Telling Stories with Jupyter notebook

Kevin Lee, Clindata Insight, Moraga, CA

ABSTRACT

The Jupyter Notebook is an open-source web application that allows programmers and data scientists to create and share documents that contain live code, visualizations and narrative text. Jupyter Notebook is one of most popular tool for data visualization and machine learning, and it is the perfect tool for story telling tool for data scientist.
First, the paper will start with the introduction of Jupyter Notebook and why it is the most popular tool for data scientist to show, share and visualize the data and analysis. The paper will show how data scientist uses Python programming language in Jupyter Notebook. The paper will show how data scientists import data into Jupyter Notebook using Pandas. The paper will introduce Python data visualization library, matplotlib, and show how data scientists use matplotlib to easily create scatter plot, line, histograms, Kaplan Meier curves and many more.

The paper will present how data scientist use Jupyter notebook for image recognitions with visualization and machine learning. The paper will show how data scientists can convert images into numeric array. Then, the paper will show how data scientist can use this numeric data to visualize and train machine learning model for image recognition.

NOTE: The paper is written in Jupyter Noteboor and exported in pdf.

INTRODUCTION OF JUPYTER NOTEBOOK

Jupyter Notebook is a free programming platform that comes from a concept of a notebook, which contains ordinary text and calculation and/or graphics. It performs the data analysis in real time. Jupyter is a loose acronym meaning Julia, Python and R since they are the first target languages. Nowadays, Jupyter notebook also supports many languages (e.g., SAS, Perl, PHP, Octave, Matlab, C++, Java, C, etc). Jupyter notebook run your program in a web browser.

WHY JUPYTER NOTEBOOK?

Jupyter notebook is different from other programming platforms because it can contain beyond comments, codes and results. One of the main powers and reasons behind the popularity of Jupyter notebook is how well it packages different medium in one simple solution. A data scientist can code, write and visualize in one place – Jupypter notebook. It greatly simplifies the sharing of programming process, especially for collaborative purposes.

Jupyter notebook contains the followings.

- Rich texts
- Graphs
- Links
- Equations
- Video
- Codes
- Results
- Visualization

Jupyter notebook runs analysis in real time, so the data scientist can show and explain what he or she does with the audience more interactively. Many data scientist present their codes and results in Jupyter Notebook in meetings and conferences.

HOW TO USE JUPYTER NOTEBOOK

The data scientist can tell stories using many features of Jupyter notebook. The paper will some how data scientists can use Jupyter notebook in the followings.

- Import image from local drive
- Import video from local drive
- how to import SAS datasets from local drive
- Python Data Visualization in Jupyter Notebook - matplotlib
- Kaplan Meier Curves creation using lifelines package
- MNIST Dataset Machine Learning Binary Classification
- Import image file

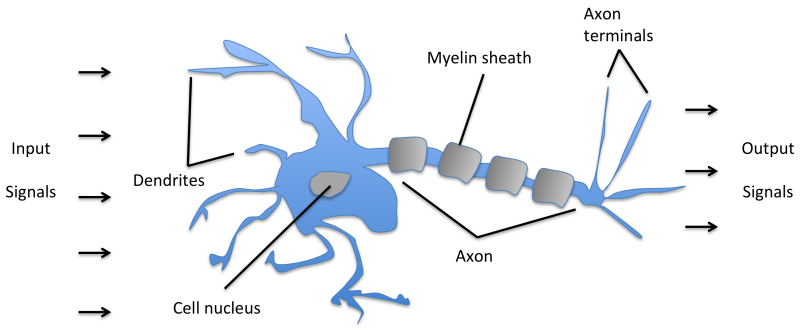
Import image from local drive

The data scientist can import Ipython package and use Image function to display the local image in Jupyter notebook.

In [1]:
```
from IPython.display import Image
```

In [2]:
```
Image(filename='./images/02_01.png', width=500)
```

Out[2]:



In [3]:
```
Image(filename='./images/PharmaSUG_2018.png', width=500)
```

Out[3]:

Import video from local drive

The data scientist can also import video in Jupyter Notebook.

0:00 / 1:00

how to import SAS datasets from local drive

The data scientist can import many formats of data sets in Python Kernel of Jupyter notebook such as csv, json, xml, sas, texts, images, videos and many more.
The paper will show how data scientist can import SAS dataset. First, The data scientist should import pandas and numpy package. Pandas is a Python package providing fast, flexible, and expressive data structure designed to make working relational data both easy and intuitive.

In [4]:
```
### Import Pandas and Numpy package
import pandas as pd
import numpy as np
```

Using read_sas function, the data scientist can import ADSL datasets.

In [5]:
```
## Read ADSL datasets
adsl = pd.read_sas('./data/SAS/ADaM/adsl.xpt')
adsl.head()
```

Out[5]:

	USUBJID	STUDYID	DOMAIN	SITEID	SITEGRP	SUBJID	VISIT1DT	RANDDT	TRTSTDT	RFSTDTC	...	EDUCLVL	HE
0	b'01-701-1015'	b'CDISCPILOT01'	b'ADSL'	b'701'	b'701'	b'1015'	19718.0	19725.0	19725.0	b'2014-01-02'	...	16.0	
1	b'01-701-1023'	b'CDISCPILOT01'	b'ADSL'	b'701'	b'701'	b'1023'	19196.0	19210.0	19210.0	b'2012-08-05'	...	14.0	
2	b'01-701-1028'	b'CDISCPILOT01'	b'ADSL'	b'701'	b'701'	b'1028'	19550.0	19558.0	19558.0	b'2013-07-19'	...	16.0	
3	b'01-701-1033'	b'CDISCPILOT01'	b'ADSL'	b'701'	b'701'	b'1033'	19792.0	19800.0	19800.0	b'2014-03-18'	...	12.0	
4	b'01-701-1034'	b'CDISCPILOT01'	b'ADSL'	b'701'	b'701'	b'1034'	19898.0	19905.0	19905.0	b'2014-07-01'	...	9.0	

5 rows × 51 columns

Describe function in Pandas can work like PROC MEANS in SAS. It provides simple descriptive statistcs for numeric variables.

In [6]:
```
### quick reveiw of ADSL datasets
adsl.describe()
```

Out[6]:

	VISIT1DT	RANDDT	TRTSTDT	LSTDOSDT	ENDDT	VISNUMEN	TRTDUR	TRTPN	TRTDOS
count	254.000000	254.000000	254.000000	254.000000	254.000000	254.000000	254.000000	2.540000e+02	2.540000e+(
mean	19515.519685	19526.519685	19526.519685	19641.610236	19646.602362	9.582677	116.090551	9.921260e-01	4.464567e+(
std	177.503743	177.125058	177.125058	192.487651	191.253746	2.828262	70.295506	8.196795e-01	3.384375e+(
min	19180.000000	19183.000000	19183.000000	19233.000000	19237.000000	4.000000	1.000000	5.397605e-79	5.397605e-7
25%	19375.500000	19384.250000	19384.250000	19490.250000	19494.000000	7.250000	46.250000	5.397605e-79	5.397605e-7
50%	19511.500000	19522.500000	19522.500000	19628.000000	19631.500000	11.000000	133.500000	1.000000e+00	5.400000e+(
75%	19655.000000	19669.250000	19669.250000	19797.500000	19802.250000	12.000000	183.000000	2.000000e+00	8.100000e+(
max	19964.000000	19968.000000	19968.000000	20152.000000	20152.000000	12.000000	212.000000	2.000000e+00	8.100000e+(

8 rows × 22 columns

In [ ]:
```
### list of variables
list(adsl)
```

The data scientist can divide ADSL into "Placebo" and "Control groups" and put them into two different datasets.

In [8]:
```
### ADSL with Placebo
adsl2 = adsl[adsl.TRTP == b'Placebo']
### ADSL with Study Drug
adsl3 = adsl[adsl.TRTP != b'Placebo']
adsl2[:20]
print(adsl2.describe())
print(adsl3.describe())
```

```
           VISIT1DT        RANDDT       TRTSTDT      LSTDOSDT         ENDDT  \
count     86.000000     86.000000     86.000000     86.000000     86.000000
mean   19527.558140  19538.604651  19538.604651  19686.674419  19690.058140
std      186.722292    186.771410    186.771410    193.510750    191.709176
min    19180.000000  19183.000000  19183.000000  19237.000000  19238.000000
25%    19380.750000  19393.500000  19393.500000  19544.250000  19545.750000
50%    19543.500000  19551.000000  19551.000000  19684.000000  19684.000000
75%    19684.500000  19698.500000  19698.500000  19811.500000  19819.250000
max    19964.000000  19968.000000  19968.000000  20152.000000  20152.000000

        VISNUMEN      TRTDUR         TRTPN       TRTDOSE         AVGDD  \
count  86.000000   86.000000  8.600000e+01  8.600000e+01  8.600000e+01
mean   10.732558  149.069767  5.397605e-79  5.397605e-79  5.397605e-79
std     2.378454   60.295506  0.000000e+00  0.000000e+00  0.000000e+00
min     4.000000    7.000000  5.397605e-79  5.397605e-79  5.397605e-79
25%    11.000000  131.000000  5.397605e-79  5.397605e-79  5.397605e-79
50%    12.000000  182.000000  5.397605e-79  5.397605e-79  5.397605e-79
75%    12.000000  183.000000  5.397605e-79  5.397605e-79  5.397605e-79
max    12.000000  210.000000  5.397605e-79  5.397605e-79  5.397605e-79

         ...            AGE    AGEGRPN      RACEN      BMIBL      DISONSET  \
count    ...      86.000000  86.000000  86.000000  86.000000     86.000000
mean     ...      75.209302   2.186047   1.232558  23.636047  18230.267442
std      ...       8.590167   0.694713   0.777241   3.671926    928.183861
min      ...      52.000000   1.000000   1.000000  15.100000  14043.000000
25%      ...      69.250000   2.000000   1.000000  21.200000  17933.000000
50%      ...      76.000000   2.000000   1.000000  23.400000  18491.000000
75%      ...      81.750000   3.000000   1.000000  25.600000  18805.500000
max      ...      89.000000   3.000000   5.000000  33.300000  19456.000000

           DURDIS    EDUCLVL    HEIGHTBL    MMSETOT   WEIGHTBL
count   86.000000  86.000000   86.000000  86.000000  86.000000
mean    42.650000  12.581395  162.573256  18.046512  62.759302
std     30.241572   2.948440   11.522361   4.272778  12.771544
min      7.200000   6.000000  137.200000  10.000000  34.000000
25%     24.325000  12.000000  154.000000  15.000000  53.625000
50%     35.300000  12.000000  162.600000  19.500000  60.550000
75%     50.100000  14.750000  171.175000  22.000000  74.175000
max    183.100000  21.000000  185.400000  23.000000  86.200000

[8 rows x 22 columns]
           VISIT1DT        RANDDT       TRTSTDT      LSTDOSDT         ENDDT  \
count    168.000000    168.000000    168.000000    168.000000    168.000000
mean   19509.357143  19520.333333  19520.333333  19618.541667  19624.357143
std      172.842224    172.223037    172.223037    188.391135    187.717787
min    19182.000000  19194.000000  19194.000000  19233.000000  19237.000000
25%    19373.750000  19382.500000  19382.500000  19473.750000  19478.750000
50%    19499.500000  19510.500000  19510.500000  19588.500000  19590.000000
75%    19649.500000  19658.250000  19658.250000  19767.250000  19767.750000
max    19898.000000  19905.000000  19905.000000  20087.000000  20087.000000

         VISNUMEN      TRTDUR       TRTPN     TRTDOSE       AVGDD     ...      \
count  168.000000  168.000000  168.000000  168.000000  168.000000     ...
mean     8.994048   99.208333    1.500000   67.500000   62.801190     ...
std      2.865230   69.202026    0.501495   13.540359   10.516574     ...
min      4.000000    1.000000    1.000000   54.000000   54.000000     ...
25%      7.000000   37.000000    1.000000   54.000000   54.000000     ...
50%      9.000000   81.000000    1.500000   67.500000   54.000000     ...
75%     12.000000  181.250000    2.000000   81.000000   75.050000     ...
max     12.000000  212.000000    2.000000   81.000000   78.600000     ...

              AGE     AGEGRPN       RACEN       BMIBL      DISONSET  \
count  168.000000  168.000000  168.000000  167.000000    168.000000
```

```
mean    75.023810    2.166667    1.333333   25.205988  18152.916667
std      8.090027    0.606024    0.994997    4.204139    853.704253
min     51.000000    1.000000    1.000000   13.700000  15171.000000
25%     71.000000    2.000000    1.000000   22.650000  17693.250000
50%     77.000000    2.000000    1.000000   24.800000  18342.000000
75%     81.000000    3.000000    1.000000   27.800000  18833.000000
max     88.000000    3.000000    6.000000   40.100000  19617.000000

           DURDIS     EDUCLVL    HEIGHTBL     MMSETOT    WEIGHTBL
count  168.000000  168.000000  168.000000  168.000000  167.000000
mean    44.599405   12.839286  164.626786   18.190476   68.650299
std     27.475408    3.590290   10.315180    4.190026   14.414082
min      2.200000    3.000000  135.900000   10.000000   41.700000
25%     23.975000   12.000000  157.500000   15.000000   56.300000
50%     36.600000   12.000000  164.950000   19.000000   68.000000
75%     59.700000   16.000000  171.800000   22.000000   78.500000
max    135.000000   24.000000  195.600000   24.000000  108.000000

[8 rows x 22 columns]
```

Just like groupby function in PROC SQL, the data scientist can create the count of race using Pandas groupby and count function.

In [9]:
```
### race counts
print(adsl.groupby(['RACE']).RACE.count())
```

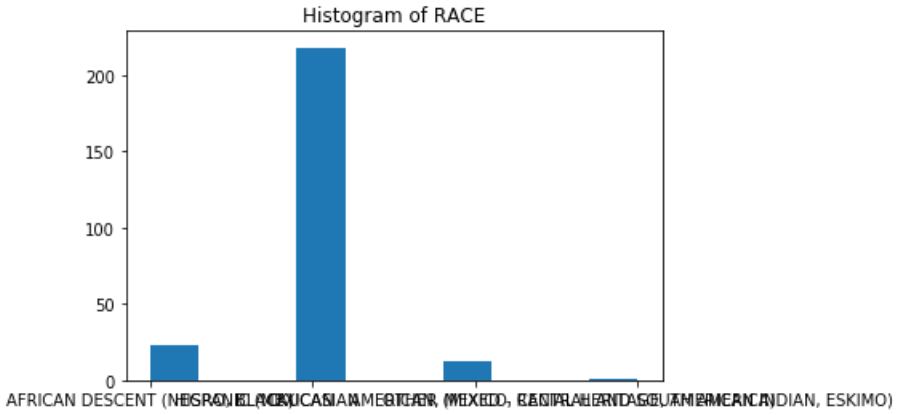
```
RACE
b'AFRICAN DESCENT (NEGRO, BLACK)'                                      23
b'CAUCASIAN'                                                          218
b'HISPANIC (MEXICAN - AMERICAN, MEXICO, CENTRAL AND SOUTH AMERICA)'    12
b'OTHER (MIXED - RACIAL HERITAGE, AMERICAN INDIAN, ESKIMO)'             1
Name: RACE, dtype: int64
```

Python Data Visualization in Jupyter Notebook - matplotlib

The data scientist can import matplotlib for simple data visualization in Jupyter notebook. Matplotlib is a Python plotting library which provides line plot, histogram, scatter plot and many more. It is one of the most popular graphing package in Jupyter notebook.
The data scientist can create the simple histogram using hist function using ADSL.RACE variables.

In [10]:
```
%matplotlib inline
### Import matplotlib for data visualization
import matplotlib
import matplotlib.pyplot as plt

### Histogram plot for race
plt.title('Histogram of RACE')
plt.hist(adsl.RACE)
```

Out[10]:
```
(array([ 23.,   0.,   0., 218.,   0.,   0.,  12.,   0.,   0.,   1.]),
 array([0. , 0.3, 0.6, 0.9, 1.2, 1.5, 1.8, 2.1, 2.4, 2.7, 3. ]),
 <a list of 10 Patch objects>)
```



For further analysis, the data scientist can calcualte the mean of AGE by each RACE group.

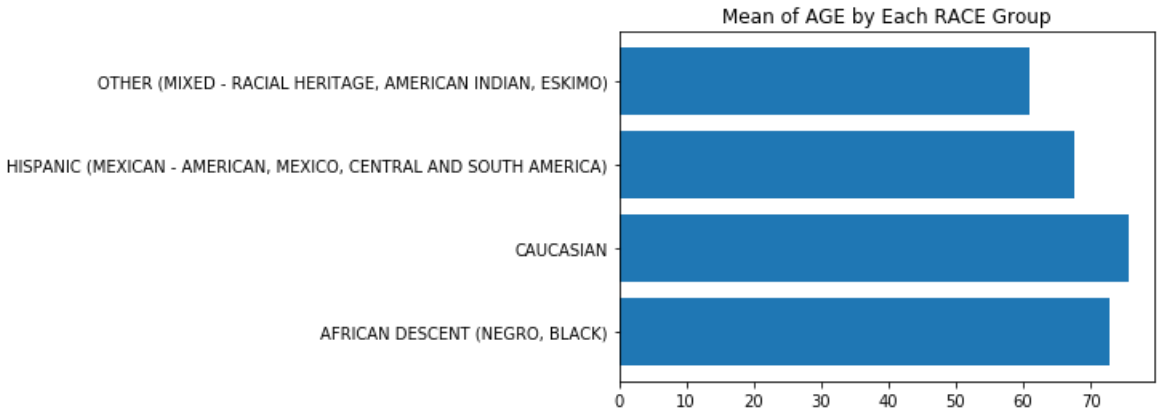
In [11]:
```
### Calculate the means of age by Race Group
mean_age = adsl.groupby(['RACE'])['AGE'].mean()
print(mean_age)
```

```
RACE
b'AFRICAN DESCENT (NEGRO, BLACK)'                                    72.869565
b'CAUCASIAN'                                                         75.793578
b'HISPANIC (MEXICAN - AMERICAN, MEXICO, CENTRAL AND SOUTH AMERICA)'  67.666667
b'OTHER (MIXED - RACIAL HERITAGE, AMERICAN INDIAN, ESKIMO)'          61.000000
Name: AGE, dtype: float64
```

Using matplotlib barh (horizontal bar) function, the data scientist can create horizontal bar graph of the mean of AGE by each RACE group.

In [12]:
```
### Import matplotlib for data visualization
import matplotlib
import matplotlib.pyplot as plt

### Horizontal bar the means of age by Race Group
plt.title('Mean of AGE by Each RACE Group')
plt.barh( mean_age.index.values, mean_age)
```

Out[12]:
```
<Container object of 4 artists>
```



Kaplan Meier Curves creation using lifelines package

The data Scientist can also create Kaplan Curves using ADTTEOS datasets. First, data scientist need to import Time to Event ADaM datasets from the local drive.

In [13]:
```
#import Time to Event ADaM datsets
adtteos = pd.read_sas('./data/SAS/ADaM/adtteos.sas7bdat')
adtteos.head()
```

Out[13]:

	SUBJID	SITEID	FASFL	SAFFL	TRTP	TRTPN	PARAM	PARAMCD	PARAMTYP	AVISIT	AVISITN	AVAL	STARTDT	A
0	b'310-001'	b'310'	b'Y'	b'Y'	b'Control'	1.0	b'Days to Death'	b'DEATH'	b'DERIVED'	b'Double-Blind Period'	91.0	1.0	2007-11-12	20(11-
1	b'310-002'	b'310'	b'Y'	b'Y'	b'Control'	1.0	b'Days to Death'	b'DEATH'	b'DERIVED'	b'Double-Blind Period'	91.0	83.0	2008-01-11	20(04-
2	b'310-003'	b'310'	b'Y'	b'Y'	b'Study Drug'	2.0	b'Days to Death'	b'DEATH'	b'DERIVED'	b'Double-Blind Period'	91.0	84.0	2008-02-01	20(04-
3	b'310-004'	b'310'	b'Y'	b'Y'	b'Study Drug'	2.0	b'Days to Death'	b'DEATH'	b'DERIVED'	b'Double-Blind Period'	91.0	9.0	2008-02-07	20(02-
4	b'310-005'	b'310'	b'Y'	b'Y'	b'Study Drug'	2.0	b'Days to Death'	b'DEATH'	b'DERIVED'	b'Double-Blind Period'	91.0	51.0	2008-02-21	20(04-

The data scientist can divide time to event ADaM dataset into "control" group and "study drug" group adam datasets.

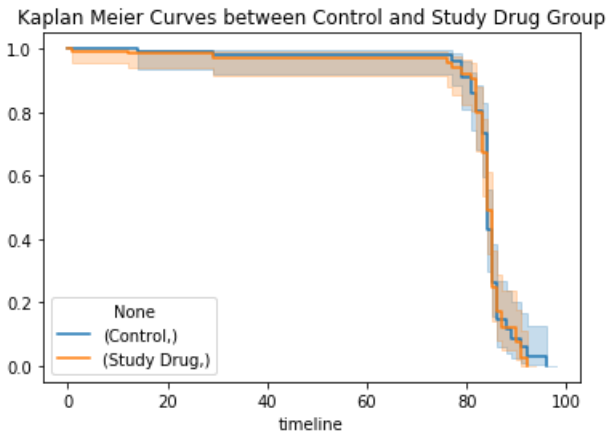
In [14]:
```python
### ADTTEOS with Placebo
adtteos_c = adtteos[adtteos.TRTP == b'Control']

### ADTTEOS with Study Drug
adtteos_sd = adtteos[adtteos.TRTP == b'Study Drug']

print(adtteos_c.head())
print(adtteos_sd.head())
```

```
        SUBJID  SITEID FASFL SAFFL        TRTP  TRTPN             PARAM  \
0   b'310-001'  b'310'  b'Y'  b'Y'  b'Control'    1.0  b'Days to Death'
1   b'310-002'  b'310'  b'Y'  b'Y'  b'Control'    1.0  b'Days to Death'
5   b'310-006'  b'310'  b'Y'  b'Y'  b'Control'    1.0  b'Days to Death'
7   b'310-008'  b'310'  b'Y'  b'Y'  b'Control'    1.0  b'Days to Death'
10  b'310-011'  b'310'  b'Y'  b'Y'  b'Control'    1.0  b'Days to Death'

     PARAMCD    PARAMTYP                 AVISIT  AVISITN  AVAL    STARTDT  \
0   b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0   1.0 2007-11-12
1   b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0  83.0 2008-01-11
5   b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0  62.0 2008-03-10
7   b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0   3.0 2008-07-11
10  b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0  85.0 2009-02-24

          ADT ADTF  CNSR                          EVNTDESC
0  2007-11-12  NaN   0.0                          b'DEATH'
1  2008-04-02  NaN   1.0  b'COMPLETE PERIOD WITHOUT EVENT'
5  2008-05-10  NaN   0.0                          b'DEATH'
7  2008-07-13  NaN   0.0                          b'DEATH'
10 2009-05-19  NaN   1.0  b'COMPLETE PERIOD WITHOUT EVENT'
       SUBJID  SITEID FASFL SAFFL           TRTP  TRTPN             PARAM  \
2  b'310-003'  b'310'  b'Y'  b'Y'  b'Study Drug'    2.0  b'Days to Death'
3  b'310-004'  b'310'  b'Y'  b'Y'  b'Study Drug'    2.0  b'Days to Death'
4  b'310-005'  b'310'  b'Y'  b'Y'  b'Study Drug'    2.0  b'Days to Death'
6  b'310-007'  b'310'  b'Y'  b'Y'  b'Study Drug'    2.0  b'Days to Death'
8  b'310-009'  b'310'  b'Y'  b'Y'  b'Study Drug'    2.0  b'Days to Death'

    PARAMCD    PARAMTYP                 AVISIT  AVISITN  AVAL    STARTDT  \
2  b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0  84.0 2008-02-01
3  b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0   9.0 2008-02-07
4  b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0  51.0 2008-02-21
6  b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0  69.0 2008-04-28
8  b'DEATH'  b'DERIVED'  b'Double-Blind Period'     91.0   5.0 2008-07-30

         ADT ADTF  CNSR  EVNTDESC
2 2008-04-24  NaN   0.0  b'DEATH'
3 2008-02-15  NaN   0.0  b'DEATH'
4 2008-04-11  NaN   0.0  b'DEATH'
6 2008-07-05  NaN   0.0  b'DEATH'
8 2008-08-03  NaN   0.0  b'DEATH'
```

The data scientist need to import lifeline package for Kaplan Meier Curve plotting.

In [15]:
```
### Import lifelines and Kaplan Meier Curves
from lifelines.estimation import KaplanMeierFitter
kmf = KaplanMeierFitter()

### Prepare Kaplan Meier Curves
ax = plt.subplot(111)

kmf.fit(adtteos_c.AVAL, event_observed=adtteos_c.CNSR, label=['Control'])
kmf.plot(ax=ax)
kmf.fit(adtteos_sd.AVAL, event_observed=adtteos_sd.CNSR, label=['Study Drug'])
kmf.plot(ax=ax)

plt.title('Kaplan Meier Curves between Control and Study Drug Group')
```

Out[15]:
```
Text(0.5,1,'Kaplan Meier Curves between Control and Study Drug Group')
```



MNIST Dataset Machine Learning Binary Classification

Jupyter notebook is very popular for Machine Learning Project. Below, we will show how data scientis can conduct Machine Learning project in Jupyter notebook. The followings are the typical steps of Machine Learning project.

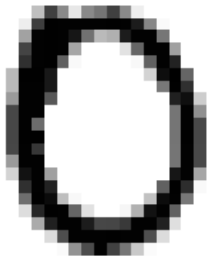
- Obtain the data
- Visualize and reveiw the data
- Preprocess the data
- Prepare train and test data
- Pick the algorithm
- Train the model
- Predict with the trained model

In [16]:
```
### import MNST datasets using sklearn API
from sklearn.datasets import fetch_mldata
mnist = fetch_mldata('MNIST original')
mnist
```

Out[16]:
```
{'COL_NAMES': ['label', 'data'],
 'DESCR': 'mldata.org dataset: mnist-original',
 'data': array([[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]], dtype=uint8),
 'target': array([0., 0., 0., ..., 9., 9., 9.])}
```

In [17]:
```python
### prepare X and Y dataset
X, y = mnist["data"], mnist["target"]
print('Shape of X: ', X.shape)
print('Shape of y: ', y.shape, 'value of y: ', y[1200])

### Pick one image and reshape to 28 by 28 for image
digit_image1 = X[1200].reshape(28, 28)

### Show that image
plt.imshow(digit_image1, cmap = matplotlib.cm.binary,
           interpolation="nearest")
plt.axis("off") ## remove the outside frame
plt.show() # show plot
```

```
Shape of X:  (70000, 784)
Shape of y:  (70000,) value of y:  0.0
```



The digit_image1 is actually 64 by 64 array of number. As seen here, its number display shows zero-like shape. Also note that the higher the number, the darker image in the display.

```
[[  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  64 191 118  24 118 138 191 118  17   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  17 193 253 252 252 252 252 253 252 227 120   5   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0  93 252 253 252 233 141  69  79 227 252 252 137   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0 187 252 253 252  89   0   0   0  29 154 252 252  95   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0 207 252 253 157   6   0   0   0   0   7 158 252 220   0   0   0   0   0   0]
 [  0   0   0   0   0   0  70 253 253 231   0   0   0   0   0   0   0  43 241 255 154   0   0   0   0   0]
 [  0   0   0   0   0   0  70 252 252 230   0   0   0   0   0   0   0   0 135 253 206   0   0   0   0   0]
 [  0   0   0   0   0   0 101 252 252 167   0   0   0   0   0   0   0   0   0 222 240  50   0   0   0   0]
 [  0   0   0   0   0   0 184 252 252   0   0   0   0   0   0   0   0   0   0 138 252 111   0   0   0   0]
 [  0   0   0   0   0   0 184 252 147   0   0   0   0   0   0   0   0   0   0 138 252 183   0   0   0   0]
 [  0   0   0   0   0   0 185 253 253   0   0   0   0   0   0   0   0   0   0 139 253 184   0   0   0   0]
 [  0   0   0   0   0   0 184 252 200   0   0   0   0   0   0   0   0   0   0 138 252 183   0   0   0   0]
 [  0   0   0   0   0   0 101 252 252   0   0   0   0   0   0   0   0   0   0 138 252 183   0   0   0   0]
 [  0   0   0   0   0   0  44 236 252  95   0   0   0   0   0   0   0   0   0 233 252  89   0   0   0   0]
 [  0   0   0   0   0   0   0 207 252 220   0   0   0   0   0   0   0   0   0 158 253 210   6   0   0   0]
 [  0   0   0   0   0   0   0 145 253 244  65   7   0   0   0   0   0  62 243 252 135   0   0   0   0   0]
 [  0   0   0   0   0   0   0  34 217 253 252 154  30   0   0   9  78 236 252 157   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0  84 222 252 252 227 184 185 197 252 252 221  32   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0  36 177 252 252 252 253 252 252 176  35   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   5 107 179 252 190 137  54   4   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]
 [  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0]]
```

The data scientist can prepare test and train data. And, the data scientist will pick the the alrgorithm and train the model with the trained data. Here, we pick the simple SGDClassifier, but the data scientist can pick other model like Support Vector Machine, Deep Neural Network and Convolutional Neural Network.

In [18]:
```
### Prepare test and train data
from sklearn.model_selection import train_test_split
X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.2, random_state=42)

### import the algorithm
from sklearn.linear_model import SGDClassifier

### Train the model
sgd_clf2 = SGDClassifier(max_iter=5, random_state=42)
sgd_clf2.fit(X_train2, y_train2)
```

Out[18]:
```
SGDClassifier(alpha=0.0001, average=False, class_weight=None, epsilon=0.1,
       eta0=0.0, fit_intercept=True, l1_ratio=0.15,
       learning_rate='optimal', loss='hinge', max_iter=5, n_iter=None,
       n_jobs=1, penalty='l2', power_t=0.5, random_state=42, shuffle=True,
       tol=None, verbose=0, warm_start=False)
```

The data scientis can validate the accuracy of the trained model. Here is the average of accuracy is aboutt 87%, which is not that bad. More advanced model like CNN can provide better accuracy.

In [19]:
```
### Validate the accuracy
from sklearn.model_selection import cross_val_score
cross_val_score(sgd_clf2, X_train2, y_train2, cv=3, scoring="accuracy")  ### With train data
cross_val_score(sgd_clf2, X_test2, y_test2, cv=3, scoring="accuracy")  ### With test data
```

Out[19]:
```
array([0.85628614, 0.88303342, 0.85545786])
```

With the trained model, the data scientist can predit the result of the test data.

In [20]:
```
print('predicted y value: ', sgd_clf2.predict([X_test2[111]]))
print('actual y value: ', y_test2[111])
```

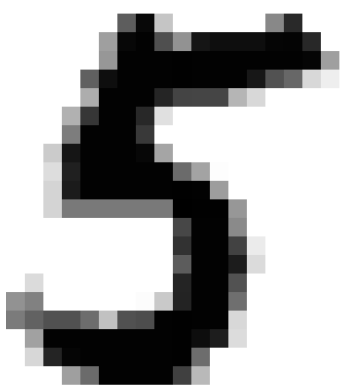
```
predicted y value:  [7.]
actual y value:  7.0
```

Import image file

The scientist can import png file into Jupyter notebook as a number

In [21]:
```
Image(filename='./images/mnist_input_5.png', width=300)
```

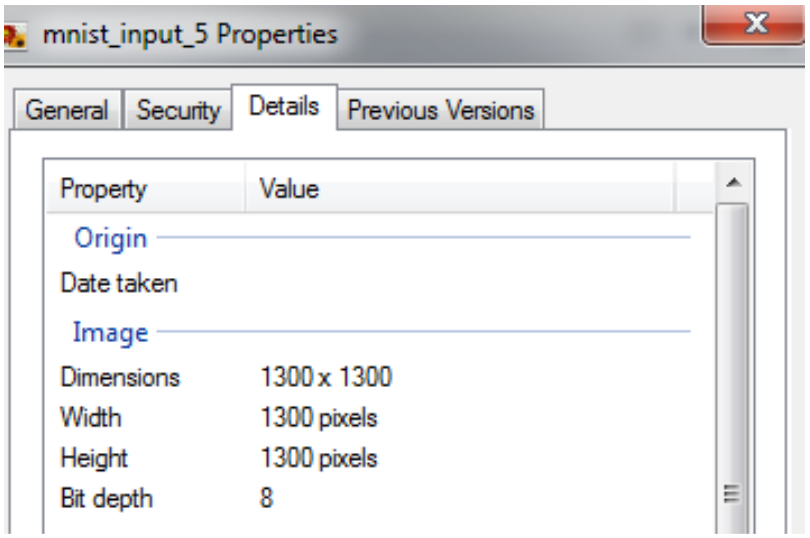
Out[21]:



The property of image 5 is 1300 X 1300 pixels. When it is uploaded into Jupyter notebook, it is represented by number in array of 1300 and 1300. Its path is "C:\zother\paper\43 Telling Stories with Jupyter Notebook\codes\images"

In [22]:
```
Image(filename='./images/mnist_input_5_property.png', width=500)
```

Out[22]:



The data scientist can import skimage package and use imread function to read the external png image into data. The data will be represted in number. image 5 will be 1300 X 1300 array just like its pixel size of a image in local drive.

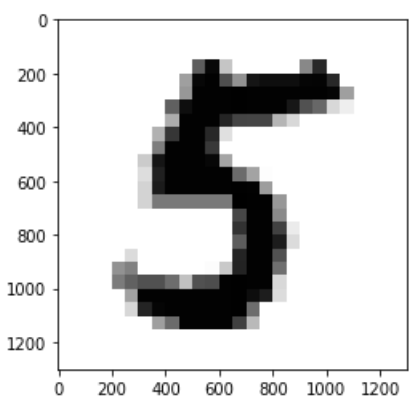
In [23]:
```
### import imread function
from skimage.io import imread

image_5 = imread(fname='./images/mnist_input_5.png')
print('Shape of Image 5 : ', image_5.shape)
image_5
```

```
Shape of Image 5 :  (1300, 1300)
```

Out[23]:
```
array([[255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       ...,
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255],
       [255, 255, 255, ..., 255, 255, 255]], dtype=uint8)
```

The data scientist can display data in image using imshow function.

In [24]:
```
### Dispaly image_5(in number) to picture 5 in Jupyter notebook
plt.imshow(image_5, cmap="gray")
plt.show()
```



Basically, here is the process of this project.

1. Read image file from local drive using imread.
2. Convert image file to data in number.

3. Display data in number as image using imshow.

CONCLUSION

The role of a data scientist has changed from complex coding to presenting the solution. The best way to present the findings and results is telling stories with data, contents and visualization. Jupyter notebook starts from our school day notebook, containing notes, pictures and drawings as well as problems and solutions. Jupyter notebook provides the data scientist more weapons to share our ideas with the audience. The data scientist can tell more compelling stories with Jupyter Notebook.

GITHUB LINK

https://github.com/kevinlee1004/telling_stories_with_Jupyter_notebook (https://github.com/kevinlee1004/telling_stories_with_Jupyter_notebook)

REFERENCES

http://jupyter.org/ (http://jupyter.org/) Jupyter notebook home page
https://www.python.org/ (https://www.python.org/) Python home page
https://matplotlib.org/ (https://matplotlib.org/) Python matplotlib home page
Python for Data Analysis

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:
Kevin Lee
Director of Data Science at Clindata Insight
Klee@clindatainsight.com
Tweet: @HelloKevinLee
LinkedIn: www.linkedin.com/in/HelloKevinLee/

In [ ]: