CSC 501 - Assignment 2

Joe Howie

November 27, 2020

Contents

List of Figures

List of Tables

Abstract

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

---

1 Creativity

Theme We want to expose networks of people that related based on the financial decisions they make with regards to loan payments. By constructing a *user centric "4-star" bipartite graph* from the relational data, we will ask particular questions to relate client-accounts to each other. These questions are designed display key financial decisions by different users. The answer to each question will be a User Network Graph (UNG).

Question 1 *Find clusters of users that are loaned money at the same age.*

Question 2 *Find clusters of users that are loaned similar amounts of money.*

Question 3 *Find clusters of users that paid similar amounts of their loan back on time.*

Goal By answering all these questions and superimposing the answers, we obtain a multi-graph G_F relating users to one another based on three key financial decisions. The multi-graph G_F is the superposition of three UNG's with three distinct edge sets—one from each question. The graph can be used to readily predict answers to questions banks are interested in knowing like: *"how likely a new user would be to repay a loan on time based on similar users behaviour in the network?"*. One could build a machine learning program to make such predictions using G_F as training data.

2 Relational Model

2.1 The Data

We are presented with a normalized relational data modelling financial information. We want to ask question that relate users to each other. Analyzing the data set we see accounts take out loans, and account has clients. Accounts have at most two clients, which only come in male, female pair; hence we assume these pairs are married couples.

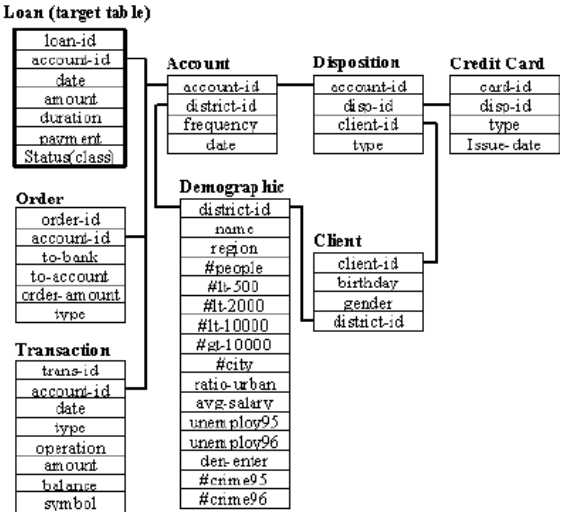

Figure 1: ER Data Diagram. Source: here

Creating Users For the analysis we want to have users with a birthday and sex—along with a unique id. Since couples make joint financial decisions we merge them into one user with sex as "c" for couple, and take the males age. To get age and sex we need to join account with disposition on account_id, followed by client on client_id. Now we have a set of client-accounts that make up our user base; we have the three desired characteristics: birthday, sex, and unique id—use account_id.

Reduced Relation Now the ER diagram consists of three relations, with several relationship between. There is a 1:N relationship between users and loans—we checked that each user had only one loan with a simple query. An 1:N relationship is equivalent to a bipartite graph where one of the sets has exactly one edge for each vertex. Since all three questions have to do with different aspects of loans and how they relate to users, we get four 1:N bipartite relationships car-

rying relevant loan information. Hence, the earlier term *user centric "4-star" bipartite graph*:

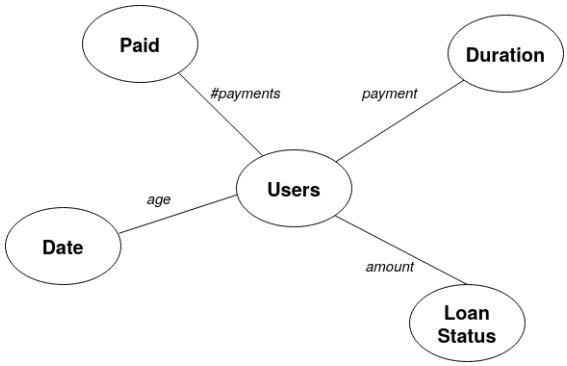

Figure 2: *user centric "4-star" bipartite graph*:
Users—unique id, birthday, and sex
Date—loan application date
Loan Status—loan quality (A, B, C, or, D)
Duration—loan duration (12, 24, 36, 48, 60) months
Paid—average payment amount, and total amount paid
age—edge between users and date vertices
amount—edge between users and loan status with weight equal to loan amount
payment—edge between users and duration with weight equal to payment amount
#payments—edges between users and paid with weight equal to number of payments made

The fact table yielding all nine of these attributes—with account_id a.k.a. user id as a primary key—is exactly the adjacency list for the *user centric "4-star" bipartite graph*. Each vertex in figure 2 represents a set isolated nodes, but by applying our questions to this graph we will generate a user financial network multi-graph.

2.2 Procedure

see code sections 1.

3 Graph Data Modelling

3.1 User centric "4-star" bipartite graph

The Query Graph Let $G_{4B} = (V, E)$ be the *user centric "4-star" bipartite graph* from the relational model.

Vertex Set Let $V = U \cup D_a \cup L \cup D_u \cup P$.

Edge Set Let $E \subseteq V \times V$ and $E \subseteq (U \times D_a) \cup (U \times L) \cup (U \times D_u) \cup (U \times P)$.

3.1.1 Vertex Sets

Users Nodes Let U be the set of users in G_{4B}, where each node has a unique ID value, a birthday (yyyy-mm-dd), and sex (m/f/c) assignment. That is, $U = \{v : v \in V \text{ s.t. } v \text{ is a 3-tuple with (id, bday, sex)}\}$ and $U \subseteq V$.

Date Nodes Let D_a be the set of dates that users applied for loans, where each node has a unique date (yyyy-mm-dd). That is, $D_a = \{v : v \in V \text{ s.t. } v \text{ has a date value only}\}$ and $D_a \subseteq V$.

Loan Status Nodes Let L be the set of loan status'. That is $L = \{A, B, C, D\}$, and $L \subseteq V$.

Duration Nodes Let D_u be the set duration of loans in months. That is, $D_u = \{12, 24, 36, 48, 60\}$ and $D_u \subseteq V$.

Paid Nodes Let P be the set of payments made on a loan, which contains two values, average payment, and total paid. That is, $P = \{v : v \in V \text{ s.t. } v \text{ is a 2-tuple with (ave paid, total paid)}\}$ and $P \subseteq V$.

3.1.2 Edge Sets

Age Edge Let $E_{age} \equiv E \cap (U \times D_a)$. For each $e \in E_{age}$, then e has capacity in range $(0, 160)$ corresponding to the age of the user when they applied for the loan, written as $c_{age}(e)$. *NB: 160 was chosen as an arbitrary upper bound on age, in the data set the oldest person taking out a loan was 63.*

Amount Edge Let $E_{amt} \equiv E \cap (U \times L)$. For each $e \in E_{amt}$, then e has capacity in range $(0, 500000)$ dollars (in whichever currency) representing the amount loan written as $c_{amt}(e)$. *NB: 500000 was chosen as an arbitrary upper bound on amount, in the data set the largest loan taking out a was $270000.*

Payment Edge Let $E_{pay} \equiv E \cap (U \times D_u)$. For each $e \in E_{pay}$, then e has capacity in range $(0, 20000)$ dollars (in whichever currency) showing the amount to be paid each month (for the loan to be paid on time) written as $c_{pay}(e)$. *NB: 20000 was chosen arbitrarily, in the data set the largest payment was $9910.*

#Payments Edge Let $E_{times} \equiv E \cap (U \times P)$. For each $e \in E_{times}$, then e has capacity in the set $\{0, 1, \ldots, 60\}$ where the value represents the number of payments made on th loan written $c_{times}(e)$.

3.1.3 Graph Properties

Notice that G_{4B} only has edges from U to the other vertex sets: D_a, L, D_u, P. Hence, there are no edges between nodes from the same set. Thus we term the graph *user centric "4-star" bipartite graph* since it has four bipartite graphs modelled. Furthermore, each bipartite sub-graph models a 1:N relation ship, meaning that each client node has exactly four edges—one of each type. So we get the resulting properties:

1) $|E| = 4\,|U|$
2) $|U| = |U \times D_a| = |U \times L| = |U \times D_u| = |U \times P|$
3) $|U| \approx |P| \approx |D_a|$

That is, P and D_a are nearly the same size as $|U|$, this does not necessarily hold true for larger data set—many people are born on the same day. These properties will be exploited in answering the three questions.

3.2 Edge Induced Graphs (EIG)

Each question is answered by taking a specific edge induced sub-graph, then looking for similar users based on the bipartite relationship. At the end of each question we will have a new user only graph $G = (U,\, E')$, where $E' \subseteq U \times U$ and each $e \in E'$ has an associated capacity function $c(e)$.

Question 1 *Find clusters of users that are loaned money at the same age.* We take only the only the age edges and nodes that are attached, that is $G_{Q1} = \big(U \cup D_a,\, E_{age}\big)$.

Question 2 *Find clusters of users that are loaned similar amounts of money.* We take only the amount edges and the attached nodes, that is $G_{Q2} = \big(U \cup L,\, E_{amt}\big)$.

Question 3 *Find clusters of users that paid similar amounts of their loan back on time.* We take both #payments edges and payment edges along with the attached nodes, that is $G_{Q3} = \big(U \cup P \cup D_u,\, E_{pay} \cup E_{times}\big)$.

3.3 User Network Graphs (UNG)

Edge Cuts The answer to each question is a network graph that connects users by a specific financial parameter carried by the edge weight. For each question's EIG, we make connected based on edges parameters.

Question 1 *Find clusters of users that are loaned money at the same age.*

1) Read edge list E_{age} and create a set C^1 of edge capacity values.

2) $\forall\, c_i^1 \in C^1$ create a set $C_i^1 = \{u : u \in U \text{ s.t. } \exists\,(u, d, c_i) \in E_{age},\ d \in D_a\}$.

3) Since each $u \in U$, G_{Q1} only has one edge, then $C_i^1 \cap C_j^1 = \emptyset\ \forall i, j \in \{1, \ldots, |C^1|\},\ i \neq j$.

4) Thus each set $C_1^1, \ldots, C_{|C^1|}^1$ of user constitutes a collection of complete graphs $G_{A1} \equiv \{K_{|C_1^1|}, K_{|C_2^1|}, \ldots, K_{|C_{|C^1|}^1|}\}$, where graph $K_{|C_i^1|}$ has $|C_i^1|$ nodes, and edges capacities equal to c_i^1.

Question 2 *Find clusters of users that are loaned similar amounts of money.*

1) Read edge list E_{amt} and create a set C^2 of edge capacity ranges.

2) $\forall\, c_i^2 \in C^2$ create a set $C_i^2 = \{u : u \in U \text{ s.t. } \exists\,(u, d, c_i) \in E_{age},\ d \in D_a\}$.

3) Since each $u \in U$, G_{Q2} only has one edge, then $C_i^2 \cap C_j^2 = \emptyset\ \forall i, j \in \{1, \ldots, |C^2|\},\ i \neq j$.

4) Thus each set $C_1^2, \ldots, C_{|C^2|}^2$ of user constitutes a collection of complete graphs $G_{A2} \equiv \{K_{|C_1^2|}, K_{|C_2^2|}, \ldots, K_{|C_{|C^2|}^2|}\}$, where graph $K_{|C_i^2|}$ has $|C_i^2|$ nodes, and edges capacities equal to c_i^2.

Question 3 *Find clusters of users that paid similar amounts of their loan back on time.*

1) Read adjacency list of G_{Q3} and partition the nodes into two adjacency lists: U_{over} and $U_{ongoing}$.

2) For the nodes in both adjacency lists each entry has the following forum: $u_i :\ (p_i, t_i),\ (d_i, pm_i)$, where $p_i \in P,\ d_i \in D_u,\ t_i \in E_{times},\ pm_i \in E_{pay}$. Recall the nodes $p_i \in P$ carry two values: ave payment, and total paid. Let $p_i = (ap_i, tp_i)$.

3) If $\frac{t_i}{d_i} < 1$ then $u_i \in U_{ongoing}$, and if $\frac{t_i}{d_i} = 1$, the $u_i \in U_{over}$. For each list, calculate percentage paid on time pp_i.

4) For $u_i \in U_{over},\ pp_i = \frac{tp_i}{d_i \cdot pm_i}$. For $u_i \in U_{ongoing},\ pp_i = \frac{(tp_i + (ap_i \cdot (d_i - t_i)))}{d_i \cdot pm_i}$, where we estimate the remaining payments based on the average payment made thus far. Save all (u_i, pp_i) pairs to set P_p.

5) Create a set C^3 of pp_i ranges. $\forall c_i^3 \in C^3$ create a set $C_i^3 = \{u : (u, pp) \in P_p\ s.t.\ pp \in c_i\}$.

6) Since each $u \in U$, G_{Q3} only appears once in P_p because each u has exactly two edges one in each set E_{pay}, E_{times} which were used to calculate pp. Hence, $C_i^3 \cap C_j^3 = \emptyset\ \forall i, j \in \{1, \ldots, |C^3|\},\ i \neq j$.

7) Thus each set $C_1^3, \ldots, C_{|C^2|}^3$ of user constitutes a collection of complete graphs $G_{A3} \equiv \{K_{|C_1^3|}, K_{|C_2^3|}, \ldots, K_{|C_{|C^3|}^3|}\}$, where the edges connecting the nodes in each complete graph $K_{|C_i^3|}$ have c_i weight.

3.3.1 Complete Connected Component (CCC) Graphs

Each question gave rise to a graph consisting of complete connected components, that is users clustered to a series of K_l graphs where l is the number of similar users (nodes in the complete graph K_l). In each graph G_{A1}, G_{A2}, G_{A3} the set of vertices are the same, U. We can now combine the edge sets and construct the multi-graph G_F.

3.4 Multi-Graph G_F

From answering all three questions we now have G_{A1}, G_{A2}, and G_{A3}, which are all CCC graphs. For each $i = 1, 2, 3$ we have $G_{Ai} = \{K_{|C_1^i|}, K_{|C_2^i|}, \ldots, K_{|C_{|C^i|}^i|}\}$ where $K_{|C_j^i|}$ is the

jth complete graph with $|C_j^i| = n$ nodes, $\frac{n(n-1)}{2}$ edges each with weight c_j^i. Then each $G_{Ai} = (U, E_{Ai})$, where E_{Ai} are all the edges in $K_{|C_1^i|}$, $K_{|C_2^i|}$, ..., $K_{|C^i_{|C^i|}|}$ and has a size of

$$|E_{Ai}| = \sum_{j=1}^{m} \frac{|C_j^i|\big(|C_j^i| - 1\big)}{2}$$

Constructing G_F To obtain G_F, simply concatenate the Adjacency Lists of $G_{A1} = (U, E_{A1})$, $G_{A2} = (U, E_{A2})$, and $G_{A3} = (U, E_{A3})$. Thus, $G_F = (U, E_{A1} \cup E_{A2} \cup E_{A3})$, where $e \in E_{A1}$ has capacity $c_{age}(e)$, E_{A2} has capacity $c_{amt\_range}(e)$ that is a pair of values in c_{amt} range, and E_{A3} has capacity $c_{percent\_range}(e)$ that is a pair of values in range (0, 1]. The Adjacency List has the form of table 1, where each cell is a Minimum Spanning Tree (MST) of the complete from whence it came. Table 2 gives a summary of the notation in this section.

Table 1: Adjacency List Format of G_F

$u \in U$	E_{A1}	E_{A2}	E_{A3}
u_1	$[(u_i, c_{age}(u_1, u_i)), \dots]$	$[(u_j, c_{amt\_range}(u_1, u_j)), \dots]$	$[(u_k, c_{percent\_range}(u_1, u_k)), \dots]$
$\vdots$	$\vdots$	$\vdots$	$\vdots$

3.4.1 Properties of G_F

Immediate facts we notice about G_F are the APL, Clusters, and Degree Distribution meanings in the model.

APL By construction users are 1, 2, 3 hops from any other user or the users are not connected (no similar financial behaviour).

Clusters We maintain the CCC from G_{A1}, G_{A2}, and G_{A3}, and gain multi-connected clusters—nodes with two, or three edge type connections).

Degree Distribution The node degree on one edge type tells us the size of that community, where the community is defined by the edge value.

Vertex Cover A minimum vertex covering set would define classes of users based on financial.

Table 2: Graph Notation Summary

Symbol	Name	Reference
G_{4B}	*user centric "4-star" bipartite graph*	See Section 3.1
V	Vertex Set of G_{4B}	3.1.1
E	Edge Set of G_{4B}	3.1.1
U	Set of User Nodes	3.1.1
D_a	Set of Date Nodes	3.1.1
L	Set of Loan Status Nodes	3.1.1
D_u	Set of Duration Nodes	3.1.1
P	Set of Paid Nodes	3.1.1
P	Set of Paid Nodes	3.1.1
E_{age}	Set of Age Edges	3.1.2
E_{amt}	Set of Amount Edges	3.1.2
E_{pay}	Set of Payment Edges	3.1.2
E_{times}	Set of #Payments Edges	3.1.2
G_{Qi}	EIG for Question i	3.2
C^i	Set of Capacities from edges in G_{Qi} edge set	3.3
c^i_j	Element of C^i	3.3
C^i_j	Set of User with edge capacities c^i_j	3.3
$K_{\lvert C^i_j \rvert}$	The jth CCC of size $\lvert C^i_j \rvert$ and edge values c^i_j	3.3.1
G_{Ai}	UNG for Question i	3.3
G_F	Multi-Graph from superimposing all G_{Ai}	3.4

4 Algorithm Cost

The three questions work together to generate a Multi-Graph User Network, G_F, to model users financial similarity. G_F can be used to predict likelihoods of future users behaviour based on financial decisions which were not explicitly modelled in the relational database. There were four algorithmic steps taken to generate G_F from the normalized data:

1) Construct the *user centric "4-star" bipartite graph*, G_{4B}, from the normalized relational database.
2) Induce G_{Q1}, G_{Q2}, and G_{Q3} from G_{4B}.
3) Construct G_{A1}, G_{A2}, and G_{A3} from G_{Q1}, G_{Q2}, and G_{Q3} respectively.
4) Superimpose G_{A1}, G_{A2}, and G_{A3} to create G_F.

A run time analysis of steps 2 and 3 follows from the procedures described in section 3.2, and 3.3 respectively.

4.1 Theoretical Analysis

EIG Run Times Obtaining EIG meant for each question selecting particular edge sets from G_{4B}. The adjacency list indexed by the set U is read and selects only desired edges and corresponding nodes. This takes $\Theta(|U|)$, and is done three times—once for each question.

UNG Run Times Constructing UNG meant relating users to each other based on common edge values connecting to specific vertex sets. The adjacency list indexed by the set U of the EIG G_{Qi} for $i = 1, 2, 3$ is read, and an array is constructed of values or range of values for

all the different edge values or function of values seen. Then for each value in the previously constructed array of size p, p different complete graphs are constructed by reading the same adjacency list again and inserting nodes into the complete graph associated the the edge value. This sounds like a lot, but we are only reading the edge list twice each time doing a constant number of operations for each question graph. Hence, this is also $\Theta(|U|)$.

Central Cluster problem The construction the CCC graphs has the potential to be $O(|U|^2)$ if there is only a hand full of edge values—ie a small number of CCC graphs. Formally, if the number of edge values ev is such that $\frac{ev}{|U|} << 1$, then the asymptotic run time becomes $\Theta(|U|^2)$. However, there is an easy trick to avoid this case, which is good because a small number of CCC graphs is not interesting to us.

Resolution Solution Each CCC graph is created based on commonalities in three different attributes which can be thought of a continuous random variables. We connected the CCC graphs based on if two users were in the same range on one of the random variables. Thus, if many users are in one range, increase range resolution to reduce the number of users in one CC graph, which subsequently gives more granular information about each group.

4.2 Experimental Analysis

The run time analysis above, showed that when the graphs are represented as adjacency list of the user node set that steps 2 and 3 run in $\Theta(|U|)$. Recall that in G_{4B}—and subsequently G_{Q1}, G_{Q2}, and G_{Q3}—U is a vertex cover, and hence the graph is fully represented by an adjacency list indexed by user nodes. Furthermore only user nodes appear in G_{A1}, G_{A2}, and G_{A3}—and by extension G_F. Additionally, there was a limit in G_{4B} on the number of edges, $|E| = 4|U|$, so reading adjacency lists of a single user was constant in G_{4B}—and subsequently G_{Q1}, G_{Q2}, and G_{Q3}.

Scaling In python, the function which called steps 2 and 3 was timed for increasing number of users from 10 to just under 700 in steps of 10. The times were stored along with the number of users, which was plotted to show how the algorithm scaled as $|U|$ gets very large. Figure 3 shows the data points in red and a best fit line was calculated with numpy.polyfit and plotted in blue. Hence, up to $|U| \approx 1000$ the algorithm runs in linear time agreeing with the theoretical analysis.

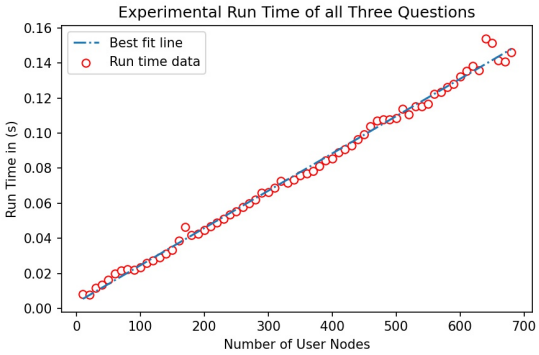

Figure 3: Scaling of all three algorithms

5 Spatial Modelling

G_F as a Vector Space The multi graph can be represented as a vector space, where each user is a point. Let $u = \langle$age, amount range, percent range$\rangle$ be a point in the vector space of age cross amount range cross percentage range. This is a discrete vector space, where multiple users land in on the same point in space. We can visualize slices of the space with two dimensional histograms.

Density Plot The colorbar shows the normalized occurrence of users at that point in the vector space. Further, the occurrences are shown

on a logarithm scale so that low value bins can be observed. The three plots below show all the two dimensional slices of the three dimensional vector space, providing a full picture. These density plots highlight the multi-connected clusters of G_F, which can be used to calculate future users likelihood on of paying back loans on time.

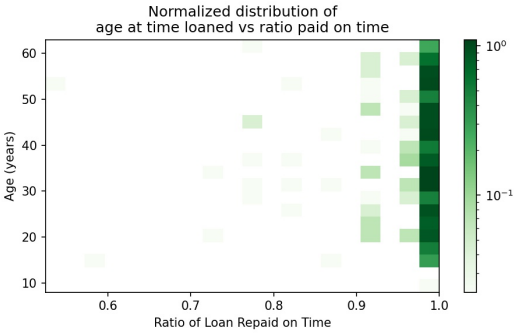

Figure 5: Age loaned vs. Ratio paid on time

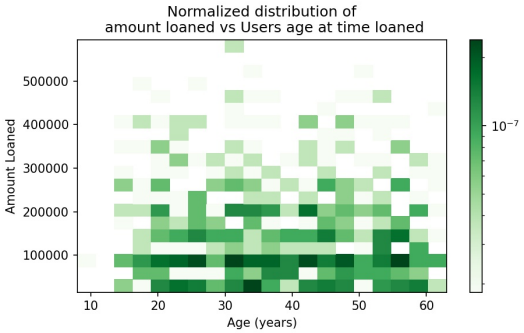

Figure 4: Amount Loaned vs Age loaned

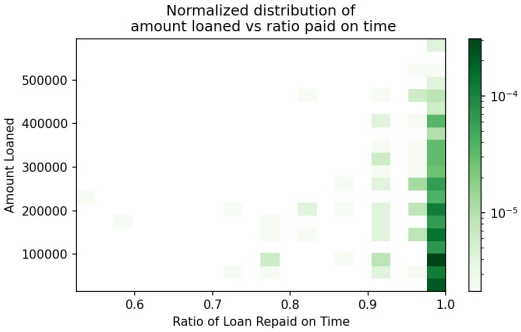

Figure 6: Amount Loaned vs. Ratio paid on time

6 Acknowledgements

Special thanks to the numpy , pandas , and matplotlib docs, without their help this code would have been horribly inefficient. Thank you to Dr. Sean Chester for all the help and guidance on this assignment. And of course thanks to the collective knowledge of people on troubleshooting forums such as: Stack Exchange , Geeks for Geeks , and many more. Finally, the fountain of all wisdom, Wikipedia for quick graph notation referencing such as: Complete Graphs, Bipartite Graphs, and Graph Theory .

7 Appendix: Code

See full code on my GitHub page.