Q1.a:
(i). Describe the equations, the inputs, and how we can classify out-of-sample items with the output.
(ii). Describe the importance of priors in NBC.
A:

(i). The most important assumption that a Naive Bayes Classifier make is the conditional independence. By making such an assumption the number of parameters to be estimated reduced significantly.
The conditional independence is defined as followed, if variable X is conditionally independent of Y given Z, then:

$$\forall i, j, k \quad P(X = x_i | Y = y_i, Z = z_k) = P(X = x_i | Z = z_k)$$

Applying the assumption in Naive Bayes algorithm, given the goal of learning $P(Y|X)$ where $X =< X_1, X_2, \cdots, X_n >$ is a feature vector, X_i is independent of each other of X_ks given Y. More generally, we have

$$P(X_1, \cdots, X_n | Y) = \prod_{i=1}^{n} P(X_i | Y)$$

In our case Y only have two classes, yes (approve to loan) or no (reject to loan). $Y =< y_1, y_2 >=< 0, 1 >$ Now, given the feature vector for an instance (the observation), the expression for the probability that Y will take the kth possible value according to Bayes rule is:

$$P(Y = y_k | X_1, X_2, \cdots, X_n) = \frac{P(Y = y_k) P(X_1, \cdots, X_n | Y = y_k)}{\sum_j P(Y = y_j) P(X_1, \cdots, X_n | Y = y_j)}$$

Applying the conditional independence we obtain:

$$P(Y = y_k | X_1, \cdots, X_n) = \frac{P(Y = y_k) \prod_i (X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}$$

How to use Naive Bayes Classifier to Make Decision
Now assume that we have a new instance $X^{new} =< X_1, \cdots, X_n >$, the decision to classify it into class k is defined as:

$$Y \leftarrow \arg\max_{y_k} \frac{P(Y = y_k) \prod_i (X_i | Y = y_k)}{\sum_j P(Y = y_j) \prod_i P(X_i | Y = y_j)}$$

Which is basically:

$$Y \leftarrow \arg\max_{y_k} P(Y = y_k) \prod_i P(X_i | Y = y_k)$$

The input of NBC
We have encoded the FICO range and Loan Purpose as 1-of-K binary vector (Just trying to be relatively consistent on encoding features for all classifier throughout this assignment). In order not to leave any feature out, we have encoded all other numeric features the same way as 1-of-K vectors. All features except for the loan title has been used. For each of the numeric feature, we first perform min-max normalization based on the training data. That is:

$$normalize\_data = \frac{data - \min(tarining\_data)}{\max(training\_data) - \min(training\_data)}$$

We then classify the normalized data into 10 bins:

$$(-\infty, 0.1), [0.1, 0.2), \cdots, [0.9, \infty)$$

As now each data point will be assigned a bin within its attributes, we can easily convert that into a 1-of-K encoding. The total length of feature vectors for one instances is 111 (36 bits for FICO range, 15 bits for

laon purpose, 10 bits for each of the remaining 6 numerical features: 35 + 16 + 60 = 111) As we have 3481 instances for training, the feature matrix we use for training has the dimensionality of 3481 $\times$ 111 where the row i is the feature for instance i, $i \in \{1, 2, \cdots, 3481\}$.
After the NBC classifier is trained we can simply format the features of testing data in similar way. As we will have 2000 instances for testing we will have 2000 $\times$ 111 instance-feature-matrix for testing. Apply the NBC classifier on testing matrix we will be able to obtain 2000 decisions, each decision will be associated with the instance j, $\quad j \in \{1, 2, \cdots, 2000\}$.

Further discussion, as we have encoded all features into 1-of-K vectors we can use NBC with Bernoulli Model. If we choose to consider the continous numeric values as input feature than we need to model the conditional probability using Gaussian distribution or anything else that may suit the dataset well.

Reference used: **Machine Learning, Tom. Mitchell, 2015 Revision Draft**
`https://www.cs.cmu.edu/ tom/mlbook/NBayesLogReg.pdf`

(ii). Prior is critical for Naive Bayes Classifier.

From lecture notes (lecture 12), we know that assume X_i is a feature, $P(y_k)$ is the prior:

$$P(y_k|X_1, X_2, \cdots, X_n) \propto P(X_1|y_k)P(X_2|y_k)\cdots P(X_n|y_k)P(y_k)$$

This implies that class with small prior will likely to be biased against (for example, if $P(y_0) \approx 0$ and $P(y_1) \approx 1$, then we will very unlikely to choose $y_k = y_0$). Take another quite extreme example, if $P(X_1|y_0) = P(X_1|y_1)$, $P(X_2|y_0) = P(X_2|y_1), \cdots, P(X_n|y_0) = P(X_n|y_1)$, then the decision will solely based on prior $P(y_k)$. In general, the prior of classes if using Naive Bayes Classifier will have a significant impact upon our classification results, especially for cases where: $P(X_i|y_0) \approx P(X_i|y_1)$.

Similarly, if $P(X_i|y_0) = 0$ *and* $P(X_i|y_1) = 1$: similar issue of huge bias could occur. (We have never seen a feature occur in class y_k hence once this feature show up it must not belong to y_k).

To conclude, the class conditional assumption produces skewed probability estimates, and the decision can be biased given the prior is skewed.

Q1.b:
(i). Describe the equations, the inputs, and how we can classify out-of-sample items with the output.
(ii) What happens if $\Phi^T R\Phi$ is singular? What is the reason and how can you solve this issue?
A:

(i). Logistic regression is usually used for two classes, the posterior probability of class $\mathcal{C}_1$ can be written as a logistic sigmoid acting on a linear function of feature vector ϕ:

$$p(\mathcal{C}_1|\phi) = y(\phi) = \sigma(\mathbf{w}^T\phi)$$

where:

$$\sigma(\alpha) = \frac{\exp(\alpha)}{1+\exp(\alpha)}$$

We can use maximum likelihood to determine the parameters of the logistic regression model. To do it, we first use the derivative of the logistic sigmoid function:

$$\frac{d\sigma}{d\alpha} = \sigma(1-\sigma)$$

For data $\{\phi_n, t_n\}$ where $t_n \in \{0,1\}$ is the decision and ϕ_n is the feature vector, the likelihood function can be written as:

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}\{1-y_n\}^{1-t_n}$$

where $\mathbf{t} = (t_1, \cdots, t_N)^T$ and $y_n = p(\mathcal{C}_1|\phi_n)$. The log-likelihood would then be:

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\}$$

where $y_n = \sigma(\alpha_n)$ and $\alpha_n = \mathbf{w}^T\phi_n$.
To start with, we first focus on the linear regression model:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(\mathbf{w}^T\phi_n - t_n)\phi_n = \mathbf{\Phi^T\Phi w} - \mathbf{\Phi^T t}$$

The Hessian matrix can then be defined as:

$$\nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N}\phi_n\phi_n^T = \mathbf{\Phi^T\Phi}$$

The Newton-Raphson update then takes the form of:

$$\mathbf{w^{(new)}} = (\Phi^T\Phi)^{-1}\Phi^T t$$

We will use the above as the initial solution.

We can iteratively update $\mathbf{w}$ based on logistic regression model:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n = \mathbf{\Phi^T}(\mathbf{y}-\mathbf{t})$$

The Hessian matrix can then be defined as:

$$\nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^T = \mathbf{\Phi^T R\Phi}$$

Where:

$$R_{nn} = y_n(1 - y_n)$$

. Finally, using Newton-Raphson we can iterative update the logistic regression model as:

$$\mathbf{w}^{(new)} = (\mathbf{\Phi^T R \Phi})^{-1}\mathbf{\Phi^T R z}$$

where $\mathbf{z}$ is an N-dimensional vector with elements:

$$\mathbf{z} = \mathbf{\Phi w^{old}} - \mathbf{R^{-1}}(\mathbf{y} - \mathbf{t})$$

`Reference used above: Lecture slides and 'Pattern Recognition and Machine Learning' by Christopher M. Bishop`

The Input to Logistic Regression: We will use the same 1-of-K encoding as has done for NBC. That is to say, break all the numeric values into bins and use 1-of-K encoding for all features, that yields the total length 111 for the feature vector.
The following detailed description is just repeating what we have described in previous part: We have encoded the FICO range and Loan Purpose as 1-of-K binary vector. (As we are will be using similar encoding for logistic regression and SVM, we want to have relatively consistent encoding.) To take more features into consideration, we have encoded all other numeric features into the same way as 1-of-K vectors. All features except for the loan title has been used. For each of the numeric feature, we first perform min-max normalization based on the training data. That is:

$$normalize\_data = \frac{data - \min(tarining\_data)}{\max(training\_data) - \min(training\_data)}$$

We then classify the normalized data into 10 bins:

$$(-\infty, 0.1), [0.1, 0.2), \cdots, [0.9, \infty)$$

As now each data point will be assigned a bin within its attributes, we can easily convert that into a 1-of-K encoding. The total length of feature vectors for one instances is 111 (36 bits for FICO range, 15 bits for laon purpose, 10 bits for each of the remaining 6 numerical features: 35 + 16 + 60 = 111) As we have 3481 instances for training, the feature matrix we use for training has the dimensionality of 3481 $\times$ 111 where the row i is the feature for instance i, $i \in \{1, 2, \cdots, 3481\}$.
After the NBC classifier is trained we can simply format the features of testing data in similar way. As we will have 2000 instances for testing we will have 2000 $\times$ 111 instance-feature-matrix for testing. Apply the NBC classifier on testing matrix we will be able to obtain 2000 decisions, each decision will be associated with the instance j, $\quad j \in \{1, 2, \cdots, 2000\}$.

How to classify using logistic regression: Once we have obtained the logistic regression classifier, which is the weight matrix $\mathbf{w}$. The plane that determine if which classes to classify is:

$$w^T\phi = 0$$

This is equivalent to:

$$\sigma(\alpha) = 0.5$$

However by doing this the logistic sigmoid function becomes infinitely steep in feature space and hence might be overfitting the data. Instead, we use a threshold of 0.5 for α. Note that although we can use the threshold of our preference, there is always a possibility we are overfitting the parameters. We will discuss this later.

(ii). If $\Phi^T R \Phi$ is singular, then obviously we would be able to update for the weight matrix. Further more, as R_{nn} is not supposed to be singular (unless $w^T\phi = -\infty$ or $w^T\phi = \infty$, which is unlikely and should not happen). The solution to singularity issue is using pseudo inverse of the $\Phi^T R \Phi$ matrix instead of trying to solve for $(\Phi^T R \Phi)^{-1}$ directly. The issue of singularity should be caused by very similar features of different instances in our training data. For example if there exists $\phi_i = \phi_j$ where $i \neq j$ then likely we will end up with a matrix $\Phi^T R \Phi$ that is singular. We can also use Quasi-Newton optimization technique if $\Phi^T R \Phi$ is singular. For the assignment, we just use pseudo inverse of the matrix as it is not that computational inefficient.

Q1.c:
(i). Describe the equations, the inputs, and how we can classify out-of-sample items with the output.
(ii). What is the advantage of a SVMs over linear regression for classification?
(iii). Which kernel, linear or Gaussian seems to work best with the data? Can you give a short likely explanation of why?
A:

(i). The fundamentals of SVM is perceptron:

$$f(x) = \sum_{i=1}^{m} w_i x_i + b$$

$$y = sign[f(x)]$$

Now, if we try to maximize the margins of classifier, we will arrive at SVM (support vector machine).

$$y(x) = w^T \phi(x) + b$$

where $\phi(x)$ is a non-linear features and $y(x)/||w||$ determines the distance to the hyper plane. The maximum margin solution would then be:

$$\arg\max_{w,b}\{\frac{1}{||w||}\min_n[t_n(w^T\phi(x)+b)]\}$$

To make the computation easier we bring the constraints:

$$t_n(w^T\phi(x_n)+b) \geq 1$$

As:

$$\arg\max ||w||^{-1} = \arg\min ||w||^2$$

We convert the problem into the following form:

$$\arg\min_{w,b} 0.5||w||^2,$$

$$s.t.\ t_n(w^T\phi(x_n)+b) \geq 1$$

Lagrange Duality: The above constrained optimization can be converted into dual problem using Lagrange mutipliers $a_n \geq 0$:

$$L(w,b,a) = \frac{1}{2}||w||^2 - \sum_{n=1}^{N} a_n\{t_n(w^T\phi(x_n)+b)-1\}$$

Setting up the derivative of $L(w,b,a)$ with regard to the w and b to zero we can obtain:

$$w = \sum_{n=1}^{N} a_n t_n \phi(x_n)$$

$$0 = \sum_{n=1}^{N} a_n t_n$$

Then the optimization can then become:

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{n=1}^{N} a_n a_m t_n t_m k(x_n, x_m)$$

$$s.t.\ a_n \geq 0$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

Where the kernel $k(x, x') = \phi(x)^t \phi(x')$
Furthermore, if the data is non-linearly separable, then we should instead use:

$$\tilde{L}(a) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{n=1}^{N} a_n a_m t_n t_m k(x_n, x_m)$$

$$s.t.\ C \geq a_n \geq 0$$

$$\sum_{n=1}^{N} a_n t_n = 0$$

Where C is a penalty for misclassification. If we set $C \to \infty$ then we will be approaching the linearly separable solution.

The input of the SVM

Similarly to the feature matrix we used for logistic regression and NBC. We encoded the FICO range and loan purpose as the 1-of-K vector. That consume 51 bits of binary features. As we can adjust the the kernels of the SVM and can use Gaussian Kernel, instead of using 1-of-K vectorization for the remaining 6 features, we only perform a min-max normalization. Hence the entire length for a feature of an instance would be 57. (51 encoded 1-of-K + 6 numeric scaled features.)

To scale the feature we use:

$$normalize\_data = \frac{data - \min(tarining\_data)}{\max(training\_data) - \min(training\_data)}$$

This applies to both linear kernel and Gaussian kernel that we will test.

How we can classify use SVM

Once the SVM classifier has been trained using training data with dimension of 3481 $\times$ 57 (3481 instances for training), we will apply the SVM classifier on testing data which has the dimension of 2000 $\times$ 57 (2000 instances for testing).

(ii). The linear regression classifier is defined by discriminant weight w:

$$y(x) = w^T x + w_0$$

where the decision boundary is:

$$y = 0$$

To obtain the linear regression classifier, we simply try to minimize the least square error:

$$\sum_{x \in Training\ Classes} (y_c - y(x_c))^2$$

The linear regression classifier can be impacted easily by unusual data points, as the hyperplane does not aim to have the maximized margins to classify different classes.
The objective for SVM, is to maximize the margins of the classifier, hence the SVM is having a great advantage over linear regression classifier. Furthermore, to better accommodate the data we can are able to change the type of kernel that we want to fit. Other than the kernel type, we also have control over the penalty factor C. The features that SVM has simply make SVM much superior to linear regression classifier.

(iii). Based on both prediction and empirical results the Gaussian kernal works better for our data. As in real world, usually non of the features are distributed linearly. Especially for the floating numeric features includes loan amount, etc. As most of the features in our world can be modelled relatively well using Gaussian model, it has been both predicted and verified on empirical results that Gaussian kernel would work better.

Our assumption can also be supported by the average F1 score: SVM with Gaussian Kernel: K = 10, mean(F1) = 0.641, var(F1) = 0.0013
SVM with Gaussian Kernel: K = 5, mean(F1) = 0.634, var(F1) = 0.0010
SVM with Linear Kernel: K = 10, mean(F1) = 0.626, var(F1) = 0.0016
SVM with Linear Kernel: K = 5, mean(F1) = 0.629, var(F1) = 0.0014

Q1.d: Use K-fold validation to estimate the average F1 score of each classifier.
A:
K-Fold Validation: To conduct the K fold validation, we first convert the entire set training data into K subsets. Each time, we leave one subset of the K subsets out as testing data, and combine the remaining K - 1 subsets as the training dataset. The goal is to use each of the subset as the testing data once. Hence, to conduct K-fold validation the training data size will be $\frac{K-1}{K} \times sizeof(Training\ Data)$ and the size of testing data will be $\frac{1}{K} \times sizeof(Training\ Data)$. The above process will be conduced K times in total. If we use K = N - 1, this is actually very tricky. (If K = N then it would be trivial as it becomes a leave one out scenarios). `My intuition on using K = N - 1 would be always leave 2 outs. Each element will be leave out only once and in total we need to validate` $sizeof(Training\ Data)/2$ `times. However, K = N -1 has been removed from this exam.`
We use F1 score to measure how good our classifier perform, the F1 score is defined as:

$$F1 = 2 \times \frac{precision * recall}{precision + recall}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

To obtain the average F1 score, we simply the the average of the K F1 scores using a specific classifier. The F1 scores for each classifier using K = 5 and K = 10 are defined as followed:
SVM with Gaussian Kernel: K = 10, mean(F1) = 0.641, var(F1) = 0.0013
SVM with Gaussian Kernel: K = 5, mean(F1) = 0.634, var(F1) = 0.0010
SVM with Linear Kernel: K = 10, mean(F1) = 0.626, var(F1) = 0.0016
SVM with Linear Kernel: K = 5, mean(F1) = 0.629, var(F1) = 0.0014
Logistic Regression: K = 10, mean(F1) = 0.665, var(F1) = 0.0004
Logistic Regression: K = 5, mean(F1) = 0.666, var(F1) = 0.0002
Naive Bayes Classifier: K = 10, mean(F1) = 0.587, var(F1) = 0.0014
Naive Bayes Classifier: K = 5, mean(F1) = 0.590, var(F1) = 0.0009

The True Positive rate is defined as the sensitivity:

$$Sensitivity = \frac{TP}{(TP + FN)}$$

The False Positive rate is defined as the (1 - specificity)

$$Specificity = \frac{TN}{(TN + FP)}$$

Summing up the AUC regions the final results we obtain is: 0.672092880459. Hence the AUC score is: 0.672092880459

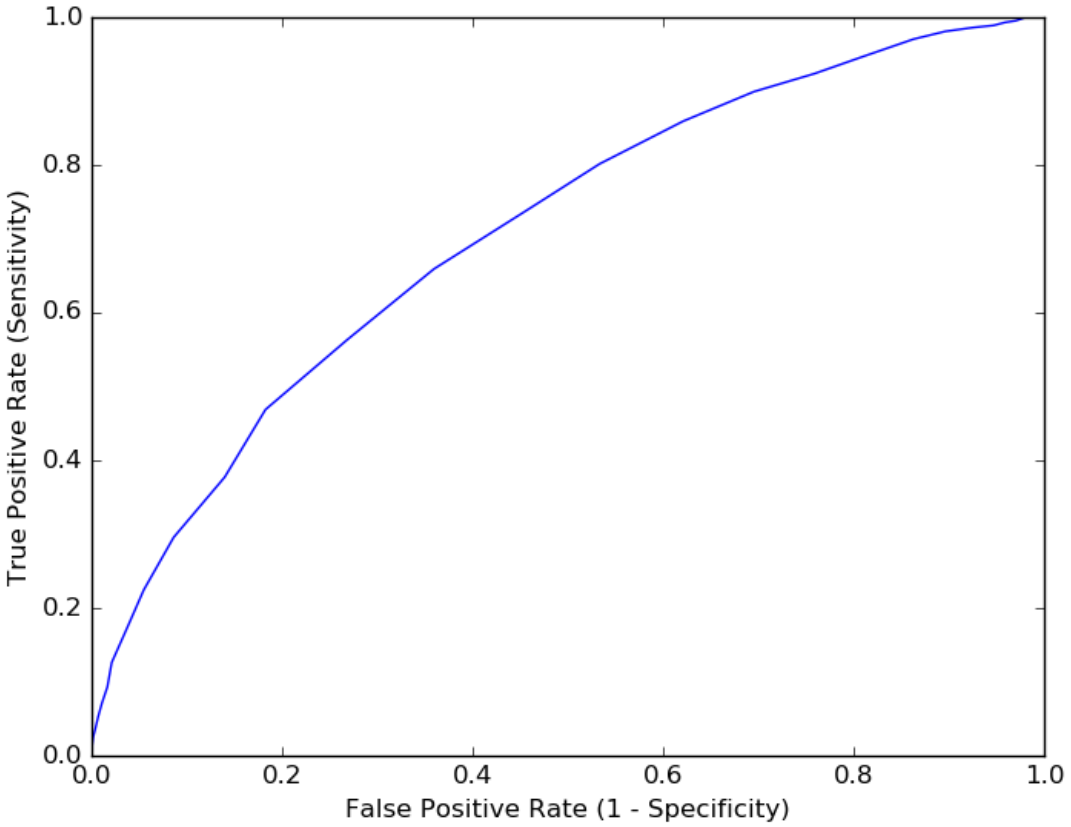

Figure 1: The ROC curve of the logistic regression. Summing up the AUC regions the final results we obtain is: 0.672092880459.

Q1.e: Using 50-fold validation perform a paired t-test.
A:

We set our α level to be 0.05.The t_d is then set to 1.65 accordingly not to reject the Null hypothesis. The t_d value can be obtained by

$$t_d = \frac{(\bar{x}_1 - \bar{x}_2) - d}{SE}$$

Where $d = 0$ in our case and:

$$SE = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$$

Our paired Null Hypothesis are as followed, inferred from the previous section and their corresponding t_d values obtained are:
'Linear SVM better than NBC': $3.37 > 1.65$
'Gaussian SVM better than Linear SVM': $0.609 < 1.65$
'Gaussian SVM better than NBC': $4.0275 > 1.65$
'Logistic Regression better than Gaussian SVM': $2.1188 > 1.65$
'Logistic Regression better than Linear SVM': $2.6973 > 1.65$
'Logistic Regression better than NBC': $6.1824 > 1.65$
Based on the pair-t test of Logistic regression with regard to all other three classifier, based on the F1 score statistics it is indeed significantly better than the other three.

Q1.f: To achieve best results we will use results obtained from SVM with Gaussian kernel. This best models most of real world distribution, and is expected to have a smaller chance to over-fit. (Assuming we are using a reasonable penalty factor). Hence the best solution (is from Gaussian kernel SVM) is actually not from the 'best' classifier (logistic regression) in this case.
Although based on our paired-t test that logistic regression should be the most promising one (using our chosen parameters), Kaggle result indicates the opposite. This might be due to the reason that parameters selected are over-fitting the logistic regression.

Q2.a:
(i). Describe how we can use Shortest Paths, Common Neighbors, Jaccard Similarity, Adamic-Adar score, Katz score, and PageRank to obtain candidates for these missing edges. Can you give advantages and disadvantages of each these methods?
A:
To move forward we first need to construct a graph. We assign each user as a node and if they are linked then we connect the two nodes together using an unweighted edge.

Shortest Path: The fundamental idea behind shortest path is to find the the shortest path between node u and node v. If they have not been connected then we assume there is a missing edge.

Common Neighbours: Common neighbours is focused on how many common neighbours that vertices u and v have. The common neighbour score is defined as:

$$score(u, v) = |N(u) \cap N(v)|$$

Where $N(\cdot)$ is defined as the neighbours of $(\cdot)$. The vertices u and v that are not connected which have the largest score will be assumed to be the missing edges.

Jaccard Similarity: Jaccard Similarity fixes the issue of common neighbours that what if nodes v and u are too active (have too many connections compared to regular nodes). Jaccard score is defined as:

$$score(u, v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

Similarly, the vertices u and v that are not connected which have the largest score will be assumed to be the missing edges.

Adamic-Adar: Adamic-Adar score put more weight to neighbours that are not shared with many other:

$$score(u, v) = \sum_{z \in N(u) \cup N(v)} \frac{1}{\log |N(z)|}$$

Similarly, the vertices u and v that are not connected which have the largest score will be assumed to be the missing edges.

Katz Centrality: The Kats score is based on the binary adjacency matrix A, where for each entry (u, v) will be assigned '1' if (u,v) is connected and '0' otherwise. The Katz centrality score can be defined as:

$$score(u, v) = \sum_{l=1}^{\infty} \alpha^l (A^l)_{u,v}$$

Katz centrality score the exponentially weighted sum of number of paths of length l. Note that α must satisfy that $\rho(\alpha A) < 1$ where $\rho(\alpha A)$ is the spectral radius of αA. $(A^l)_{u,v}$ reflects the total number of l degree connection between nodes i and j. Similarly, the vertices u and v that are not connected which have the largest score will be assumed to be the missing edges.

PageRank: Page Rank is a technique based on modelling the hitting time of a random walk. (A random walk assume that we can travel from one node u to another v following transition probability $P(u, v)$). A stationary probability walker is at v under the following random walk: With probability α, jumps back to u and with probability 1 - α go to a random neighbour of current node.

$$score(u, v) = \pi_v^{(u)}$$

Similarly, the vertices u and v that are not connected which have the largest score will be assumed to be the missing edges.

Reconstructing Adjacency Matrix via SVD Decomposition: We can also obtain the the missing links using the reconstructed adjacency matrix upon Singular Value Decomposition of the original matrix. To do so, we first construct the adjacency matrix as described in Katz Centrality measure. Then decompose the matrix using SVD. There are N principal values after decomposition. During the reconstruction, we only use the K principal values for reconstruction, assigning the rest of $N - K$ principal values to be 0. In the reconstructed matrix, those entries (i,j) that has the largest number assign to them (while i and j is not connected) will be considered as the missing links.

Q2.b:
Discussion of which one I believe is the best and the reason:
Not to use shortest path: The shortest path has the issue of the diameters are usually small and distribution is concentrated. In empirical experiment we have found that there are two many pairs of nodes can be considered as missing links. However we can not rank the probability of which ones are most likely as the shortest path = 2. The worst case scenario is the one node is connected to everyone else, and hence the shortest path between any two nodes will be 2 at maximum.

Not using Common Neighborus: Common neighbours failed to consider those overly-active nodes. For example, if one node is overly active, more likely it will share more neighbours with other nodes.

Not using Jaccard Similarity: Although the Jaccard Similarity measure takes care of the nodes that are too active, each node is still treated equally. This is not case in reality as some silent users only prefer to connect to nodes that they know very well and active users connect to whomever they might know. We should weigh the silent user more (as they are more selective building edges) as oppose to an active user (not so selective building edges).

Using Adamic-Adar: Adamic-Adar gives more weight to neighbours that are not share with many other. We will this Adamic-Adar as it carries the most reasonable explanation of changing the weight on neighbours. It has also been tested to have the best accuracy against other scores in the work:
[1]: LibenNowell D, Kleinberg J. The linkprediction problem for social networks. Journal of the American society for information science and technology. 2007 May 1;58(7):1019-31.

Not using Katz Score nor SVD reconstruction: Although indicated by the previous work[**1**] that using Katz score and SVD reconstruction can yield very similar accuracy as Adamic-Adar score, the empirical computational cost for Katz score is relatively high. If times allows I would like to test on Katz score and SVD reconstruction as to compare against Adamic-Adar.

Not using PageRank: Although PageRank is worth investigating however only limited time is allowed. Furthermore indicated in the work[**1**], rooted PageRank has a lower accuracy even compared to common neighbours, hence we will not investigate PageRank in this assignment.