Summary Page of Portfolio Reports

For CSE 578 Data Visualization, CSE 531 Multiprocessor and Distributed Operating Systems, and CSE 575 Statistical Machine Learning

Hannah O. Ajoge
hajoge@asu.edu and ohuajo@gmail.com

***Abstract*— This summary page is an overview of the portfolio reports I submitted as part of my fulfillment of the requirements for the award of the Maters of Computer Science (MCS) at Arizona State University. The three reports utilize Python to address questions in the subfields of Big Data, Cybersecurity and Artificial Intelligence respectively.**

I. Introduction

Projects executed by students, including capstone project involves creativity, critical thinking, and advanced problem-solving skills. Such projects are required of the students, to prove the students' abilities, attained skills, and solve relevant problem in their field. [1]. I thus present in this Portfolio, a sample of my acquired/enhanced abilities, attained skills and the problems solved in the field of Computer Science as a result of my undertaking the journey of studying for an MCS degree.

II. CSE 578 Data Visualization

This was a team project, in which we used diverse data visualization techniques to identify and present six top attributes that are suitable for predicting income. These attributes are aimed at being a useful strategy for marketing and increasing enrollment for a hypothetical UVW College. We specifically used stacked bar charts, grouped bar charts, pie charts, mosaic plots, histogram, and sunburst plot for visualizing the top attributes. As a result of these visualizations, we showed that being married, being a wife, possessing a college degree, being younger than 46, working more hours, and being a male is associated with earning more than $50,000. We suggested further analyzes of these top attributes with more complex analysis like k-means with L2-normalization, principal component analysis or an autoencoder as a necessary implementation as part of "UVW" next marketing strategy.

Our team adopted the flexible Kanban software development strategy. It was my responsibility and privilege to oversee the project activities as an Agile team facilitator. In addition to my team-building activities, I initiated and drafted a significant portion of the system documentation and executive report. Out of the 13 visualizations produced by my team of six students (myself inclusive), I specifically assessed race (pie charts), sex (sunburst plot), and native country (choropleth maps) as factors that can be used to predict income.

From this project I learnt how to execute visualization methods like choropleth maps, exploded pie charts, treemaps, sunburst charts, etc. I also enhanced my Python skill using Jupyter notebook. Furthermore, I had the opportunity to develop my people management skills as an Agile team facilitator.

III. CSE 531 Multiprocessor and Distributed Operating Systems

In this course I executed three projects, in which I built a distributed banking systems using gRPC (general/Google Remote Procedure Call) API. I first built a distributed banking system in which customers communicate with specific branches. The branches executed customer requested transactions and maintained data-centric consistency. Then I bult upon the first project by implementing Lamport's logical clock algorithm to maintain logical time in an asynchronous message-passing system. Finally, I built on the first project but deferred its aim of data-centric consistency to client-centric consistency.

From these projects, I learnt how to implement a distributed system using gRPC. I understood and implemented different consistency models. I also enhanced my Python programming skill, by understanding how to implement Python class, and learnt to utilize the multiprocessing and the os libraries.

IV. CSE 575 Statistical Machine Learning

In this project I implemented both k-means and k-means++ algorithms from scratch in Python, and observed the superiority of k-means++ over basic k-means.

From this project, I acquired adequate experience in executing unsupervised machine learning algorithms. The project also enhanced my Python programming skill.

References

[1] F. K.A. Salem, I. W. Damaj, L. A. Hamandi, and R. N. Zantout. "Effective Assessment of Computer Science Capstone Projects and Student Outcomes." International Journal of Engineering Pedagogy, vol. 10, no. 2, International Society for Engineering Education (IGIP), Kassel University Press, 2020, pp. 72–93, doi:10.3991/ijep.v10i2.11855.

Individual Portfolio Report

For CSE 578: Data Visualization

Hannah O. Ajoge
hajoge@asu.edu and ohuajo@gmail.com

I. Introduction

Data visualization is critical for exploring and communicating findings from diverse data types. Thus researchers have to organize and describe the design space of data visualizations strategically to inform design choices [1]. This strategic construction and analysis of visualization design space entail in-depth analysis of literature visualizations[1].

Thus, in this project, we used analysis of literature and visualization to identify six top attributes that are suitable for predicting income. These attributes are aimed at being a useful strategy for marketing and increasing enrollment for a hypothetical UVW College. In this team project, the stakeholders were thus the UVW college executives, to whom the project outcome was to be presented. The executives are the stakeholders because the executives have legitimate claims on the college, are affected, and can be affected by the college's objectives [2]. In an Agile team project, there are usually different roles like product owner or scrum master. However, our team adopted the Kanban software development strategy, this is more flexible than Scrum and thus no need for such roles [3]. It was then a joint responsibility by the team members to maximize efficiency and reduce the time taken to complete tasks. It was thus my responsibility and privilege to oversee the project activities as an Agile team facilitator.

II. Description Of Solution

Our team communicated the importance of our selected attributes in a visually appealing way to the executive team. This was done by first numbered our variables in alphabetical order. Then we visualization the 13 user stories, viz: Age, Capital Gain-Loss, Total Education, Education-num, FNLWGT, Hours-per-week, Marital Status, Native Country, Occupation, Race, Relationship, Sex, and Working Class as user stories 1 to 13, respectively. Finally, we utilized the correlation coefficients to identify our six top user stories (attributes) in decreasing priority to be Marital Status, Relationship, Education, Age, Hours-per-week, and Sex. Below (figures 1 to 7) are our top six visualizations and the correlation matrix used for deciding the selection. Our code is available on https://github.com/ohuajo/Falcons_Income_Estimator.

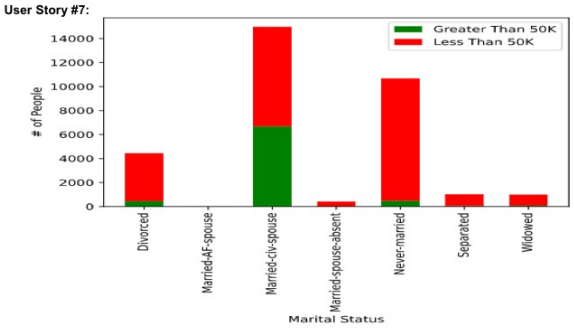

Figure 1: Stacked Bar Chart for Marital Status Vs Income

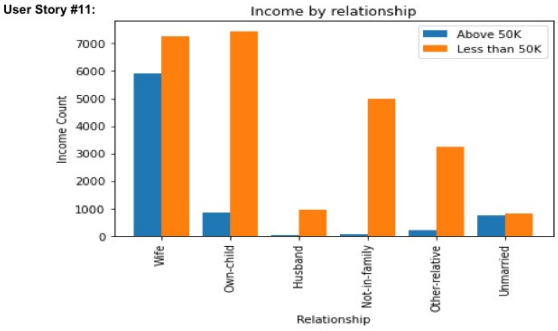

Figure 2: Grouped Bar Chart for Relationship Vs Income

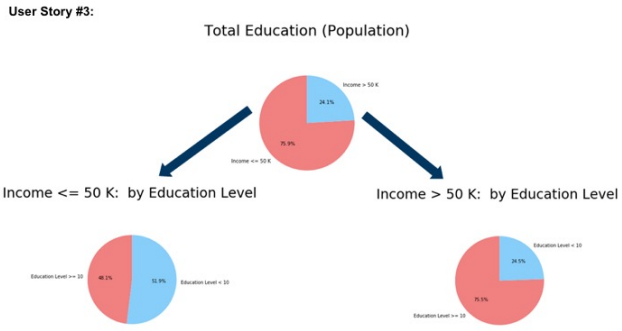

Figure 3: Pie Charts for Total Education Vs Income

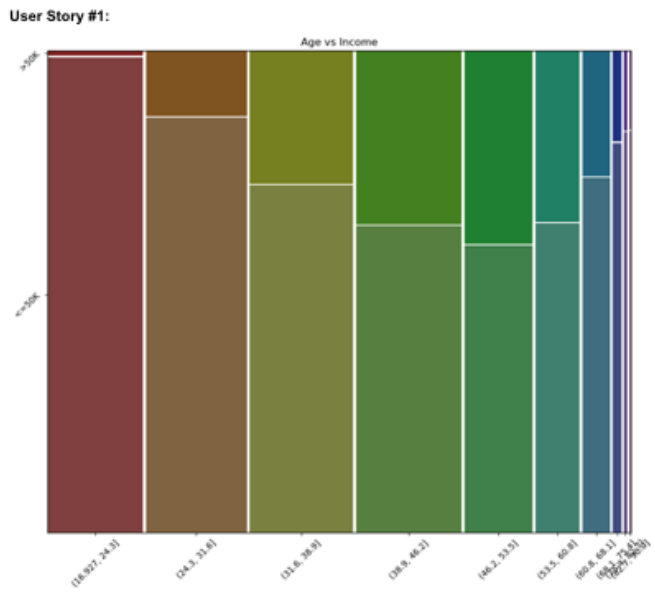

Figure 4: Mosaic Plot for Age Vs Income

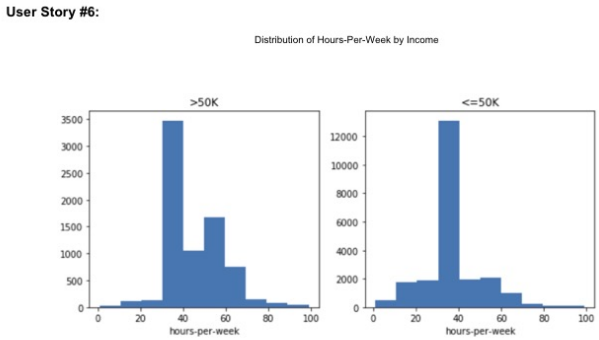

Figure 5: Histogram for Hours-per-week Vs Income

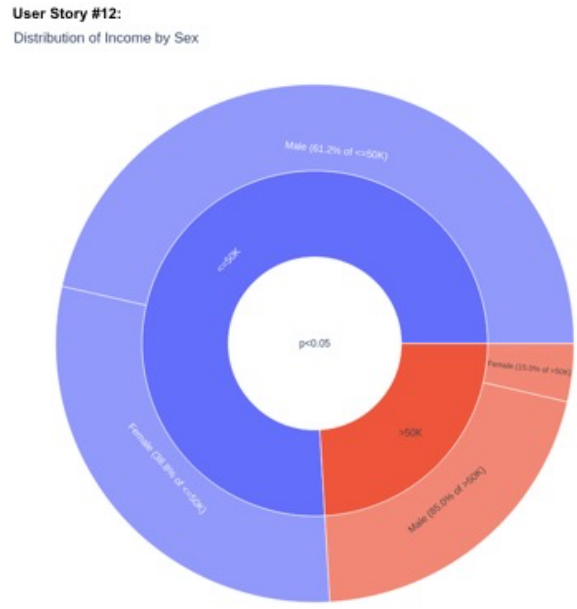

Figure 6: Sunburst Plot for Sex Vs Income

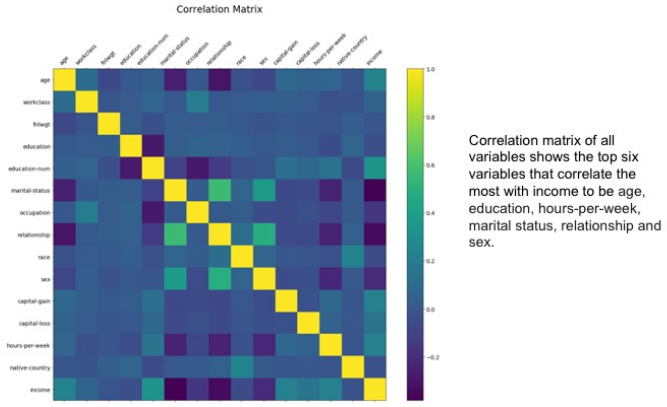

Figure 7: Correlation Matrix Showing the Strength of the Relationship Between the Variables (User Stories)

III. RESULTS

The visualizations above showed how the six top chosen attributes relate to the income of a group; and how the attributes will steer UVW's marketing strategies. Below are the interesting findings.

User Story 7 (visualization by Intzar Singh)

Marital status was selected as the top attribute because it had the strongest correlation with income. This attributed was visualized using a stacked bar chart. The chart shows what proportion of the people in each marital status category make more or less than the $50,000 threshold. The only group that shows a significant portion making more than $50,000 is the group of people who are married. This would make a compelling choice for further evaluation of how significant the difference in someone's marital status could be on their income level.

User Story 11 (visualization by Gad Asare)

Relationship had the second-strongest correlation with income. This attribute was visualized with grouped bar charts. From the chart, it could be concluded that wife and own-child were the most people who earned less than $50,000. However, wife also was the only group with a significant number of people making above $50,000.

User Story 3 (visualization by Pierre LeBlanc)

Total education was regarded as the third top attribute. This was visualized with pie charts. A pie chart was used to show that 75% of individuals earn less than $50,000. A pie charts of the two different income levels showed that having higher education (> level 10) increases your chance of earning above $50,000.

User Story 1(visualization by Michael Salzarulo)

Age ranked fourth among the top user stories. A mosaic plot showed that after age of the 46 the width of

the bars rapidly deteriorates. The older the people the more likely to earn $50,000. Further analysis of age is warranted, especially with a third parameter to identify a deeper relationship to individual income.

User Story 6 (visualization by Jiteng Xu)

Hours-per-week ranked as the fifth most important variable. Histograms showed that the people who worked over 40 hours were more likely to earn above $50,000.

User Story 11 (visualization by Hannah Ajoge)

Sex ranked as the sixth top variable. Here I used a sunburst chart to show that males dominate both income groups. Males were more likely to earn above $50,000 (p= 0.0 by chi-square). Although this was the least correlated among the top six, it will make a good marketing strategy to market to both males and females.

As a result of these visualizations, we concluded that:

- Our visualizations have identified six top variables that strongly correlate with income.
- We showed that being married, being a wife, possessing a college degree, being younger than 46, working more hours, and being a male is associated with earning more than $50,000.
- Further analyzes of these top attributes with more complex analysis like k-means with L2-normalization, principal component analysis or an autoencoder will be necessary before implementation as part of UVW next marketing strategy.

IV. My Contribution

I attended all meetings; and I actively took part in brainstorming, strategizing and discussing the projects. I started the team slack channel, hosted the zoom meetings, co-chaired meetings, and functioned as an Agile team facilitator. I suggested our team name of Falcons because of my vision of speedy delivery. As an Agile team facilitator, my vision as a coach was to ensure that my team is high performing. From the agile coach's point of view, high-performing teams exhibit two qualities, delivery and continuous improvement [4]. Thus my focus was to deliver tasks in speedily and with continuous improvement. I was able to achieve this through enabling leadership, by enhancing the quality of interactions amongst team members.

In addition to my team-building activities, I initiated and drafted a significant portion of the system documentation and executive report. Out of the 13 visualizations produced by my team, I specifically assessed race (pie charts), sex (sunburst plot), and native country (choropleth maps) as factors that can be used to predict income. Among the three attributes I assessed, sex made it to be one of the top six attributes identified by the team. In addition to Sex, which I have already explained above, below (figures 8 to 9) are the other two visualizations I implemented and the interesting findings.

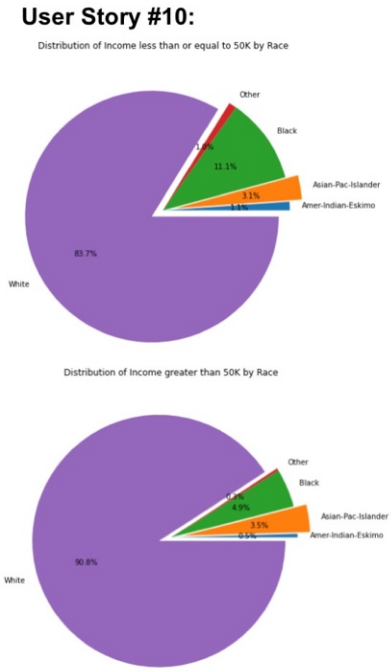

Figure 8: Pie Charts with Exploded Slices for Race Vs Income

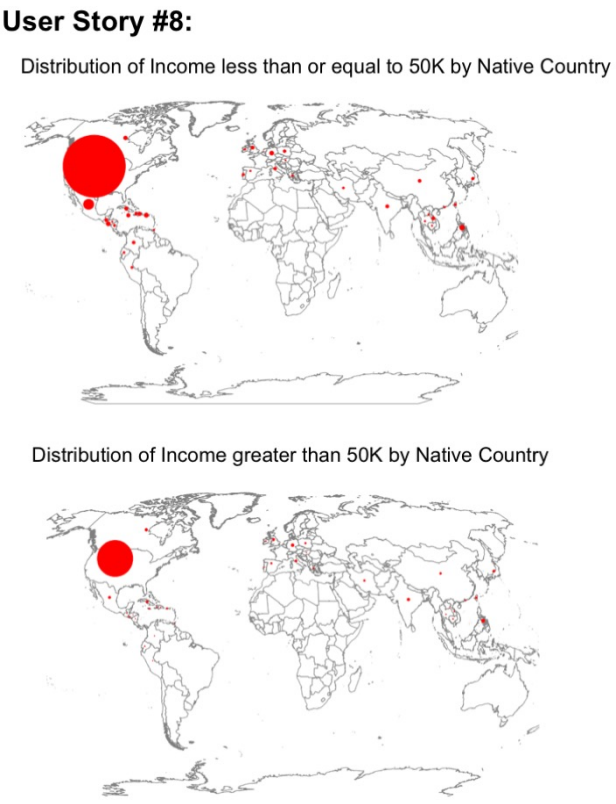

Figure 9: Choropleth Maps for Race Vs Income

User Story 10 (visualization by Hannah Ajoge)

Race was the next significant attribute among the three that I visualized. I used pie charts with exploded slices to show that the percentage of Whites and Asian-Pac-Islanders increases significantly from earning less than or equal to $50,000 to earning more than $50,000. This was found to be significant (p=2.30e-70 by chi-square). However, this was not a top priority because the significance could not be supported by the strength of the relationship with income.

User Story 8 (visualization by Hannah Ajoge)

Here I used choropleth maps to show that the native country of the people who participated in the survey was predominantly the United States. The same pattern of native countries could be seen in the two groups of incomes. Most of the people came from North America, South America, Europa, and Asia. Participants did not come from Africa nor Australia. Though the choropleth map was aesthetically significant, the lack of a strong relationship with the income disqualified the attribute of being among the top six attributes considered to be important to the business objective.

As a result of my three visualizations, I concluded that :

- My visualizations have contributed to the identification of one of the six top variables that strongly correlate with income.
- I showed that being a male, being a white person, and being native to the United States is associated with earning more than $50,000.
- Further analyzes of these three variables and the five other top attributes with more complex analysis like k-means with L2-normalization, principal component analysis or an autoencoder will be necessary before implementation as part of UVW next marketing strategy.

V. LESSONS LEARNED

The new expertise I obtained through this project include the following:

1. I learned visualization methods like choropleth maps, exploded pie charts, treemaps, sunburst charts, and others I wasn't aware of previously.
2. I have also learned how to use Jupyter notebook to execute these visualizations. I also realized that the Python skills I have acquired in the course were also useful in another course I was taking at the same time.
3. Being an Agile facilitator in a team project setting. Although, managing people challenges is more of an art than a science; it is still a critical part of software development in a team setting [5]. The sources of team problems could be the organization, the project, the team, or an individual. An example of such problems includes decision-making [5]. Thus it is very important that a competent Agile coach/facilitator or Scrum master ensures effective team decision-making. In my role as a facilitator, I came across situations in which we had to make timely decisions. I usually restore to voting on the decision. To me, my most important experiential learning came from my leadership role. I made a deliberate effort to enhance my leadership skill and also to seek feedback from teammates individually (one on one) and as a group. I have also suggested connecting through LinkedIn and have been able to connect with two teammates. I intend to continue to collaborate with them and to consciously try to get feedback that can enhance my leadership skill.

My most important learning, as I have noted above, comes from my role as an Agile facilitator. I was excited to have worked with a group of five other teammates who were responsive, responsible, and cooperative. These great gentle people who made our teamwork fruitful in alphabetical order of first names are Gad Asare, Intzar Singh, Jiteng Xu, Michael Salzarulo, and Pierre LeBlanc

REFERENCES

[1] A. Crisan, J. L. Gardy and T. Munzner. "A Systematic Method for Surveying Data Visualizations and a Resulting Genomic Epidemiology Visualization Typology: GEViT." Bioinformatics 35.10 (2019): 1668–1676.

[2] M. H. Yip, and T. Juhola. "Stakeholder Involvement in Software System Development – Insights into the Influence of Product-Service Ratio." Technology in society 43 (2015): 105–114.

[3] E. L.-C. Law and M. K. Lárusdóttir. "Whose Experience Do We Care About? Analysis of the Fitness of Scrum and Kanban to User Experience." International journal of human-computer interaction 31.9 (2015): 584–602.

[4] G. Bäcklander. "Doing Complexity Leadership Theory: How Agile Coaches at Spotify Practise Enabling Leadership." Creativity and innovation management 28.1 (2019): 42–60.

[5] K. Conboy, S. Coyle, X. Wang and M. Pikkarainen People over Process: Key Challenges in Agile Development. IEEE software. [Online] 28.4 (2011): 48–57. Minna

Individual Portfolio Report

For CSE 531: Distributed and Multiprocessor Operating Systems

Hannah O. Ajoge
hajoge@asu.edu and ohuajo@gmail.com

I. Introduction

This report covers three projects that I executed. In the first project, the aim was to build a distributed banking system in which customers communicate with specific branches. The branches receive requests for transactions, execute the transactions and also ensure the propagation of the balance across all the branches in order to maintain data-centric consistency. The second project builds on the first project by implementing Lamport's logical clock algorithm on the first project so as to maintain logical time in an asynchronous message-passing system. Finally, the third project builds on the first project but defers in its aim of client-centric consistency instead of the aim of data-centric consistency as in the previous two projects. What is central to the three projects, is that the distributed banking systems were built using gRPC

gRPC (general/Google Remote Procedure Call) API (application programming interface) is a cross-platform, language and platform independent, general-purpose infrastructure RPC-based protocol that was published by Google in 2015 [1,2]. gRPC was developed for the web and it runs over HTTP/2.0. gRPC is faster than REST (REpresentational State Transfer) API and API developers do not need HTTP commands in their codes. The API developers only need to select procedures to call and the right parameters as well[3].

II. Description Of Solution

My implementation of the three projects are similar in the first four subsections and defer in the last (implementation of the server). For each project, I executed my implementation in a single folder. Below are summarized version of my implementations:

A. Definition of service in a .proto file: I implemented my .proto file, named as example.proto. This file comprised of definitions to describe the service that the server provides. I described the syntax to use protocol buffer version 3 (proto3), because proto3 allows the use of the full range of gRPC-supported languages, as well as avoids compatibility issues with proto2/3 clients-servers communication[4]. I then defined a service named RPC which defines the interaction of the servers with the clients. In the first project, I used a single RPC called MsgDelivery for customer to branch communication and a file system for branch to branch communication for the propagation. In the subsequent projects, I used a second RPC called ClockUpdate to handle the branch to branch communication. Finally, I implemented the message schema's data types to be structured as strings.

B. Automatic generation of client and server stubs for service using protocol buffer compiler: The result was two files - example_pb2.py (containing generated protocol buffer code) and example_pb2_grpc.py (containing client and server classes corresponding to protobuf-defined services). I executed the following command from within the folder in the terminal: python -m grpc_tools.protoc -I./ --python_out=. --grpc_python_out=. example.proto

C. Input test file: Each provided test input was copied and pasted into a file and saved as "input.json". The json files contain a list of customers and branch processes.


```
[
  {
    "id" : 1,
    "type" : "customer",
    "events" :  [{ "id": 1, "interface":"query", "money": 400 }]
  },
  {
    "id" : 2,
    "type" : "customer",
    "events" :  [{ "id": 2, "interface":"deposit", "money": 170 },{ "id": 3, "interface":"query", "money": 400 }]
  },
  {
    "id" : 3,
    "type" : "customer",
    "events" :  [{ "id": 4, "interface":"withdraw", "money": 70 },{ "id": 5, "interface":"query", "money": 400 }]
  },
  {
    "id" : 1,
    "type" : "branch",
    "balance" : 400
  },
  {
    "id" : 2,
    "type" : "branch",
    "balance" : 400
  },
  {
    "id" : 3,
    "type" : "branch",
    "balance" : 400
  }
]
```


Fig. 1: Input file (input.json) for project 1 and 2


```
[
  {
    "id" : 1,
    "type" : "customer",
    "events" :  [{"interface":"deposit", "money": 400,"dest":1  },{"interface":"withdraw", "money": 400,"dest": 2 }, {"interface":"query", "dest": 2 }]
  },
  {
    "id" : 1,
    "type" : "bank",
    "balance" : 0
  },
  {
    "id" : 2,
    "type" : "bank",
    "balance" : 0
  }
]
```


Fig. 2: Input file of monotonic writes example for project 3


```
[
  {
    "id" : 1,
    "type" : "customer",
    "events" :  [{"interface":"deposit", "money": 400,"dest":1  }, {"interface":"query", "dest":2  }]
  },

  {
    "id" : 1,
    "type" : "bank",
    "balance" : 0
  },

  {
    "id" : 2,
    "type" : "bank",
    "balance" : 0
  }

]
```


Fig. 3: Input file of read-your-writes example for project 3

D. Implementing the client (Customer): For projects 2 and 3 I updated the createStub and the executeEvents methods of the given Customer class for creating the customer stub and sending out events to the branches. For the first project I did not use the createStub method but implemented the algorithm under the *if__name == "__main__"* idiom. To send out requests to the appropriate branch, a valid request message is created and then a call is made inside executeEvents. The executeEvent method sends a string with the information of (1) a unique identifier of the customer, (2) an event/transaction of deposit/withdrawal/query, and (3) a write set in the case of project 3. This write set is critical for enforcing client-centric consistencies.

To ensure that my requests from the Customer are multi-processed for the third project, I created a function called setupRequest outside the Customer class. The setupRequest function accepts a list and passes the list elements to Customer class. Then the createStub and the executeEvents methods of the Customer class are executed.

Finally, I ensured that on running the Customer.py file, the "input.json" file is read and process for sending requests to the appropriate server. Then response to produce the output file ("output.json"). Besides the requirement, I ensured that after each request has been responded to and written to the output file. The client should shut down the appropriate servers.

E. Implementing the servers (Branches): The major differences in the projects are peculiar to the execution of the branches. Different methods and functions were utilized in the branches, depending on the project. For project one, a file system was utilized in achieving propagation bank balances across the branches. For project 2, software counter Lamport's algorithm was utilized in ensuring logical clock timestamps. For project 3 the write sets that came along with the request enforce two client-centric consistency models, the monotonic writes and read-your-writes. The codes for the implementations are available on GitHub as following.

1. Project 1: *https://github.com/ohuajo/gRPC-based-distributed-banking-system*

2. Project 2: *https://github.com/ohuajo/Implementing-Lamport-s-Logical-Clock*

3. Project 3: *https://github.com/ohuajo/Client-Centric-Consistency*

III. RESULTS

Project 1

On executing the implementation process explained in the preceding section, an output file (output.json) with the expected output format is generated. In addition, files containing the processing IDs are generated (in the test case Branch1_IDs.txt, Branch3_IDs.txt, and Branch2_IDs.txt). The final balance is also stored in a file (old_balance.txt). While the justification for each step of the implementation has been explained along with each step in the preceding section, I wish to succinctly explain the overall results and the justification:

a) Using the implemented Branch.py, Customer.py, example.proto, example.py files and the appropriate input file (input,json); a distributed banking system was built. This system ensured that specific customers communicate with specific branches; and carry out transactions (query on balance, withdrawal, and deposit) appropriately based on unique IDs.

b) The introduction of the "balance.txt" database system and the use of Locks, ensured that as each customer's request is executed in a specific branch, the global balance is updated in a nonconcurrent manner. At the end of the process, the expected global balance is achieved and graciously retire as "old_balance.txt".

c) The implementation ensured that the branches are the servers and the customers are the clients in a bidirectional distributed network communication. Finally, to mimic the real-life scenario of exiting connection with a bank branch once clients have successfully completed their transactions, each client stub terminates the server after writing the expected result to the output.json file.


```
{"id":1, "recv":[{"interface": "deposit", "result": "success", "money": 500}]}
{"id":2, "recv":[{"interface": "deposit", "result": "success"}, {"interface": "deposit", "result": "success", "money": 500}]}
{"id":3, "recv":[{"interface": "deposit", "result": "success"}, {"interface": "deposit", "result": "success", "money": 500}]}
```


Fig. 4: Output file ("output.json") for project 1

Project 2

As with project 1 executing the implementation process explained in the preceding section, an output file (output.json) with the expected output is generated. Below are the overall results and the:

a) A distributed banking system was built as in project 1. Most importantly a software and counter-based Lamport's logical clock was implemented. This system ensured that specific customer communicates with specific branches and carry out transactions (query on balance, withdrawal, and deposit) appropriately based on unique IDs; and log the sub-events and the timestamps.

b) Using three methods to implement two each of the sub-events, I produced a successful result that ensures a happened-before relationship. Pivotal to this accomplishment is the use of multiprocessing, locks, and the efficient branch to branch communication. The branch to branch communication followed Lamport's clock algorithm and served as both client (propagating server) and server (receiving branch) as needed. This

agrees with the concept of multi-tiered architectures of the centralized distributed system organization, in which an application server can act as both client and server [5].

c) My implementation's algorithm was more geared to maintaining the needed happened-before relationship over the type of sub-event or the type of transaction. Thus, I implemented my succinct four methods in the branch to execute the six type of sub-events and the three types of transactions appropriately.

d) The implementation ensured that the branches are the servers and the customers are the clients in a bidirectional distributed network communication. Finally, to mimic the real-life scenario of exiting connections with bank branches once a client has successfully completed their transactions, each client stub terminates the server after writing the expected result to the output.json file.

The output file for the second project is not shown because it is bulky. However, it is available in the project GitHub link above.

Project 3

An output file (output.json) with the expected output is also generated as previously. Below are the overall results and the justification:

a) Using the implemented Branch.py, Customer.py, example.proto, and the input file (input.json); a distributed banking system was built. Using a write set that has to be the same for the incoming message and the branch's write set; I enforced both the monotonic writes and the read-your-writes. It was fascinating to see that an implementation could fulfill both consistency models. This system ensured that different branches can handle requests from the same process (from a single customer) and achieve both client-centric models of consistencies.

b) My implementation's algorithm was more geared to maintaining the needed client-centric consistencies and not dwelling on different methods for different transaction types.

c) The implementation ensured that the branches are the servers and the customers are the clients in a bidirectional distributed network communication. Finally, to mimic the real-life scenario of exiting connections with bank branches once a client has successfully completed their transactions, the client stub terminates the server after writing the expected result to the output.json file.

```
1   [{"id": 1, "balance": 0}]
```

Fig. 4: Output file of monotonic writes example for project 3

```
1   [{"id": 1, "balance": 400}]
```

Fig. 4: Output file of read-your-writes example for project 3

IV. Lessons Learned

The new expertise I got through this project include:

1. I learned how to implement a distributed system using gRPC. Before now I had taken no course in operating systems, but this course made me consider studying some needed background knowledge in operating systems and networking.
2. Understanding Python class is critical to implementing the projects. Because the projects need the object-oriented programing features of Python classes. My Python scripting capability was quite limited before starting this course. But as I proceeded through the project, I got a better grip on the implementation of Python classes. A good look into my projects will show an increasing and better utilization of the object-oriented properties of the Python class.
3. Another Python feature I got the opportunity of learning and utilizing is the multiprocessing library. This library, which is usually shipped along with Python distributions, is great for spawning multiple processes and leveraging multiple processors in a single machine. The same library also provided lock feature, which was very useful for my implementing consistency.
4. Another library I learned and utilized is the os. This powerful Python library provided a portable way for using different system-dependent functionalities.
5. My implementations of the projects were initially buggy. I had to learn and utilize print statements in debugging both the client and server scripts. As my debugging skill, which was hitherto limited, improved, I realized I could complete the projects much faster.
6. To ease my implementation of the projects, I had to learn to use Visual Studio Code. VS Code is a code editor that is powerful for building and debugging different applications.
7. Finally, becasue of the lectures and the projects, I have been able to understand and implement various consistency models. Both data-centric models and client-centric models. I have also learned that the level and type of consistency that should be implemented is application-dependent.

REFERENCES

[1] S. Du, J. Lee, and K. Kim. “Proposal of GRPC as a New Northbound API for Application Layer Communication Efficiency in SDN.” Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication. ACM, 2018. 1–6.

[2] S. Kiraly and S. Szekely. “Analysing RPC and Testing the Performance of Solutions.” Informatica (Ljubljana) 42.4 (2018): 555–561.

[3] A.Fang. REST vs gRPC: Understanding Two Very Different API Styles. Accessed March 30th 2021, Last updated February 22, 2021. https://rapidapi.com/blog/rest-vs-grpc-understanding-two-very-different-api-styles/

[4] The gRPC Documentation. https://grpc.io/docs/what-is-grpc/introduction/

[5] Arizona State University Spring B 2021 CSE 531: Distributed and Multiprocessor Operating Systems lecture notes.

Individual Portfolio Report

For CSE 575: Statistical Machine Learning

Hannah O. Ajoge
ohuajo@gmail.com

***Abstract*— This report is the individual portfolio report for Online MSC Course, CSE 575 Statistical Machine Learning. This report explain the k-means project.**

I. INTRODUCTION

The aim of data clustering is to divide a set of objects into groups of objects such that objects in the same group/cluster are more similar to each other (according to measured or perceived intrinsic characteristics) than to objects from other group/clusters [1-2]. Clustering analysis is an unsupervised learning process, one of the fundamental methods of discovering and understanding underlying patterns embodied in data and is often used as a preliminary step for data analytics [1-2]. Clustering analysis is widely utilized by many disciplines that include image bioinformatics, financial analysis, segmentation, text mining and wireless sensor networks [2]. The three main purposes of data clustering are understanding the: (1) underlying structure of the data - gain insight into data, generate hypotheses, detect anomalies, and identify salient features; (2) natural classification of the data - identify the degree of similarity among forms or organisms; and (3) compression of data - as a method for organizing the data and summarizing data through cluster prototypes [3]. Among the many clustering algorithms that have been developed in the past sixty years, the k-means algorithm is one of the oldest and most popular clustering algorithms [1]. This is because k-means clustering is simple, efficient and easy to implement [2].

K-means algorithm (adapted from Jain 2010 [3])

Let X={xi}, i=1,…,n be the set of n d-dimensional points to be clustered into a set of clusters, C={ck,k=1,…,K}. K-means algorithm finds a partition such that the squared error between the empirical mean of a cluster and the points in the cluster is minimized. Let μk be the mean of cluster ck. The squared error between μk and the points in cluster ck is defined as:

$$J(c_k) = \sum_{x_i \in c_k} ||x_i - \mu_k||^2$$

The goal of K-means is to minimize the sum of the squared error over all K clusters,

$$J(C) = \sum_{k=1}^{K} \sum_{x_i \in c_k} ||x_i - \mu_k||^2$$

Despite the popularity of k-means clustering, it suffers from a number of limitations, such as initialization sensitivity, susceptibility to noise and vulnerability to undesirable sample distributions [1-2]. To compensate for these limitations, the basic k-means algorithm has been extended in different ways [3]. One such extension is the k-means++ algorithm, which reduces the problem of sensitivity to initialization [4].

K-means++ was first an augmentation to K-means that was introduced by David Arthur and Sergei Vassilvitskii [5]. According to Arthur and Sergei 2007, K-means++ outperforms K-means in terms of both accuracy and speed. They obtained an algorithm that is O(log k)-competitive to K-means algorithm with the optimal clustering. They proposed "a variant that chooses centers at random from the data points, but weighs the data points according to their squared distance squared from the closest center already chosen".

K-means++ algorithm (adapted from Arthur and Sergei 2007 [5])

A specific way of choosing centers for the k-means algorithm was proposed. In particular, let D(x) denote the shortest distance from a data point to the closest center that have already been chosen. Then, they defined the following algorithm, which they called k-means++.

1. Take one center c_1, chosen uniformly at random from X .
2. Take a new center c_i, choosing $x \in X$ with probability $\frac{D(x)^2}{\sum_{x \in \mathcal{X}} D(x)^2}$
3. Repeat Step 2. until k centers have been taken altogether.
4. Proceed as with the standard k-means algorithm.

They called the weighting used in Step 2. simply as "D2 weighting".

Applications of K-means++ algorithm

Other authors have shown K-means++ as a superior algorithm and of useful application. One of such authors is Michael Schindler, who reviewed several clustering algorithms[6]. Schindler explained that it can be shown:

> that if this [K-means++] is run 2^k times, it is likely for the best solution found to be within a constant factor of the optimal. Their empirical results on both synthetic and natural datasets showed k-means++ is a good improvement in both running time and solution quality over k-means. [6]

Lee *et al* 2007[7] utilized k-means++ to obtain the best possible initial set of seed points for geographical cluster calculation. These geographical calculations were used for "analysis of the correlation of annotated information unit (textual) tags and geographical identification metadata geotags"[7].

Objective of Project

In this project my objectives were to implement both k-means and k-means++ algorithms from scratch in Python. I was also interested in observing the superiority of k-means++ over basic k-means.

II. SOLUTIONS

In this project, I implemented two strategies of k-means algorithm - the basic k-means algorithm and the k-means++ algorithm (an extension of the basic k-means algorithm). The implementation was done from scratch in Python. I utilized the Jupyter notebook (lab) provided within Coursera platform for my coding.

Strategy 1

I entered the four last digits of my student ID and with that generated my set of initial centroids. I first viewed the initial utilized the initial centroids locations on the data through scatter plot. From time to time I visualized my centroid and assigned clusters through colored scattered points.

Then I created a function "dist_toCentroid" that takes a set of centroids and a data frame with two columns (x and y). The function then generate a new set of centroids and a data frame. The generated data frame contains distances of each point from the centroids, a column known as closest which indicate the cluster that each point is assigned, and a column named color which contains letters respective to the assigned cluster (the letter can be used to assign colors to data points in scatter plots).

Then I produced while loop that ensures that the function "dist_toCentroid" is used to iteratively generate new centroids until the centroids don't change (when no single data points change clusters anymore).

Finally I calculated the objective function, by calculating the squared errors for each cluster data point to the cluster's centroid, and then summed up the squared errors. This sum of squared errors is the objective function or J(C).This process was executed for both k1 (K=3) and k2 (K =5).

Strategy 2

For Strategy two I started with only the first centroid and then computed the remaining initial centroids by assigning data point that is furthest from the previous centroid(s). After obtaining the remaining initial centroids, I computed the final centroids and the objective function as in Strategy1.

Strategy 2a

Since the assigned Ks for the two strategies are different – 3 and 5 (for strategy 1) and 4 and 6 (for strategy 2); it may not be plausible to say that the extension to the basic k-mean algorithm provided better result (lower objective function). To remove this implausibility, I implemented strategy 2 again which I now call strategy 2b. In strategy 2b, I reimplemented strategy 2 but: (1) utilized the same Ks as in strategy – 3 and 5; and (2) utilized the same first initial centroids as in strategy 1.

III. RESULTS

Below are the resulting final centroids and objective functions of the two strategies.

Strategy 1

K1 (K=3) Result

- Final centroid: {1: [[7.2397511895844495, 2.4820826910731952]], 2: [[4.833753175392286, 7.316058236043574]], 3: [[3.2489642305948876, 2.5802769113756954]]}
- Objective function: 1338.1059838

K2 (K=5) Result

- Final centroid: {1: [[2.8749081274874264, 7.010822811156844]], 2: [[2.681986334188929, 2.094615867800809]], 3: [[7.556167822397726, 2.235167959857534]], 4: [[5.257224105388598, 4.255186256593418]], 5: [[6.775311757464789, 8.115401936406947]]}

- Objective function: 598.678798534.

Strategy 2

K1 (K=4) Result

- Final centroid: {1: [[7.252626831256577, 2.4001582635520533]], 2: [[3.2285300905383707, 2.5240486292057867]], 3: [[6.6259253846324615, 7.576149167622678]], 4: [[2.9054774114449513, 6.905122763339948]]}
- Objective function: 789.237972218

K2 (K=6) Result

- Final centroid: {1: [[4.022935100912096, 3.870824728369343]], 2: [[7.756483249146484, 8.556689279063415]], 3: [[2.825447560760127, 1.6546753620598549]], 4: [[2.512257357886769, 7.057170033892222]], 5: [[7.477682316761515, 2.2990031525317582]], 6: [[5.464277356727894, 6.837713536435891]]}
- Objective function: 484.04926087

Strategy 2b

K1 (K=3) Result

- Final centroid: {1: [[7.2397511895844495, 2.4820826910731952]], 2: [[4.830919584356354, 7.299599586723327]], 3: [[3.234890046359086, 2.5530321964002036]]}
- Objective function: 1338.10760165

K2 (K=5) Result

- Final centroid: {1: [[5.339072124427115, 4.465511754739297]], 2: [[7.299749694765005, 8.413318379491713]], 3: [[2.681986334188929, 2.094615867800809]], 4: [[3.1507276114491836, 7.121929064073764]], 5: [[7.556167822397726, 2.235167959857534]]}
- Objective function: 592.528384259

IV. CONCLUSION

In this project I showed that:

1. The extension to the basic k-mean algorithm (k-mean ++) is capable of providing slightly better clustering result or in other words lesser objective function. This can be seen when comparing strategy 1 and strategy 2b, in which the K=5 for strategy 2b provided better result.
2. However the basic K-mean algorithm still performs quite similar to k-mean ++), as there was no difference in objective function when K was 3.
3. As expected, increase in K leads to better clustering result – lesser objective function.

V. LESSONS LEARNED

The new expertise I obtained through this project include ability to:

1. Understanding the concept of k-mean clustering algorithm, as well as it's extension - k-mean ++.
2. Implementing from scratch in Python, the algorithm necessary to solve a k-mean clustering problem.

REFERENCES

[1] G. M. Gan and K.-P. Ng, "k-means clustering with outlier removal," Pattern Recognition Letters, vol. 90, pp. 8-14, 2017, http://www.sciencedirect.com/science/article/pii/S0167865517300740

[2] H. Xie, L. Zhang, C. P. Lim, Y. Yu, C. Liu, H. Liu and J. Walters, "Improving K-means clustering with enhanced Firefly Algorithms," Applied Soft Computing, vol. 84, Article 105763, 2019, http://www.sciencedirect.com/science/article/pii/S1568494619305447

[3] A. K. Jain, "Data clustering: 50 years beyond K-means," Pattern Recognition Letters, vol, 31, no 8, pp. 651-666, 2010, http://www.sciencedirect.com/science/article/pii/S0167865509002323

[4] Lecture videos

[5] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding," Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms. 2007.

[6] M. Shindler, "Approximation Algorithms for the Metric k-Median Problem," https://web.archive.org/web/20110927100642/http://www.cs.ucla.edu/~shindler/shindler-kMedian-survey.pdf

[7] S. S. Lee, D. Won and D. McLeod, "Discovering Relationships among Tags and Geotags," Association for the Advancement of Artificial Intelligence (www.aaai.org), 2007,

https://web.archive.org/web/20160303223350/http://sir-lab.usc.edu/publications/2008-ICWSM2LEES.pdf