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 – Jupyter 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]:

![Image](./images/02_01.png)
```

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

```
Out[3]:

![PharmaSUG Seattle 2018](./images/PharmaSUG_2018.png)
```

**Import video from local drive**

The data scientist can also import video in Jupyter Notebook.
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

```python
import pandas as pd
import numpy as np
```

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

In [5]:
### Read ADSL datasets

```python
adsl = pd.read_sas('./data/SAS/ADaM/adsl.xpt')
adsl.head()
```

Out[5]:

<table>
<thead>
<tr>
<th>USUBJID</th>
<th>STUDYID</th>
<th>DOMAIN</th>
<th>SITEID</th>
<th>SITEGRP</th>
<th>SUBJID</th>
<th>VISIT1DT</th>
<th>RANDDT</th>
<th>TRTSTDT</th>
<th>RFSTDTC</th>
<th>...</th>
<th>EDUCLVL</th>
<th>HEI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0  b'01-701-1015'</td>
<td>b'CDISCIPL01'</td>
<td>b'ADSL'</td>
<td>b'701'</td>
<td>b'1015'</td>
<td>19718.0</td>
<td>19725.0</td>
<td>19725.0</td>
<td>b'2014-01-02'</td>
<td>...</td>
<td>16.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1  b'01-701-1023'</td>
<td>b'CDISCIPL01'</td>
<td>b'ADSL'</td>
<td>b'701'</td>
<td>b'1023'</td>
<td>19196.0</td>
<td>19210.0</td>
<td>19210.0</td>
<td>b'2012-08-05'</td>
<td>...</td>
<td>14.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2  b'01-701-1028'</td>
<td>b'CDISCIPL01'</td>
<td>b'ADSL'</td>
<td>b'701'</td>
<td>b'1028'</td>
<td>19550.0</td>
<td>19558.0</td>
<td>19558.0</td>
<td>b'2012-08-05'</td>
<td>...</td>
<td>16.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3  b'01-701-1033'</td>
<td>b'CDISCIPL01'</td>
<td>b'ADSL'</td>
<td>b'701'</td>
<td>b'1033'</td>
<td>19792.0</td>
<td>19800.0</td>
<td>19800.0</td>
<td>b'2014-03-18'</td>
<td>...</td>
<td>12.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4  b'01-701-1034'</td>
<td>b'CDISCIPL01'</td>
<td>b'ADSL'</td>
<td>b'701'</td>
<td>b'1034'</td>
<td>19898.0</td>
<td>19905.0</td>
<td>19905.0</td>
<td>b'2014-07-01'</td>
<td>...</td>
<td>9.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5 rows × 51 columns

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

In [6]:
### quick review of ADSL datasets

```python
adsl.describe()
```

Out[6]:

<table>
<thead>
<tr>
<th>VISIT1DT</th>
<th>RANDDT</th>
<th>TRTSTDT</th>
<th>LSTDOSDT</th>
<th>ENDDT</th>
<th>VISNUMEN</th>
<th>TRTDUR</th>
<th>TRTPN</th>
<th>TRTDOS</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>254.000000</td>
<td>254.000000</td>
<td>254.000000</td>
<td>254.000000</td>
<td>254.000000</td>
<td>254.000000</td>
<td>2.540000e+02</td>
<td>2.540000e+02</td>
</tr>
<tr>
<td>mean</td>
<td>19515.519685</td>
<td>19526.519685</td>
<td>19526.519685</td>
<td>19641.610236</td>
<td>19646.602362</td>
<td>9.582677</td>
<td>116.090551</td>
<td>9.921260e+00</td>
</tr>
<tr>
<td>std</td>
<td>177.503743</td>
<td>177.125058</td>
<td>177.125058</td>
<td>192.487651</td>
<td>191.253746</td>
<td>2.828262</td>
<td>70.295506</td>
<td>8.1964567e+00</td>
</tr>
<tr>
<td>min</td>
<td>19180.000000</td>
<td>19183.000000</td>
<td>19183.000000</td>
<td>19233.000000</td>
<td>19237.000000</td>
<td>4.000000</td>
<td>1.000000</td>
<td>5.397605e-79</td>
</tr>
<tr>
<td>25%</td>
<td>19375.500000</td>
<td>19384.250000</td>
<td>19384.250000</td>
<td>19490.250000</td>
<td>19494.000000</td>
<td>7.250000</td>
<td>46.250000</td>
<td>5.397605e-79</td>
</tr>
<tr>
<td>50%</td>
<td>19511.500000</td>
<td>19522.500000</td>
<td>19522.500000</td>
<td>19628.000000</td>
<td>19631.500000</td>
<td>11.000000</td>
<td>133.500000</td>
<td>5.397605e-79</td>
</tr>
<tr>
<td>75%</td>
<td>19655.000000</td>
<td>19669.250000</td>
<td>19669.250000</td>
<td>19797.500000</td>
<td>19802.250000</td>
<td>12.000000</td>
<td>183.000000</td>
<td>5.397605e-79</td>
</tr>
<tr>
<td>max</td>
<td>19964.000000</td>
<td>19968.000000</td>
<td>19968.000000</td>
<td>20152.000000</td>
<td>20152.000000</td>
<td>12.000000</td>
<td>212.000000</td>
<td>5.397605e-79</td>
</tr>
</tbody>
</table>

8 rows × 22 columns

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

In [ ]: 
### List of variables

```python
list(adsl)
```
In [8]:

```python
### ADSL with Placebo
ads12 = ads1[ads1.TRTP == b'Placebo']

### ADSL with Study Drug
ads13 = ads1[ads1.TRTP ! = b'Placebo']

ads12.describe()
ads13.describe()
```

```
<table>
<thead>
<tr>
<th>VISIT1DT</th>
<th>RANDDT</th>
<th>TRTSTDT</th>
<th>LSTDOSDT</th>
<th>ENDDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>86.000000</td>
<td>86.000000</td>
<td>86.000000</td>
<td>86.000000</td>
</tr>
<tr>
<td>mean</td>
<td>19527.558140</td>
<td>19538.604651</td>
<td>19538.604651</td>
<td>19686.674419</td>
</tr>
<tr>
<td>std</td>
<td>186.722292</td>
<td>186.771410</td>
<td>186.771410</td>
<td>193.510750</td>
</tr>
<tr>
<td>min</td>
<td>19180.000000</td>
<td>19183.000000</td>
<td>19183.000000</td>
<td>19237.000000</td>
</tr>
<tr>
<td>25%</td>
<td>19380.750000</td>
<td>19393.500000</td>
<td>19393.500000</td>
<td>19544.250000</td>
</tr>
<tr>
<td>50%</td>
<td>19543.500000</td>
<td>19551.000000</td>
<td>19551.000000</td>
<td>19684.000000</td>
</tr>
<tr>
<td>75%</td>
<td>19684.500000</td>
<td>19698.500000</td>
<td>19698.500000</td>
<td>19811.500000</td>
</tr>
<tr>
<td>max</td>
<td>19964.000000</td>
<td>19968.000000</td>
<td>19968.000000</td>
<td>20152.000000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VISNUMEN</th>
<th>TRTDUR</th>
<th>TRTPN</th>
<th>TRTDOSE</th>
<th>AVGDD</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>86.000000</td>
<td>86.000000</td>
<td>86.000000</td>
<td>86.000000</td>
</tr>
<tr>
<td>mean</td>
<td>10.732558</td>
<td>149.069767</td>
<td>5.397605e-79</td>
<td>5.397605e-79</td>
</tr>
<tr>
<td>std</td>
<td>2.378454</td>
<td>60.295506</td>
<td>0.000000e+00</td>
<td>0.000000e+00</td>
</tr>
<tr>
<td>min</td>
<td>4.000000</td>
<td>7.000000</td>
<td>5.397605e-79</td>
<td>5.397605e-79</td>
</tr>
<tr>
<td>25%</td>
<td>11.000000</td>
<td>131.000000</td>
<td>5.397605e-79</td>
<td>5.397605e-79</td>
</tr>
<tr>
<td>50%</td>
<td>12.000000</td>
<td>182.000000</td>
<td>5.397605e-79</td>
<td>5.397605e-79</td>
</tr>
<tr>
<td>75%</td>
<td>12.000000</td>
<td>183.000000</td>
<td>5.397605e-79</td>
<td>5.397605e-79</td>
</tr>
<tr>
<td>max</td>
<td>12.000000</td>
<td>210.000000</td>
<td>5.397605e-79</td>
<td>5.397605e-79</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>AGE</th>
<th>AGEGRPN</th>
<th>RACEN</th>
<th>BMIBL</th>
<th>DISONSET</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>86.000000</td>
<td>86.000000</td>
<td>86.000000</td>
<td>86.000000</td>
</tr>
<tr>
<td>mean</td>
<td>75.209302</td>
<td>2.186047</td>
<td>1.232558</td>
<td>23.636047</td>
</tr>
<tr>
<td>std</td>
<td>8.590167</td>
<td>0.694713</td>
<td>0.777241</td>
<td>3.671926</td>
</tr>
<tr>
<td>min</td>
<td>52.000000</td>
<td>1.000000</td>
<td>1.000000</td>
<td>15.100000</td>
</tr>
<tr>
<td>25%</td>
<td>69.250000</td>
<td>2.000000</td>
<td>1.000000</td>
<td>21.200000</td>
</tr>
<tr>
<td>50%</td>
<td>76.000000</td>
<td>2.000000</td>
<td>1.000000</td>
<td>23.400000</td>
</tr>
<tr>
<td>75%</td>
<td>81.750000</td>
<td>3.000000</td>
<td>1.000000</td>
<td>25.600000</td>
</tr>
<tr>
<td>max</td>
<td>89.000000</td>
<td>5.000000</td>
<td>5.000000</td>
<td>33.300000</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DURDIS</th>
<th>EDUCLVL</th>
<th>HEIGHTBL</th>
<th>MMSETOT</th>
<th>WEIGHTBL</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>86.000000</td>
<td>86.000000</td>
<td>86.000000</td>
<td>86.000000</td>
</tr>
<tr>
<td>mean</td>
<td>42.650000</td>
<td>12.581395</td>
<td>162.573256</td>
<td>18.046512</td>
</tr>
<tr>
<td>std</td>
<td>30.241572</td>
<td>2.948440</td>
<td>11.522361</td>
<td>4.272778</td>
</tr>
<tr>
<td>min</td>
<td>7.200000</td>
<td>6.000000</td>
<td>137.200000</td>
<td>10.000000</td>
</tr>
<tr>
<td>25%</td>
<td>24.325000</td>
<td>12.000000</td>
<td>154.000000</td>
<td>15.000000</td>
</tr>
<tr>
<td>50%</td>
<td>35.300000</td>
<td>12.000000</td>
<td>162.600000</td>
<td>19.500000</td>
</tr>
<tr>
<td>75%</td>
<td>50.100000</td>
<td>14.750000</td>
<td>171.175000</td>
<td>22.000000</td>
</tr>
<tr>
<td>max</td>
<td>183.100000</td>
<td>21.000000</td>
<td>185.400000</td>
<td>23.000000</td>
</tr>
</tbody>
</table>
```

[8 rows x 22 columns]

```
<table>
<thead>
<tr>
<th>VISIT1DT</th>
<th>RANDDT</th>
<th>TRTSTDT</th>
<th>LSTDOSDT</th>
<th>ENDDT</th>
</tr>
</thead>
<tbody>
<tr>
<td>count</td>
<td>168.000000</td>
<td>168.000000</td>
<td>168.000000</td>
<td>168.000000</td>
</tr>
<tr>
<td>mean</td>
<td>19509.357143</td>
<td>19520.333333</td>
<td>19520.333333</td>
<td>19624.357143</td>
</tr>
<tr>
<td>std</td>
<td>172.842224</td>
<td>172.223037</td>
<td>172.223037</td>
<td>187.717787</td>
</tr>
<tr>
<td>min</td>
<td>19182.000000</td>
<td>19194.000000</td>
<td>19194.000000</td>
<td>19237.000000</td>
</tr>
<tr>
<td>25%</td>
<td>19373.750000</td>
<td>19382.500000</td>
<td>19382.500000</td>
<td>19473.750000</td>
</tr>
<tr>
<td>50%</td>
<td>20350.000000</td>
<td>20483.000000</td>
<td>20483.000000</td>
<td>20588.000000</td>
</tr>
<tr>
<td>75%</td>
<td>21325.000000</td>
<td>21458.000000</td>
<td>21458.000000</td>
<td>21561.500000</td>
</tr>
<tr>
<td>max</td>
<td>22300.000000</td>
<td>22433.000000</td>
<td>22433.000000</td>
<td>22556.000000</td>
</tr>
</tbody>
</table>
```

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

### race counts

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

```python
import matplotlib.pyplot as plt

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

```text
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 calculate the mean of AGE by each RACE group.

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

```text
RACE                       mean
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.
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.

### Import Time to Event ADaM datasets
```python
adtteos = pd.read_sas('./data/SAS/ADaM/adtteos.sas7bdat')
adtteos.head()
```

<table>
<thead>
<tr>
<th>SUBJ</th>
<th>SITEID</th>
<th>FASFL</th>
<th>SAFSL</th>
<th>TRTP</th>
<th>TRTPN</th>
<th>PARAM</th>
<th>PARAMCD</th>
<th>PARAMTYP</th>
<th>AVISIT</th>
<th>AVISITN</th>
<th>AVAL</th>
<th>STARTDT</th>
<th>AI</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>b'310-001'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Control'</td>
<td>1.0</td>
<td>b'Days to Death'</td>
<td>b'DEATH'</td>
<td>b'DERIVED'</td>
<td>91.0</td>
<td>1.0</td>
<td>2007-11-12</td>
<td>20C 11-</td>
</tr>
<tr>
<td>1</td>
<td>b'310-002'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Control'</td>
<td>1.0</td>
<td>b'Days to Death'</td>
<td>b'DEATH'</td>
<td>b'DERIVED'</td>
<td>91.0</td>
<td>83.0</td>
<td>2008-01-11</td>
<td>20C 04-</td>
</tr>
<tr>
<td>2</td>
<td>b'310-003'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Study Drug'</td>
<td>2.0</td>
<td>b'Days to Death'</td>
<td>b'DEATH'</td>
<td>b'DERIVED'</td>
<td>91.0</td>
<td>84.0</td>
<td>2008-02-01</td>
<td>20C 04-</td>
</tr>
<tr>
<td>3</td>
<td>b'310-004'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Study Drug'</td>
<td>2.0</td>
<td>b'Days to Death'</td>
<td>b'DEATH'</td>
<td>b'DERIVED'</td>
<td>91.0</td>
<td>9.0</td>
<td>2008-02-07</td>
<td>20C 02-</td>
</tr>
<tr>
<td>4</td>
<td>b'310-005'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Study Drug'</td>
<td>2.0</td>
<td>b'Days to Death'</td>
<td>b'DEATH'</td>
<td>b'DERIVED'</td>
<td>91.0</td>
<td>51.0</td>
<td>2008-02-21</td>
<td>20C 04-</td>
</tr>
</tbody>
</table>

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())
```

<table>
<thead>
<tr>
<th>SUBJID</th>
<th>SITEID</th>
<th>FASFL</th>
<th>SAFFL</th>
<th>TRTP</th>
<th>TRTPN</th>
<th>PARAM</th>
<th>ADT</th>
<th>ADTF</th>
<th>CNSR</th>
<th>EVNTDESC</th>
</tr>
</thead>
<tbody>
<tr>
<td>b'310-001'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Control'</td>
<td>1.0</td>
<td>b'Days to Death'</td>
<td>2007-11-12</td>
<td>NaN</td>
<td>0.0</td>
<td>b'DEATH'</td>
</tr>
<tr>
<td>b'310-002'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Control'</td>
<td>1.0</td>
<td>b'Days to Death'</td>
<td>2008-02-02</td>
<td>NaN</td>
<td>0.0</td>
<td>b'COMPLETE PERIOD WITHOUT EVENT'</td>
</tr>
<tr>
<td>b'310-004'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Study Drug'</td>
<td>2.0</td>
<td>b'Days to Death'</td>
<td>2008-02-15</td>
<td>NaN</td>
<td>0.0</td>
<td>b'DEATH'</td>
</tr>
<tr>
<td>b'310-008'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Study Drug'</td>
<td>2.0</td>
<td>b'Days to Death'</td>
<td>2008-04-11</td>
<td>NaN</td>
<td>0.0</td>
<td>b'DEATH'</td>
</tr>
<tr>
<td>b'310-011'</td>
<td>b'310'</td>
<td>b'Y'</td>
<td>b'Y'</td>
<td>b'Study Drug'</td>
<td>2.0</td>
<td>b'Days to Death'</td>
<td>2008-07-05</td>
<td>NaN</td>
<td>0.0</td>
<td>b'DEATH'</td>
</tr>
</tbody>
</table>

The data scientist need to import lifeline package for Kaplan Meier Curve plotting.
In [15]:

```python
### 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')

### Import MNST datasets using sklearn API
```python
from sklearn.datasets import fetch_mldata
mnist = fetch_mldata('MNIST original')
```

Out[16]:

```python
{'COL_NAMES': ['label', 'data'],
'DESCRIPT': '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]], dtype='uint8'),
'target': array([0., 0., 0., ..., 9., 9., 9.])}
```
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.

The data scientist can prepare test and train data. And, the data scientist will pick the the algorithm 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.
The data scientist can validate the accuracy of the trained model. Here is the average of accuracy is about 87%, which is not that bad. More advanced model like CNN can provide better accuracy.

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

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:\other\paper\43 Telling Stories with Jupyter Notebook\codes\images"
The data scientist can import skimage package and use imread function to read the external png image into data. The data will be represented in number. Image 5 will be 1300 X 1300 array just like its pixel size of a image in local drive.

In [23]:
```python
### 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)
```
```
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]:
```python
### Display 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

REFERENCES

http://jupyter.org/ Jupyter notebook home page
https://www.python.org/ Python home page
https://matplotlib.org/ Python matplotlib home page

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/