble)
  - Gradient Boosting (Ensemble)
- Added HGBM to give it a try.

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html

HGMB *Why use it?*

- Because it bins, it should be faster.
- Allows for null-values in data




```
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.ensemble import HistGradientBoostingClassifier
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
/var/folders/4b/cmmlzn5s5r1g4wc7kn98_7gh0000gp/T/ipykernel_16912/3750044628.py in <module>
      1 from sklearn.ensemble import GradientBoostingClassifier
----> 2 from sklearn.ensemble import HistGradientBoostingClassifier

ImportError: cannot import name 'HistGradientBoostingClassifier' from 'sklearn.ensemble' (/Users/charlieapigian/opt/a
naconda3/lib/python3.9/site-packages/sklearn/ensemble/__init__.py)
```

Charlie's Demo

Let’s Demo:

Histogram-based Gradient Boosting Models

Demo *Histogram-based Gradient Boosting Models*

```
%%time

vartrain = X_train_n
vartest = X_test_n
model = GradientBoostingClassifier(random_state=21)

shorttraintest(vartrain, vartest, y_train_n, y_test_n, model)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<timed exec> in <module>

/var/folders/4b/cmmlzn5s5r1g4wc7kn98_7gh0000gp/T/ipykernel_29931/2610170750.py in shorttraintest(vartrain, vartest, y
_train, y_test, model)
      2
      3     #Fit the model
----> 4     model.fit(vartrain, y_train)
      5
      6     #Predict with the model

~/opt/anaconda3/lib/python3.9/site-packages/sklearn/ensemble/_gb.py in fit(self, X, y, sample_weight, monitor)
    427         # trees use different types for X and y, checking them separately.
    428
--> 429         X, y = self._validate_data(
    430             X, y, accept_sparse=["csr", "csc", "coo"], dtype=DTYPE, multi_output=True
    431         )

~/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py in _validate_data(self, X, y, reset, validate_separatel
**check_params)
    552                 y = check_array(y, input_name="y", **check_y_params)
    553             else:
--> 554                 X, y = check_X_y(X, y, **check_params)
    555             out = X, y
```

GBM

HGBM

In [59]:

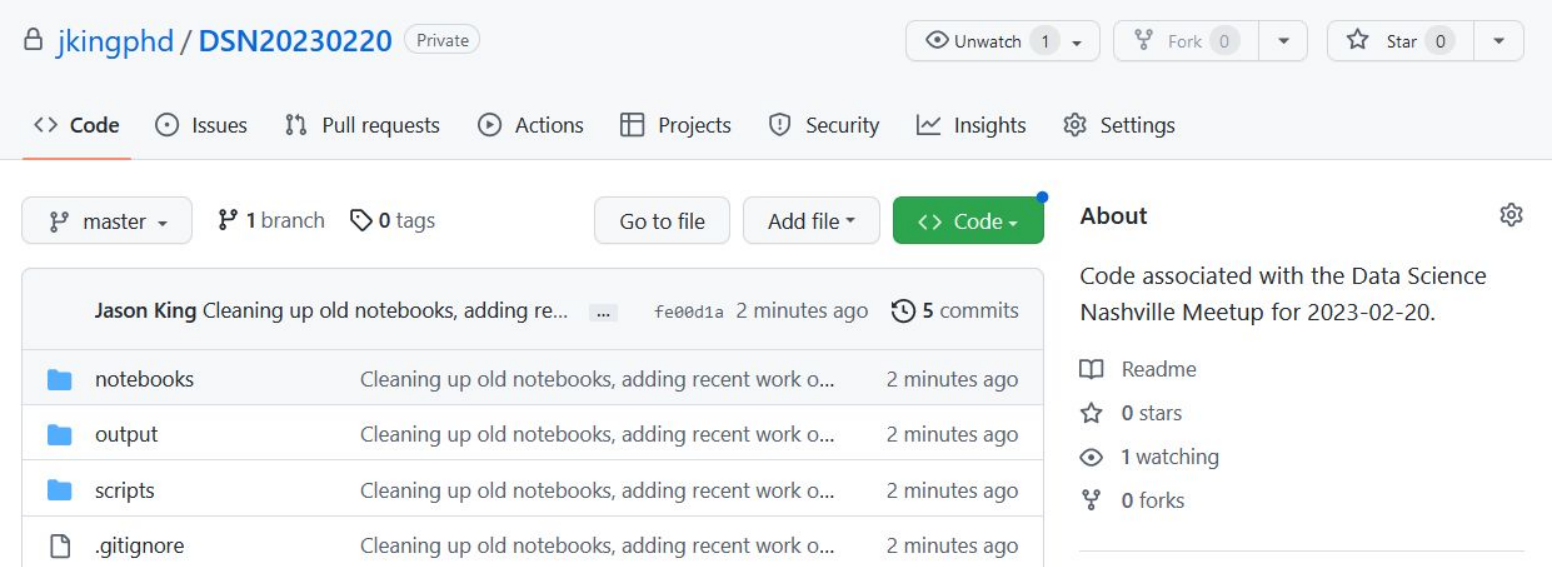
```
%%time

vartrain = X_train_n
vartest = X_test_n
model = HistGradientBoostingClassifier(random_state=21)

shorttraintest(vartrain, vartest, y_train_n, y_test_n, model)
```

```
Confusion Matrix:
[[358  10]
 [ 12  65]]

Accuracy Score: 0.9506
Recall Score: 0.8442
Precision Score: 0.8667
ROC AUC Score: 0.9861

CPU times: user 6min 21s, sys: 10.5 s, total: 6min 31s
Wall time: 5min 46s
```

Demo *Jason's Turn*

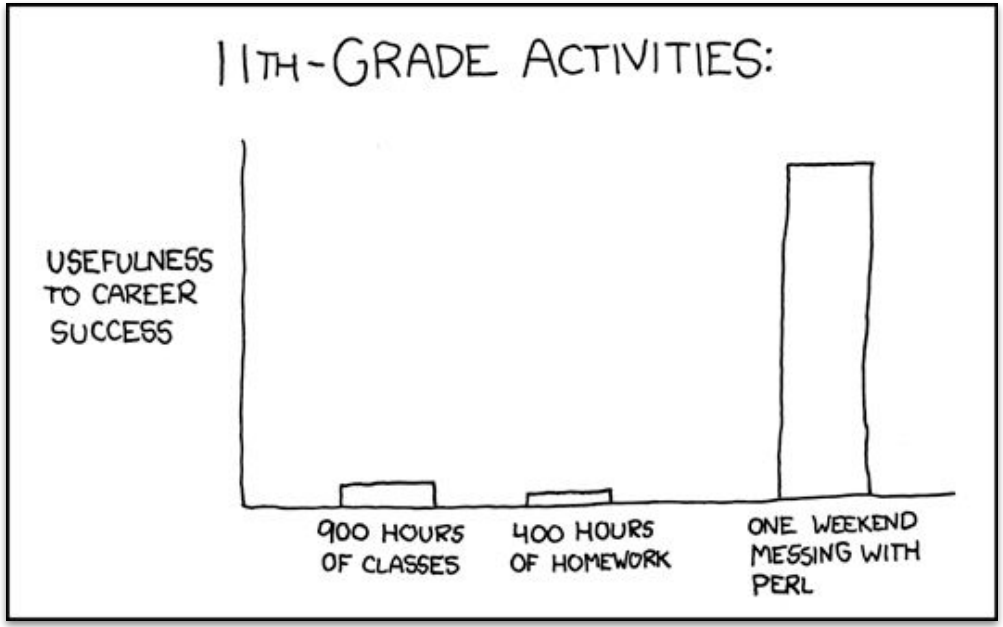
- Quality-of-life additions
- MiniBatchNMF and online learning
- SGDOneClassSVM (time permitting)

Jason's Demo



Takeaways

- Personal growth takes effort
  - Force yourself to build something from scratch or use a new tool/method
  - Don’t be afraid to revisit old code and/or libraries



DATA SCIENCE NASHVILLE

Jason King

jason@kinglambda.com

Charlie Apigian

charles.apigian@belmont.edu

Charlie's Demo

Jason's Demo

DATA SCIENCE NASHVILLE

Upcoming Meetups

Monday, March 20th at 6pm

Scott Hawley - Belmont University and Harmonai.org

Mathemangling Audio Embeddings for Fun and (Perhaps) Profit

Wednesday, April 19th at 6pm

David Ciommo - Humana

Data Storytelling: Where Art, Science & Data Literacy Meet

Monday, May 15th at 6pm

Anti-Social - TBD