TestJ2m

June 11, 2020

1 How to turn your Jupyter notebook to a Medium article in one Click

1.1 Introduction

For those of you who have been using Jupyter notebook to document in detail your Machine Learning models and experiments with Data Science you'll find this feature very useful.

I have tried myself to publish notebooks by re-writing content specifically for Medium by editing images, importing, sorting out format etc… and it's very tedious process that takes attention and time away from developing the notebooks themselves.

1.2 Step 1 - Get the Library

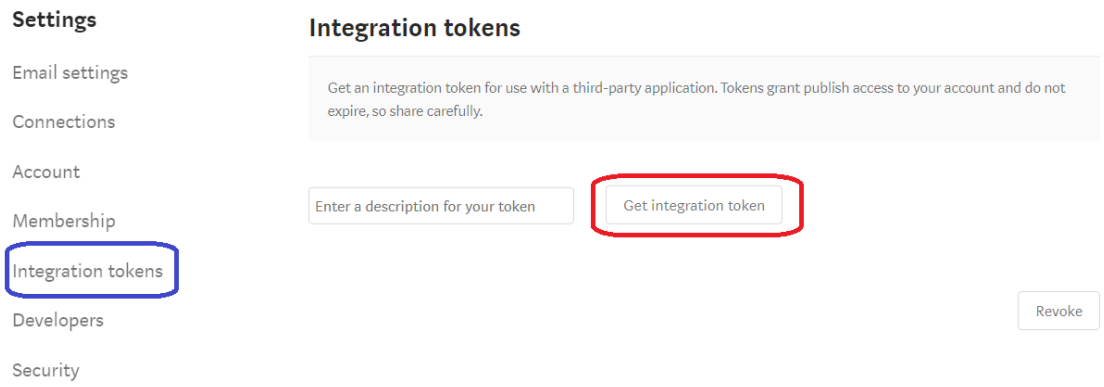
Luckily @Ted Petrou has published a library that can do that in one Click, and guess its name:"jupyter-to-medium".

To install, run the following command under your Linux shell:

pip install jupyter-to-medium

1.3 Step 2 - Medium Integration Token

To start with, you need to get an integration token from Medium. Go to your settings and select "Integration tokens" section. From there you can self-issue a token and give it a description as per screenshot below:



This is the so called "Self-issued" access tokens, which is described below in detail (taken from Medium's API documentation https://github.com/Medium/medium-api-docs):

Self-issued access tokens (described in user-facing copy as integration tokens) are explicitly designed for desktop integrations where implementing browser-based authentication is non-trivial, or software like plugins where it is impossible to secure a client secret. You should not request that a user give you an integration token if you don't meet these criteria. Users will be cautioned within Medium to treat integration tokens like passwords, and dissuaded from making them generally available.

Users can request an access token by emailing yourfriends@medium.com. We will then grant access on the Settings page of their Medium account.

You should instruct your user to visit this URL and generate an integration token from the Integration Tokens section. You should suggest a description for this token - typically the name of your product or feature - and use it consistently for all users.

Self-issued access tokens do not expire, though they may be revoked by the user at any time.

Alternatively, you can check the same page on **Browser based** authentication to do it programatically

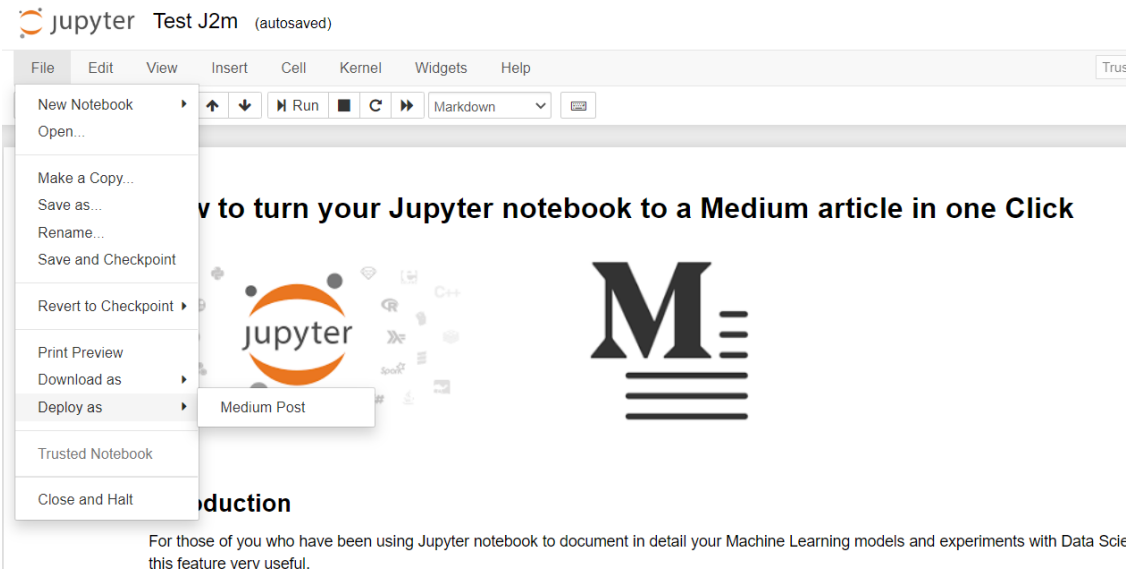
1.3.1 Save your token

Once you've generated the token, it needs to be saved locally under the following folder in your home directory:

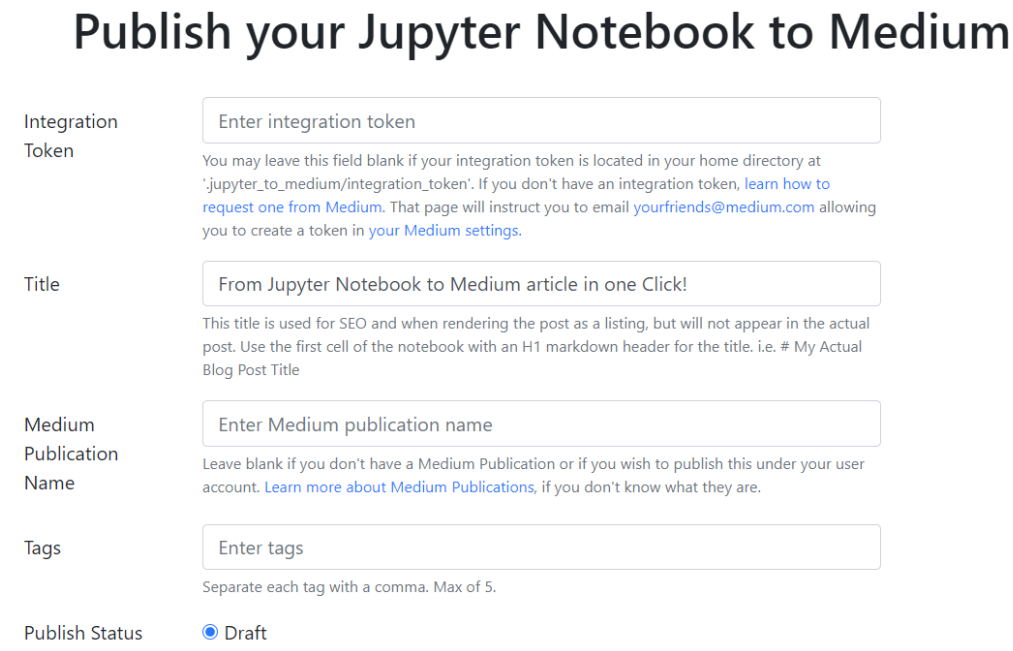
.jupyter_to_medium/integration_token

1.4 Step 3 - Publish via Notebook

Once you've completed your work, navigate to the the menu File –> Deploy as –> Medium Post :



This will launch a new tab in your browser with the following screen at Medium website:



As per instructions, you don't have to manually enter the token if you've taken Step 2 above. You only need to change the Title and hit publish to turn your notebook into a Draft publication to Medium.

Once you've done that, you'll get a confirmation message and a link to navigate to your post:

Published as a draft to Medium!

Link to your Medium Post: Test Deploying JN as draft post

Navigate to your post on Medium to finalize publishing.

URL - https://medium.com/@etzimopoulos/a3a31d02aeb

Canonical URL -

Publish Status - draft

Tags - []

License - all-rights-reserved

License URL - https://medium.com/policy/9db0094a1e0f

[ ]:

1.5 Step 4 - Linear Regression Example

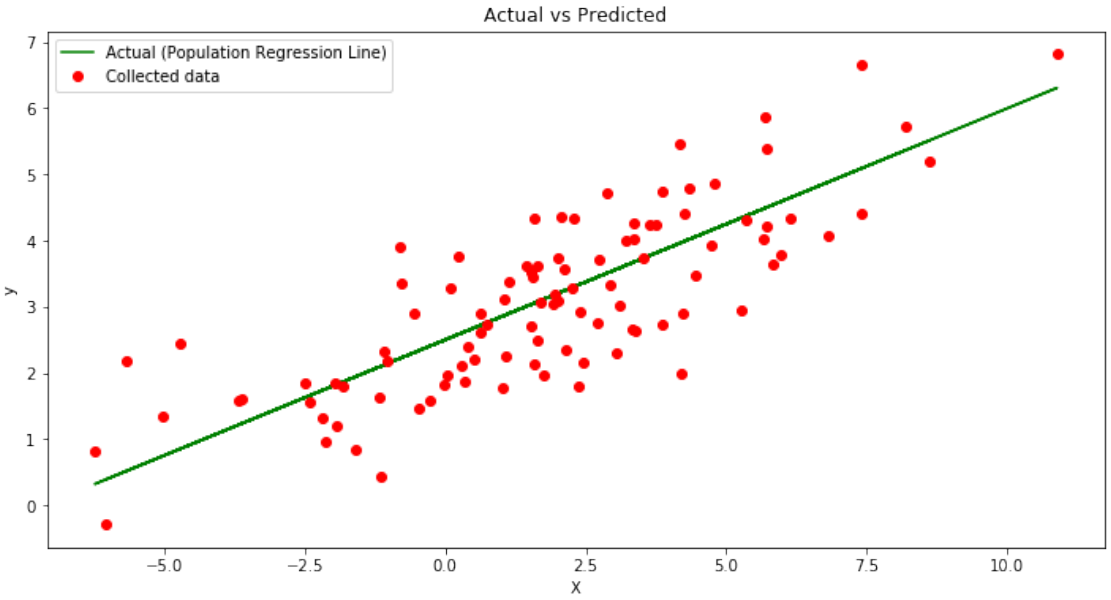
In the example below, we will: * Create a sample regression line based on a linear equation y = aX + b * Generate sample X and y by adding noise to simulate a sample dataset of X,y pairs * Store the dataset into a Pandas datafre * Plot the regression line and the sampled population

[1]:
```
import numpy as np
import pandas as pd
from numpy import array
#from numpy.linalg import inv
from matplotlib import pyplot as plt
#from scipy import stats
```

[2]:
```
# Number of Samples
n = 100

# Create r and r1, random vectors of 100 numbers each with mean = 0 and␣
 ↪standard deviation = 1
r = np.random.randn(n)
r1 = np.random.randn(n)

# Create random Input vector X using r
```

```
# mean = 3
# stddev = 2
X = 3 * r + 2

# Create random Residual term Res using r
# mean = 0
# stddev = 0.8
res = 0.8 * r1

# Generate Y values based on the simulated regression line and error/noise
# Population Regression Line
yreg = 2.5 + 0.35 * X
# Adding noise/error
y = yreg + res

# Storing Population Regression Line "RegL", data points X and y in a data frame
rl = pd.DataFrame(
    {'X': X,
     'y': y,
     'RegL':yreg}
)

# Show the first five rows of our dataframe
rl.head()
```

[2]:
[3]:
```
# Plot regression against actual data
plt.figure(figsize=(12, 6))
# Population Regression Line
plt.plot(X,rl['RegL'], label = 'Actual (Population Regression␣
 ↪Line)',color='green')
# Least squares line
#plt.plot(X, ypred, label = 'Predicted (Least Squares Line)', color='blue')
# scatter plot showing actual data
plt.plot(X, y, 'ro', label ='Collected data')
plt.title('Actual vs Predicted')
plt.xlabel('X')
plt.ylabel('y')
plt.legend()
plt.show()
```



[ ]:

1.6 Step 5 - Publish on Medium

Once you've published a draft on Medium, you can edit and make changes as required but essentially, all the pictures, gifs, dataframes and plots should be automatically tranfered without the need to manually do this again.

1.7 References

Original Medium article by @Ted Petrou - https://medium.com/dunder-data/jupyter-to-medium-initi

[ ]:

[ ]:

[ ]: