Introduction to data science

Patrick Shafto

Department of Math and Computer Science

Plan for today

- HW 2
- Programming for data science
- Superfast intro to python

HW 2

- Write a short tutorial on dicts (yes, again), functions, and classes
- Cover basic functionality with worked examples
- Due Sunday by 11:59 pm

- Evaluations due Tuesday by 11:59pm
    - Use the rubric!
    - Justify your scores on each section of the rubric!

HW 2: Example evaluations

- Criteria 1: 19/24
- Simple examples.Easy to understand.Need to add more examples for dictionary and class.More application based examples needs to be added other then simple add and subtract operations.Please add references for the work done.
- Overall -3-Interpretation,3-Representation,3-Analysis,3-Calculation,3-Assumptions,3-Communications

HW 2: Example evaluations

- Criteria 1: 22/24
- Theoretical part is explained nicely with good examples and comments. Need to add more application based examples in classes.
- Overall: 4-Interpretation,4-Representation,4-Analysis,3-Calculation,3-Assumptions,4-Communications

HW 2: Example evaluations

- Criteria 1: 24/24
- Interpretation - 4
- Representation - 4
- Calculation - 4
- Application / Analysis - 4
- Assumptions - 4
- Communication - 4
- All concepts are explained properly with multiple examples. Presentation is quite good. Appropriate comments are present.

HW 2: Example evaluations

- Criteria 1: 24/24
- Interpretation - 4
- Representation - 4
- Calculation - 4
- Application / Analysis - 4
- Assumptions - 4
- Communication - 4
- Clear and concise examples and explanations used. Presentation is also good.

HW 2: Example evaluations

- Criteria 1: 18/24
- Interpretation: Explanation is slightly vague
- Representation: Could have given more examples about concepts and built in methods for Dictionaries, rest is okay
- Calculation:Does not apply
- Application/Analysis:Does not apply
- Assumptions:Does not apply Communication:Satisfactory usage of various functionalities of dicts, functions and classes, could have been more detailed.
- Please note: Evaluation marks is taking into consideration the error messages not handled while program execution.

HW 2: Example evaluations

- Criteria 1: 23/24
- Calculations: N/A
- Assumption: N/A
- Communication 4: Covered the topics in depth with examples
- Representation 3: Showed relevant information but needed to go over the use of the 'self' keyword and constructors, '__init__' which is essential to the basic use of classes.
- Interpretation 4: provided accurate explanation of information presented
- Application/Analysis 4: properly understood the information and applied it with examples of their own

HW 2

- Write a short tutorial on dicts (yes, again), functions, and classes
- Cover basic functionality with worked examples
- Due Sunday by 11:59 pm

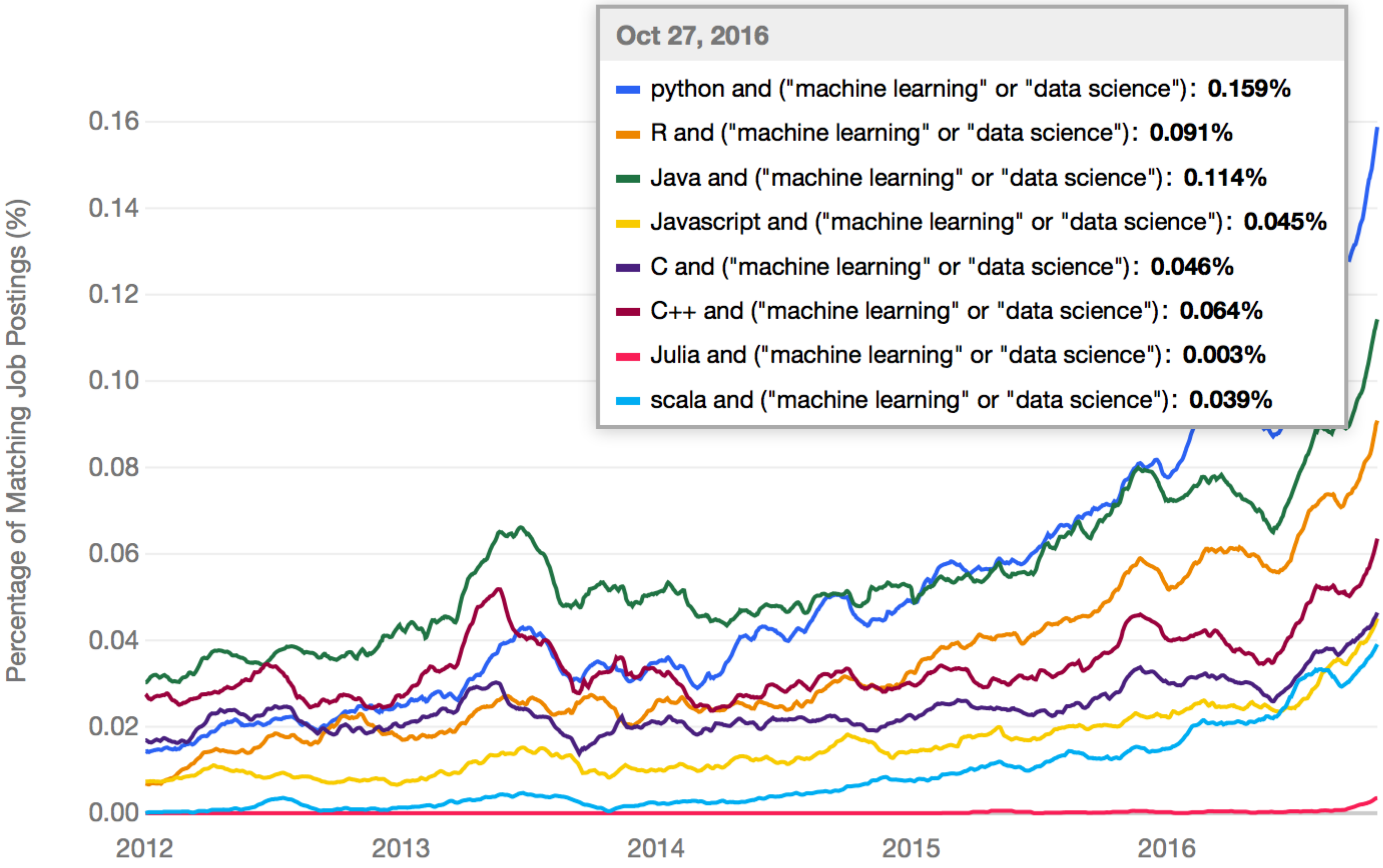
I will be grading you on your evaluations!
Provide quality feedback: use the rubric, provide comments

- Evaluations due Tuesday by 11:59pm
    - Use the rubric!
    - Justify your scores on each section of the rubric!

Programming for data science

What languages and packages are being used?

How is this changing over time?



Oct 27, 2016
python and ("machine learning" or "data science"): 0.159%
R and ("machine learning" or "data science"): 0.091%
Java and ("machine learning" or "data science"): 0.114%
Javascript and ("machine learning" or "data science"): 0.045%
C and ("machine learning" or "data science"): 0.046%
C++ and ("machine learning" or "data science"): 0.064%
Julia and ("machine learning" or "data science"): 0.003%
scala and ("machine learning" or "data science"): 0.039%
Percentage of Matching Job Postings (%)
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
2012
2013
2014
2015
2016

Python for data science

Libraries for python:

- **NumPy** stands for Numerical Python. The most powerful feature of NumPy is n-dimensional array. This library also contains basic linear algebra functions, Fourier transforms, advanced random number capabilities and tools for integration with other low level languages like Fortran, C and C++
- **SciPy** stands for Scientific Python. SciPy is built on NumPy. It is one of the most useful library for variety of high level science and engineering modules like discrete Fourier transform, Linear Algebra, Optimization and Sparse matrices.
- **Matplotlib** for plotting vast variety of graphs, starting from histograms to line plots to heat plots.. You can use Pylab feature in ipython notebook (ipython notebook –pylab = inline) to use these plotting features inline. If you ignore the inline option, then pylab converts ipython environment to an environment, very similar to Matlab. You can also use Latex commands to add math to your plot.
- **Pandas** for structured data operations and manipulations. It is extensively used for data munging and preparation. Pandas were added relatively recently to Python and have been instrumental in boosting Python's usage in data scientist community.
- **Scikit Learn** for machine learning. Built on NumPy, SciPy and matplotlib, this library contains a lot of effiecient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

- **Statsmodels** for statistical modeling. Statsmodels is a Python module that allows users to explore data, estimate statistical models, and perform statistical tests. An extensive list of descriptive statistics, statistical tests, plotting functions, and result statistics are available for different types of data and each estimator.
- **Seaborn** for statistical data visualization. Seaborn is a library for making attractive and informative statistical graphics in Python. It is based on matplotlib. Seaborn aims to make visualization a central part of exploring and understanding data.
- **Bokeh** for creating interactive plots, dashboards and data applications on modern web-browsers. It empowers the user to generate elegant and concise graphics in the style of D3.js. Moreover, it has the capability of high-performance interactivity over very large or streaming datasets.
- **Blaze** for extending the capability of Numpy and Pandas to distributed and streaming datasets. It can be used to access data from a multitude of sources including Bcolz, MongoDB, SQLAlchemy, Apache Spark, PyTables, etc. Together with Bokeh, Blaze can act as a very powerful tool for creating effective visualizations and dashboards on huge chunks of data.
- **Scrapy** for web crawling. It is a very useful framework for getting specific patterns of data. It has the capability to start at a website home url and then dig through web-pages within the website to gather information.

Introducing python: Google

(open up your computers and fire up jupiter)

```
>>> a = 6         ## set a variable in this interpreter session
>>> a             ## entering an expression prints its value
6
>>> a + 2
8
>>> a = 'hi'      ## 'a' can hold a string just as well
>>> a
'hi'
>>> len(a)        ## call the len() function on a string
2
>>> a + len(a)  ## try something that doesn't work
Traceback (most recent call last):
  File "", line 1, in
TypeError: cannot concatenate 'str' and 'int' objects
>>> a + str(len(a))  ## probably what you really wanted
'hi2'
>>> foo           ## try something else that doesn't work
```

Introducing python: Google

Demonstrate your skills:

1) Use markdown to create a section header saying “In-class assignment 09-20-17
2) Write your name and anyone else who is collaborating with you in the second cell
3) Type a math equation using LaTeX
4) Add two numbers
5) Divide two whole numbers
6) Write an if statement using binary logic
7) Write an if statement checking if the number 15 is true
8) Check the type of 15 and ’15’
9) Check the length of 15 and ’15'

```python
# Defines a "repeat" function that takes 2 arguments.
def repeat(s, exclaim):
    """
    Returns the string 's' repeated 3 times.
    If exclaim is true, add exclamation marks.
    """


    result = s + s + s # can also use "s * 3" which is faster (Why?)
    if exclaim:
        result = result + '!!!'
    return result
```

Introducing python: Strings

```
s = 'hi'
print s[1]          ## i
print len(s)        ## 2
print s + ' there'  ## hi there
```

```
pi = 3.14
##text = 'The value of pi is ' + pi
text = 'The value of pi is '  + str(pi)
```

Introducing python: Strings

s.lower(), s.upper() -- returns the lowercase or uppercase version of the string

s.strip() -- returns a string with whitespace removed from the start and end

s.isalpha()/s.isdigit()/s.isspace()... -- tests if all the string chars are in the various character classes

s.startswith('other'), s.endswith('other') -- tests if the string starts or ends with the given other string

s.find('other') -- searches for the given other string (not a regular expression) within s, and returns the first index where it begins or -1 if not found

s.replace('old', 'new') -- returns a string where all occurrences of 'old' have been replaced by 'new'

s.split('delim') -- returns a list of substrings separated by the given delimiter. The delimiter is not a regular expression, it's just text. 'aaa,bbb,ccc'.split(',') -> ['aaa', 'bbb', 'ccc']. As a convenient special case s.split() (with no arguments) splits on all whitespace chars.

s.join(list) -- opposite of split(), joins the elements in the given list together using the string as the delimiter. e.g. '---'.join(['aaa', 'bbb', 'ccc']) -> aaa---bbb---ccc

Demonstrate each method in your notebook

Introducing python: Strings

Slicing! We talked about….

```
# % operator
text = "%d little pigs come out or I'll %s and %s and %s" % (3,
'huff', 'puff', 'blow down')
```

Introducing python: Lists

```
colors = ['red', 'blue', 'green']
print colors[0]    ## red
print colors[2]    ## green
print len(colors)  ## 3
```

```
b = colors   ## Does not copy the list
```

```
squares = [1, 4, 9, 16]
sum = 0
for num in squares:
  sum += num
print sum  ## 30
```

Introducing python: Lists

```
list = ['larry', 'curly', 'moe']
if 'curly' in list:
  print 'yay'
```

```
## print the numbers from 0 through 99
for i in range(100):
  print i
```

modify first bit of code to also print out the location in list

Introducing python: Lists

```
list = ['larry', 'curly', 'moe']
list.append('shemp')         ## append elem at end
list.insert(0, 'xxx')        ## insert elem at index 0
list.extend(['yyy', 'zzz'])  ## add list of elems at end
print list  ## ['xxx', 'larry', 'curly', 'moe', 'shemp', 'yyy', 'zzz']
print list.index('curly')    ## 2

list.remove('curly')         ## search and remove that element
list.pop(1)                  ## removes and returns 'larry'
print list  ## ['xxx', 'moe', 'shemp', 'yyy', 'zzz']
```

Introducing python: Lists

```
list = ['a', 'b', 'c', 'd']
print list[1:-1]   ## ['b', 'c']
list[0:2] = 'z'    ## replace ['a', 'b'] with ['z']
print list         ## ['z', 'c', 'd']
```

Introducing python: Dicts

```
## Can build up a dict by starting with the the empty dict {}
## and storing key/value pairs into the dict like this:
## dict[key] = value-for-that-key
dict = {}
dict['a'] = 'alpha'
dict['g'] = 'gamma'
dict['o'] = 'omega'

print dict  ## {'a': 'alpha', 'o': 'omega', 'g': 'gamma'}

print dict['a']     ## Simple lookup, returns 'alpha'
dict['a'] = 6       ## Put new key/value into dict
'a' in dict         ## True
```

Introducing python: Dicts

```
## By default, iterating over a dict iterates over its keys.
## Note that the keys are in a random order.
for key in dict: print key
## prints a g o

## Get the .keys() list:
print dict.keys()  ## ['a', 'o', 'g']

## Likewise, there's a .values() list of values
print dict.values()  ## ['alpha', 'omega', 'gamma']

## Common case -- loop over the keys in sorted order,
## accessing each key/value
for key in sorted(dict.keys()):
  print key, dict[key]
```

HW 3

- Write a short tutorial on numpy and scipy
- Cover basic functionality with worked examples
- Due Sunday by 11:59 pm

- Evaluations due Tuesday by 11:59pm
    - Use the rubric!
    - Justify your scores on each section of the rubric!