Final Task Assessment

By - Ashish Gusai
Employee ID - 10200
Mentors - Chetan Khatri
& Babu Prabhakar

List of Tasks

1. Add Airflow on Ambari HDP using mpack.
2. Work with Local executor ,Sequential Executor ,Celery Executor.
3. Work with BranchPythonOperator
4. PartitionBy work with multiple category and apply filters ,aggregation ,groupBy ,orderBy on dataframes.
5. Using master=local instead of YARN for SparkContext
6. Work with kaggle datasets and apply some business logic to get meaningful data.
7. Upload code and presentation at GitHub.

Airflow using MPack

- What is MPack?
  - add an "add-on" service (custom service) to the stack definition, one has to manually add the add-on service to the stack definition on an Ambari Server. There is no release vehicle that can be used to ship add-on services.
- Problem Statement :- Airflow webserver UI is not responding.
- Solution :- No solution found yet.

Airflow Sequential Executor (default)

- Airflow uses a sqlite database, which you should outgrow fairly quickly since no parallelization is possible using this database backend. It works in conjunction with the SequentialExecutor which will only run task instances sequentially.

Airflow Local executor

- LocalExecutor executes tasks locally in parallel. It uses the multiprocessing Python library and queues to parallelize the execution of tasks.
- Need to install PostgreSQL or MySql to support parallelism.

Setting-up :-

```
$gedit ~/airflow/airflow.cfg
#change configurations
executor = LocalExecutor
result_backend = db+mysql://airflow:airflow@localhost:3306/airflow
```

Airflow Celery Executor

- CeleryExecutor is one of the ways you can scale out the number of workers. For this to work, you need to setup a Celery backend (RabbitMQ, Redis, ...) and change your airflow.cfg to point the executor parameter to CeleryExecutor and provide the related Celery settings.

  Setting-up :-

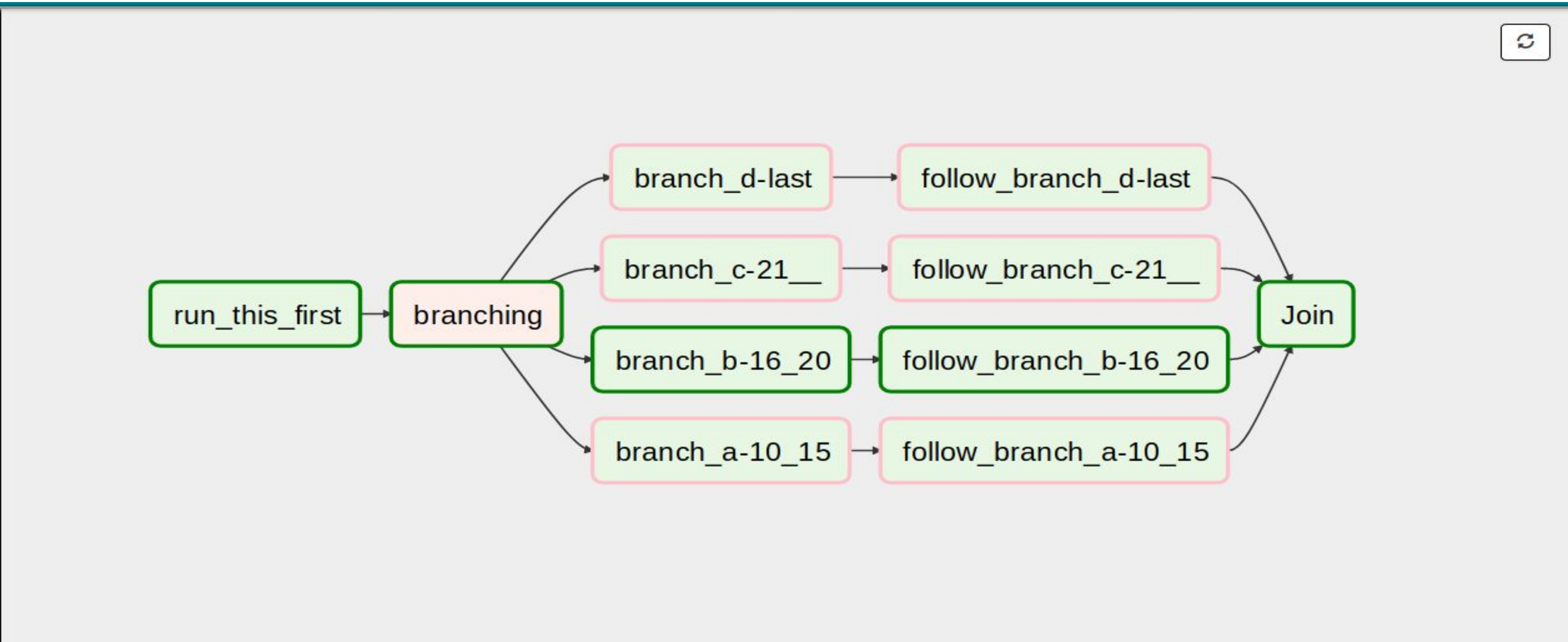
  ```
  $gedit ~/airflow/airflow.cfg
  #change configurations
  executor = CeleryExecutor
  result_backend = db+mysql://airflow:airflow@localhost:3306/airflow

  [celery]
  broker_url = pyamqp://guest:guest@localhost:5672/airflow
  celery_result_backend = pyamqp://guest:guest@localhost:5672/airflow
  ```

BranchPythonOperator

- The BranchPythonOperator is much like the PythonOperator except that it expects a python_callable that returns a task_id. The task_id returned is followed, and all of the other paths are skipped. The task_id returned by the Python function has to be referencing a task directly downstream from the BranchPythonOperator task.

- Code:-BranchPythonOperator

BranchPythonOperator



Working with Kaggle Dataset

- Dataset:- DonorsChoose
- Problem Statement:-
    - DonorsChoose.org has funded over 1.1 million classroom requests through the support of 3 million donors, the majority of whom were making their first-ever donation to a public school. If DonorsChoose.org can motivate even a fraction of those donors to make another donation, that could have a huge impact on the number of classroom requests fulfilled.

Dataset Overview

1. Donations.csv: For every project in the Projects.csv dataset, there are one or more donations. This dataset contains each donation from a citizen donor and is joined with the dataset above using the “Project ID” column.test test1
2. Donors.csv
3. Resources.csv: For every project in the Projects.csv file, there are one or more resources that is requested. This dataset contains the names of each resource in the project request and is joined with the dataset above using the “Project ID” column.
4. Projects.csv
5. Schools.csv: More information at a school level. Each row represents a single school. This dataset is joined with the project data using the “School ID” column.
6. Teachers.csv: More information at a teacher level. Each row represents a single teacher. This dataset is joined with the project data using the “Teacher ID” column.

Programs and Dataflow

DataFlow :- WorkFlow

Code :- DonorChoose

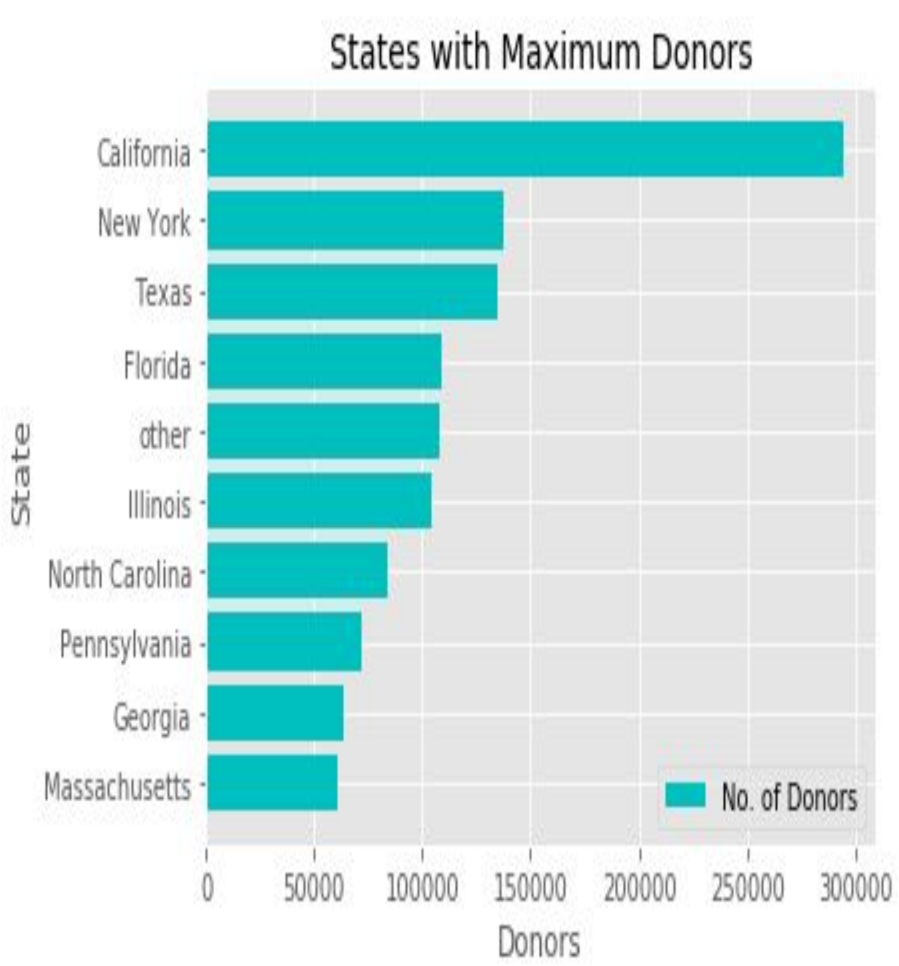
Notebook code :- kaggle_data_visualization.ipynb

YAML Encryption

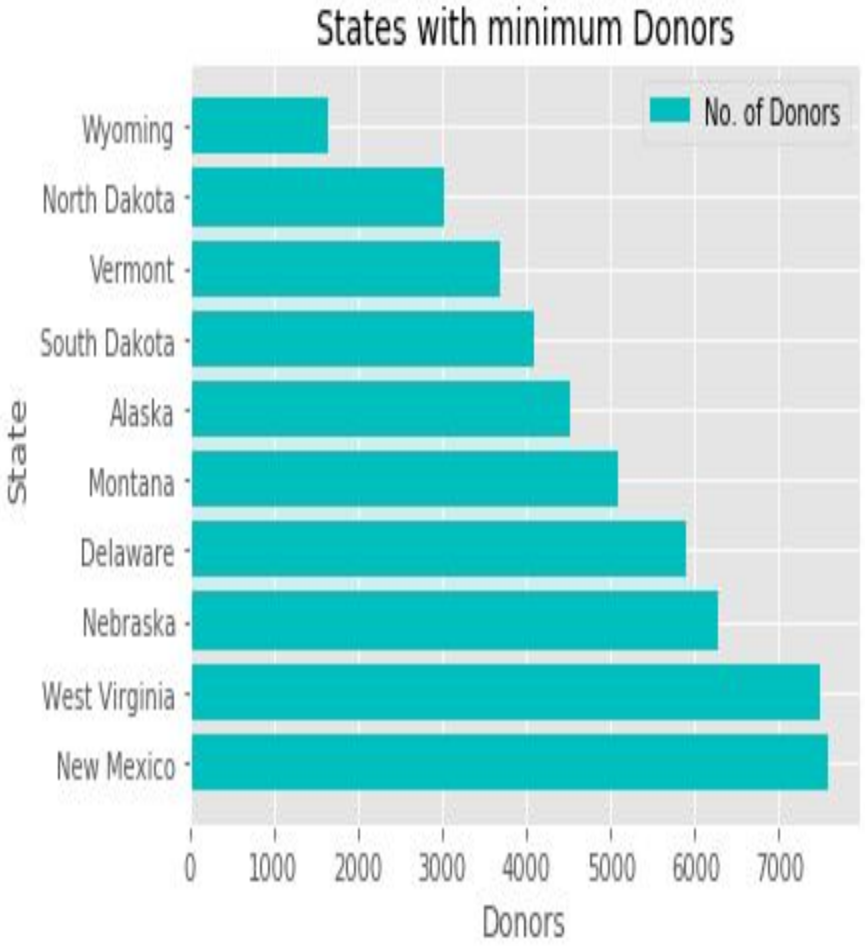
- For encrypting YAML i have used yaycl-crypt 0.4.0 library.
- It uses python Cryptography libraries for encryption.

Code:- YAML config

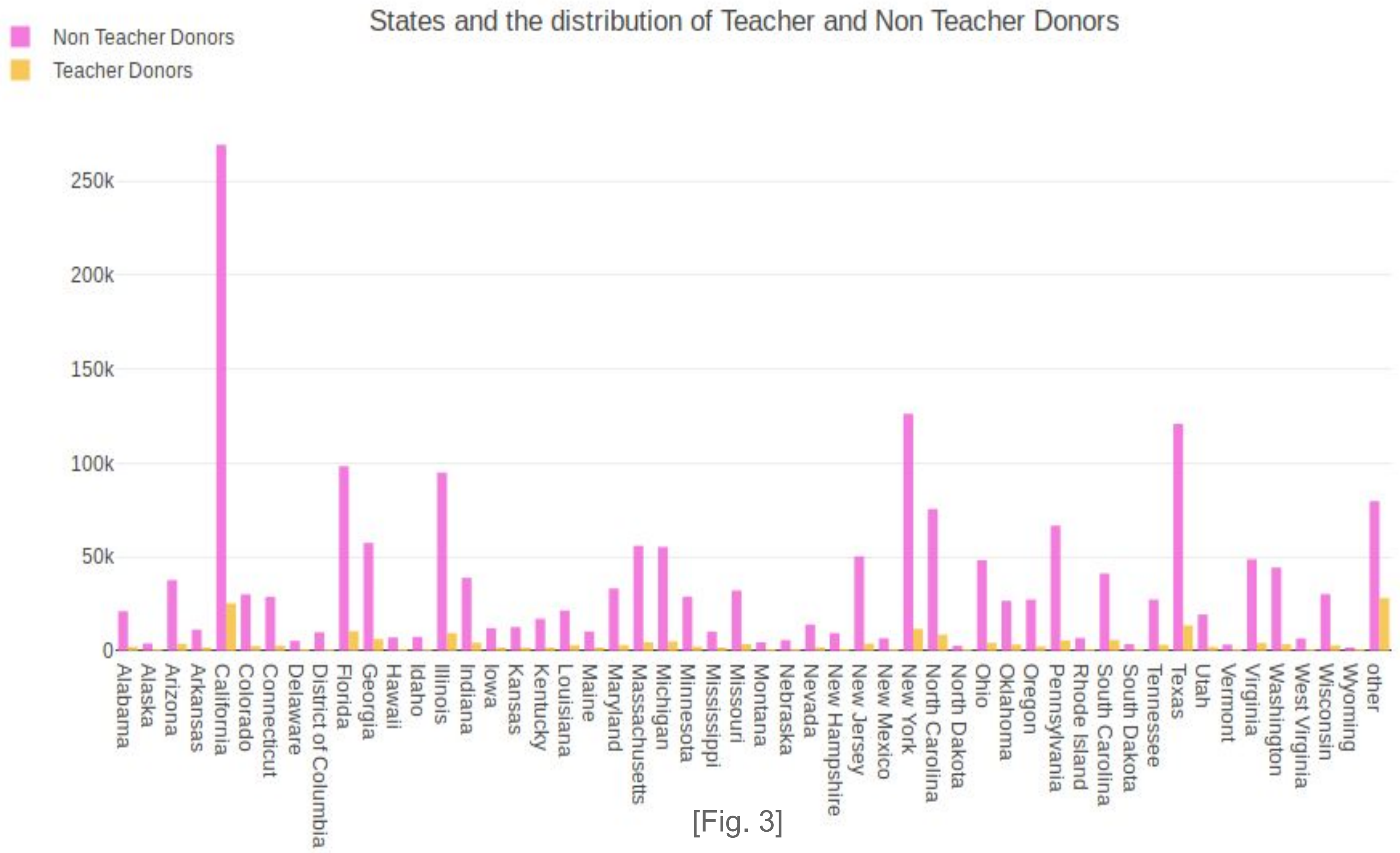
Chart Analysis


States with Maximum Donors
California
New York
Texas
Florida
other
Illinois
North Carolina
Pennsylvania
Georgia
Massachusetts
State
No. of Donors
0
50000
100000
150000
200000
250000
300000
Donors

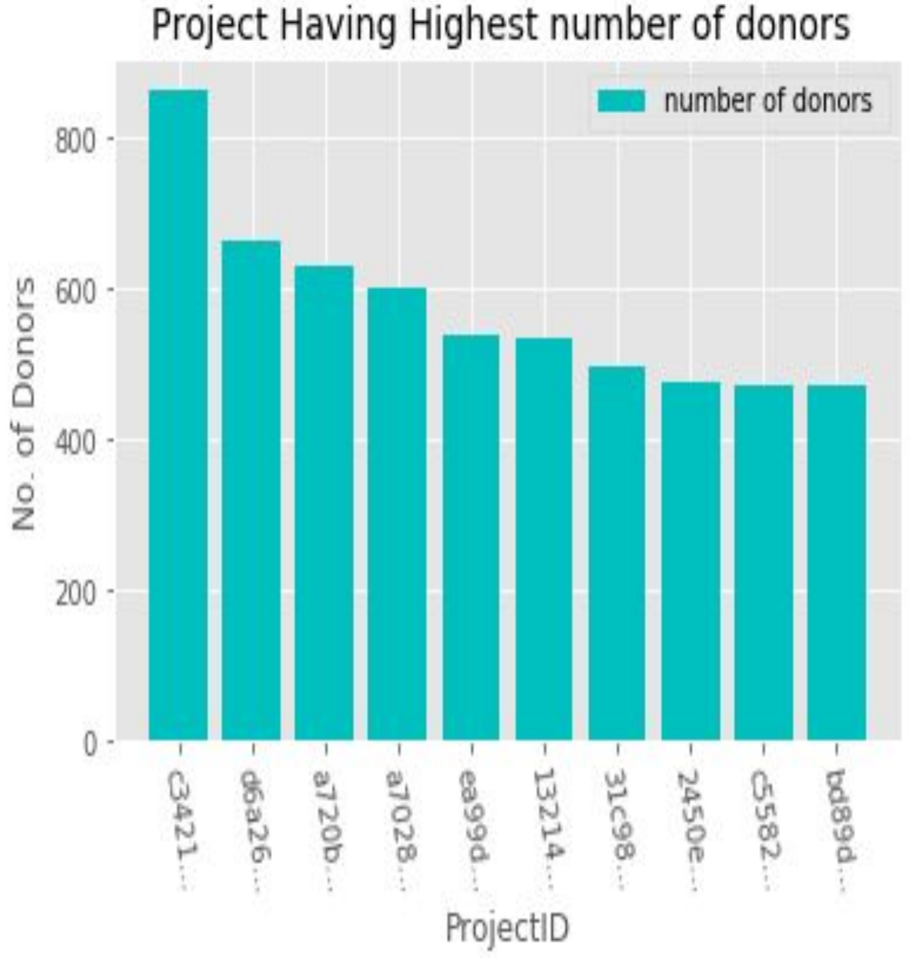
[Fig. 1]


States with minimum Donors
Wyoming
North Dakota
Vermont
South Dakota
Alaska
Montana
Delaware
Nebraska
West Virginia
New Mexico
State
No. of Donors
0
1000
2000
3000
4000
5000
6000
7000
Donors

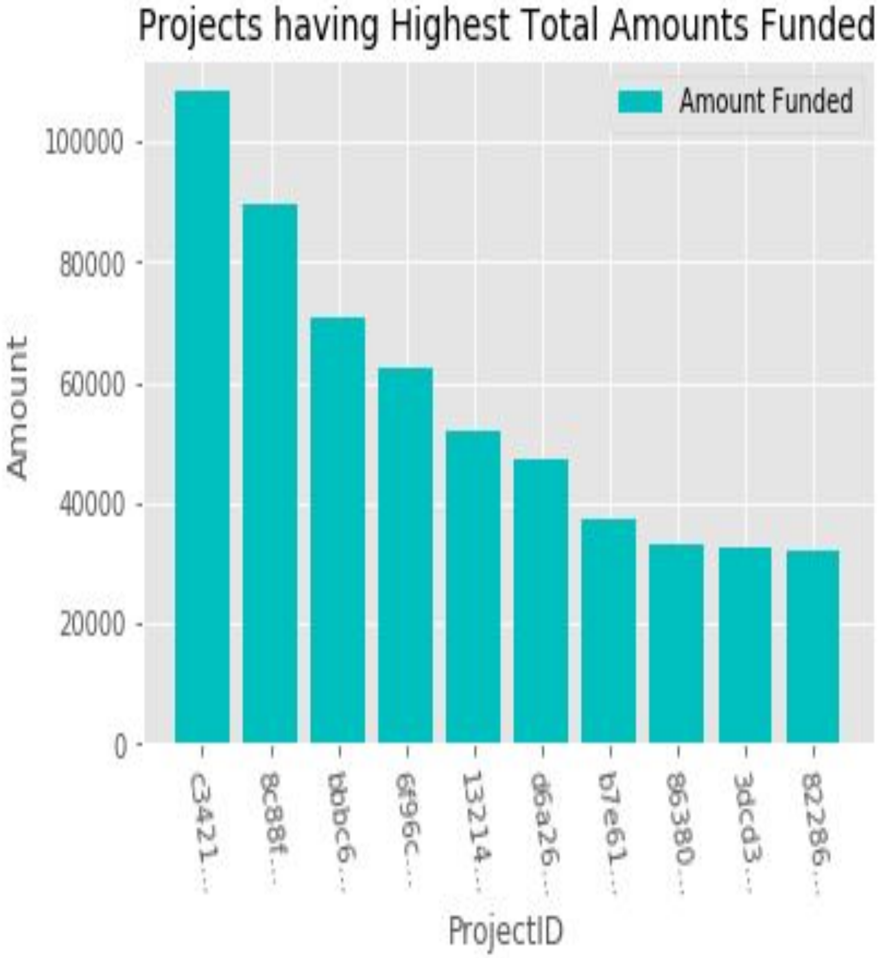
[Fig. 2]


States and the distribution of Teacher and Non Teacher Donors
Non Teacher Donors
Teacher Donors
250k
200k
150k
100k
50k
0
Alabama
Alaska
Arizona
Arkansas
California
Colorado
Connecticut
Delaware
District of Columbia
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
Massachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
New Hampshire
New Jersey
New Mexico
New York
North Carolina
North Dakota
Ohio
Oklahoma
Oregon
Pennsylvania
Rhode Island
South Carolina
South Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Washington
West Virginia
Wisconsin
Wyoming
other

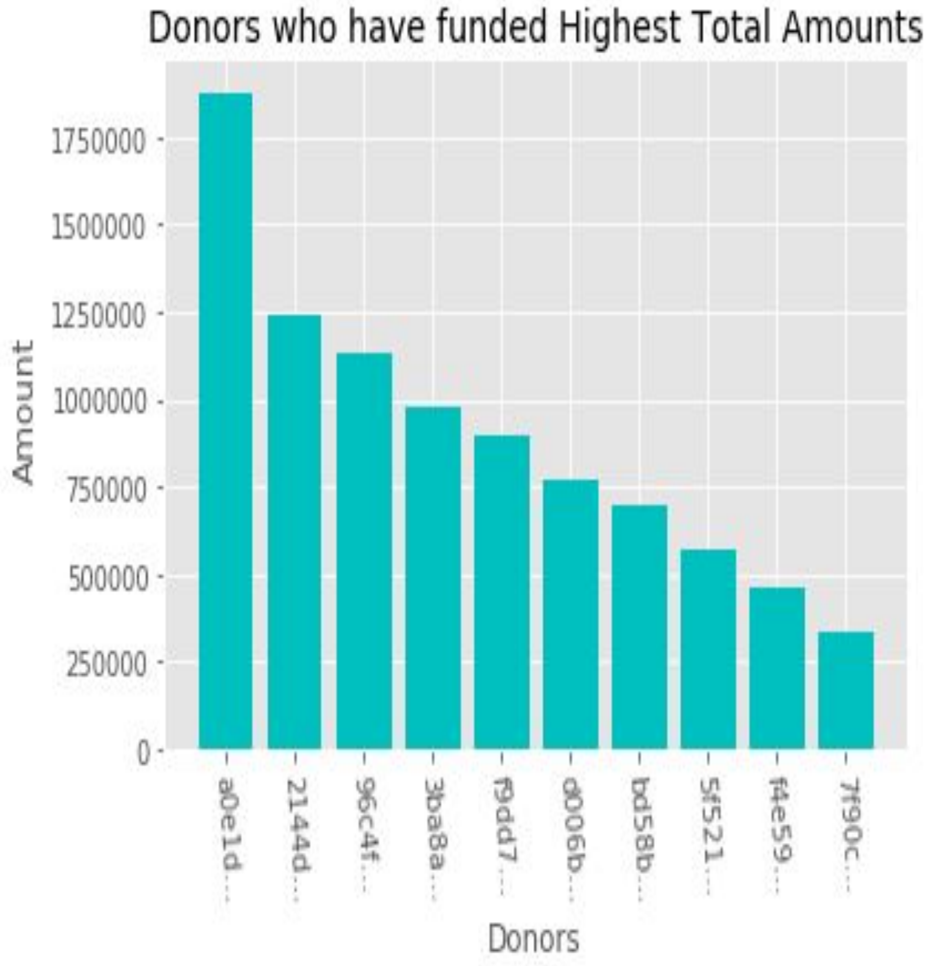
[Fig. 3]


Project Having Highest number of donors
number of donors
No. of Donors
0
200
400
600
800
c3421…
d6a26…
a720b…
a7028…
ea99d…
13214…
31c98…
2450e…
c5582…
bd89d…
ProjectID

[Fig. 4]

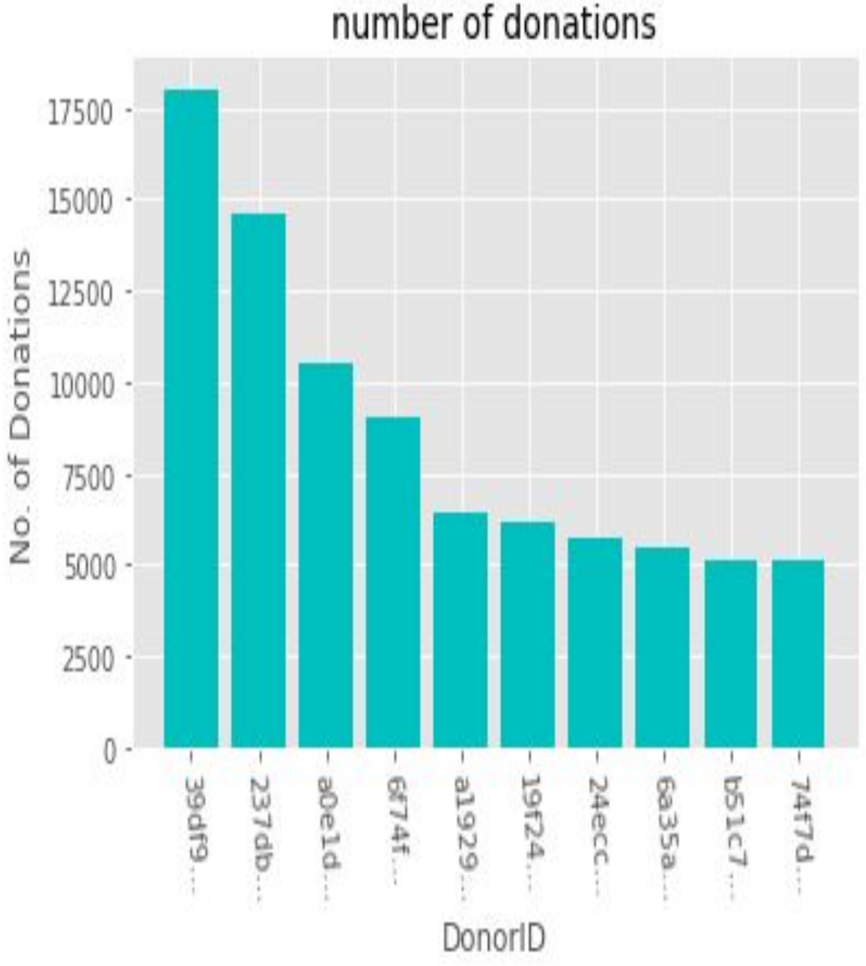

Projects having Highest Total Amounts Funded
Amount Funded
Amount
0
20000
40000
60000
80000
100000
c3421…
8c88f…
bbbc6…
6f96c…
13214…
d6a26…
b7e61…
86380…
3dcd3…
82286…
ProjectID

[Fig. 5]



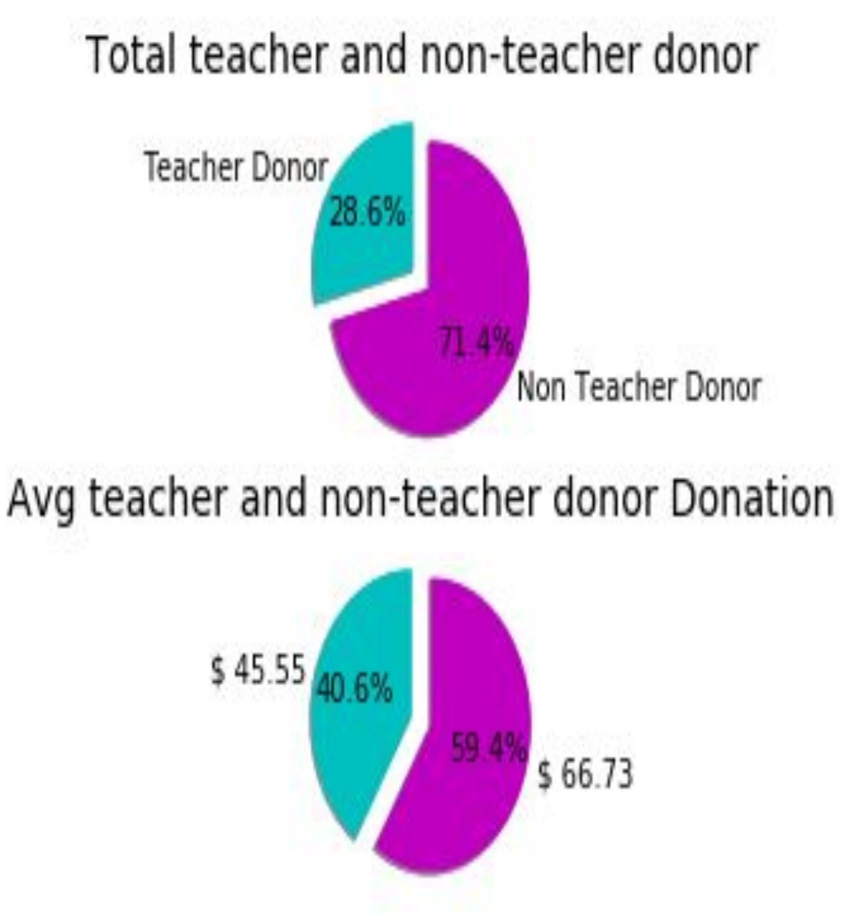
Donors who have funded Highest Total Amounts
Amount
1750000
1500000
1250000
1000000
750000
500000
250000
0
a0e1d...
2144d...
96c4f...
3ba8a...
f9dd7...
d006b...
bd58b...
5f521...
f4e59...
7f90c...
Donors

[Fig. 6]

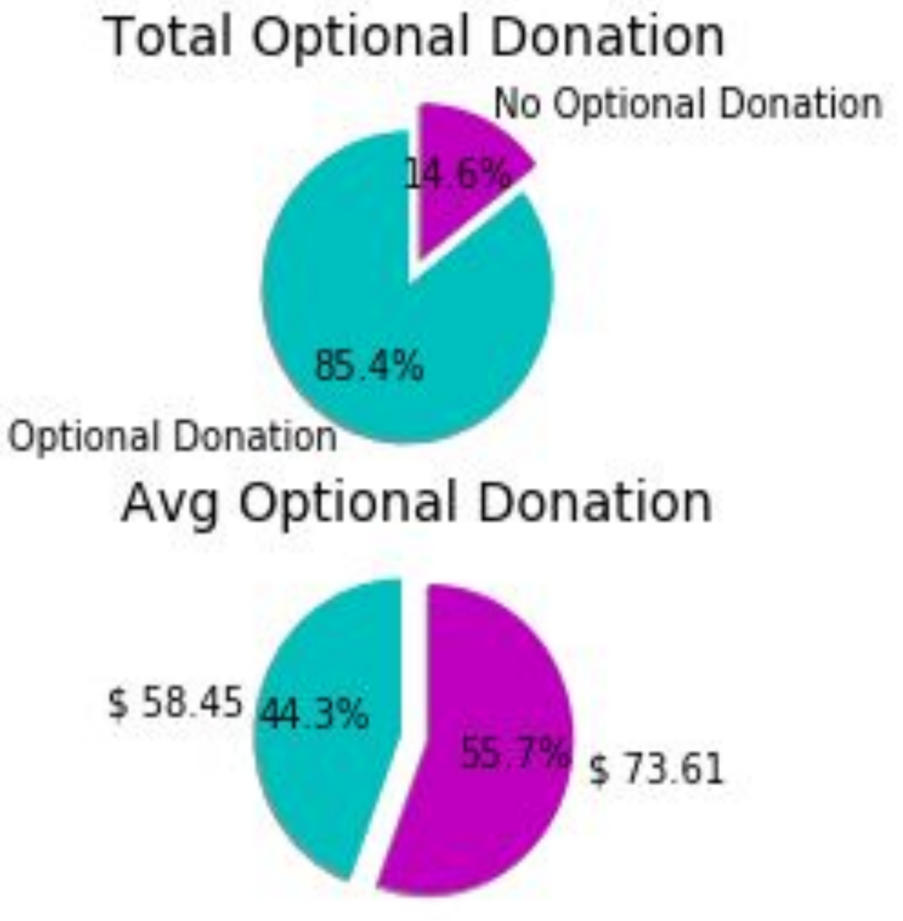


number of donations
No. of Donations
17500
15000
12500
10000
7500
5000
2500
0
39df9...
237db...
a0e1d...
6f74f...
a1929...
19f24...
24ecc...
6a35a...
b51c7...
74f7d...
DonorID

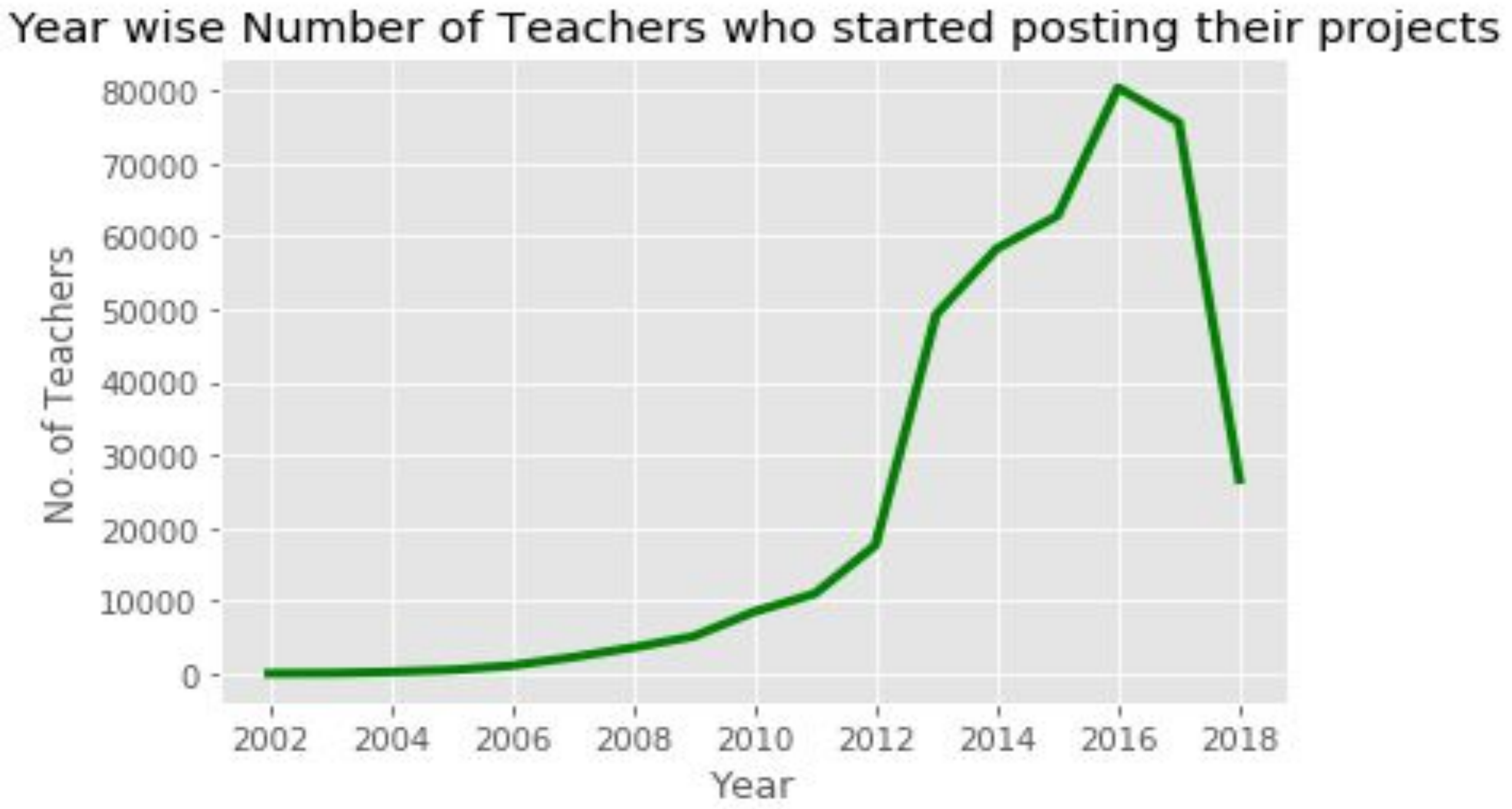
[Fig. 7]


Total teacher and non-teacher donor
Teacher Donor
28.6%
71.4%
Non Teacher Donor
Avg teacher and non-teacher donor Donation
$ 45.55
40.6%
59.4%
$ 66.73

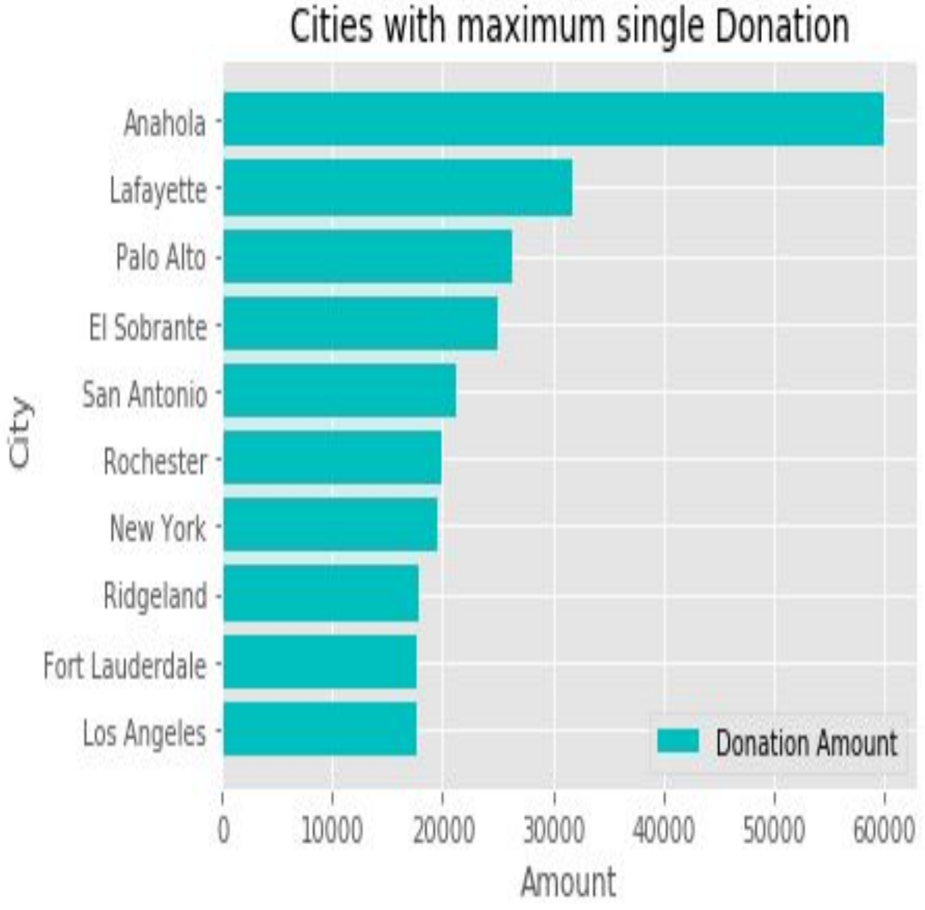
[Fig. 8]


Total Optional Donation
No Optional Donation
14.6%
85.4%
Optional Donation
Avg Optional Donation
$ 58.45
44.3%
55.7%
$ 73.61

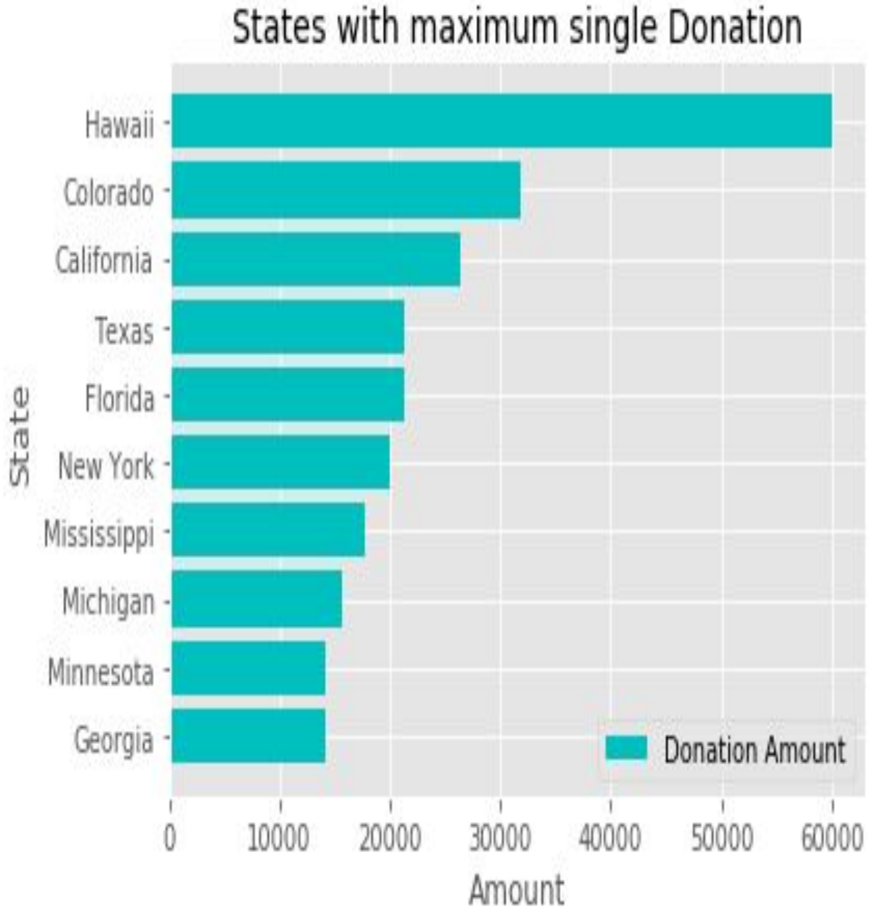
[Fig. 9]


Year wise Number of Teachers who started posting their projects
No. of Teachers
Year
80000
70000
60000
50000
40000
30000
20000
10000
0
2002
2004
2006
2008
2010
2012
2014
2016
2018

[Fig. 10]

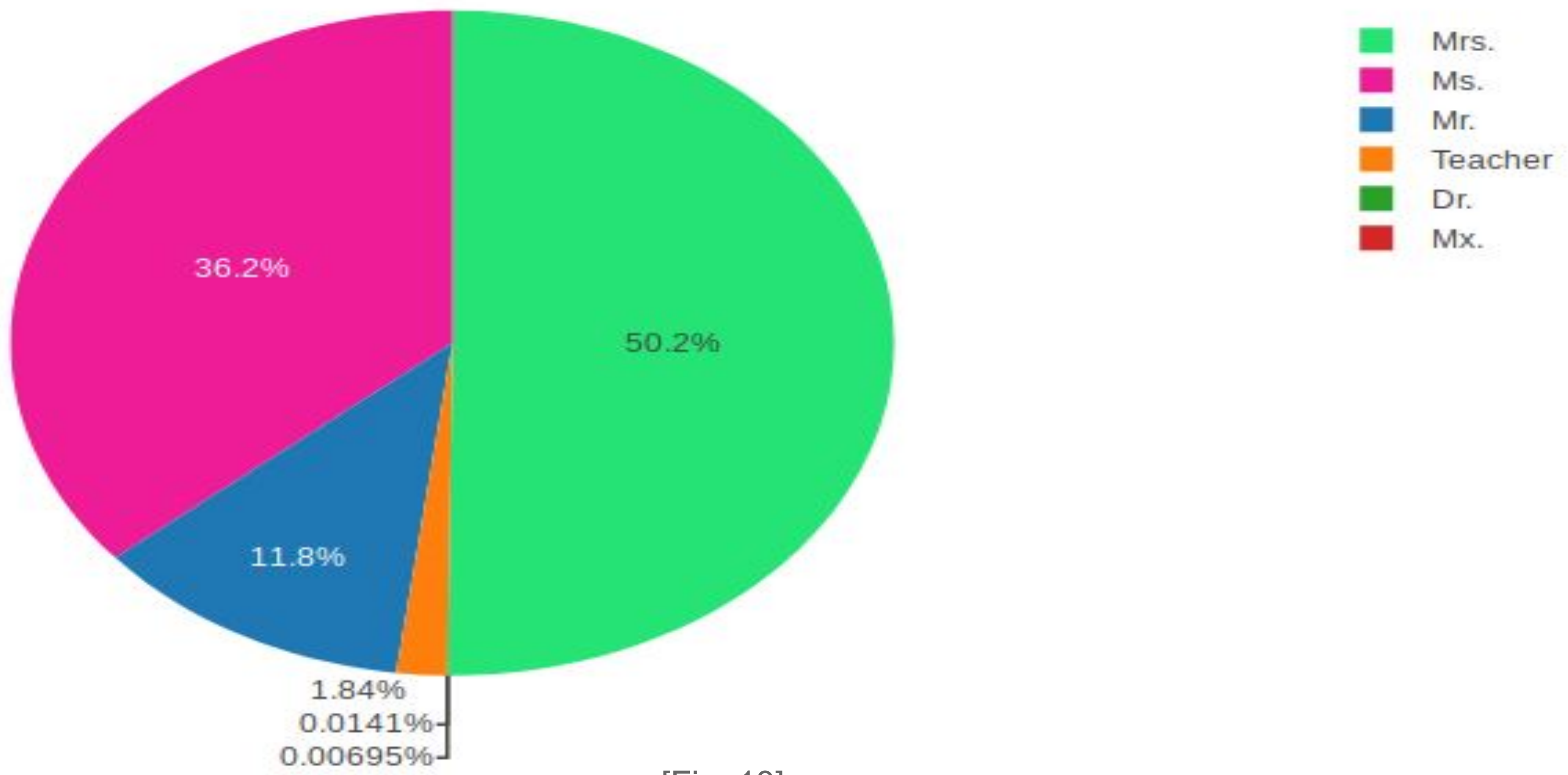

Cities with maximum single Donation
City
Anahola
Lafayette
Palo Alto
El Sobrante
San Antonio
Rochester
New York
Ridgeland
Fort Lauderdale
Los Angeles
Donation Amount
0
10000
20000
30000
40000
50000
60000
Amount

[Fig. 11]

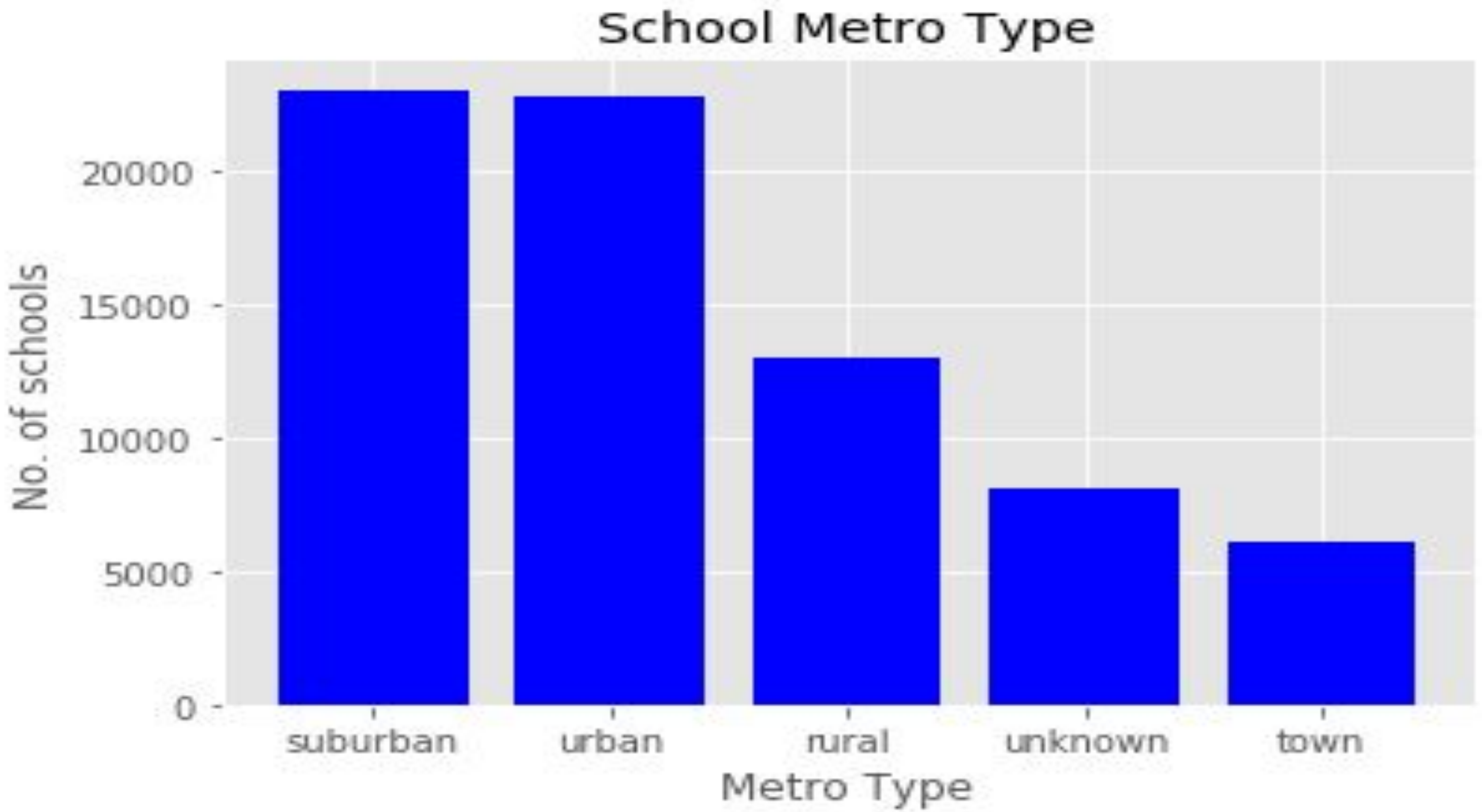

States with maximum single Donation
State
Hawaii
Colorado
California
Texas
Florida
New York
Mississippi
Michigan
Minnesota
Georgia
Donation Amount
0
10000
20000
30000
40000
50000
60000
Amount

[Fig. 12]

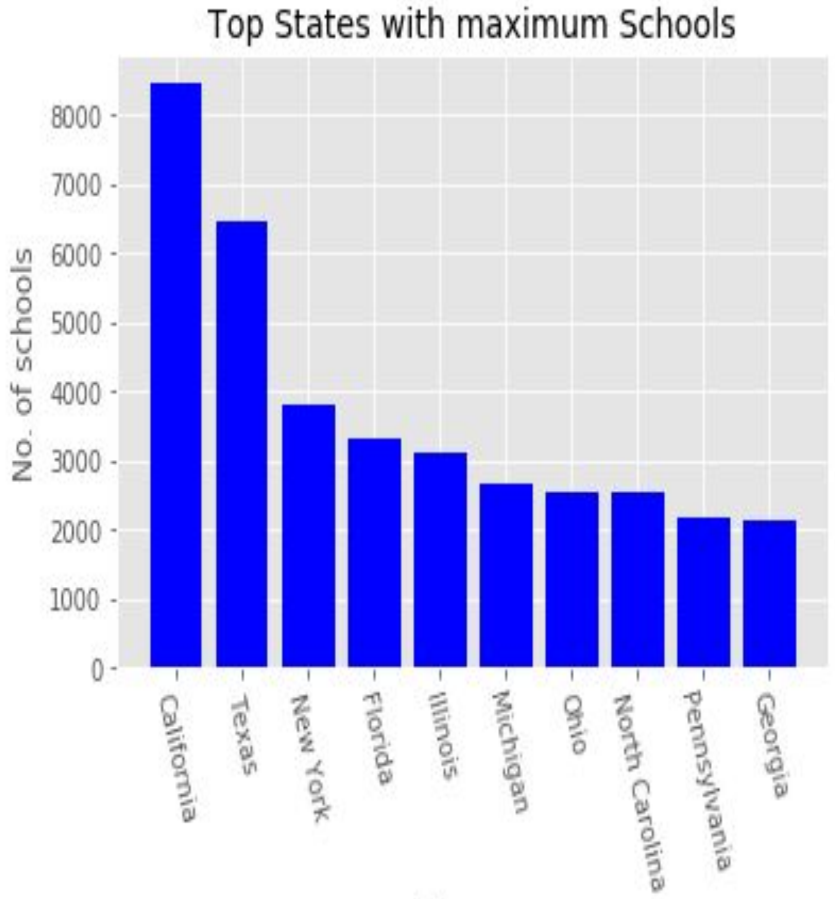
Number of Donations with Optional Donations



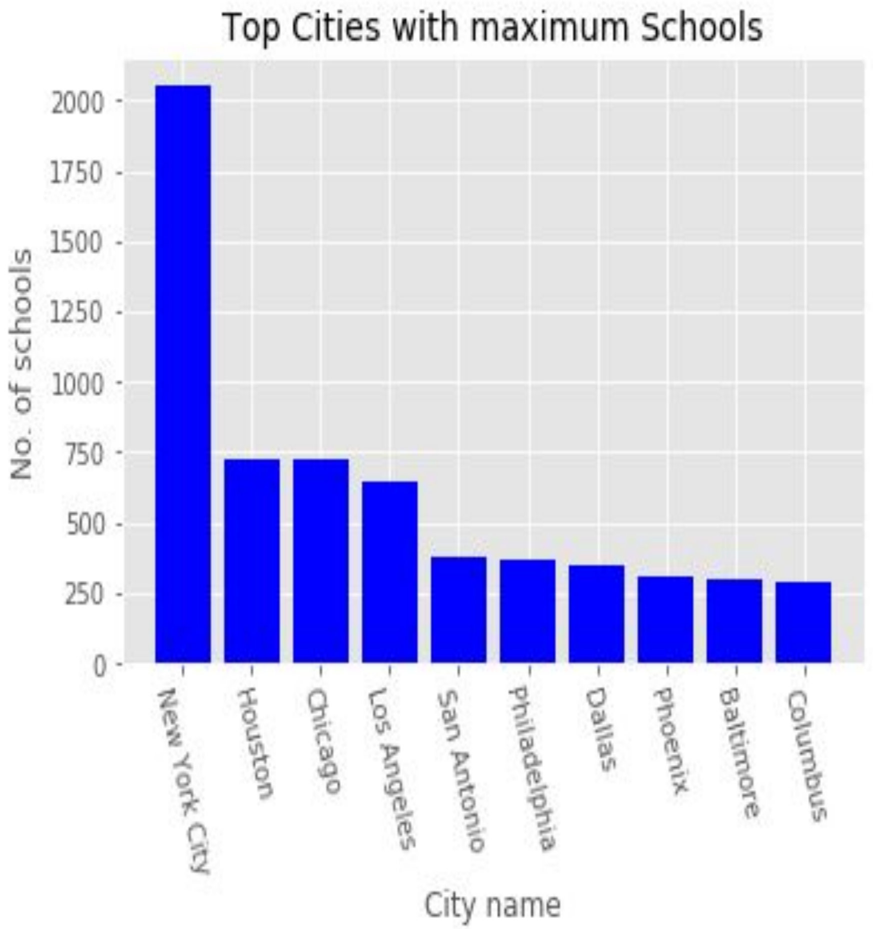
[Fig. 13]


School Metro Type
No. of schools
0
5000
10000
15000
20000
suburban
urban
rural
unknown
town
Metro Type

[Fig. 14]


Top States with maximum Schools
No. of schools
0
1000
2000
3000
4000
5000
6000
7000
8000
California
Texas
New York
Florida
Illinois
Michigan
Ohio
North Carolina
Pennsylvania
Georgia
state name

[Fig. 15]


Top Cities with maximum Schools
No. of schools
0
250
500
750
1000
1250
1500
1750
2000
New York City
Houston
Chicago
Los Angeles
San Antonio
Philadelphia
Dallas
Phoenix
Baltimore
Columbus
City name

[Fig. 16]

References

- MPack-Ambari:-https://medium.com/@mykolamykhalov/integrating-apache-airflow-with-apache-ambari-ccab2c90173
- https://github.com/miho120/ambari-airflow-mpack
- https://medium.com/@mykolamykhalov/integrating-apache-airflow-with-apache-ambari-ccab2c90173
- https://medium.com/a-r-g-o/installing-apache-airflow-on-ubuntu-aws-6ebac15db211
- Sequential Executors:-https://airflow.apache.org/start.html
- Local executors:-http://airflow.apache.org/_modules/airflow/executors/local_executor.html
- Celery Executor:-https://airflow.apache.org/howto/executor/use-celery.html
- BranchPythonOperator:-https://airflow.apache.org/concepts.html
- Kaggle Dataset:-https://www.kaggle.com/donorschoose/io/home
- Yaml encryption:- https://pypi.org/project/yaycl-crypt/