STAT 29000 Project 2

Topics: python3, numpy, scipy, pandas

Motivation: Knowing Python has many useful, built-in utilities to solve data driven problems is great, however, we would be remiss to not explore some of the core libraries for scientific computing. There are about 5 extremely popular packages to do scientific computing in Python: numpy, scipy, pandas, tensorflow, and last but certainly not least pytorch. Each package takes some getting used to and will most likely involve a good deal of searching the documentation to find methods that do what you want. With that being said, gaining experience using these packages will set you up for success in industry and academics.

Context: We are gaining familiarity using Python to solve data driven problems. In this project we will get some experience with 3/5 of the aforementioned scientific computing packages.

Scope: Python is a top programming language for statistics, data science, and machine learning applications. numpy, scipy, and pandas are scientific computing packages that enable users to do perfomant, repeatable data analysis.

If you were to look at the versions of `numpy`, `scipy`, and `pandas` on scholar, you will see that they are outdated. In scholar, open up a shell and type the following:

```
# the python interpreter notebook.scholar.rcac.purdue.edu uses is /opt/anaconda3/bin/python
# this will show the versions of the preinstalled packages, as you can see,
# numpy (1.12.1), scipy (0.19.0), and pandas (0.20.1), all need updated
/opt/anaconda3/bin/python -m pip list

# we will go ahead and install the up-to-date versions in our local site-packages folder
python3.6 -m pip install pandas scipy numpy --user

# you can see the packages here
ls ~/.local/lib/python3.6/site-packages
```

In addition, we must install a missing package called backoff. In scholar, open up a shell and type the following:

```
python3.6 -m pip install backoff --user
```

For the following questions, please copy and paste the following code at the top of your Jupyter notebook. We are simply importing useful Python modules that are required for this project. For now, think of these statements as the `library()` function in R. We will address this in more detail at a later date. Be sure to replace "PURDUEALIAS" with your Purdue username. Note that if you get an error after running this chunk of code, you may need to click on Kernel->Restart.

```
# the following two lines tell notebook.scholar.rcac.purdue.edu's default python interpreter
# that it should look in ~/.local/lib/python3.6/site-packages for packages as well
# if you do not include these two lines at the very top of your notebook, you won't
# be able to import and use the packages we installed earlier.
import sys
sys.path.append("/home/PURDUEALIAS/.local/lib/python3.6/site-packages")

import io
import requests
import numpy as np
import pandas as pd
from PIL import Image
```

```
from scipy import stats
import backoff
```

The `/class/datamine/data/spring2020/stat29000project02examples.ipynb` notebook will provide you with useful examples that will help you solve problems given in this project.

Use the template found here or on scholar: `/class/datamine/data/spring2020/stat29000project02template.ipynb` to submit your solutions.

After each problem, we've provided you with a list of keywords. These keywords could be package names, functions, or important terms. Each keyword will point you in the right direction when trying to solve the problem, and give you accurate terminology that you can further look into online.

Official documentation links: numpy, scipy, pandas

Question 1: numpy

1a. *(1 pt)* `numpy` is a core scientific computing library. Among other things, it provides n-dimensional arrays, and a variety of efficient linear algebra operations. Both the scipy and pandas libraries are built using numpy. One of the most useful features is the ability to randomly sample from a variety of different statistical distributions.

You work for a company that has developed a model that accepts ratings data as one of its inputs. You were asked to generate data for this input for simulations. Generate an array of 100000 random samples from a normal distribution with a mean of 4 and variance of .06.

Verify that the data you've sampled could have likely come from a normal distribution with a mean of 4 and variance of .06.

Hint: Be careful and read the descriptions of the arguments to ensure you are providing the functions the correct inputs.

Keywords: *random, normal, sample*

1b. *(1 pt)* The group had a meeting and one of the statisticians determined that the ratings actually seem to follow a gamma distribution with the same parameters (mean of 4 and variance of .06). Generate an array of 100000 random samples from a gamma distribution with a mean of 4 and variance of .06.

Verify that the data you've sampled could have likely come from a gamma distribution with a mean of 4 and variance of .06.

Hint: k is shape, θ is scale

$$mean = k\theta$$
$$variance = k\theta^2$$

Keywords: *random, gamma, sample*

Question 2: judging books by their colors

2a. *(1 pt)* In this series of questions we are going to explore the books.csv dataset further using pandas and scipy. In the first project we read in the dataset using the requests library and csv library. This time, use the `pandas` package to read the dataset in. As a reminder the data can be found here. How many rows and how many columns does the dataset have?

Keywords: *shape, pandas, read_csv*

2b. *(1 pt)* Below, we've provided you with a series of comments and hints to help you write python code to create a new column called `cover_images`. This column will contain the book cover image from goodreads.com for each book in the dataset. For the sake of practicality, we are going to ask you to take a subset of the first 10 rows of data, and download the data for those 10 books only. In (2c), we will provide you with directions to load the full dataset containing all of the cover images. The links to the images can be found in the column called `image_url`.

```
# First import some packages to download webpages, parse web pages, and deal with image files
import requests
import io
from PIL import Image

# Since we are scraping a live website, let's not overwhelm the website and get blocked. We will
# use exponential backoff when scraping webpages.
import backoff
@backoff.on_exception(backoff.expo, requests.exceptions.RequestException)
def get_url(url):
  return requests.get(url)

# this scrapes the webpage from the internet
response = get_url(image_url)

# get the stream of the content of the response object
image_file = io.BytesIO(response.content)

# let's convert to an image
img = Image.open(image_file)

# you can view the image to confirm things are working
img.show()
```

Keywords: *iterrows*

2c. *(1 pt)* We've provided you with the full dataset that has been serialized. You can run the code below in order to load the dataframe containing the full dataset.

```
import pandas as pd

# to load the data at a later date
myDF = pd.read_pickle("/class/datamine/data/spring2020/books_with_images")

# display a cover image
myDF.loc[0, 'cover_images'].show()
```

It has been shown that cartons of ice cream where warm colors are present sell fewer than those without. Your manager read this and wants to know whether or not the color of the book covers has anything to do with the ratings – specifically if ratio of warm to cool colors effects a book's average rating. We've provided examples of getting matrices representing the pixels in an image. Use these example to further supplement our dataset with 2 new variables: `percent_warm`, and `percent_cool`. Use the following function to determine whether or not a pixel is warm or cool.

```
def is_warm(r,g,b):
    return r > g or (r >= 128 and g > b)
```

Keywords: *PIL Image size, PIL convert RGB, PIL getpixel*

2d. *(1 pt)* Get the row(s) of the dataframe where the cover has the maximum % of warm colors, call the

resulting dataframe `warmest`. Get the row(s) of the dataframe where the cover has the maximum % of cool colors, call the resulting dataframe `coolest`. How many rows are in the new dataframes? Run the code below to see a density plot of `percent_warm`:

```
data['percent_warm'].plot.kde()
```

Hint: This may be more than 1 row if there is a tie for maximum %.

Keywords: *loc method, min, max, len*

2e. *(1 pt)* (2d) provides enough oddity to explore the warmest covers. Get a sample of 5 random covers from the `warmest` dataframe. Loop through the sampled books and display the cover. What do you notice?

Keywords: *show, sample*

2f. *(1 pt)* In (2e) we make an observation about what we can presume to be a large percentage of our `warmest` dataframe. Let's remove every row from our `data` dataframe where our observation holds true.

Hint: You can compare two Images using the `==` operator. For example, if the images at index 0 and 1 are the same, `data['cover_images'].iloc[0] == data['cover_images'].iloc[1]` will result in `True`.

Hint: Be careful. We learned to sample a subset in (2e). I'd recommend sampling a small subset to test and confirm your method works as you expect.

Hint: There are multiple approaches to this problem.

Keywords: *drop, index*

2g. *(1 pt)* Create a list of book titles where the percentage of warm colors is in the range from 49-51%. How many titles are in the list? Print the first 5 `original_title`'s elements in the list sorted alphabetically.

Keywords: *loc, between*

2h. *(1 pt)* Calculate the spearman r between the percent of warm colors in the book covers and the average rating.

Keywords: *scipy, stats, spearmanr*

Project Submission:

Submit your solutions for the project at this URL: https://classroom.github.com/a/FRNWXq9I using the instructions found in the GitHub Classroom instructions folder on Blackboard.

Important note: Make sure you submit your solutions in both .ipynb and .html formats.