# INFO 208 Quiz #2 Lab Exploration Component

Your Name: \_\_Megha Rajam Rao\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Note: Solutions and Screenshots from Page 2\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The following Lab items should be carried out by you using your regular Operating System (Windows, macOS, or perhaps Linux) and a browser such as Chrome, Firefox, or Safari.

We will **not** be using your **Hortonworks HDP Sandbox** VM for this Quiz. But, remember, you were asked to keep your HDP Sandbox VM for later use — both in this course and for continuing education beyond this course. You will probably be using the Hortonworks HDP Sandbox for the Final Exam Practical

The purpose of this **Lab Exploration Component of Quiz #2** is to give you more practice with Python 3 & Jupyter and then to ***introduce*** you to the Julia language — remember that this is one of the languages as the basis for Jupyter.

**Open up the area after each question / section** in this MS Word file and type in (or paste) your answer(s) and your comments. Mail the resulting file to me at [glen.mules@sjsu.edu](mailto:glen.mules@sjsu.edu)

Your email **Subject** line should be:

**INFO 208 Quiz #2**

And the attached MS Word file sent by email should be named:

**INFO\_208\_Quiz2-*YourLastName*.docx**

The file naming is important as it causes my email software to sort your *Quiz Response Email* into a directory for me so that I don’t lose amongst my other daily email.

You will be able to upload your Practical / Lab Exploration Component into Quiz #2 on Canvas as the last item, if you have time.

**First Lab.**  **Using an existing Python 3 Jupyter Notebook.** You are not expected to be an expert in Python 3, but, hopefully, you have an introductory background.

**Steps:**

\_\_1. Go to the Gallery of interesting Jupyter Notebooks at <https://github.com/jupyter/jupyter/wiki/A-gallery-of-interesting-Jupyter-Notebooks#programming-and-computer-science> and select **two notebooks** which ***do not have*** “Introductory” in their title or description. Your choices should be beyond the mere introductory level.

My suggestion is that you select your notebooks from an area of your own personal experience, your undergraduate major or minor, or your work/business background, e.g.: Programming and Computer Science, Statistics, Machine Learning and Data Science, Mathematics, Physics, Chemistry, Biology, Earth Science and Geo-Spatial data, Social data, Psychology and Neuroscience, Machine Learning, Statistics and Probability, Physics, Chemistry and Biology, Economics and Finance List here:

\_\_2. Work your way through these two Python-based Notebooks and cut-and-paste some snippets here to illustrate that you worked them and achieved completion of the work in the notebooks themselves.

\_\_3. Comment here on **what** you learned both about the topic chosen and the Jupyter Notebook approach to documenting work (***at least a half dozen sentences***):

**Second Lab.** No prior knowledge of programming in Julia is expected. The intent is to introduce you to the Julia language. Some features of Julia:

* Language syntax is similar to Python.
* But whereas Python is interpreted, Julia is compiled down to CPU executable statements that run often as fast as C / C++.
* “Looks like Python, feels like Lisp, runs like C”

**Steps for the Second Lab:**

\_\_1. Do a Google search on the topic “Julia Language” and read several of the articles including the Wikipedia article. List here which articles (atleast 3) you read:

\_\_2. Look through the accompanying file **Intro\_Julia\_tutorial\_slides\_ODSC\_West.pdf** to get an overview of the main reasons for another computer language and why Julia could be that language for you.

\_\_3. Bookmark the following pages for use as references:

* Introducing Julia  
  https://en.wikibooks.org/wiki/Introducing\_Julia
* Introducing Julia/print  
  <https://en.wikibooks.org/wiki/Introducing_Julia/print#Introduction>

\_\_4. For this lab you will be working with Julia notebooks *remotely* using a browser. You do **not** need to install Julia on your local system (**nor** in your Sandbox VM).

Create an account at JuliaBox (<https://juliabox.com/>) using your sjsu.edu email account and then work your way through *several* of the “Getting started” Julia Notebooks: e.g., 00, 01, …, 09, …

\_\_5. Document here your work (using cut & paste or similar): e.g., 😺 + 😞 == 😀  
 *Expand this space as necessary*.

\_\_6. Comment here on **what** you learned both about the Julia language and the Jupyter Notebook approach to working with Julia (***at least a half dozen sentences***):

\_\_\_\_\_\_\_\_\_\_\_**I hope that you were able to add a smile to your life with Julia:** 😺 + 😞 **==** 😀\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Solutions and Screenshots**

**Steps completed:**

* List here your chosen notebooks -
  + NYC Taxi data - Plotting very large datasets meaningfully using datashader –

<https://anaconda.org/jbednar/nyc_taxi/notebook>

* + Labeled faces in the wild recognition

<http://nbviewer.jupyter.org/github/ogrisel/notebooks/blob/master/Labeled%20Faces%20in%20the%20Wild%20recognition.ipynb>

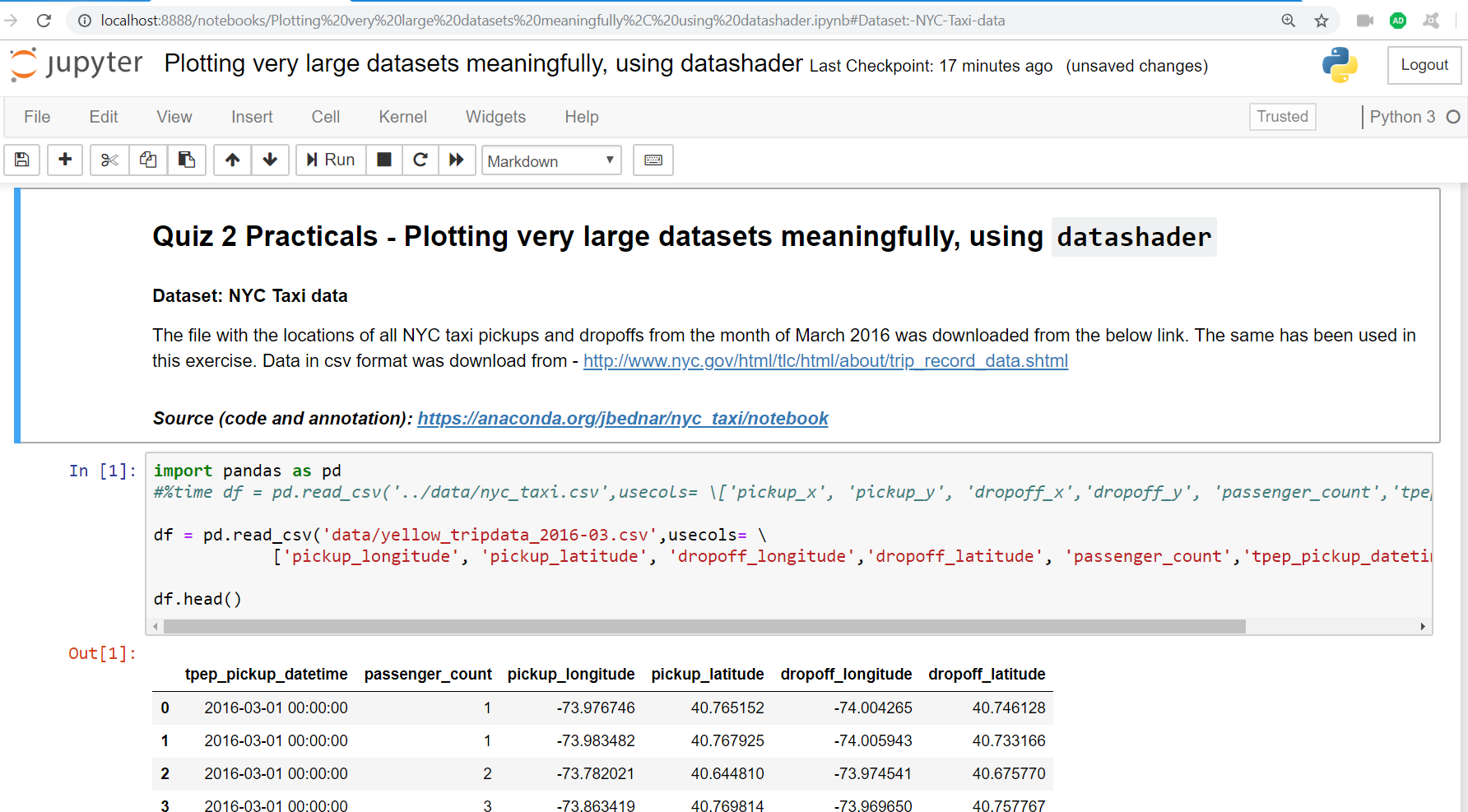
* Work your way through these two Python-based Notebooks and cut-and-paste some snippets here to illustrate that you worked them and achieved completion of the work in the notebooks themselves.
* **Notebook 1 - NYC Taxi data - Plotting very large datasets meaningfully using datashader**
  + Link to the reference python notebook – we choose a different dataset from March 2016.

<https://anaconda.org/jbednar/nyc_taxi/notebook>

* + After downloading the notebook and reading about the exercise, I downloaded the file with all locations of all NYC taxi pickups and dropoffs from the month of March 2016 from the below link. It had over 12 million entries.

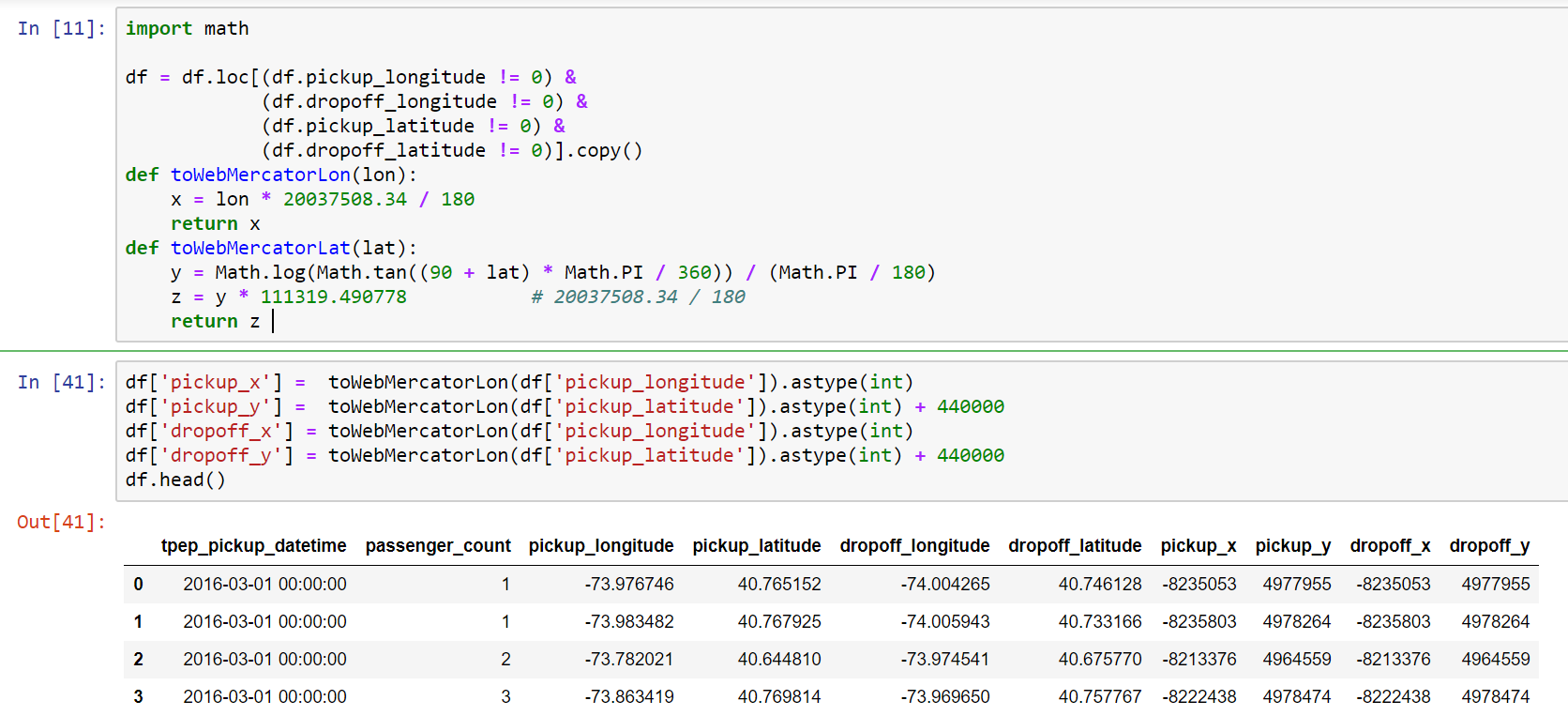
<http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml>

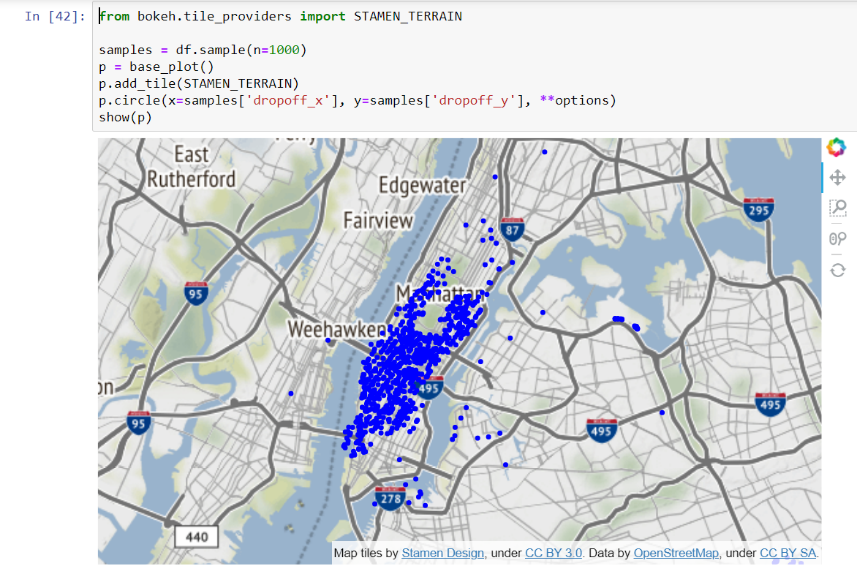
* + Few columns from the original file were imported into a dataframe using pandas module. Initially, there were only latitude and longitude columns in it as shown below.

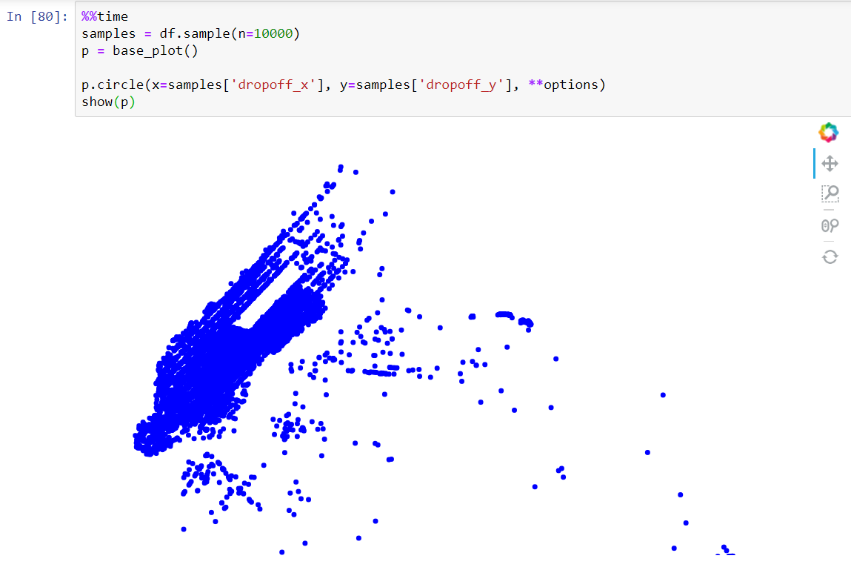


* + To map the co-ordinate information in a map, I had to convert latitude and longitude to web Mercator co-ordinates (used in google maps and other web applications). This turned out to be a time-consuming endeavor as there are many confusing co-ordinate systems.

I had to create a user-defined function, after filtering out the null values in the data. Math turned out to be best package for this job as other packages such as pyproj and utm took hours to process 12 million entries. This piece of code was missing in the original notebook.

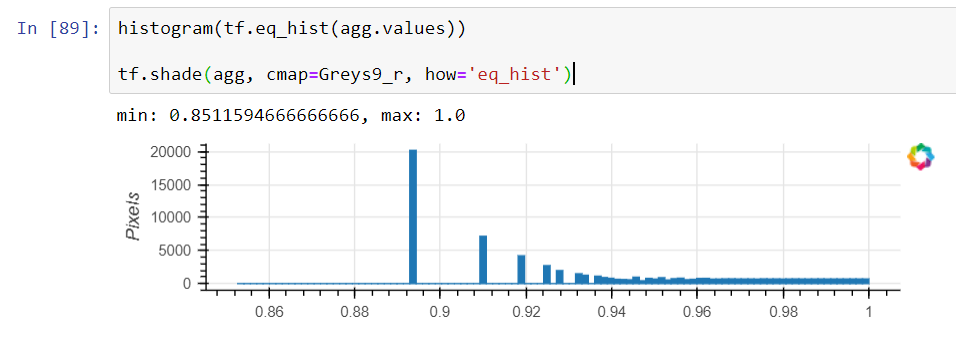
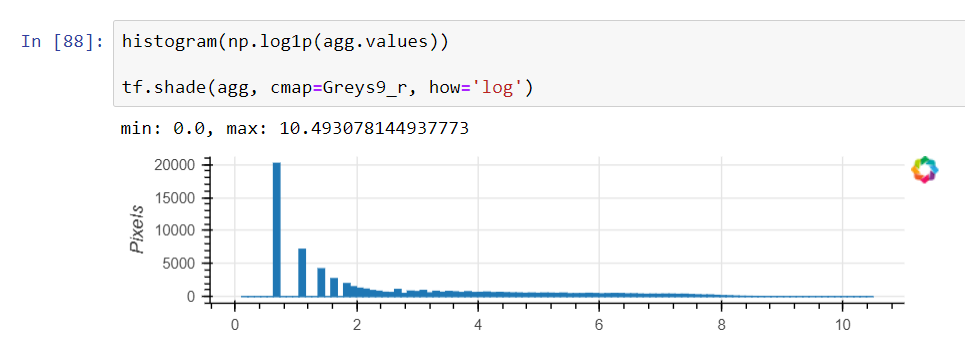
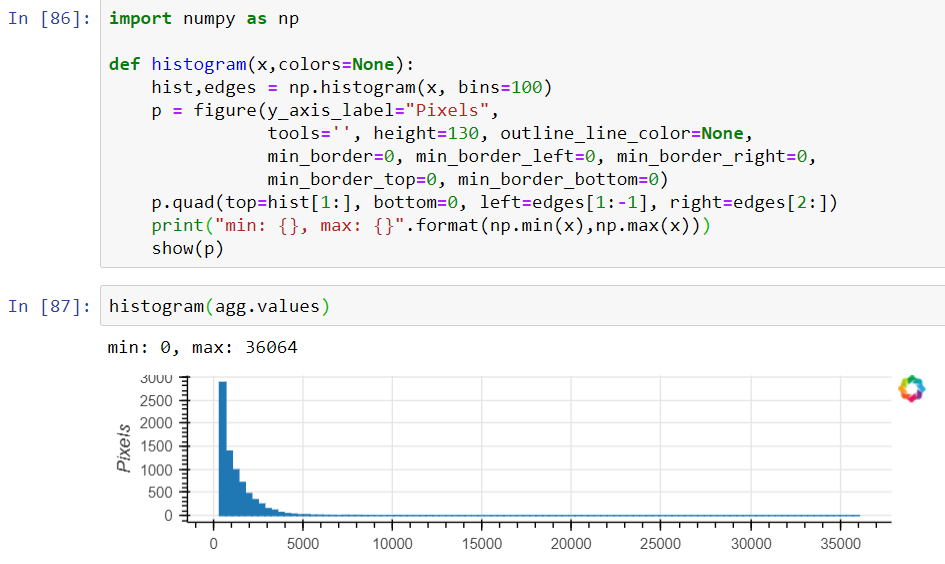
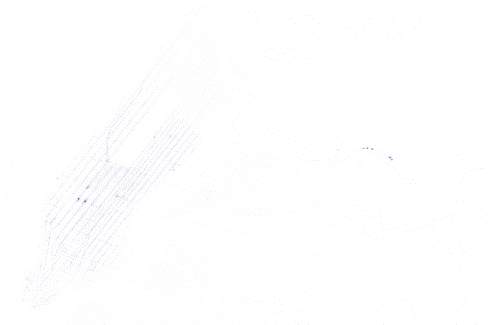
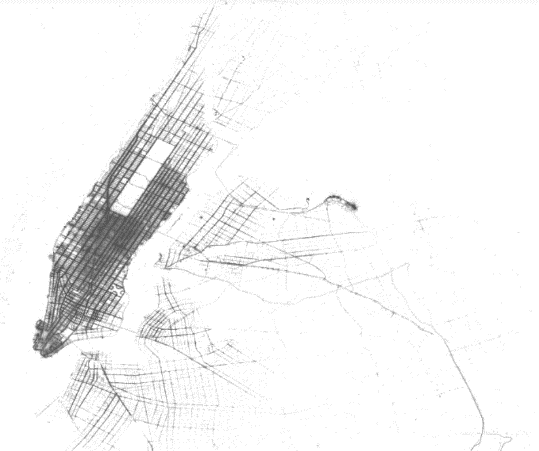
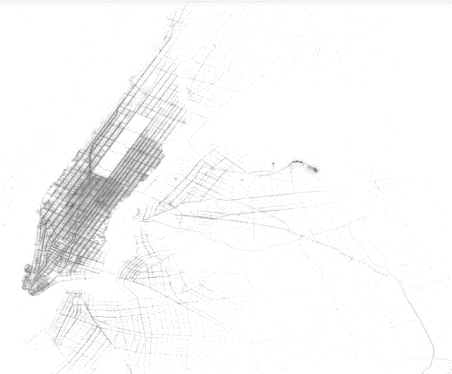


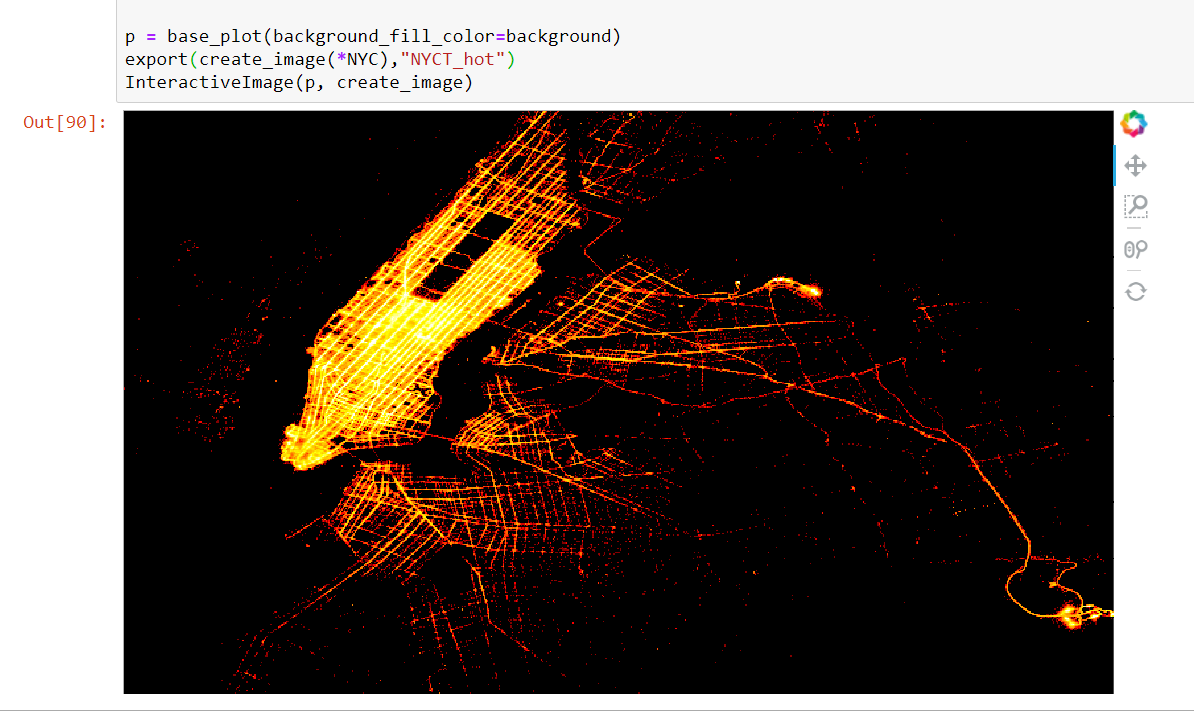
* + **1000-point scatterplot** - It was interesting plotting with 1000 datapoints in the sample. As only 0.000083 %of the data from 12 million entries were represented, it lead to a phenomenon called undersampling shown in Figure (a).
  + **10,000-point scatterplot** – lead to a phenomenon called over-sampling and it was hard to view along with the map as shown in Figure (b).
  + **100,000-point scatterplot** – lead to saturation as the points became tiny and visible at popular locations. (Figure (c)).

 ***Figure (a) above, figure (b) below.***

***Figure (c)***

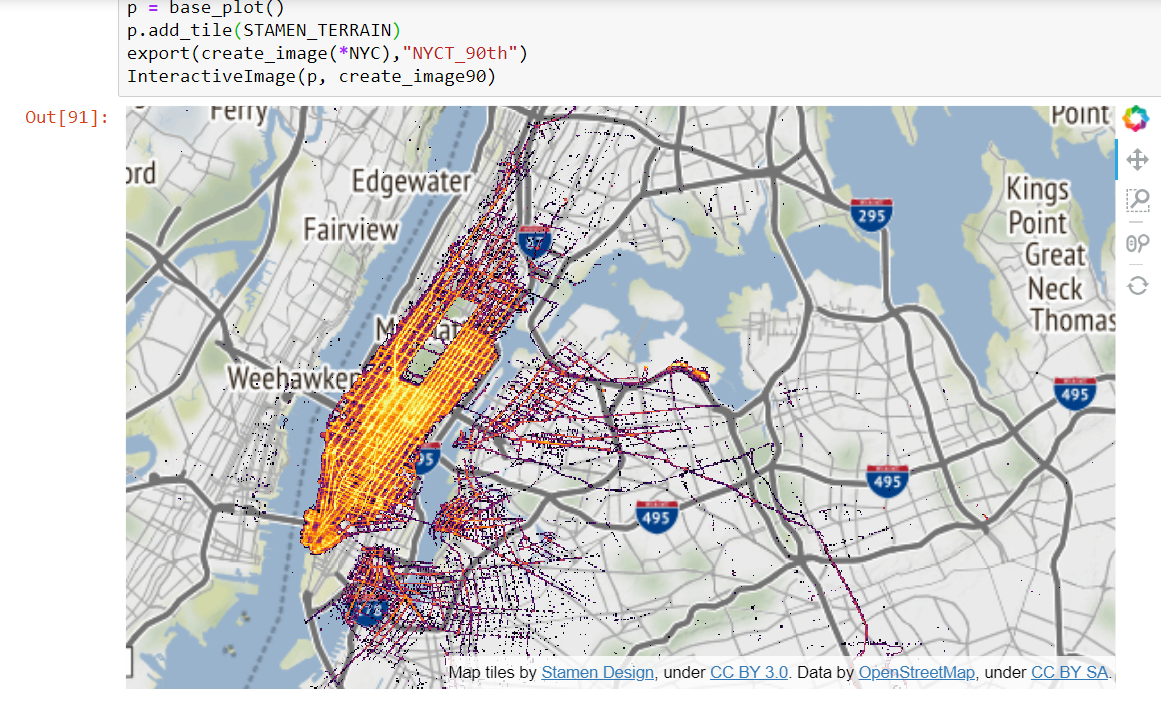
### 10 million-point datashaded plots - with datashader, there comes features such as auto-ranging and we can plot large datasets without over-sampling, undersampling or saturation. Below are some of the plots of 10 million datapoints with varying range and parameters. They are generated in few seconds.



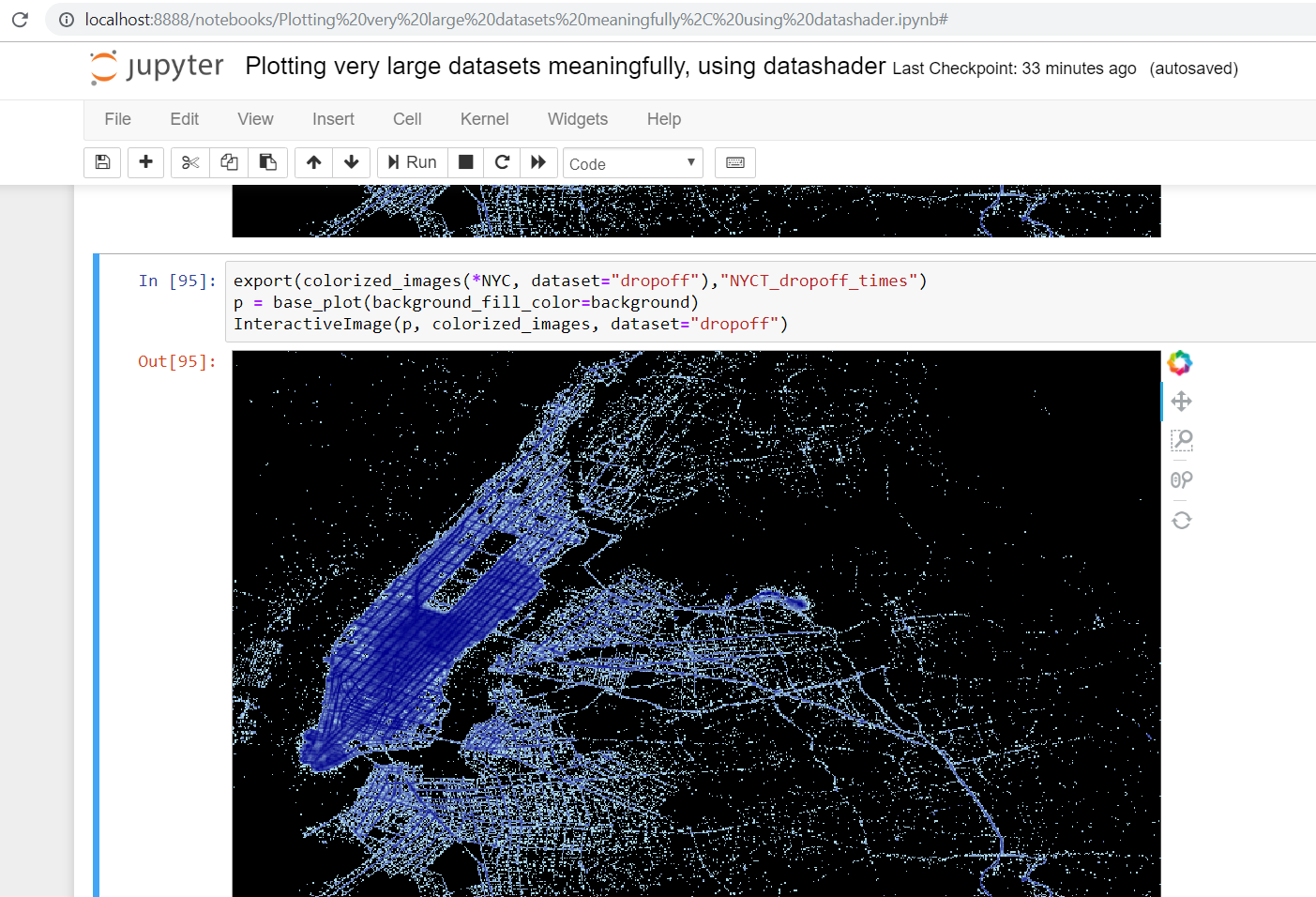


* **Interactive datashaded plots –** Here is the unveiling of the full strength of datashader. It is interactive showing the different levels of the large dataset.

It reveals local structures that are invisible through global view.

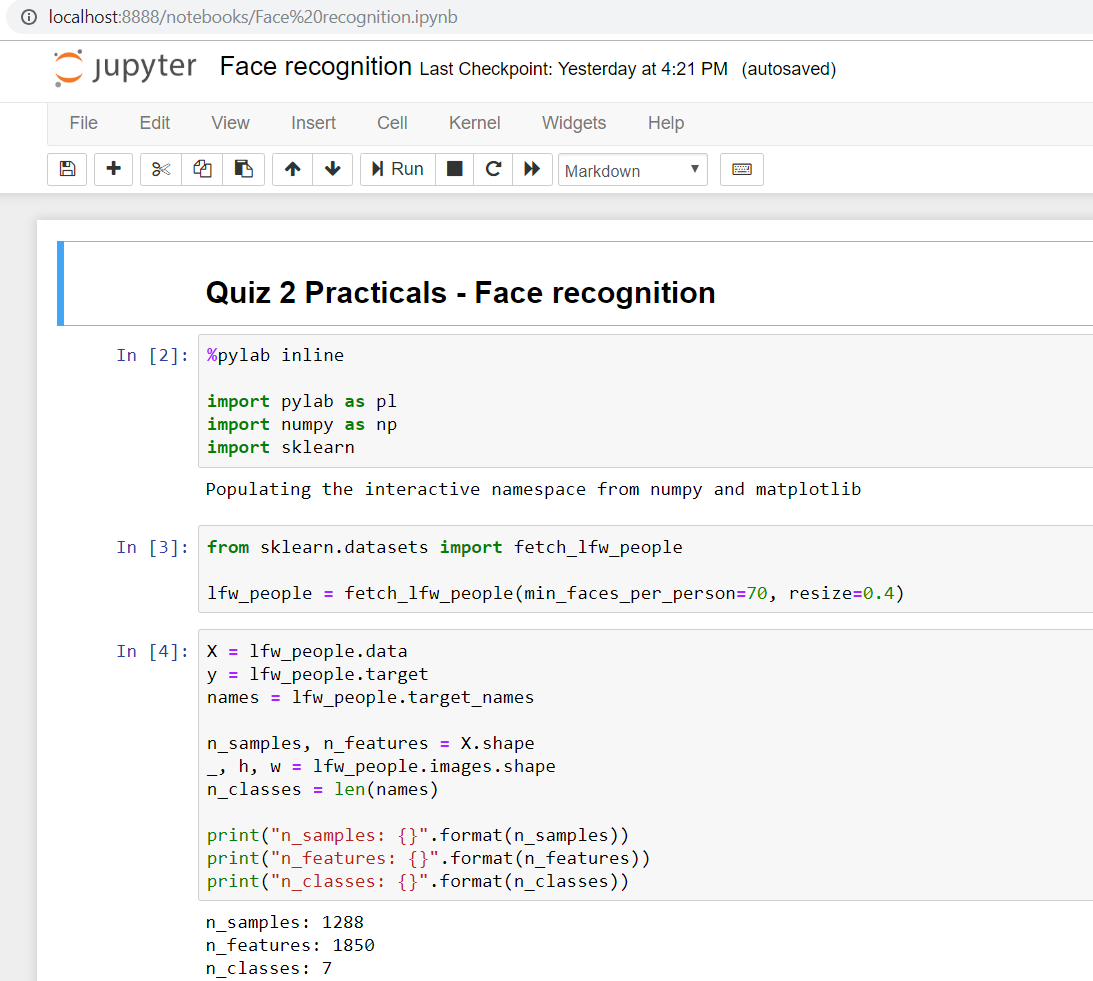
Furthermore, it performs automatic operations and auto-ranging and optimates our output.

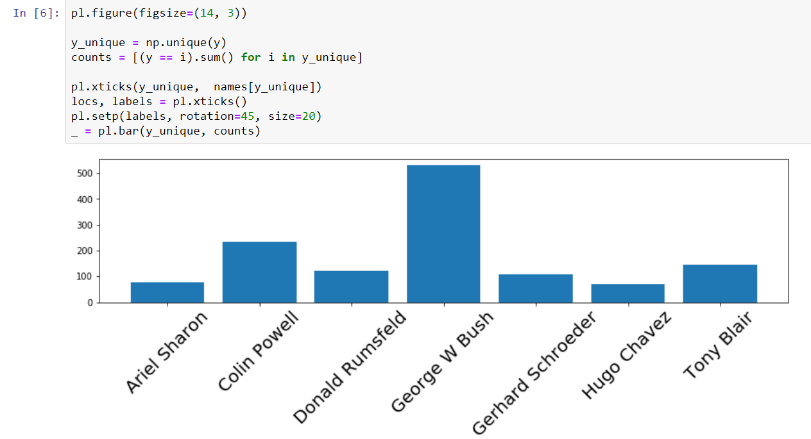
* We can further customize the plot by inserting Bokeh tile- STAMEN\_TERRAIN (apparently by Stamen design) that inserts a map of the area under study / visualization.
* There were several errors during these steps due to an empty array passed from the aggregate parameter. Turned out to be incorrect x and y range in the plot definition.
* The final plot looks surreal and beautiful. The interactive features of bokeh and datashader can be used to view local structures. Even though the timing of the trip is color-coded (example: red at night, blue in the evening), there are busy hours that stand-out.

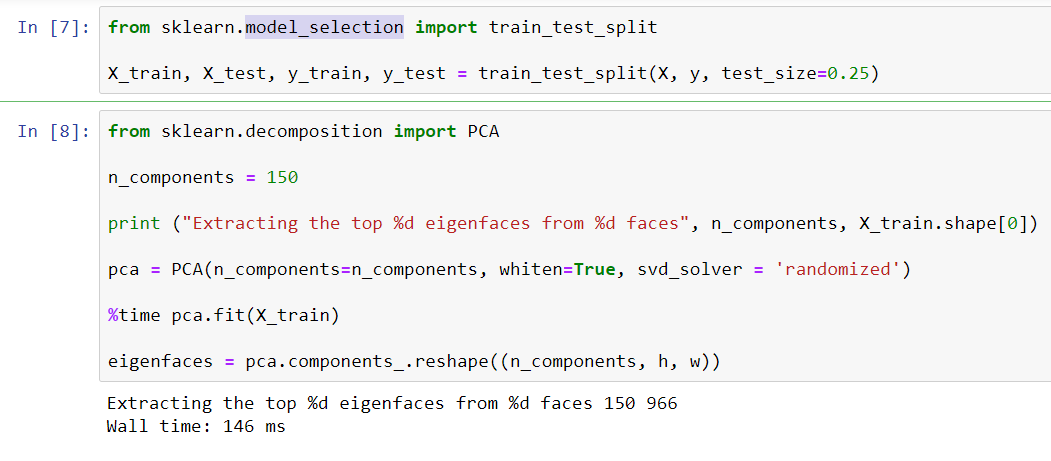
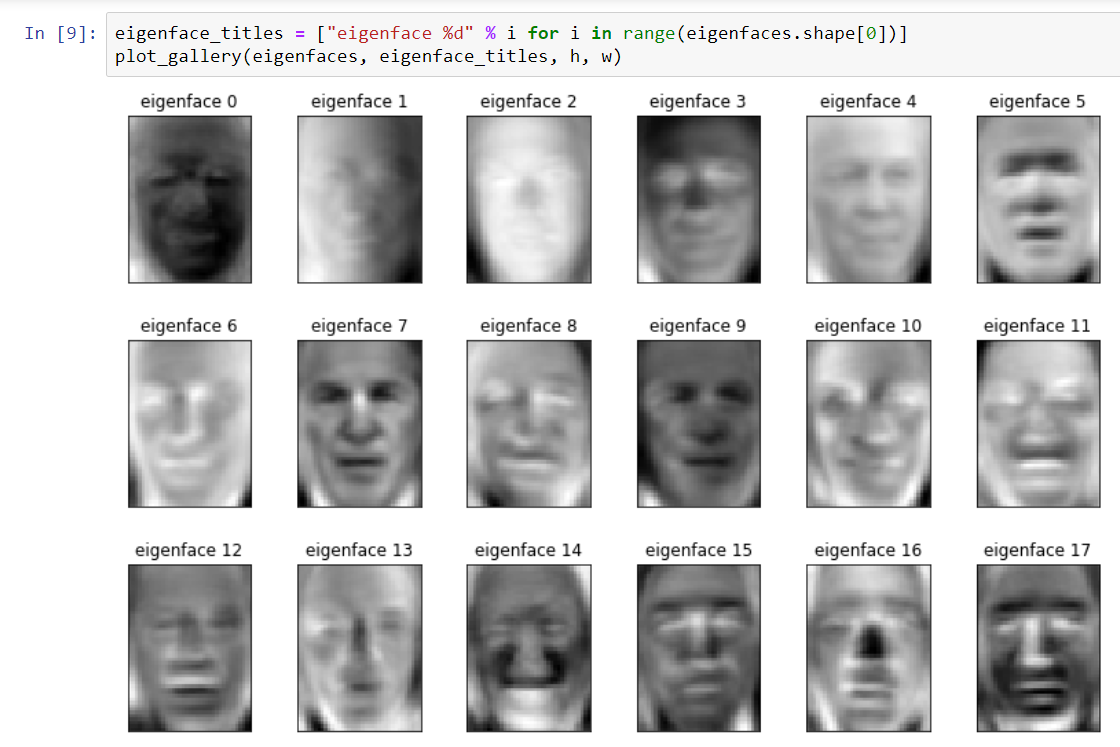
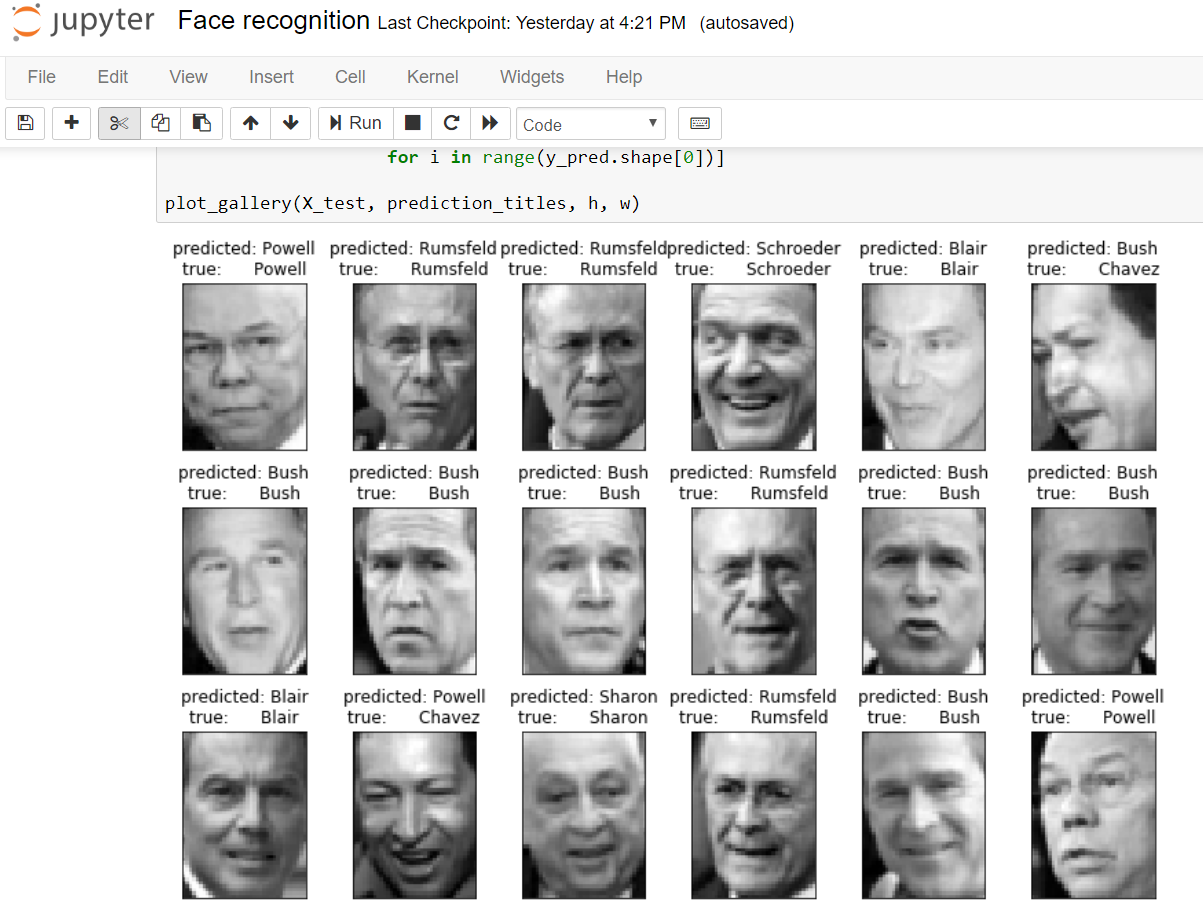


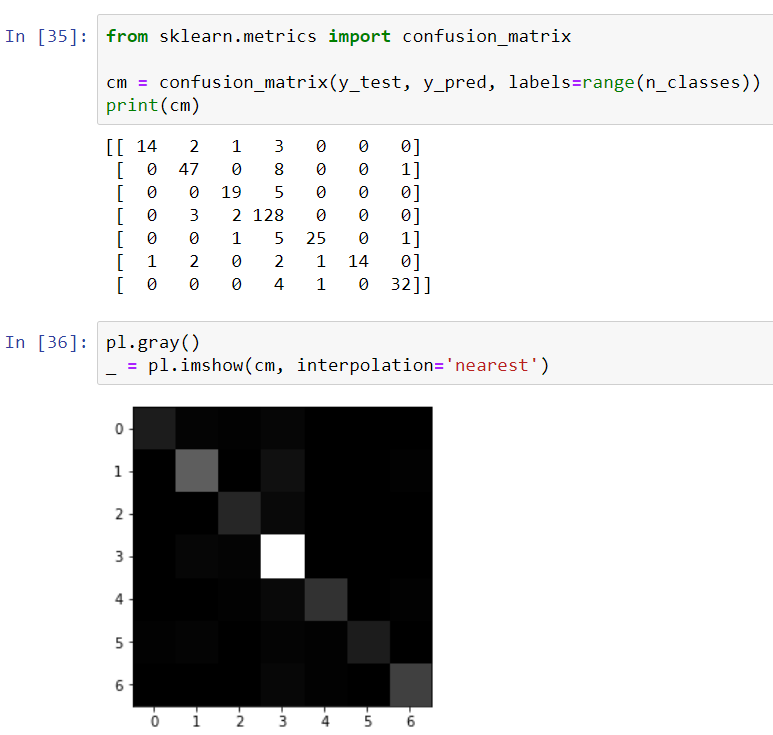
* **Notebook 2:** **Labeled faces in the wild recognition**
* Link of the reference document

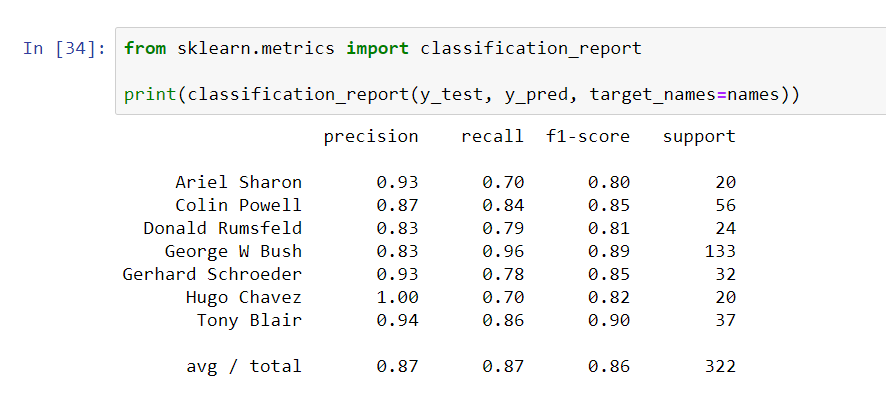
<http://nbviewer.jupyter.org/github/ogrisel/notebooks/blob/master/Labeled%20Faces%20in%20the%20Wild%20recognition.ipynb>

* I followed the reference document and imported pylab, NumPy and sklearn.
* There were countless errors as the notebook is old and many classes have been depreciated.
* It was fun exploring the various options through documentations, google search and Stackoverflow forum.
* A pre-existing dataset from sklearn was importing for this study.
* There were 1288 samples and 1850 features.
* After an initial display of some of the samples with some of the biggest names of this century, the uniqueness in features was plotted using a histogram.



* Then came the quintessential training and test split using model\_selection from sklearn. Few depreciated class was replaced during this process.
* There was considerable syntax change in the next step (RandomizedPCA) used to generate top eigenfaces.
* The complex steps involved in AI and machine learning look simple in this powerful language (python).
* Mathematical and statistical functions and estimations are performed (for example SVM or support vector machines and SVC).
* The output below is truly incredible.
* The precision scores are displayed below along with the confusion matrix.





* Comment here on **what** you learned both about the topic chosen and the Jupyter Notebook approach to documenting work (***at least a half dozen sentences***):

|  |  |
| --- | --- |
| Topic | What I learned? |
| Why I choose these notebooks? | * As a former bioinformatics engineering student, I was interested in how we can display large datasets efficiently. It was always cumbersome displaying genetic information. Even at my workplace (at Accenture), we worked with millions of data entries in a single table and I never found a complete and efficient representation technique. Hence, this notebook grabbed my attention. Later, I discovered that this notebook was part of Kaggle competition. * Face recognition and machine learning are technological breakthroughs that are highly interesting. I heard an AI painting sold at a record price at an auction recently and this triggered a greater interest. |
| **Notebook 1** - NYC Taxi data - Plotting very large datasets meaningfully using datashader | * I learnt about installing and learning about new packages from documentation. * I learnt how powerful some of the modern plotting tools can be, capable of visualizing 150-200 million datapoints at once. * I understood more about web Mercator system of coordinates and how important it is in plotting. I learnt that math module is way faster and efficient than utm or pyproj, the hard way. * I learned to sample and plot various sizes of datapoints and the concept of undersampling, oversampling and saturation. * I was intrigued to see the10 million-point datashaded plots with datashader, that come with features such as auto-ranging. I learnt that there is no need to adjust color and figure parameters hence eliminating the probability of undersampling, oversampling and saturation. Even 100 million entries can be plotted. * Steps involved in interactive datashader plotting: creating a canvas object with the shape of the proposed plot, aggregating all points into a set of bins, and mapping the resulting counts into a visible color from a specified range to make an image. * I learned to troubleshoot issues that arise while handling huge datasets. I understood how to modify dataframes. * I learned to set and tweak parameters to optimize the final visualization. |
| **Notebook 2** - Labeled faces in the wild recognition | * I learned the art of machine learning and face recognition. * I understood how to import pre-existing datasets and plot image galleries in python. * I learned about the various mathematical and statistical methods and estimations used. * I learned how to split data into training and test set and generate the desired outcome. * I understood how to check the precision and other characteristics of the resulting output. * I learned how to replace depreciated packages and classes and redo the code. * I learned how to handle errors by modifying the existing code and removing obsolete parameters from commands. |
| **Jupyter Notebook** | * In the era of sharing there can be no better way of sharing information and educating others. * I was enthralled when I first used LaTeX for representing a formula for web Mercator co-ordinates. I preceded and succeeded few formulae with $$ symbol.      * Markdown code column is excellent in documenting information otherwise it would be difficult using # and other symbols for each annotation. * Markdown is a treasure trove of features. We can generate various levels of headings, italics and highlight information and insert images, graphics and videos. * Upon exploring the plethora of features in Jupyter notebook, I understood why so many notebooks were available in this notebook – it is user-friendly, offers numerous features and supports several formats and connections and most importantly, offers features for effectively documenting information. * It is a fantastic approach for documenting codes, code snippets and related information. * Despite the errors, without reference, it would have been extremely difficult to accomplish these tasks. Provided a much-needed base for coding. |

**Steps for the Second Lab:**

* Do a Google search on the topic “Julia Language” and list here which articles (atleast 3) you read:
  + <https://en.wikipedia.org/wiki/Julia_(programming_language)>
  + <https://www.infoworld.com/article/3284380/data-science/what-is-julia-a-fresh-approach-to-numerical-computing.html>
  + <https://www.infoworld.com/article/3241107/python/julia-vs-python-julia-language-rises-for-data-science.html>
  + <https://www.techrepublic.com/article/is-julia-the-next-big-programming-language-mit-thinks-so-as-version-1-0-lands/>
* Machine generated alternative text:
  ::Julia 
  computmg 
  Registration successful. You may log in now. I read through the file **Intro\_Julia\_tutorial\_slides\_ODSC\_West.pdf** and understood main reasons for another computer language and why Julia could be that language to explore. The suggested links introducing Julia were bookmarked as advised.
* Juliabox account creation was completed successfully.

Machine generated alternative text:
C 
Julia 
BOX" 
juliabox.com 
Dashboard Jupyter 
O Help 
Buy more Compute 
Assignments 
megharajam.rao@sjsu.edu • 
C Jupyter 
Files 
Running 
Nbextensions 
Select items to perform actions on them. 
o 
0 
o 
o 
o 
o 
o 
o 
o 
o 
o 
o 
o 
o 
o 
E / tutorials I introductory-tutorials I intro-to-julia 
D. 
CD images 
CO short-version 
Name 
00. 
02. 
03. 
04. 
05. 
06. 
07. 
08. 
09. 
10. 
Jupyter_notebooks.ipynb 
Getting started. ipynb 
Strings.ipynb 
Data structures.ipynb 
Loops.ipynb 
Conditionals.ipynb 
Functions.ipynb 
Packages.ipynb 
Plotting.ipynb 
Julia is fast.ipynb 
Multiple dispatch.ipynb 
10.1 Multiple dispatch.ipynb 
Last Modified 
seconds ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
a minute ago 
File size 
2.27 kB 
6.16kB 
8.61 kB 
16.3 kB 
5.66 kB 
13.3 kB 
5.49 kB 
6.39 kB 
16.3 kB 
9.47 kB 
11.6kB 

* Here is the dashboard with introductory Jupyter notebooks.
* Document here your work (using cut & paste or similar): e.g., 😺 + 😞 == 😀

|  |  |  |
| --- | --- | --- |
| **First tutorial –** Jupyter notebook  **Second tutorial –** Getting started | | * Learned how to suppress the otherwise default print of last line using a semicolon. * Learned how to get more documentation on a function (other than shift+tab) and even examples by preceding the function with a question mark. * Learned to execute shell commands using a semi-colon before the command. * I learned how to print, comment, assign variables and find the variable type using typeof () function. * I understood basic math and convert syntax. * Interesting to know that instead of using # each time, we can comment multi-line using #= *comment*=#. * I learned how to insert a smiley and assign values to it. Simply mind-blowing! * In this funny illustration, a cat is assigned a positive 1 score while a sad face gets a negative one score, sum of which is zero (happy face).   😺 + 😞 == 😀  I have never worked with a language uses smilies as variables. Truly powerful and futuristic. |
| **Third tutorial –** Strings | | * I learned about string definition using double quote. Single quote defines a char variable. * For simply printing multi-line text, we can use 3 double quotes. * To add existing variable into a string, we can use dollar sign ($). * Even though asterisks (\*) and string () function facilitates concatenation, string () functions goes a step further and converts even non-strings into strings. * The repeat function prints the variable repeatedly n specified times. |
| **Forth Tutorial** – Data structures | * We graduated to basic data structures such as Tuples, dictionaries and arrays. Learned how to define them. * I understood that Tuple is an ordered list, (enclosed in normal brackets) while a namedTuple has name for each element. The first element is numbered as 1 not 0. Tuple.element gives value. * I learned about dictionary (related data represented as key value pairs) and related operations. * I learned to simultaneously display and delete using !pop function. * Arrays are mutable, enclosed in square brackets, and multi-dimensional and highly flexible. I learned about rand () functions used to generate random numbers and copy () functions. | |
| **Fifth Tutorial** – Loops | * I understood the syntax for while and for loops. * I learned how to use nested loops and multiple variables in them. * I learnt to use loops inside an array assignment. That was intriguing! * I learnt to code an intermediate level loop printing squares of numbers from 1 to 100. | |
| **Sixth Tutorial** – Conditionals | * I understood the syntax of if, elseif, else conditions. * I learned to use ternary operators. * I understood short-circuit evaluation (using && and Boolean logic). Side-effect is that for nested query, it will not execute the nested segment that comes second if first statement is false. It is fair to say it jumps to conclusions! :D * I learned to use double pip operator (||) which seems very handy. | |
| **Seventh Tutorial** – Functions | * I learned how to declare functions and perform operations with the same. * I learned how to duck – type using Julia. The sayhi () function is one of a kind and says hi to any variable we specify. * I understood that mutating functions followed by! can alter their contents while non-mutating cannot (no ! symbol). * Map is same as in python and applies function to every element of the data structure. Similar to map is broadcast. | |
| **Eighth Tutorial** – Packages | * I learnt that the keyword in Julia for importing packages is ‘using’. First time we need to use the package manager to add it. * We can also call languages python and R with PyCall and RCall. * It was exciting viewing the elaborate color palette available in Julia. * Even error messages have long descriptions pointing towards the correct command or operation. | |
| **Ninth Tutorial -** Plotting | * I learnt that Julia has several plotting packages such as Plot.jl, gr () and plotlyjs() and pyplot. * Two simple commands with plot() and scatter! () and viola!, we get a plot. However, I learned how important ‘!’ symbol after a function is to render it changeable (mutating function). * I learnt label axes, flip them and assign title to the plot. * I tried my hand at uniplots and subplots. There were quite easy. | |

* Comment here on **what** you learned both about the Julia language and the Jupyter Notebook approach to working with Julia (***at least a half dozen sentences***):
  + Rich, terse, expressive and straightforward syntax makes it highly user-friendly. Even non-geeks can easily play with it. It is interactive and flexible in nature.
  + There are numerous help aids. Even error messages have long descriptions pointing towards the correct command or operation. Over 1900 packages work well together in synchrony.
  + Julia bears resemblance to Python and it is hard to believe it is a compiled language like C.
  + While converting co-ordinates in the first example, I realized how slow python was. Julia is supposedly adept at scientific computing, data processing and analytics. I am yet to explore its capabilities.
* Python, R, C, Fortran libraries can be accessed. Furthermore, it allows metaprogramming or programs which generate and modify programs. That is truly advanced.
  + As little as smiley variable, duck-typing to metaprogramming, Julia seems futuristic and immensely superior and consummate. Especially when we compare it to the likes of Python.
  + My take on Julia based on my understanding– “She might look simple, but she can outsmart her peers.”
* From my reading, I learned that -
* Julia has built-in for parallelism and uses multiple dispatch enabling object oriented and functional programming.
* I was fascinating to know that the just-in -time(JIT) compiled language, at its best, can be at par with C (in terms of speed). It is in the “petaflop club” because it can handle petaflops of data and simulate over 188 million stars, galaxies and astral objects.
* It enables dynamic typing (chameleon-like nature).
* MIT, reputed universities and companies have already endorsed Julia.
* It is slated to power self-driving cars, AI advancements, modern advancements such as 3-D printers, augmented reality, genomics and machine learning.
* Jupyter notebook approach
  + I will start with a confession – when I started python, we learned complex statistical functions skipping basics and I had to browse and troubleshoot when I encountered several loop-related errors. That was painful and time-consuming.
  + It was nice to go through basics right in the beginning along with illustration and demonstration. It felt like a well-supplemented basic course.
  + Enlightening, interactive and educational, feels like ‘machine teachers’ are taking over. Jokes aside, it is highly helpful and facilitates learning.
  + Exercises and practice gave good hands-on experience. Stays in memory longer too. Hence, I feel fairly confident for a beginner.
  + 10 years ago, when I had to learn a language, it was hard. Now, the introductory lessons in interactive notebooks have made learning fun. Also, we can easily go back, revisit and revise the topics covered.
  + There is ample documentation in markdown cells and eliminates the need to make hard and soft copies of explanation, code snippets and basic commands.
  + Furthermore, it can be used as a reference whenever necessary. For example: Exploring the Jupyter notebooks with NYC Taxi (large dataset) plotting and Face recognition was highly interesting. Without them, it would have been difficult to start coding.
  + Jupyter notebooks make it easy to share, collaborate and educate. Julia won me over with its simplicity, speed and computational prowess. Both of them, definitely brought a smile to our faces 😀.