**Title:**

Citation prediction for papers published in 2013 using ArnetMiner Dataset.

**Group Information:**

***Team Name–*** *Dark Knights*

***Team Id –*** *10*

***Team Members -***

Aniket Ghodke (ghodke.a@husky.neu.edu)

Akshay Sawant ([sawant.ak@husky.neu.edu](mailto:sawant.ak@husky.neu.edu))

Chintan Pathak ([pathak.c@husky.neu.edu](mailto:pathak.c@husky.neu.edu))

Kunal Bohra ([bohra.k@husky.neu.edu](mailto:bohra.k@husky.neu.edu))

**Abstract:**

To reveal information hiding in link space of bibliographic networks, link analysis has been studied from different perspectives in recent years. In this project we are addressing a novel problem namely citation prediction, we are given information about authors, topic, target publication venue as well as time of certain research paper, we are finding and predicting the citation relationship between a query paper and a set of previous papers. Considering the huge size of relevant papers, the loosely connected citation relation distribution, citation prediction is more complex than just prediction the relationship.

**Introduction:**

In this project we address a commonly known problem: Given the information about the Author, Topic, Publication venue, Year and abstract of a paper and some sample data set, we have to generate a model which will predict the reference of a given input based on learning from the sample data set. It is obvious from our intuition that whenever a new paper is written in some domain the authors tends to site the authors, from the same domain (venue in this case), who are highly related to the given paper. But considering the size of the network it has almost become impossible for researcher to read and site all the papers published in the given venue. So we have to develop a system, which will take the current paper as input and will find most relevant paper to this paper.

We can implement topic modeling but considering the size of the network, there can be situation where thousands of paper will share same topic but then we also have to find most relevant amongst these thousands of papers.

We also have to consider a situation where if a researcher has done some research on the venue before then in future when he will write a paper in same domain he will refer to his own work first. Also we have to capture a relationship when an Author has higher citations then chances of his paper to come in references is more than a comparatively new author in the same domain (venue).

In this project we use a two different approaches to solve the problem. This first being Discriminative Term Bucket based approach referred from [1] which captures both document and topic similarity as well as hidden network structures that are sensitive to citation relationship and use this model to predict the references for given input test paper. Given author information, publication venue and abstract and title of the paper our project returns a list of relevant papers, ranked by probability of citation in the query paper.

This first approach includes:

1. We build a discriminative term buckets, which captures document and topic similarity without breaking citation relationships and put the papers in different term buckets, this step helps in reducing the search space.
2. We set up a meta-path base feature space to interpret hidden information and define citation prediction with meta-path features.

The second and our main approach is based on Topic and Document Similarity combined with User Guided Search on training data. It’s a two-step approach:

1. First after data preprocessing, for every test paper we find the most similar papers from the training data based on topic and document similarity. Every test paper is assigned a small paper bucket which is supposed to capture the possible citation relations for it.
2. Then for each test paper and its corresponding bucket pair, we define 3 different ranking functions which rank the papers in the buckets according to most probable citation relations. We use these functions to assign possible citation predictions to a given test paper and repeat the process for complete testing data set.

**Problem Statement**

Goal is to predict top 10 references for a given paper published in 2013 using information such as abstract, authors, venue and title of paper.

**Problem Formalization**

Given – ArnetMiner Publication’s dataset of papers published. For each paper in the training dataset, index, title, authors, year, publication venue, abstract and id of references of this paper is provided. For each data set in testing dataset we are given the same format but not references i.e. index, title, authors, year, and publication venue, abstract. The papers in testing dataset will be of the ones published in 2013. We have to predict the top 10 references for those papers.

**Data Description:**

The data in training dataset is taken from ArnetMiner, the data is in following format:

#index ---- index id of this paper

#\* ---- paper title

#@ ---- authors (separated by semicolons)

#t ---- year

#c ---- publication venue

#% ---- the id of references of this paper (there are multiple lines, with each indicating a reference)

#! ---- abstract

**Example dataset:**

#index 1083734

#\* ArnetMiner: extraction and mining of academic social networks

#@ Jie Tang;Jing Zhang;Limin Yao;Juanzi Li;Li Zhang;Zhong Su

#t 2008

#c Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining

#% 197394

#% 220708

#% 1250184

#! This paper addresses several key issues in the ArnetMiner system, which aims at extracting and mining academic social networks. Specifically, the system focuses on: …

Testing dataset will not include the references. i.e. “#%” delimiter and contents corresponding to it.

**Data Preprocessing:**

1. Before beginning the analysis of the data, we have preprocessed the dataset as follows –
2. Data Cleaning – We have converted all data to lowercase to maintain consistency.
3. We have also removed unwanted special character, converted to UTF format, performed sanity checks on data (such as year should be 4 digit numerical).
4. We have also applied Stop wording to remove words, which do not provide specific information corresponding to paper (e.g. and, to, from, the).
5. We have also removed special characters form the data. Like (, and. Etc.…)
6. Data Integration – We have applied stemming algorithm (Porter2) to convert different forms of a word to standard root word.
7. We have changed the data from the paragraph format to matrix format with Column1: Index, Column2: Authors (these are “,” delimited), Column3: Year, Column4: Title, Column5: References (these are “,” delimited), Column6: Abstract.
8. We have also made sure the Authors, Title, References, and abstract can be null but not index.
9. We have also removed the words that occur frequently like proceedings, data, article, years, etc. it in almost all the titles so it would not add to the value of citation relationship and unnecessary increase bucket size.

**Methods description:**

We tried several methods and devised algorithms to solve the given problem. After testing them and comprehensive comparison, we devised two algorithms and approaches to solve the given problem. They are described in detail below:

**Approach: Meta-path and Discriminative Term Bucket based Prediction System**

We now formally describe our approach to solve the problem using Data mining algorithms. After careful analysis of problem statement, we did a thorough research and came to a conclusion that our approach to solve the problem would involve a multi-step approach and algorithms, not just one algorithm. Our approach is largely influenced by work done by J. Han et al. [1].

We kicked off the project with all data pre-processing which involved all the steps described in previous section. Since we interpret the problem and data as a heterogeneous information network, we first converted the training data set given to us in the form of a heterogeneous information network using Paper’s index # as a primary key. From the definition of heterogeneous information network given in [1] we treated author names, titles, venues and abstract as primary values for keys where keys were paper’s index #. Several maps were constructed were the key values were paper’s index # and values were abstract, title, author names and venues.

After molding the data into the desired form, we came to conclusion that we would design the solution as a query prediction system where given an input test paper, out system would treat the input test paper as a query and generate a list of papers from training dataset ranked in order of probability that will be cited the by the query paper. To achieve this task, we referred Han’s et al. work from [1].

Looking at the massive size of given training data set our first task was to reduce the search space for the query processing system. To achieve this, we first created discriminative term buckets on the training data set.

Discriminative Term Buckets: It’s a special data structure from [1] where we club all the papers together that are of similar topics and also have high document similarity along with citation relationship. It is to be noted that one paper can belong to multiple term buckets given the inter-disciplinary nature of papers and citation probability.

After constructing term buckets, we now confine our search space to only those group of papers which have same topic similarity as of test/query paper. We do a classic classification algorithm – Binomial Logistic Regression on these term buckets to do a prediction of possible papers that can be cited by the query paper. Our detailed design and flow is explained in next section.

**Approach: Topic and Document Similarity based User Guided Prediction System**

In parallel to discriminative term bucket and meta-paths based prediction system defined above, we devised our own algorithm to solve the given problem, which is based on predicting results for a given test query paper using User Guided Prediction System and concept of document similarity. In this approach, we started with same pre-processed data as we did in our previous approach. For a given test query paper, we took its title and tried matching it with titles of papers in training data. This idea originated from the fact that for a given test query paper, if we are looking for papers that are probable to be cited by it, then these ‘probable, to be cited’ papers must have some sort of document similarity as well as topic similarity with the test query paper. Thus, to do this we first retrieved top ranked papers (ranking function and design is described below) from the training data for a given test paper. Similar to term buckets data structure we discussed earlier, we form a cluster of 500 documents for a given test query papers. Stripping top 500 ranked documents from the training data forms this cluster. Then, we formed some User Guided search functions for the test paper to use and find the most probable ‘to be cited’ papers from this particular paper bucket. We repeat this process for all test query papers from 2013.

The complete details and design for this method is discussed below.

**Design:**

**Approach: Meta-path and Discriminative Term Bucket based Prediction System.** The design for this first solution is multi-step and is divided in following stages:

Stage 1:

Goal: Reduce the search space

Solution: Create term buckets from the training data set such that for a given query paper, we are only considered with term buckets in which the paper falls in.

To create term buckets, we process all the venues in the given complete training data set and the prune all the high and low frequency words from these venues and form a bag of words from these venue titles which later we use to create the inverted index. Then, we create following data structures:

1. We create an inverted index in all the words on venues from training data set papers that fall in between the frequency range tuned by us (explained in next section). The keys for inverted index map are words in frequency range and value for each key is set of papers that have this ‘seed words’. We call this data structure as train\_inverted\_index.
2. Create another map with keys and paper’s index number and value for each key as a set of papers (papers here are from training as well as test data) that this paper defined by key refers (has citation relations already defined). We call this data structure as References Map.
3. After creating above maps, we use them and create another data structure called Citation Discriminative Measure Map or cdm\_map. This map has keys as terms from train\_inverted\_index (same keys) and values are floating values that are numerical measure for the terms. This numerical measure is called Citation Discriminative Measure [1]. It tells us how much power/ability a certain term has. If this term occurs in a given paper, that paper is likely to be cited by a query papers. Clearly a term like ‘machine learning’ has more CDM than term like ‘observations’. To get CDM, of a given term, we find all the papers from the training data set that has that term. We call this list Pt. Now we create a matrix Pt X Pt and mark all the PiPj cell 1 that defines actual citation relation (i.e. Pi refers Pj) and rest all cells as 0.

Then using the formula : CDM (Ti) = count (+ 1) +1/ count(-1) +1, we find cdm values for all terms. Terms with CDM value greater than 0 are only used and rest terms are dropped.

Thus, we have cdm\_map data structure with keys as keywords from training data set venues and value for each key as a numerical measure call CDM.

1. Now using cdm\_map data structure, we create a new data structure called initial\_term\_bucket\_map. In this data structure we have keys as terms from training data we have been using so far and papers from training **as well as test data.** To introduce the test papers, we collect papers from test data, preprocess them exactly like training data papers. Then we assign test papers to each term if that term occurs in the given test paper. Similary, we add papers from train data as well. Thus, iterating over cdm\_map we create a new map with same length (keys) as of cdm\_map but now values as set of papers from test and train data.
2. Finally to create term\_bucket\_map, we iterate over initial\_term\_bucket\_map and expand the keys first. We do this create term buckets with multiple terms (keywords to be precise) These keywords helps us to capture the papers from test data as well and eliminates the possibility that a given test paper can be related to a given term bucket’s papers but might not fall in it for not having the keywords from train data. Thus, we expand keys using a popluar measure calld Normalized Mutual Information or NMI. This is the most critical measure in our approach for creating term buckets and reducing the search space. NMI for two given terms gives us an idea that if a term Ti has information that is critical for citation, then another Tj which occurs frequently with Ti also implies and carries that citation information. Thus, we define NMI as :

∀t ∈ D, NMI (ti, tj) = [Pr(ti, tj)]\*log[(Pr(ti, tj)/Pr(ti)\*Pr(tj)]

Where, D is bag of words for testing as well training data venues (vocabulary of keywords that fall in the frequency range defined in coming section). Pr (ti, tj) is probability of terms ti and tj occurring in a single paper which is defined as:

Pr(ti, tj) = Number of papers (test +train) in which ti and tj occurs togethter / Total number of papers (test + train)

Pr(ti) = Number of papers (test + train) in which ti occurs / Total number of papers (test + train)

Pr(tj) = Number of papers (test + train) in which tj occurs / Total number of papers (test + train)

After expanding the keys/ terms our term\_bucket\_map has keys as string of words that have strong effect on citation relationship. Now in second step, we assign values to these keys. Values here are set paper for each key. This set of papers are papers that have one or more keywords from the keys of of term\_bucket\_map.

Stage 2:

Goal: Creating Numerical Measures for Learning Model

Solution: For each query paper, we learn a model using Logistic Regeression as mentioned above. But

to train and create the model, we need numerical feauteres for each paper Pi in the search space

(one or more term buckets). This for all given a test query paper Pt and a paper Pi in the term buckets

we define certain meta paths based feature space defined in [1] We take following meta paths:

P - A - P: For a given pair of two papers, see if they have common authors.

P - T - P: For a given pair of two papers, see if they have common terms in their abstract.

P - C - P: For a given pair of two papers, see if they have common citation relations

Apart from above meta-paths, we define additional purely numerical features in the feature space

namely, H-index of the papers in the term buckets and citation count of papers in term buckets as well.

Clearly, if a paper’s author has higher H-index and this paper has high citation count then, this paper

is likely to be cited by a test query paper.

We call these 4 + 2 = 6 feature as our feature space. After defining the feature space, we define the

measures. From [1] we define, following measures:

Count and PathSim [2]

Thus, for a pair Pt and Pj, where Pt is the test query paper and Pj being one of the papers from term

bucket papers, we have 6\*2 = 12 features/numerical attributes.

To sum up, we have F = P X M

Where, F is our meta-path based feature space, P is our meta-paths (6 defined above) and M as

Measures (2 defined above). F is Cartesian product of P and M.

Stage 3:

Goal: Create and learn the citation prediction model

Solution: In order to create and learn the citation prediction model, we first define how we process a

test query paper Pt.

For a given test paper Pt, we find all the buckets in which the paper Pt falls. We call this set of buckets

B (Pt). Then we find all the papers in this bucket sets and call the set of papers as B\*(Pt). This is our

Final search space for the query paper Pt. We classify all the papers in B\*(Pt) as positive and negative

based on existing citation relationships between them using references we have from training data set.

Now, we have 12 numerical attributes and a class label for each paper pair (One train paper with a

given test or query paper). We do a L2 regularized logistic regression on the paper set and find a

citation score for each paper pair Pt, Pj where Pt is test query paper and Pj is the paper from B\*(Pt).

We find all citation probability for possible pairs formed by combining Pt with all papers from B\*(Pt).

Calculating citation probability = Pr(label = 1|Pt, Pj; θ) = (e^z)/(e^z + 1)

Where z = Σfi∈F ′θi· fi. Pr(label = 1| Pt, Pj; θ) is the probability that paper Pt cites paper Pj and label

being 1 denotes positive class label that there might be citation relationship already hidden between Pt

Pj.

After deriving citation probability for all papers, we finally define final citation scores for each paper:

Citation Score = log (1 + cbn (Pt), Pj)) · Pr(label = 1|Pt, Pj; θˆ)

Where cbn denotes the number of term buckets shared by Pt and Pj and θˆ is weight/coefficient we get

After performing logistic regression on training data for Pt (papers from B\*(Pt) with class labels as 0

or 1 and numerical attributes defined above).

Since we are interested in finding top 10 papers that are probable to be cited by the query test paper Pt,

We sort the list of all scores for pairs in Pt, Pj and take top 10 Pj from it as our answer to the query

paper.

**Approach: Topic and Document Similarity based User Guided Prediction System**

In this second approach, we aimed at solving the problem in a very minimalistic and small algorithm

Based on User Guided Search among a smaller search space for a given test paper. The major steps of this approach are described below:

1. Our first task was to narrow down the search space for a given test paper. For a given test paper we first took its topic/title and formed a bag of words from it. Similarly, for the complete training data set, we formed a map of keys as paper indexes and values as set of words from topic/titles of each paper/key. We call this data structure as paper-title-map.
2. For every test paper, we matched its words from the topic with every paper from the training data and formed a cluster of 500 papers whose title matches most with the test paper’s topic. The ranking function for most match between a pair of given test paper and training paper is simply the cardinality of set of words formed by intersection of sets of words from topic of test and training paper respectively.
3. Since the search is very small, one can be suspicious that these clusters actually do contain the probable papers that can be cited by the test paper. To validate this, we formed our test paper set of around 5000 papers from training data set and took all papers from 2012 and treated the rest of training data with papers till 2011 as sample training data.
4. We formed approx. 500 papers sized clusters for every 2012 test papers and since we already have actual citation relation for them, we analyzed how much actual citation relationships are captured by these paper clusters and approx. more than half of the buckets (4000 out of 5000) captured more than 6 actual citation relation for a given 2012 query paper.
5. With this very limited search space, it turned out to be a good ratio. At this stage after validating the term buckets, we incorporated three different User Guided Functions to guide the search algorithm and predict possible citations for the query paper.
6. Our three functions are as follows:

Topic and document similarity based ranking: In this function, we generate at most half of the total predictions for a given test query paper. The heuristic involved is that, if a given paper is similar to some other paper in topic as well as content, it is likely that they are addressing the same

research problem and the new paper is likely to cite the old paper. Considering this, we take this function of highest priority and try to provide maximum predictions using this function. If we are still short of require 10 predictions, then we take the next function to generate predictions.

Author collaboration based ranking: In this function, we first take the author/s names of a given test paper Pt and fetch its 500 paper sized cluster. In this cluster we find all those papers whose authors have collaborated with the author of test paper in past. To do this, we form an author-author collaboration map data structure from the training data set. If found, we fetch all papers that hold this relationship and mark them as subset of possible citations. We do this since heuristic says that authors who have collaborated in past are more likely to cite each other’s papers in future. It is second highest weighted function. We aim to provide maximum possible predictions from this function.

Citation count based ranking: In this ranking function, we take all the papers from the bucket for a given test query paper and sort them in decreasing order of their citation count. The heuristic involved here is that high citation count of a paper denotes highly important and revered paper in a given topic. So a test query paper is likely to cite the paper with same topic and high citation count. This ranking function is given the least priority.

1. After getting papers from first function, we check if they are less than 10. If so, we use second ranking function and find more papers to complete the result set. If not, then finally we call third ranking function and form a 10 paper set as result for a given test query paper.
2. We repeat this process for all test query papers, find the final solution to the given problem.

**Evaluation:**

**Approach: Meta-path and Discriminative Term Bucket based Prediction System**

We now present an evaluation of our Query Processing System using Meta Path based features from [1] and Discriminative term buckets concept from [1].

Evaluation of NMI:

Normalized Mutual Information is one of the most important parameter we have in our design. For a low NMI value, we can get a very large term bucket and our goal of creating term buckets to reduce search space would hold no value. On the other hand, if we keep NMI threshold as high, our bucket size can be too small which can result in loss of possible citation papers. Thus after much analysis for various values of NMI show below, we set NMI as 0.003. To achieve this value, we took all the papers from 2012 as our new test data and papers up to 2011 as our training data. Then for each test paper from 2012, we found the optimal bucket size that captures most citation relations which we already have as ground truth labels and found that for NMI value near 0.003, the bucket size is such that it captures most of the citation relations.

Evaluation of Frequency range:

We created term buckets initially without defining any frequency range for seed terms and found that almost every bucket was flooded or spammed with papers which have terms like ‘proceedings’, ‘data’ , ‘process’ etc. in them. We removed these high frequency words and also set a lower frequency threshold for words that occur very rarely in the whole training data set venues. We came to a optimal range of frequency value of 70 – 100000. By setting this value our bucket sizes reduced to 60% original value and capturing same amount of citation relations from test set of 2012, clearly indicating that papers that were removed due to high or low frequency words had no importance in the term buckets.

Evaluation of Results:

For our own test data for papers from 2012, we stripped 3500 papers that 8 or more references already defined. We kept our training data with papers only till 2011 from the original data set. Then we built our query processing system exactly as defined above. Following are the major observations:

1. The total number of seed terms which we get from taking venues of all papers is around 50000. After creating the CDM Map, we were left with 4000 approx. terms. The term buckets had 4000 keys/strings of keywords formed from seed terms and then expanding them as described above.
2. Out of above 4000 buckets, average size of each bucket was initially 2,50,000. Then we pruned all those papers who were never cited before and whose authors had a very low H index. We took these two heuristics because if a paper has never been cited before, it is very unlikely to be a part of 10 paper result set which we pose for a query. Similarly, a lower H index author is very unlikely to be cited by a new paper. By doing this, the average size of buckets in term bucket map reduced to 1, 25,000 to 1, 50,000.
3. Now, 4000 buckets with each bucket size being approx. 1, 50, 000 is still huge to search top 10 papers though we did a remarkable job by narrowing search space from 1.9 million to 0.15 million. That is we pruned 92% of search space for a given test query paper. To validate our term buckets, we ran our validation scripts which takes papers from our own 2012 test paper set and for each paper since we already have citations, we check how many of the citations for a given 2012 paper falls in these 4000 buckets. Repeating this process for every test paper from 2012, we found that, all buckets were capturing more than 80% of original citation relations. This clearly indicated that out term buckets were capturing high proportion of expected citation relations.
4. For a given test paper query from original test data given to us, we ran the query processing system. We used Logistic Regression module from Scipy Python library. Average turnaround time for creating term buckets was 350 seconds. Average time for finding top 10 citation results for a test paper was 25 seconds. To overcome timing constraint, we divided our test data of 6205 papers into 6 parts and each part of 1000 test papers were ran in parallel to compute the final results which turned out to be an effective as we processed all queries efficiently and less time.

**Approach: Topic and Document Similarity based User Guided Prediction System**

In this approach, our algorithm mainly relied on following numerical parameters:

1. Threshold for cluster size for each test paper: As already mentioned above, we form our own test data set of papers from 2012 and training data set of papers till 2011. Then we ran a validation script to check how many clusters out of 5000 (one approx. 500 sized cluster for each test paper of 2012) captures more than 50% of actual citation relations. Approx. 4000 out of 5000 clusters could capture more than 50% such actual citation relations. Getting more than 50% of actual citation relation in a cluster of size 500 out of 1.9 million papers is a very good indication that our clusters were formed with papers that were similar to topic, problem addressed and similarity with the test paper.
2. Threshold for citation count in user guided ranking function: For every cluster and test paper, we sort the bucket papers in decreasing order of citation count using our Citation map data structure and take the top cited papers in the bucket as a part of the predictions.2
3. The search space for prediction (bucket size) was really small as compared to our previous approach which greatly reduced the running time of the algorithm. We generated the 6204 buckets, one for each test paper in around 150 minutes with rate being 1 bucket per 3 seconds.
4. The running time for algorithm after creating buckets turned out to be 15 seconds which is way faster than our previous approach. The reason here is that our buckets of possible citation papers is very small compared to previous approach but with this trade off we lose precision to some extent. But looking at the practical nature of project and large test dataset, we choose this approach as our main approach for the course project.

**Discussion:**

In lieu of our two approaches discussed above to solve the problem for predicting top references for papers in a bibliographic network, we observed following:

1. In addition to coming up with various data mining approaches for prediction, this problem consisted of combining various algorithms and not just one to propose the solution of the problem.
2. Topic modelling and clustering similar papers from training data was the first step for both of our approaches. Both approaches differs drastically in the way of how we form similar paper clusters.

The second approach being simple than the first one but more effective in running time complexity.

1. For prediction, various algorithms were tried like Logistic Regression, Personalized Page Rank and User Guided Search with Topic and Document Similarity.
2. From aforementioned three algorithms, User Guided Search was the fastest one due smallest search space, then logistic regression and then personalized page rank. We finally included meta path based logistic regression approach and User Guided Search as our final choice for solving the given problem.
3. The most critical step in both of our approaches were deciding the size and formation of term/paper buckets. In the first discriminative term bucket based approach, almost all buckets could capture more than 90% of citation relations on 2012 test data but took longer to process the whole output. Whereas, in our second approach we form very small buckets of paper to reduce our prediction time but it captures a bit less citation relation on the same 2012 data.
4. Clearly there is time and precision trade off involved in both approaches. In interest of time, we decided to move on with second approach: Topic and Document Similarity based User Guided Prediction System for predicting citation relations.

**Conclusion:**

After implementing various Data Mining Algorithms like Topic Modelling, P-PageRank, Logistic Regression and User Guided Search, we finalized our main approach as Topic and Document Similarity based User Guided Prediction System. We decided to take this approach for one main reason: Running time complexity of algorithm. Our system has following highlights:

|  |  |
| --- | --- |
| Running Time (sec.) | Function |
| 350 | Data Preprocessing |
| 19000 | Generating Term/Paper buckets for a set of test papers using Topic Similarity between paper pairs |
| 20 | For generating 10 prediction for 6205 test queries. |

Clearly this running time is very efficient in comparison to Discriminative Term bucket based approach which had more precision but very high computational complexity with 25 seconds to generate 10 citations for 1 test paper. Here in the same time, we generate all predictions.

**References:**

We have referred to the following papers:

1. Citation prediction in Heterogeneous Bibliographic Network – Xiao Yu, Quanquan Gu, Mianwei Zhou and Jiawei Han

**Task Distribution:**

|  |  |
| --- | --- |
| Data Cleaning | Aniket, Chintan, Kunal and Akshay |
| Designing the Approach 1 | Kunal and Akshay |
| Designing the Approach 2 | Chintan and Aniket |
| Implementation of Approach 1 | Kunal,Akshay and Aniket |
| Implementation of Approach 2 | Chintan and Aniket |
| Report | Kunal,Akshay and Chintan |