

Final Project

James Dedon

SEIS736

Examining Peer Institutions in Higher Ed

Getting Started

When the assignment was first assigned I started digging through datasets online. Just a few weeks before Google had launched its Dataset Search tool, which threw me into a deep rabbit hole of data sets that made me extremely nervous about the project and my ability to complete the job successfully. Most of the datasets that I found earlier on were cryptic tables of binary, hex and numeric data that I just didn't understand.

I came across the College Scorecard dataset from the U.S. Department of Education. A version of the set is pre-compiled on Kaggle and documented with a full data dictionary and all that good stuff. Another more up to date version of the Scorecard is housed on the DoE website, but requires the user to compile the set into a single entity, as each year is separate.

The lack of documentation is a show-stopper for someone approaching a new-dataset with limited domain knowledge. Even with domain knowledge, lack of concise documentation can make things really difficult.

The College Scorecard dataset compiled data from the U.S. Department of Education as a tool to allow the public to make informed decisions about college choices. Looking into the data one can identify all sorts of indicators of school/graduate success, such as percentage of students receiving Pell grants, or retention rates for middle income family students. The dataset even contains binary data for all of the CIP code groups, which make up the classification system for different majors. This will make it possible to compare schools that offer fields like engineering vs schools that don't in terms of things like debt accumulated upon graduation and demographic differences between schools. Overall, I'm excited to be able to look into this dataset.

Dataset Structure

The data set contains merged files for each year from 1996 through 2016/17 in the most recent DoE version.

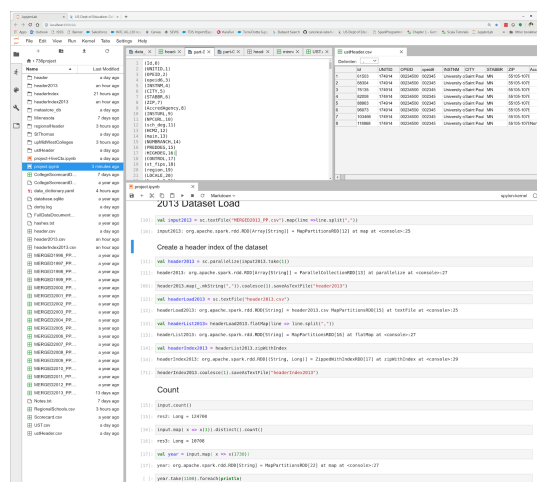
Because I went with the DoE version (more about that later), I had to do a series of CSV loads and then union them all together into one big RDD. I chose to merge 10 years of data resulting

in 74,181 rows and 1898 features. The year was mapped to the individual year RDD at load time.

Getting to Know The Data

Upon loading the data into an RDD the first thing I wanted to do was to find out exactly what is in the data and come up with a plan on how to go about analyzing the data. I decided to do the majority of my poking around in the data in a JupyterLab Notebook running the spylon kernel on Apache Spark 2.3.1.

I'm a big fan of running things in JupyterLab because it gives me quick access to everything and a canvas upon which I can lay out all of the necessary information needed to dive into the work. As you can see in the screenshot below, the tool allows for a layout of the header index, St. Thomas data, the files in the working directory, and the notebook running the Scala code in Spark.



Running highly iterative (read: broken) code in a command line interface is a big drag and having tools on your local machine to mitigate some of the pain in learning a new language is always welcome.

The first action that I did after loading the data was to create an index of the header of the csv, as the file has 1730 indicators and I need to be able to perform actions and transformations on those data points as needed.

To build the indexes, I originally exported the header files by doing a `take(1)` on the input RDD and saving it as a text file. This was because I was dumb

and was looking at my screen for too long. This is usually the case.

The truth is that I spend a good long time trying to find out how to find the index locations of the array items. A lot of the information that I could find online was for lists. It wasn't until most excellent teacher Professor Brad Rubin pointed me in the right direction towards the successful execution of the `zipWithIndex` transformation that I was able to compile the files as I wanted. Even then, I was still using my backward exported/transformed/imported method. It wasn't until the next day that I went back to clean that mess up.

Once I got the indexes up-to-snuff and began looking around in the data, I realized that a great share of the year indicator feature data was missing in the Kaggle file. It was at this point, due to this and the lack of newer data, where I made the call to switch to the DoE data and start

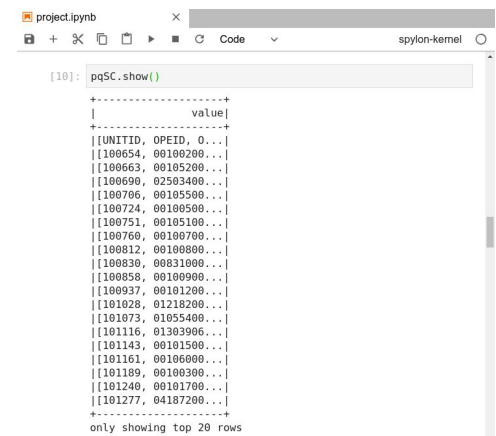
over. Upon compiling the main 10 year RDD, I recreated a UST RDD to verify the data is lined up and complete. Luckily, 10 years of data made up that RDD.

Converting to a Parquet table

I converted my compiled RDD to a Parquet table and saw a 5 fold reduction in storage size from the sum of the 10 standalone CSV files that make up the compiled RDD (1.5GB) to the stored Parquet file(s) as partitioned on my hard drive(294MB). That's something good there! I'm not sure exactly how that works, but from one of the videos that I watched online¹, I guess Parquet does some trimming of the strings and whatnot to store only what is needed to be able to build out the dataset. For example, if all of the strings in a column or index location begin with UST, it can remove that "UST", somehow store it away, and write only the individually identifying data needed to disk... At least that's how I'm understanding it to work. I'm not sure if that's part of the snappy compression algorithm, or a raw parquet feature.

After I initially created my parquet file and loaded it back in as a DF, I realized that the resulting DF was a single string stored as one value. As you might guess, this isn't what I was expecting or wanting. As you might also guess, I was utterly disappointed and heartbroken upon viewing the data frame rendering as seen to the right. I realized that this was due to the fact that it's not so straight-forward as converting an RDD to DF, even though you can totally do an `RDD.toDF()` command and get something out.

The fix for this is to create the data frame from a CSV file or other applicable input file format. The `read.csv()` function will interpret the headers and columns as needed.



```
project.ipynb | x | Code | sylon-kernel | O
[10]: pq5C.show()
+-----+
| value |
+-----+
|[UNITID, OPED, 0...|
|[00654, 00100200...|
|[00663, 00105200...|
|[00690, 02503400...|
|[00706, 00105500...|
|[00724, 00100500...|
|[00751, 00105100...|
|[00760, 00100700...|
|[00812, 00100000...|
|[00830, 00831000...|
|[00858, 00100900...|
|[00937, 00101200...|
|[01028, 01218200...|
|[01073, 01055400...|
|[01116, 01303900...|
|[01143, 00101500...|
|[01161, 00106000...|
|[01189, 00100300...|
|[01240, 00101700...|
|[01277, 04187200...|
+-----+
only showing top 20 rows
```



```
[6]: val indF = spark.read.csv("scoreCard10Year.csv")
indF.show()
```

laptop.

One thing that I wasn't ready for though was that the output was a series of almost 1900 "+"s and even more "-"s making up the line above the header line (see left). That really threw me off for a minute before I realized what was going on with that.

Now that I had to write the parquet files again and this is where I ran into some issues running it with locally running Jupyter. The write-out failed when my laptop ran out of memory and it all came crashing down in all of its pink job aborted report glory. This put me at the end of the road for moving forward on the

¹ Spark + Parquet In Depth: <https://databricks.com/session/spark-parquet-in-depth>

So, I moved the data up to the Databricks Community Cluster....

Databricks Platform and Alternative Tools

I watched the videos assigned for week 12 and decided that it would make sense to try out how things work on their clusters. The account is free, but offer no security for your data at the free tier. That's fine for my purposes here, so I decided to export a CSV to try out my code there. That would allow me to use a notebook environment on a cluster.

AWS also has the ability to fire up an instance of JupyterHub, but the time it would take to configure everything and get my browser pointed there, we'd probably be starting the spring term. So that wasn't happening.

Loading Data

The Databricks Community Edition, on the other hand let me create an account and start uploading my dataset in less than 5 minutes. Very nice!

Because I created my base working RDD by unioning 10 separate CSV files on my laptop, I decided to write out a new CSV that contained all 10 years of data in one file to make the upload and convert it to a DataFrame and then to Parquet. I initially uploaded all 10 csv files and was going to recompile the big RDD/DataFrame in the cloud, but I was confused by the data upload process in Databricks CE where it wants to convert everything upon upload either through their UI or through a notebook. Furthermore, there isn't a quick way to access your uploaded files through a GUI; only the tables that you create. This was disorienting.

At first I just went back to the upload screens, as you have access to the dbfs as a way to create your tables. However, you can access the file system via the notebook by creating a filesystem magic in a cell. I'm not sure it that's what the real term would be in this context, but that's what some of the kids these days are calling it when you run a python cell on a scala kernel and vise versa in a Jupyter notebook. You can do this by starting your cell with `"%fs"` followed by your normal command line `"ls"`, `"cp"` and all that fun stuff. You can also call the `dbutils` method in your scala code to access the file system without designating a cell to those functions.²

I uploaded the compiled unioned CSV file and created a data frame off of that. I then saved that dataframe to parquet. This was quick and easy.

I wanted to compare the sizes of the parquet file written in the cloud against the csv file, because the file I had previously written on my laptop wasn't split correctly and I didn't know if that would affect file size. Once again, we have a roughly 1.4GB csv file and a 231MB parquet

² Databricks File System <https://docs.databricks.com/user-guide/dbfs-databricks-file-system.html>

file. This was in-line with my previous experience, so the column splitting didn't affect it to a noticeable extent. The compression used was the default one in Databricks, snappy.

I then pared down my columns to a more reasonable size that's okay to look at. While it isn't really necessary, it makes a nice reference to go back to when looking for header names and for quick comparisons (right).

Parquet vs. CSV

Now that I have all of the data loaded and nicely pared down to a small subset for working with, it's time to compare the speed of the csv based dataframe to the parquet based dataframe. For my first test, I filtered my regional peers out again, just as I did in my Jupyter instance into a selection of region 4 universities with a Carnegie size and setting classification of 4 year, medium, primarily residential colleges and universities. Once I had my filtered dfs set up, I did a `show()` on the two of them. Each rdd only has 20 rows and 11 columns. The `show()` spawned 11 spark jobs to complete on the CSV storage and took roughly 20 seconds, but only 7 spark jobs to complete on the Parquet storage and 12 seconds. I have no idea why these two operations would have a different number of jobs, but they do. A simple `count()` on the data frames also displayed a significant speed advantage of the parquet dataframe with it completing in 3.35 seconds on parquet and 11.90 seconds on the CSV frame.

These were two relatively simple chains of select and filtering, so I willing to take the Pepsi™ challenge and go out on a limb to say that benefits will be even greater for larger, more complex chains of transformations and actions.

Data Types and Other Headaches

My next step is to start doing some comparisons between St. Thomas and other schools like and not like us. I did a regional SAT average comparison on an RDD early on in this exercise and had no difficulties in finding a solution to the problem of casting the strings which comprise my whole data set to integer or anything else. Data frames on the other hand are a different beast and I've given up on the conversion at the time of writing this paragraph, as the time needed to figure it out isn't going to be short enough to allow for the completion of this paper and study for the final exam. One day, maybe.....

The next day, it was! I figured out why I couldn't cast types to data frames on the NEXT day. What I didn't get was that Data Frames don't use the normal scala data types. It looks like

End 17

1 regionalPeers.show()

▼ (7) Spark Jobs

- Job 84 View (Stages: 1/1)
- Job 85 View (Stages: 1/1)
- Job 86 View (Stages: 1/1)
- Job 87 View (Stages: 1/1)
- Job 88 View (Stages: 1/1)
- Job 89 View (Stages: 1/1)
- Job 90 View (Stages: 1/1)

[UNITID]	[INSTNM]	[STABBR]	[REGION]	[CCSIZSET]	[RELAFFIL]	[ADM_RATE]	[TUTITFTE]	[PCTPELL]	[PCTFLOAN]	[2016]
155025	Esperia State Uni...	KS	4	13	NULL	0.8743	6948	0.3693	0.5541	[2016]
173124	Benjamin State Uni...	MN	4	13	NULL	0.6408	5321	0.3129	0.5677	[2016]
173065	Hamline University	MN	4	13	TA	0.7931	5244	0.3558	0.6855	[2016]
174558	Minnesota State U...	MN	4	13	NULL	0.5961	5783	0.2865	0.5924	[2016]
174817	Saint Mary's Univ...	MN	4	13	30	0.7924	1376	0.2986	0.6297	[2016]
174914	University of St...	MN	4	13	30	0.8266	1987	0.1716	0.5277	[2016]
175065	St. Catherine Univ...	MN	4	13	30	0.8083	1749	0.3791	0.741	[2016]
175078	Southwest Minneso...	MN	4	13	NULL	0.5419	3953	0.123	0.2137	[2016]
175272	Winona State Univ...	MN	4	13	NULL	0.6975	5887	0.255	0.614	[2016]
177214	Brury University	MO	4	13	NULL	0.4997	11867	0.5171	0.5844	[2016]
178387	Missouri Western ...	MO	4	13	NULL	NULL	5892	0.4083	0.4685	[2016]
178411	Missouri Universi...	MO	4	13	NULL	0.7933	11692	0.2429	0.4979	[2016]
178615	Truman State Univ...	MO	4	13	NULL	0.4768	5817	0.203	0.4143	[2016]
178624	Northwest Missour...	MO	4	13	NULL	0.7406	6591	0.3197	0.5589	[2016]
179557	Southeast Missour...	MO	4	13	NULL	0.8281	7143	0.3296	0.4557	[2016]
179894	Weber State Univers	MO	4	13	NULL	0.4658	36258	0.3475	0.5663	[2016]
181215	University of Neb...	NE	4	13	NULL	0.8586	5182	0.3336	0.4655	[2016]
181783	Wayne State College	NE	4	13	NULL	NULL	4355	0.357	0.5826	[2016]
210046	Black Hills State...	SD	4	13	NULL	0.7976	6598	0.2095	0.4913	[2016]
219471	University of Sou...	SD	4	13	NULL	0.8813	7964	0.2432	0.5426	[2016]

Command took 12.04 seconds -- by dbd0087@stthomas.edu at 12/9/2018, 4:10:47 PM on Saturday Afternoon

they're trying to build out a new paradigm for spark dataframes. It's interesting, because one of the databricks videos that I watched last weekend had a pop-up text box in the video that recommended users to avoid RDDs and stick with data frames and datasets as of Spark 2.X.X. I can't remember which version it was or what video it was, but I clearly remember seeing it.

Even though I was able to cast a type with a `select()` statement, I was still unable to calculate a mean on the dataframe, so I went back to RDD (against the recommendations of the Databricks guys, mind you). I'm a big fan of stringing a `mean()` on the end of a command and get what I want.

Data Exploration

Back when I was deciding on this dataset I had a lot of great ideas and high flying aspirations for what I was going to do with the dataset. My reasoning for picking this was that I always had doubts when we were in discussions in the office as to how we are doing compared to our "peer" institutions. My biggest doubt was that we were misidentifying our peers. It always went like this...

Somebody asks, "how are we doing compared to our peer institutions?"

Someone else replies, "Which ones are we talking about?"

A third person hops into the conversation with a list of catholic universities in the midwest and whatever other catholic universities that they can think of. Someone else will mention Loyola Marymount because they heard about that when our Provost came to St. Thomas and yet another person will mention Notre Dame, only to be shot down by the first speaker, who says "Oh, no. Their rankings are way higher than ours...."

So, I wanted to take the opportunity to set the record straight. The process would be quite simple. I'd pare down the set by different metrics, such as the different Carnegie classifications. For example, my first attempt was to select only the schools in the plains region, which includes IA, KS, MN, MO, NE, ND, and SD, with a size and setting the same as ours. Of the reporting schools (or correctly classified) only 20 institutions fell into this category, with the largest number located in Minnesota. Having this information in a concise data frame made it easy to quickly glance at the data and get an idea on how we compare to other schools like ours in terms of admission rate, tuition for full time enrollment, % of students receiving Pell grants, % with federal loans and religious affiliation.

There is so much in the dataset that it can be intimidating to look at it and make a decision.

```
1 regionalPeers.show()
```

► (7) Spark Jobs

UNITID	INSTNM	STABBR	REGION	CCSIZSET	RELAFFIL	ADM_RATE	SAT_AVG	TUITFTE	PCTPELL	PCTFLOAN	GRAD_DEBT_MDN	2016
155025	Emporia State Uni...	KS	4	13	NULL	0.8743	1029	6948	0.3693	0.5541	18568	2016
173124	Bemidji State Uni...	MN	4	13	NULL	0.6408	1030	5321	0.3129	0.5677	20433	2016
173665	Hamline University	MN	4	13	71	0.703	1104	15244	0.3558	0.6855	24400	2016
174358	Minnesota State U...	MN	4	13	NULL	0.5961	1049	5783	0.2865	0.5924	22579	2016
174817	Saint Mary's Univ...	MN	4	13	30	0.7924	1065	13376	0.2986	0.6297	23250	2016
174914	University of St ...	MN	4	13	30	0.8266	1204	19872	0.1716	0.5277	25000	2016
175005	St Catherine Univ...	MN	4	13	30	0.9083	1032	17479	0.3759	0.741	26938	2016
175078	Southwest Minneso...	MN	4	13	NULL	0.5419	967	3953	0.123	0.2137	20500	2016
175272	Winona State Univ...	MN	4	13	NULL	0.5975	1049	5887	0.255	0.614	23206	2016
177214	Drury University	MO	4	13	NULL	0.6997	1142	11067	0.5171	0.5844	23000	2016
178387	Missouri Western ...	MO	4	13	NULL	NULL	NULL	5892	0.4003	0.4605	22000	2016
178411	Missouri Universi...	MO	4	13	NULL	0.7933	1265	11602	0.2429	0.4979	24250	2016
178615	Truman State Univ...	MO	4	13	NULL	0.6769	1224	5017	0.203	0.4141	22402	2016
178624	Northwest Missour...	MO	4	13	NULL	0.7406	1047	6591	0.3197	0.5509	23557	2016
179557	Southeast Missour...	MO	4	13	NULL	0.8281	1049	7143	0.3296	0.4557	22185	2016
179894	Webster University	MO	4	13	NULL	0.4658	1110	16258	0.3475	0.5663	23225.5	2016
181215	University of Neb...	NE	4	13	NULL	0.8508	1028	5182	0.3336	0.4655	20980	2016
181783	Wayne State College	NE	4	13	NULL	NULL	NULL	4355	0.357	0.5826	19921.5	2016
219046	Black Hills State...	SD	4	13	NULL	0.7976	1009	6598	0.2995	0.4813	25500	2016
219471	University of Sou...	SD	4	13	NULL	0.8813	1050	7964	0.2432	0.5426	25000	2016

Command took 21.62 seconds -- by dedo3671@stthomas.edu at 12/9/2018, 2:52:46 PM on SunMorn

I always thought that our admission rate was very high and that we could probably be a bit more selective, but if you look at our average SAT scores, the selection is taking place before the application period, as our pool of students, while admitted at a higher rate, has the third highest average SAT score, at 1204. That's nothing to scoff at.

SAT Scores

Minnesota SAT average

```
[43]: mnColleges.filter(x => x(1899) == "2016").map(x => x(59)).filter(x => x != "NULL").map(_.toInt).mean()
[43]: res25: Double = 1102.0666666666668
```

Regional Peers SAT Average

```
[44]: regionalPeers.filter(x => x(1899) == "2016").map(x => x(59)).filter(x => x != "NULL").map(_.toInt).mean()
[44]: res26: Double = 1080.7222222222224
```

National Peers SAT Average

```
[47]: nationalPeers.filter(x => x(1899) == "2016").map(x => x(59)).filter(x => x != "NULL").filter(x => x != "SAT_AVG").map(_.toInt).mean()
[47]: res28: Double = 1040.932432432432
```

National SAT Average

```
[45]: scoreCard10Year.filter(x => x(1899) == "2016").map(x => x(59)).filter(x => x != "NULL").filter(x => x != "SAT_AVG").map(_.toInt).mean()
[45]: res27: Double = 1059.2575757575776
```

Catholic Schools Average

```
[48]: scoreCard10Year.filter(x => x(1899) == "2016").filter(x => x(35) == "30").map(x => x(59)).filter(x => x != "NULL").filter(x => x != "SAT_AVG").map(_.toInt).mean()
[48]: res29: Double = 1057.0273972602745
```


Furthermore, the average score for all reporting Minnesota schools was 1102, which is still over one hundred points below the St. Thomas score reported for that year.

As you can see on the previous page, the St. Thomas undergraduate students' average SAT score is visibly higher than all of the other measured segments.

Other indicators, such as **percent of students receiving Pell grants** reinforces the stereotype that St. Thomas is predominantly affluent undergrad students. The St Thomas percentage is just seventeen while every other grouped segment had an average percentage of more than double that. I assume that this will change with the addition of the Dougherty Family College and other current initiatives will have a positive impact on our student body and bring our number more in-line with other schools.

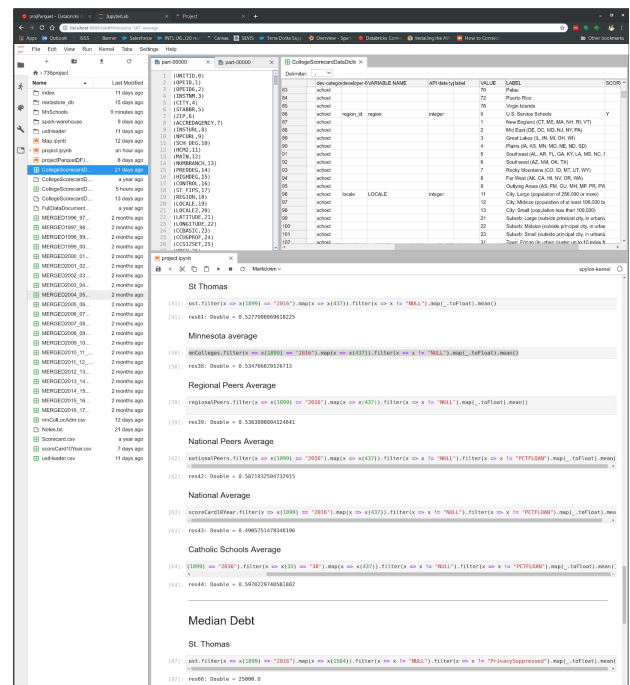
Our **percent federal loan** receiving students falls in-line with that of the other school segments measured.

The **median debt** of graduates at St. Thomas was reported to be \$25,000 in 2016. If you look at the dataframe in the databricks notebook, you can see that there definitely is a tendency to round the numbers and around \$25,000 seems to be normal for our regional peers and national peers, as well as catholic schools nationally. I'm not buying those numbers, though.

Tools Used (JupyterLab)

The main tool used in this project was a JupyterLab Notebook running locally on my laptop. I had to install the Sylon scala kernel in order to get my code to run. I'm a big fan of Jupyter and have been using the notebooks since my Intro to Statistics course a year or so back. Jupyter notebooks are especially powerful when you are exploring data and want to tell a story with the data.

With JupyterLab, which is currently in beta (although considered production ready for users³), is an improvement upon the Jupyter Notebook in that it allows users to layout different panes within the browser window for a more comprehensive view of all of the data. As you can see in the figure on the right I had access to the working directory, the index of the features of the data set, the data dictionary in tabular format, as well as the actual notebook, where I was doing all of the work. This ability to lay out your



³ JupyterLab is Ready for Users <https://blog.jupyter.org/jupyterlab-is-ready-for-users-5a6f039b8906>

data and reference tables makes it a more comprehensive working environment than most others for analyzing data.

The wonderful thing about the notebooks is that the outputs are laid out inline with the inputs and stored in the .ipynb file. This allows for sharing of notebooks and results in a manner that visually lays out your steps taken to achieve your results. The .ipynb file type is also rendered as it is locally when uploaded to GitHub. That's pretty cool.

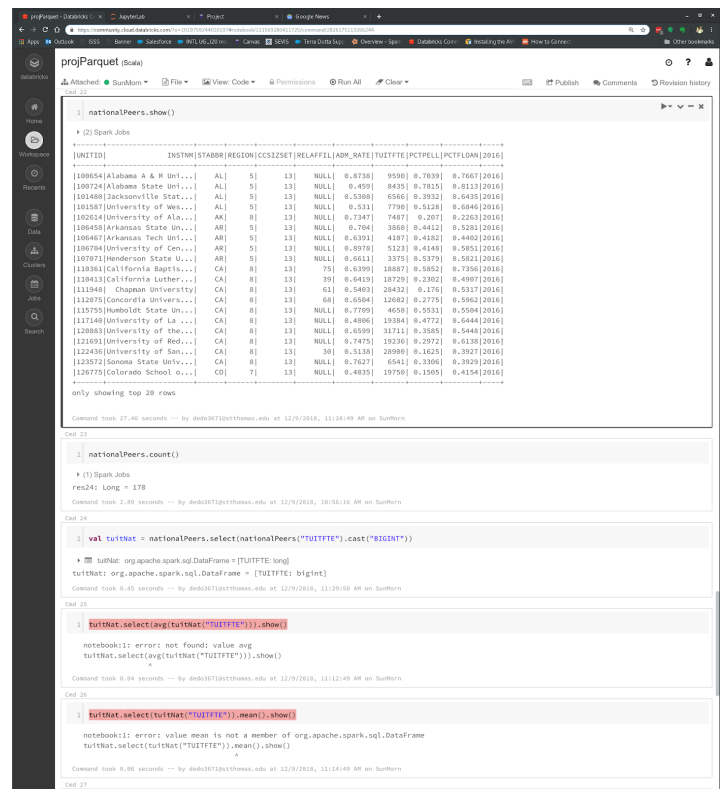
I had wanted to experiment with some data visualizations, as Jupyter Notebooks allow you to mix your code in different cells. For example, if you start a cell with %python you designate that cell as a python cell and execute python code. You can also do the same with Scala in a Python notebook. However, I ran into issues with package dependencies on my machine and decided that, while cool, the time and work needed to figure out what's missing/mis-configured wasn't going to contribute to the overall effectiveness of the project and was thus left out in the end.

Tools Used (Databricks cont....)

Databricks was founded by the smart guys who created Spark in the first place. The Community Edition is available free of charge to non-commercial users who don't need any sort of data privacy or security. The dataset that I worked with in this project is public, so that wasn't a problem.

The system allows you to fire up a small cluster running Spark 2.3.1 with Scala 2.11. The machines that you have provisioned for you aren't the most beastly of the bunch, with only 6GB of RAM and roughly 1 core available. It was able to get past the point at which my laptop, with 8GB of ram and 2 cores buckled under the pressure, though. So it was just beastly enough for my job here.

The platform gives you the ability to create a notebook similar to those of Jupyter, running either Python, Scala, R, or SQL. That's a pretty good set of options for a platform looking to serve the needs of people analyzing data. The only difference is that it limits your notebook cells to code only, so you can't mix in markdown, or



```
1 nationalPeers.show()

+ (2) Spark Jobs

+-----+-----+-----+-----+-----+-----+-----+-----+
|UNITID|INSTNM|STABBR|REGION|CCS22SET|RELAFELL|ADM_RATE|TUTFTFE|PCTPELL|PCTFLOW|2016|
+-----+-----+-----+-----+-----+-----+-----+-----+
|108802|Alabama A & M Un...|AL|51|13|NULL|0.8738|0.9080|0.7039|0.7607|2016|
|108724|Alabama State Un...|AL|51|13|NULL|0.4591|0.4435|0.7835|0.8113|2016|
|101408|Jacksonville Stat...|AL|51|13|NULL|0.5308|0.6561|0.3932|0.6435|2016|
|101587|University of Ala...|AL|51|13|NULL|0.5311|0.7798|0.5128|0.6440|2016|
|102614|University of Ala...|AK|81|13|NULL|0.7347|0.7487|0.2071|0.2203|2016|
|106458|Arkansas State Un...|AR|51|13|NULL|0.7041|0.3681|0.4412|0.5281|2016|
|106467|Arkansas Tech Un...|AR|51|13|NULL|0.4293|0.4187|0.4182|0.4402|2016|
|106704|University of Cen...|AR|51|13|NULL|0.8978|0.5323|0.4148|0.5851|2016|
|107071|Henderson State U...|AR|51|13|NULL|0.6611|0.3371|0.5379|0.5021|2016|
|118361|California Baptis...|CA|81|13|751|0.4399|0.6887|0.1082|0.7161|2016|
|118413|California Luther...|CA|81|13|391|0.6419|0.1829|0.2302|0.4907|2016|
|111948|Chapman University|CA|81|13|611|0.5403|0.2842|0.1761|0.5117|2016|
|111875|Concordia Univers...|CA|81|13|681|0.4584|0.2082|0.2775|0.5062|2016|
|115755|Humboldt State Un...|CA|81|13|NULL|0.7709|0.4651|0.5531|0.5004|2016|
|117149|University of La ...|CA|81|13|NULL|0.4886|0.1934|0.4772|0.6444|2016|
|108883|University of the ...|CA|81|13|NULL|0.4599|0.3711|0.3585|0.5448|2016|
|121691|University of Red...|CA|81|13|NULL|0.7475|0.1926|0.2972|0.6138|2016|
|122436|University of San...|CA|81|13|281|0.5138|0.2098|0.1625|0.3027|2016|
|123572|Sonoma State Univ...|CA|81|13|NULL|0.7827|0.6541|0.3386|0.3929|2016|
|126775|Colorado School o...|CO|71|13|NULL|0.4835|0.1958|0.1505|0.4154|2016|
+-----+-----+-----+-----+-----+-----+-----+-----+

only showing top 20 rows

Command took 27.46 seconds --- by ddb03871@thomas.edu at 12/9/2018, 11:20:49 AM on Sanborn

End 23

1 nationalPeers.count()

+ (1) Spark Jobs

res24: Long = 178

Command took 2.89 seconds --- by ddb03871@thomas.edu at 12/9/2018, 11:20:16 AM on Sanborn

End 24

1 val tutftat = nationalPeers.select(nationalPeers("TUTFTFE").cast("BIGINT"))

1 // tutftat: org.apache.spark.sql.DataFrame = [TUTFTFE: bigint]
tutftat: org.apache.spark.sql.DataFrame = [TUTFTFE: bigint]

Command took 0.40 seconds --- by ddb03871@thomas.edu at 12/9/2018, 11:20:58 AM on Sanborn

End 25

1 tutftat.select(avg(tutftat("TUTFTFE"))).show()

notebook1: error: not found: value avg
tutftat.select(avg(tutftat("TUTFTFE"))).show()

Command took 0.44 seconds --- by ddb03871@thomas.edu at 12/9/2018, 11:21:49 AM on Sanborn

End 26

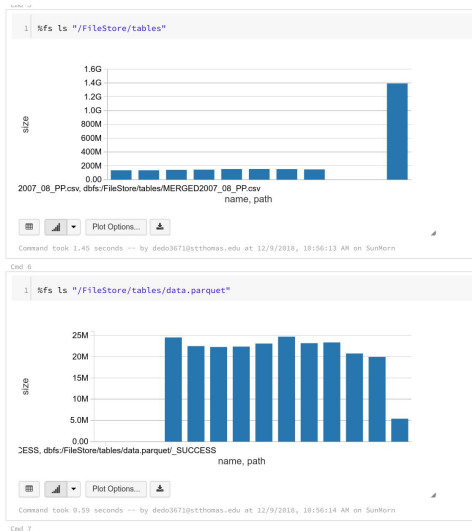
1 tutftat.select(tutftat("TUTFTFE").mean()).show()

notebook1: error: value mean is not a member of org.apache.spark.sql.DataFrame
tutftat.select(tutftat("TUTFTFE").mean()).show()

Command took 0.00 seconds --- by ddb03871@thomas.edu at 12/9/2018, 11:21:49 AM on Sanborn

End 27
```

switch a cell to raw text instead of commenting it out to keep it from running (a habit of mine, although maybe not a good one).



One cool thing about Databricks is that they built graphing right into the notebook, so any cell that results in a table (I'm not sure what kind of table) can be graphed. I was only able to use this function on the file sizes of my file system queries, but I can see how this functionality would be useful in other settings.

I appreciate that they make this product available for free. It's cool to have access to a cloud based notebook environment attached to an actual cluster.

Conclusions

Part of me is disappointed that I wasn't able to program some sort of spectacular algorithm that would analyze this data, have Alexa™ integration, and compliment me, while simultaneously causing my printer to waft out both glitter and ticker tape. Then the other part is satisfied with what I have been able to accomplish here. I'm satisfied that I have done what I set out to do with the tools available. My project mainly relied on parsing and filtering the data to get what I was looking for.

Looking back, if I were to do this whole thing again, I would have probably chosen to use this opportunity to experiment with different ways to load the table in different ways. That's a pretty lofty goal, though, as what I would want would be for all of the features to get read in with the correct data-types (int, boolean, string, etc.) and splice the tables together without the header lines and stuff. I kept running into issues with header lines (strings) throwing a wrench in the gears into type casting for other operations. Thinking about it now, I could have fixed that on the load of the files, but it wouldn't be a very robust load function and leave me with a one-trick-pony.

The Kaggle dataset was packaged with a YAML file that typed all of the features, but I'm not sure if that would be an asset in my data load here. I'll learn more about YAML in the spring semester with the DevOps class, I hear.

I wish I would have had more time to do more with this project. Altogether, I probably logged around 20 hours or so of screen time working with the set, stumbling through the execution of the operations, and trying to figure out what's wrong.

If I were to have had more time, I would have liked to go further to identify peer institutions based on majors offered, but the number of features that need to go into that and the type casting needed (they all should be boolean, but they're string) was time prohibitive.

At the outset, I also wanted to plot a map with the supplied lat/long data in the set, but I ran into some package missing hiccups with my python installation and wasn't able to get the packages working as needed.

I can see that Apache Spark is going to be the go-to tool for data manipulation for the foreseeable future and I'm glad to have learned it here this semester. Scala is probably my favorite language also, as I find the whole functional programming paradigm more fun than the object oriented slog.

I want to thank you also for the great curriculum that you have put together here in 736. This was probably one of the best classes that I have taken so far in the Data Science program.

Have a great holiday break!

Code and Outputs

The link is the code and outputs from my Jupyter Notebook. The Databricks files are a separate html file in the directory. Unfortunately, the export function in Databricks is quite limiting and converting the outputs to pdf format leaves it looking quite bad.

<https://github.com/jamesdedon/736project/blob/master/project.ipynb>