Python in Data Science Research and Education

Randy Paffenroth[§*], Xiangnan Kong[‡]

https://www.youtube.com/watch?v=EUEHOYl0mRg

Abstract—In this paper we demonstrate how Python can be used throughout the entire life cycle of a graduate program in Data Science. In interdisciplinary fields, such as Data Science, the students often come from a variety of different backgrounds where, for example, some students may have strong mathematical training but less experience in programming. Python's ease of use, open source license, and access to a vast array of libraries make it particularly suited for such students. In particular, we will discuss how Python, IPython notebooks, scikit-learn, NumPy, SciPy, and pandas can be used in several phases of graduate Data Science education, starting from introductory classes (covering topics such as data gathering, data cleaning, statistics, regression, classification, machine learning, etc.) and culminating in degree capstone research projects using more advanced ideas such as convex optimization, non-linear dimension reduction, and compressed sensing. One particular item of note is the scikit-learn library, which provides numerous routines for machine learning. Having access to such a library allows interesting problems to be addressed early in the educational process and the experience gained with such "black box" routines provides a firm foundation for the students own software development, analysis, and research later in their academic experience.

Index Terms—data science, education, machine learning

Introduction

Data Science is a burgeoning field of study that lies at the intersection of statistics, computer science, and numerous applied scientific domains. As is common within such *interdisciplinary* domains of study, Data Science education, mentoring, and research draws ideas from, and is inspired by, several other domains such as the mathematical sciences, computer science, and various businesses and application domains. Perhaps just as importantly, students who wish to pursue education and careers in Data Science come from similarly diverse backgrounds. Accordingly, the challenges and opportunities of being an educator in such a domain requires one to reflect on appropriate tools and approaches that promote educational success. It is the authors' view, and experience, that the Python scripting language can be an effective part of the Data Science curriculum for several reasons such as its ease of use, its open source license, and its access to a vast array of libraries covering many topics of interest to Data Science.

Worcester Polytechnic Institute (WPI) has recently (fall 2014) begun admitting students into its new Data Science Master's degree program and, as of spring 2015, has also initiated a Data Science Ph.D. program. Even at this early stage, the program has been quite fortunate to receive many more applications from students then it can reasonably admit. The authors have the pleasure of being faculty members in the program and have the honor of teaching a number of the courses on offer. In addition, as long time Python users (for one of them since 1997 in fact [Paf99]), the authors were intrigued by the possibility of leveraging Python in the graduate Data Science curriculum and this monograph describes some of the experiences, both successes and challenges, gained from that effort. Of course, it is much too early to make any comments on the sustained effect of using Python for graduate Data Science education, however, perhaps the reader will find some value in the authors' experiences, even at this early date.

Of course, we are not the first to suggest Python's effectiveness in an education and research environment. In fact, the Python scripting language is quite popular in numerous problem domains and Python has seen wide used in education, see e.g., [Mye07] and [Sta00]. In fact, it ranks quite highly in many surveys of programming language popularity [OGr14], it is seeing substantial growth within the Data Science community [Sin14], and is generally speaking quite easy to learn [Lut13].

However, it is not our purpose here to focus on Python in general, but rather to focus on its use in *Data Science* education and research. With that in mind, herein we will focus on a small number of case studies that provide insights into how we have leveraged Python in that domain.

In particular, herein we will discuss the use of Python at three different levels of Data Science education and research. First, one of the courses that is offered as part of our Data Science curriculum, and in which the authors and others have leveraged Python, is DS501–"Introduction to Data Science". The idea of DS501 is to provide an introductory overview of the many fields that comprise Data Science and it is intended that DS501 be one of the first classes a new student takes when entering the program. Second, one of the authors has also used Python to support MA542–"Regression Analysis". MA542 is a somewhat more advanced class that is a (core) elective in the Data Science program as well as being a class taken by many students who are seeking degrees in the Mathematical Sciences department. Finally, the authors mentor a number of students' research projects within the Data Science Program, the Mathematical Sciences Department, and the Computer Science Department. Many of these research projects leverage Python in various ways, and having access to a common code base allows the various student projects to build off of one another.

Two key themes will permeate our discussion in the following

* *Corresponding author: rcpaffenroth@wpi.edu*
§ *Worcester Polytechnic Institute, Mathematical Sciences Department and Data Science Program*
‡ *Worcester Polytechnic Institute, Computer Science Department and Data Science Program*

sections. First, the Python community provides easy access to a vast array of libraries. Even though Data Science education and research draws from many other domains, Python was always there with a library ready to support our work. Second, and perhaps more subtly, having access to a language which is easy to use, but provides access to many advanced libraries, allows one to carefully craft the difficulty and scope of homework assignments, class projects, and research problems. In particular, Python allows students to tackle specific aspects of real world problems, without being overly burdened with details that are extraneous to their particular learning objectives. Both properties make Python particularly advantageous.

Finally, in an effort to assist the reader who is not steeped in Python, we will attempt to provide a range of references so that the interested reader can learn more about the specific libraries we leverage in our work. While in certain circles the libraries we mention are well known, we thought it would be useful to collect these references together into a single document.

DS501 Introduction to Data Science

DS501–"Introduction to Data Science" is intended to be one of the first classes a new student takes when entering the Data Science program at WPI, and the goal is to provide a high level overview of a wide swath of the material that a burgeoning Data Scientist should know. In particular, the course is described as:

> This course provides an overview of Data Science, covering a broad selection of key challenges in and methodologies for working with big data. Topics to be covered include data collection, integration, management, modeling, analysis, visualization, prediction and informed decision making, as well as data security and data privacy. This introductory course is integrative across the core disciplines of Data Science, including databases, data warehousing, statistics, data mining, data visualization, high performance computing, cloud computing, and business intelligence. Professional skills, such as communication, presentation, and storytelling with data, will be fostered. Students will acquire a working knowledge of data science through hands-on projects and case studies in a variety of business, engineering, social sciences, or life sciences domains. Issues of ethics, leadership, and teamwork are highlighted. – http://www.wpi.edu/academics/catalogs/grad/dscourses.html

As one might imagine from such an ambitious description, finding the right level of detail for the course can be quite challenging. One must consider the fact that many of the students have quite varied backgrounds. Some students are experts in mathematics and have less training in computer science or software development, while others find themselves in the opposite situation.

Space does not allow for a fulsome description of the class content and, in any event, such a discussion would distract us from our focus on Python. However, in the authors' view, one important feature of such a class is that the students should be able to *get "their hands dirty" playing with real data* both early and often. Students can often find inspiration by seeing the ideas developed as part of the lectures being put to use on problems of practical interest.

With all of the above in mind, it was decided to have four interconnected *case studies* as major learning activities for the class. Each case study is intended to build upon the previous one with the students solving interesting and pertinent problems in Data Science at every step. Accordingly, our focus here will be on these case studies and the substantial role that Python had to play in their development.

Case Study One

The idea of the first case study in DS501 is to perform basic data gathering, cleaning, and collection of statistics. For this case study we choose our data source to be the Twitter Data Streaming API [Rus13], [Twi15]. Already, Python begins to demonstrate its usefulness, since it allows ready access to the Twitter API through the python-twitter library [Ptw15].

Another key feature of the case studies in DS501 is that we chose to use IPython notebooks [Per07] both to provide the assignments to the students and to have the students submit their results. Using IPython notebooks for both of these tasks provided a number of advantages. First and foremost, it let the instructors provide the students with skeleton implementations of their assignments and allowed the students to focus on their learning objectives. Second, it provide a uniform and easy to use development environment for the students' efforts. As DS501 is not a programming class, per se, leveraging IPython notebooks made the introduction of Python to those students unfamiliar with it substantially easier.

For example, in the IPython notebooks we are able to provide code examples to get the students started with their development work. For example, we could provide code similar to the following as a launching pad for their efforts (see [Twi15] for details and code example is based upon [Rus13]):

```
import twitter
#-----------------------------------------------

# Define a Function to Login Twitter API
def oauth_login():
    # Go to http://twitter.com/apps/new to create an
    # app and get values for these credentials that
    # you'll need to provide in place of these empty
    # string values that are defined as placeholders.
    # See https://dev.twitter.com/docs/auth/oauth
    # for more information on Twitter's OAuth
    # implementation.

    CONSUMER_KEY = '<Insert your key>'
    CONSUMER_SECRET ='<Insert your key>'
    OAUTH_TOKEN = '<Insert your token>'
    OAUTH_TOKEN_SECRET = '<Insert your token>'

    auth = twitter.oauth.OAuth(OAUTH_TOKEN,
                               OAUTH_TOKEN_SECRET,
                               CONSUMER_KEY,
                               CONSUMER_SECRET)

    twitter_api = twitter.Twitter(auth=auth)
    return twitter_api

#-----------------------------------------------
# Your code starts here
# Please add comments or text cells in between
# to explain the general idea of each block of the
# code.  Please feel free to add more cells below
# this cell if necessary.
```

In this example we provide a skeleton that allows the students to focus on the objective of analyzing tweets and hashtags with frequency analysis and not have to struggle with the details of Twitter authentication. Using Python, and the skeleton code provided by the instructors, the student were able to gather and

analyze many thousands of tweets and learn important lessons about data gathering, data APIs, data storage, and basic analytics.

Case Study Two

Building upon the skills gained in the first case study, the second case study asks the students to analyze the MovieLens 1M Data Set [Mov15], which contains data about how users rate movies. The key learning objectives are to analyze the data set, make conjectures, support or refute those conjectures with data, and use the data to tell a compelling story. In particular, the students are not only asked to perform several technical tasks, but they must also propose a business question that they think this data can answer. In effect, they are expected to play the role of a Data Scientist at a movie company and they must convince "upper management", who are not presumed to be technically minded, that their conjecture is correct.

While a seemingly tall order for only the second case study, Python again shows its utility. In particular, just as in case study 1, the assignment is provided in an IPython notebook, and the student is required to submit their work in the same format, thereby leveraging the skills learned in the first case study.

However, in this case study we introduce several important Python libraries that support Data Science including Numpy [Wal11], matplotlib [Hun07], and, perhaps most importantly, pandas [McK10]. As is perhaps well known to the readers of this text, Numpy provides a vast selection of routines for numerical processing, including powerful array and matrix/vector classes, while matplotlib allows for plotting of data and generation of compelling figures. Finally, pandas provides many tools for data processing, including a structure called a DataFrame (inspired by a data structure with the same name in the R language [RCT13]), which facilitates many data manipulations. Note, we are certainly not the first to consider this collection of libraries to be important for Data Science, and this particular case study was inspired by the excellent book "Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython", by Wes McKinney [McK12] (which is required reading for this particular assignment).

Many of the tasks in this case study revolve around question like:

- How many movies have an average rating over 4.5 overall?
- How many movies have an average rating over 4.5 among men? How about women?
- How many movies have a *median* rating over 4.5 among men over age 30? How about women over age 30?
- What are the ten most popular movies given a reasonable, student derived, definition of "popular"?

and the visualization of the data by way of:

- Plotting a histogram of the ratings of all movies.
- Plotting a histogram of the *number* of ratings each movie received.
- Plotting a histogram of the *average rating* for each movie.
- Plotting a histogram of the *average rating* for movies which are rated more than 100 times.
- Making a scatter plot of men versus women and their mean rating for every movie.
- Making a scatter plot of men versus women and their mean rating for movies rated more than 200 times.

Note, there are a number of important learning objectives that we wish to support. First, several terms are, intentionally, only vaguely defined in the assignment. For example, the precise definition of "popular" is left to the student to derive. As is often the case is real world Data Science, one of the key first steps of analysis is to decide precisely what the question of interest is. Second, the student is expected to make hypotheses or conjectures based upon the definitions they come up with. For example, the student might conjecture that men's and women's rating for certain genres are highly correlated, while for other genres their ratings more independent. Finally, the students must try to either prove, or just as interestingly, disprove their conjectures based upon the data.

Diving a bit more deeply into some of the specific functionality that we leverage in Python, we note that pandas [McK10] is particularly useful for these kinds of data analysis questions. In particular, to any Python aficionado, it is likely to be clear that there are many ways to process the data to answer the questions above, ranging from the brute force to the elegant.

To begin, we note that the MovieLens 1M Data Set itself is actually provided in three different files. First is a file containing the information regarding individual users, indexed by a unique *user_id*. Second is a file containing the information regarding each movie, indexed by a unique *movie_id*. Finally, and perhaps most importantly, is a file which contains ratings (and time stamps) indexed by a pair of *user_id* and *movie_id*.

Already we can perceive a thorny issue. Clearly, the questions of interest can only be answered by appropriate cross referencing between these three files. For example, all three files must be referenced to answer a question as seemingly straight forward as "how many action movies do men rate higher than 4?" While perhaps not too troublesome for students who are adept programmers, the cross referencing between the files presents an unnecessary impediment to less proficient students and overcoming this sort of impediment does not support the learning goals for this assignment.

Of course, a straightforward answer would be for the instructors to preprocess the data appropriately. However, using the power of Python one can easily arm the students with a general tool, while at the same time avoiding unnecessary hurdles. In particular, pandas has a merge function [PMe15] that provides exactly the required functionality in a quite general framework. In particular, one can use the code below to easily merge the three data files into a single DataFrame.

```
import pandas as pd
#--------------------------------------------

# Read in the user data into a DataFrame
unames = ['user_id', 'gender', 'age',
          'occupation', 'zip']
users = pd.read_table('ml-1m/users.dat',
                      sep='::', header=None,
                      names=unames)

# Read in the rating data into a DataFrame
rnames = ['user_id', 'movie_id',
          'rating', 'timestamp']
ratings = pd.read_table('ml-1m/ratings.dat',
                        sep='::', header=None,
                        names=rnames)

# Read in the movie data into a Data Frame
mnames = ['movie_id', 'title', 'genres']
movies = pd.read_table('ml-1m/movies.dat',
                       sep='::', header=None,
                       names=mnames)

# Merge all the data into one DataFrame
data = pd.merge(pd.merge(ratings,
```

```
                                users),
                    movies)
```

Of course, even once the data files have been merged, there are many places where a student might fall astray. Fortunately, pandas provides another tool which allows for elegant and compact code, namely the *pivot-table*. For example, one can imagine writing complicated loops and conditionals to perform the task of printing out all movies that have a median rating of 5 by men or women. However, using pivot-tables, such a question can be answered with just three lines of code (using the Python 2 "print" statement versus the Python 3 "print()" function):

```
# Create a pivot table to aggregate the data
mean_ratings = data[data['age'] > 30].\
     pivot_table(values='rating',
                 rows='title',
                 cols='gender',
                 aggfunc='median')
# Only print out movies with at least one rating
print (mean_ratings[mean_ratings['M'].notnull()].\
     sort('M',ascending=False)['M'] > 4.5).nonzero()
print (mean_ratings[mean_ratings['F'].notnull()].\
     sort('F',ascending=False)['F'] > 4.5).nonzero()
```

Of course, one might be tempted to argue that having students develop their own code, rather than leveraging such *black box* routines leads to a deeper learning experience. While we certainly appreciate this point of view, we wish to emphasize that the class in question is an introductory Data Science class, and not a programming or data structure class. Accordingly, using Python, and the powerful features of libraries such as Pandas, allows us to focus on the Data Science learning goals, while at the same time allowing the students to utilize large scale, real world, and sometimes messy data sources. This theme of using Python to allow for focused learning goals, using real world data, is a key message of this text.

Case Study Three

The third case study is substantially more challenging than the second case study, but builds on the foundations already laid down. While case study two focused on analyzing *numerical* movie reviews, case study three focuses on detecting positive and negative reviews from raw text using natural language processing.

In particular, in case study three, the class turns its attention to the Movie Review Data v2.0 from http://www.cs.cornell.edu/people/pabo/movie-review-data. This data set contains written reviews of movies divided into positive and negative reviews, and the goal is to learn how to automatically distinguish between the two cases.

Of course, tackling such problems is well known to be difficult, and there are many open research problems in this domain. On the other hand, such problems are clearly of importance in many domains, and it is not at all difficult to get students interested in solving them. The question remains, how can students in their very first Data Science class be expected to approach such difficult and important problems, and still be able to make meaningful progress? Of course, the answer is, again, Python.

In particular, we base this case study on the excellent scikit-learn [Ped11] Python library. Scikit-learn provides easy to use and efficient tools for data analysis. Most importantly, it provides routines for many important Data Science concepts such as machine learning, cross validation, etc. In fact, this case study is inspired by the scikit-learn tutorial "Working With Text Data" which can be found at http://scikit-learn.org/stable/tutorial/text_analytics/working_with_text_data.html.

Following our theme of leveraging Python to quickly get to interesting Data Science problems, the students in case study three are encouraged to start their work based upon various examples provided in the scikit-learn library. In particular, the students leverage the exercise_02_sentiment.py files from the directories:

- doc/tutorial/text_analytics/skeletons/
- doc/tutorial/text_analytics/solutions/

One version of the file is merely a skeleton of a natural language processing example, while the other contains the full source code.

For DS501 there are two key learning goals for this case study. First, the students need to derive *features* from the raw text that they feel would be useful in predicting positive and negative sentiments. Second, they must make predictions by processing these features using a variety of supervised machine learning algorithms.

Feature Generation: Classically, rather than attempting to do machine learning on raw text, Data Science practitioners will first process the raw text to derive features for downstream processing. A detailed description of text feature generation is beyond the scope of the current text (the interested reader may see [Raj11], and references therein, for more details). However, Python and scikit-learn [Ped11] provide easy access to the exact functionality required by the students by way of the TfidVectorizer class which implements the term frequency–inverse document frequency (TF-IDF) statistic [Raj11]. For our purposes we merely observe that there are several parameters that the student can explore to get a feel for feature generation from raw text, including *min_df* and *max_df* parameters (which control thresholds on document frequencies) and ngram_range (which controls how many words are conglomerated into a single token). Experimenting with these parameters provide many important insights for feature generation from real world text data, not the least of which is that large values of ngram_range may take a long time to run.

Supervised Machine Learning: Now, given a collection of reviews, each represented by a set of features sometimes called *predictors*, one can imagine many interesting problems. For example, a classic problem in machine learning involves using a set of reviews which have appropriate labels (in this case positive or negative) to *predict* labels of other reviews which do not already have labels. This process is called *supervised* machine learning. The idea is that the labeled data is used to *supervise* the training of an algorithm which, after training, can attempt to compute labels just from the raw features. Again, supervised machine learning is a vast subject, and space does not allow us to treat the subject even at the more superficial level here (the interested read may see [Fri01], [Jam13], [Bis06], and references therein, for more details). However, we will note that scikit-learn provides functions and classes for many standard algorithms, allowing the students to become familiar with important machine learning and Data Science concepts, without being expected to have too many prerequisites. For example, scikit-learn provides access to classic and powerful algorithms such as K-nearest neighbors, support vector classifiers, and principal component analysis [Fri01], [Jam13], [Bis06].

Using such routines, several important learning objectives can be supported, such as error estimation, by way of techniques such as cross-validation and confusion matrices. In fact, one particularly effective learning experience revolved around the following challenge. Using their favorite technique the student is asked to

find a two dimensional plot of the data where the positive and negative reviews are separated. While easy to state, practitioners of natural language processing will recognize that actually solving the problem is exceptionally difficult, and the instructors admit that they are not in possession of an actual solution. For some students this may be the first time they have been presented with a problem they are expected to tackle for which their instructor *does not know the solution*. The student's ability to begin thinking about such open problems so early in their Data Science career is substantially supported by a language such as Python and the libraries it provides.

Case Study Four

The final case study, and in some sense the capstone of the class, revolves around the Yelp Dataset Challenge http://www.yelp.com/dataset_challenge. This case study involves a large data set with approximately 42,153 business, 252,898 users, and 1,125,458 reviews in Phoenix, Las Vegas, Madison, Waterloo and Edinburgh.

Again, building off of the previous case studies, the students are expected to process the data, generate statistics, process reviews using TfidVectorizer, etc. However, for this case study the students are also expected to process the data using MapReduce [Dea08]. As is well known in certain circles, MapReduce is a programming model (with various implementations) for distributed processing of large scale data sets. Distributed processing models, and MapReduce in particular, are essential elements of modern Data Science and we would have felt remiss if students in a class such as DS501 were not able to experience, at least at some level, the beauty and power of such methods.

Fortunately, and we fear that we are repeating ourselves, Python provides precisely the functionality we required. In particular, there are several MapReduce interfaces for Python, and the mrjob package [MrJ15] was chosen to support the students learning objectives. This package is especially useful in a classroom environment since it can be used locally on a single computer (for testing) and in a cluster environment. Accordingly, the students can learn about MapReduce with the need for access to large scale computing resources.

Introductory Data Science: Final Thoughts

Of course, Python is not the only choice for an Introductory Data Science course. For example, the scripting language R [RCT13] is also a popular choice which has also used successfully in the Data Science curriculum. In particular, R offers much, if not all, of the functionality mentioned above, including interfaces to MapReduce [Usu14]. Accordingly, the choice of language for such a class may be considered a matter of taste.

However, there is mounting evidence of Python's growing popularity within the Data Science community [Sin14] and the software development community at large [OGr14]. Perhaps, if we may be forgiven a small measure of Python bias, we will merely emphasize that Python's popularity cuts across many problem domains. For example, the authors are not aware of any customer relationship management applications, system administration tools, or web servers[1], to name just a handful of areas outside of statistical and data analysis, currently being developed in R, nor many other domains in which Python has made inroads. The fact that Python is as generally applicable as it is, while perhaps still being just as popular as R for Data Science, is a testament to its advantages.

MA542 Regression Analysis

Leaving aside introductory classes, we now make brief mention of Python's usefulness in more advanced classes. In particular, one of the authors recently taught a Regression Analysis class (using the text *Applied linear regression models* [Kut04]), for the first time, with all of the development in the class being Python focused. Regression Analysis is a more advanced class with a greater concentration of mathematically focused students who take the class. In addition, many students were first time Python users, with the majority of the exceptions being Data Science students who had taken DS501–"Introduction to Data Science" previously.

Just as in DS501, Numpy [Wal11], matplotlib [Hun07], and pandas [McK10] provided almost all of the functionality the students required for the learning objectives in the class. Also as in DS501, the instructor can use Python and its vast array of libraries to carefully control the difficulty and scope of assignments. In fact, one of the challenges in this class was that Python perhaps does *too good* of a job providing functionality to the students.

In particular, Python provides so many libraries that, for example, many of the computationally oriented homework questions are trivially answerable if the students look hard enough. Accordingly, as an instructor, one needs to be careful that the ground rules are set correctly so that the learning objectives are achieved. For example, if the learning objective is for the student to understand the details of a particular mathematical concept, say the *normal equations*, rather than just a numerical procedure, such as *linear regression* on a particular data set, then the expectations for the assignment need to be carefully delineated.

Accordingly, to maintain the integrity of the learning objectives, a tactic used by the authors was to carefully delineate what parts of the assignment are allowed to be Python "black boxes" and which parts must be hand coded. In addition, we require the students to hand in their Python code, even though the code itself is *not* graded. The learning objectives of the class are mathematical, and not programming. Accordingly, the quality of the implementations is not a focus. However, having access to the code allows the instructor to verify that the desired learning objectives are being met.

As one final note, one tactic that was quite successful was to encourage the students to check their hand coded results against those provided by any black box routine they are able to use. It was quite useful for the students in debugging their own implementations and understanding of the mathematical concepts. It was quite empowering for the students when their answers would exactly match those of the black box. They then appreciated that they understood, in a deep way, what the "professionals" were doing.

Student research projects and theses

Python has had an important part to play in the authors' research since 1997 [Paf99]. Currently, we perform research involving, and mentor students in, several topics revolving around semi-supervised and unsupervised machine learning applied to several

1. We would be remiss not to at least mention the quite beautiful R web application framework Shiny [Shi14]. However, we believe our point still stands.

different domains, with a focus on cyber-defense (see, for example, [Paf13]). Accordingly, one of our key goals is to support the training of the next generation of researchers in these domains. We will not burden the reader with the mathematical details of our research directions, but just observe that our work, and the work of our students, draws from a laundry list of ideas from mathematics, statistics, and Data Science, including convex optimization [Boy04], deep learning [Den14], graphical models [Lau96], and scientific visualization [War10].

For the current purpose, it is merely important to note that Python libraries are available that support *all of these subject areas*. For example, we have:

- Statistical modeling: Statsmodels [StM15]
- Convex optimization: cvxopt [Dah06], CVXPY [Dia14]
- Deep learning: Theano [Ber11]
- Graphical models: libpgm [Kar14], pgmpy [Pgm15]
- Scientific visualization: Mayavi [Ram11], Matplotlib [Hun07], Bokeh [Bok15], Seaborn [Was14]

Accordingly, students who are trained in classes such as DS501 and MA542 can leverage that training to get a running start on their research subjects. Perhaps this is the single biggest advantage of using a language such as Python from the earliest stages of Data Science education. In addition to being easy to learn [Lut13], and providing access to many libraries that support Data Science education, Python provides ready access to a broad swath of cutting edge Data Science research.

We use all of these libraries in our work, where we are especially interested in large scale robust principle component analysis [Can11], [Paf13] and non-linear dimension reduction problems [Lee07]. These problem domains are mathematically subtle, computationally intensive, and lead to, in the authors' opinion, rather intriguing visualization problems, which are also supported by Python through libraries such as Mayavi, as shown in the figure below.

Beyond the mathematical research that Python supports, there are a vast array of computational resources that are at the fingertips of those well versed in Python. For example, our research group is interested in developing algorithms for modern distributed supercomputers that leverage GPUs to accelerate computations. Again, Python displays its usefulness with the pycuda [Klo12] and mpi4py [Dal08] libraries.

As one can see, Python is an effective tool for cutting edge Data Science research. Of course, there are many such tools, and often the specific choice of language for Data Science research is a matter of taste. However, we would respectfully submit that few languages have the broad range of support for Data Science research that Python provides.

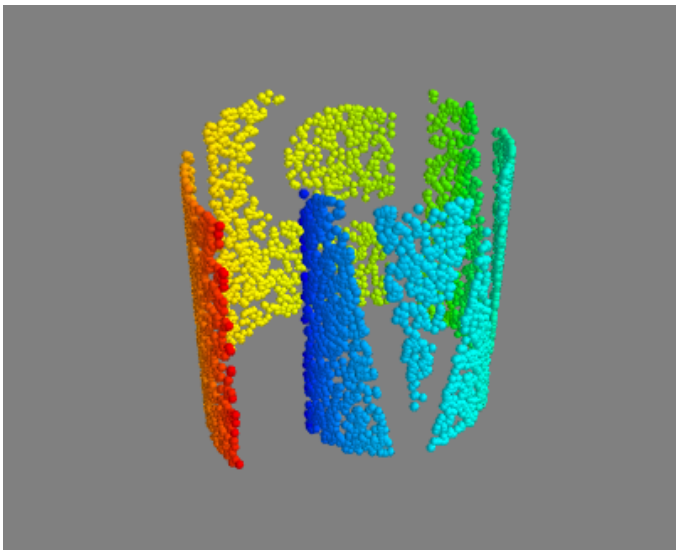

Fig. 1: *An example of a 3D visualization of a manifold using Mayavi [Ram11]. In our work we attempt to detect the non-linear dependencies in such data, even when the data is noisy and unevenly distributed. In this synthetic example we see data which is intrinsically two-dimensional (since it is a flat surface) embedded in a three-dimensional space. The two-dimensional structure is non-trivial to detect based upon the non-linear nature of the data, noise, and regions with no data points.*

Conclusion

We have discussed how Python can be used throughout the entire life cycle of a graduate program in Data Science. Python is easy to learn and use, but it also provides access to a vast array of libraries for cutting edge Data Science research. In particular, IPython notebooks, scikit-learn, NumPy, SciPy, and pandas can be used to support many aspects of the Data Science education. These libraries allow instructors to focus on desired learning objectives, while leaving many of the less important details to the libraries. Having access to such libraries allow interesting problems to be addressed early in the educational process and the experience gained with such Python libraries supports the student's own software development, analysis, and research throughout their academic career and beyond.

Acknowledgments

We wish to gratefully acknowledge several people without whom this monograph would not have been possible. In particular, the authors are deeply grateful to the other members of Data Science Steering Committee at WPI:

- Prof. Elke Angelika Rundensteiner (Director of Data Science)
- Prof. Mohamed Eltabakh
- Prof. Eleanor T. Loiacono
- Prof. Joseph D. Petruccelli
- Prof. Carolina Ruiz
- Prof. Diane M. Strong
- Prof. Andrew C. Trapp
- Prof. Domokos Vermes
- Prof. Jian Zou

without whose tireless efforts the WPI Data Science program would not be what it is today.

References

[Beh11] Stefan Behnel, Robert Bradshaw, Craig Citro, Lisandro Dalcin, Dag Sverre Seljebotn and Kurt Smith. *Cython: The Best of Both Worlds*, Computing in Science and Engineering, 13, 31-39 (2011), DOI:10.1109/MCSE.2010.118

[Ber11] Bergstra, James, et al. *Theano: Deep learning on gpus with python*. NIPS 2011, BigLearning Workshop, Granada, Spain. 2011. http://deeplearning.net/software/theano/ [Online; accessed 2015-06-08].

[Bis06] Bishop, Christopher M. *Pattern recognition and machine learning*. Springer, 2006.

[Bok15] *Bokeh* (2015), http://bokeh.pydata.org/en/latest/ [Online; accessed 2015-06-17].

[Boy04] Boyd, Stephen, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Can11] Candès, Emmanuel J., Li, Xiaodong, Ma, Yi, and Wright, John. *Robust principal component analysis?*. Journal of the ACM (JACM) 58.3 (2011): 11.

[Dah06] Dahl, Joachin, and Lieven Vandenberghe. *Cvxopt: A python package for convex optimization.* Proc. eur. conf. op. res. 2006. http://cvxopt.org [Online; accessed 2015-06-08].

[Dal08] Dalcín, Lisandro, Paz, Rodrigo, Storti, Mario, and D'Elía, Jorge (2008). *MPI for Python: Performance improvements and MPI-2 extensions*. Journal of Parallel and Distributed Computing, 68(5), 655-662.

[Dea08] Dean, Jeffrey, and Sanjay Ghemawat. *MapReduce: simplified data processing on large clusters*. Communications of the ACM 51.1 (2008): 107-113.

[Den14] Deng, Li, and Dong Yu. *Deep learning: methods and applications*. Foundations and Trends in Signal Processing 7.3–4 (2014): 197-387.

[Dia14] Diamond, Steven, Eric Chu, and Stephen Boyd. *CVXPY: A Python-embedded modeling language for convex optimization*, version 0.2." (2014). http://cvxpy.org [Online; accessed 2015-06-08].

[Fri01] Friedman, Jerome, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*. Vol. 1. Springer, Berlin: Springer series in statistics, 2001.

[Jam13] James, Gareth, et al. *An introduction to statistical learning*. New York: springer, 2013.

[Hun07] John D. Hunter. *Matplotlib: A 2D Graphics Environment*, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

[Kar14] Karkera, Kiran R. *Building Probabilistic Graphical Models with Python*. Packt Publishing Ltd, 2014.

[Klo12] Andreas Klöckner, Nicolas Pinto, Yunsup Lee, Bryan Catanzaro, Paul Ivanov, Ahmed Fasih, *PyCUDA and PyOpenCL: A scripting-based approach to GPU run-time code generation*, Parallel Computing, Volume 38, Issue 3, March 2012, Pages 157-174.

[Kut04] Kutner, Michael H., Chris Nachtsheim, and John Neter. *Applied linear regression models*. McGraw-Hill/Irwin, 2004.

[Lau96] Lauritzen, Steffen L. *Graphical models*. Oxford University Press, 1996.

[Lee07] Lee, John A., and Michel Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[Lut13] Lutz, Mark. *Programming python*. 5th edition, O'Reilly Media, Inc., 2010.

[McK10] McKinney, Wes. *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

[McK12] McKinney, Wes. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython* .O'Reilly Media, Inc., 2012.

[PMe15] *Merge, join, and concatenate* (2015), http://pandas.pydata.org/pandas-docs/stable/merging.html [Online; accessed 2015-06-08].

[Mil11] K. Jarrod Millman and Michael Aivazis. *Python for Scientists and Engineers, Computing in Science & Engineering*, 13, 9-12 (2011), DOI:10.1109/MCSE.2011.36

[Mov15] *MovieLens* (2015), http://grouplens.org/datasets/movielens/ [Online; accessed 2015-06-08].

[MrJ15] *mrjob* (2015), https://pythonhosted.org/mrjob/ [Online; accessed 2015-06-08].

[Mye07] Myers, Christopher R., and James P. Sethna. *Python for education: Computational methods for nonlinear systems.* Computing in Science & Engineering 9.3 (2007): 75-79.

[OGr14] O'Grady, Stephen. *The RedMonk Programming Language Rankings: January 2014* (2014), http://redmonk.com/sogrady/2014/01/22/language-rankings-1-14/ [Online; accessed 2015-06-08].

[Oli01] Jones, Erid, Oliphant, Travis, Peterson, Pearu, et al. *SciPy: Open Source Scientific Tools for Python*, 2001-, http://www.scipy.org/ [Online; accessed 2015-05-31].

[Oli07] Travis E. Oliphant. *Python for Scientific Computing*, Computing in Science & Engineering, 9, 10-20 (2007), DOI:10.1109/MCSE.2007.58

[Paf99] Paffenroth, Randy C. *VBM and MCCC: Packages for objected oriented visualization and computation of bifurcation manifolds.* Object Oriented Methods for Interoperable Scientific and Engineering Computing: Proceedings of the 1998 SIAM Workshop. Vol. 99. SIAM, 1999.

[Paf13] Paffenroth, Randy, Du Toit, Philip, Nong, Ryan, Scharf, Louis, Jayasumana, Anura. P., and Bandara, Vidarshana. (2013). *Space-time signal processing for distributed pattern detection in sensor networks*. Selected Topics in Signal Processing, IEEE Journal of, 7(1), 38-49. Chicago

[Ped11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. Scikit-learn: Machine Learning in Python, Journal of Machine Learning Research, 12, 2825-2830 (2011)

[Per07] Fernando Pérez and Brian E. Granger. IPython: A System for Interactive Scientific Computing, Computing in Science & Engineering, 9, 21-29 (2007), DOI:10.1109/MCSE.2007.53

[Pgm15] *Python Library for Probabilistic Graphical Models (pgmpy)* (2015), https://github.com/pgmpy/pgmpy [Online; accessed 2015-06-24].

[Ptw15] *Python Twitter* (2015), https://code.google.com/p/python-twitter/ [Online; accessed 2015-06-08].

[Raj11] Rajaraman, Anand and Jeffrey David Ullman. *Data Mining*, Mining of Massive Datasets. 1st ed. Cambridge: Cambridge University Press, 2011. pp. 1-17. Cambridge Books Online. http://dx.doi.org/10.1017/CBO9781139058452.002 [Online; accessed 2015-06-08].

[Ram11] Ramachandran, Prabhu and Varoquaux, Gael, *Mayavi: 3D Visualization of Scientific Data* IEEE Computing in Science & Engineering, 13 (2), pp. 40-51 (2011)

[RCT13] R Core Team (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. http://www.R-project.org/.

[Rus13] Russell, Matthew A. *Mining the Social Web: Data Mining Facebook, Twitter, LinkedIn, Google+, GitHub, and More*. O'Reilly Media, Inc., 2013.

[Shi14] RStudio, Inc. *shiny: Easy web applications in R* (2014), http://shiny.rstudio.com [Online; accessed 2015-06-08].

[Sin14] Singh, Harpreet. *Is Python Becoming the King of the Data Science Forest?* (2014), http://www.experfy.com/blog/python-data-science/ [Online; accessed 2015-06-08].

[Sta00] Stajano, Frank. *Python in education: Raising a generation of native speakers.* Proceedings of 8th International Python Conference. 2000.

[StM15] *Statsmodels* (2015), http://statsmodels.sourceforge.net/ [Online; accessed 2015-06-17].

[Twi15] *The Streaming APIs Overview* (2015), https://dev.twitter.com/streaming/overview [Online; accessed 2015-06-08].

[Usu14] Usuelli, Michele. *An Example of MapReduce with rmr2* (2014), http://www.milanor.net/blog/?p=853 [Online; accessed 2015-06-08].

[Wal11] Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

[War10] Ward, Matthew, Georges Grinstein, and Daniel Keim. *Interactive Data Visualization: Foundations, Techniques, and Applications*, AK Peters, Ltd., Natick, MA (2010).

[Was14] Waskom, Michael et al.. (2014). *seaborn: v0.5.0* (November 2014). Zenodo. 10.5281/zenodo.12710