Study on **“Data Structures for Machine Learning”**

K. SAI BHARGAV 20071D5807

V. RAJENDRA 20071D5811

VALLURUPALLY NAGESWARA RAO VIGNANA JYOTHI

INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE

(M. TECH)

1. ABSTRACT

Data structure is the organization of data into our memory and efficient retrieval of data from memory. Machine Learning is the process of making machine to learn and perform on its own with the given inputs and outputs. So both are different streams of computer field. So this paper brings a relation between data structures and machine learning algorithms by taking an algorithm from data structures and machine learning. In this paper we are going to discuss about two algorithms which are used in information retrieval and they are K-Nearest Neighbor(KNN) and K Dimensional - Tree (KD-Tree).**KD-Tree (K Dimensional – Tree)** is a data structure for partitioning data points into a k-dimensional space. KD-Trees are widely used in several applications like range searches and machine learning algorithm like nearest neighbor searches and creating point clouds. KD-Trees are special case of binary search trees invented by Jon Louis Bentley in the year 1975. It is an improvement over KNN. It is useful for representing data in an efficient manner. In KD-Tree the data points are organized and partitioned on the basis of some specific rules and conditions. **KNN (K-Nearest Neighbor)** is a supervised machine learning classification algorithm which gives information regarding what group something belongs to, for example, type of tumor, the favorite sport of a person etc. This paper focusses on the parallel implementation of both the KD-Tree and KNN algorithms for information retrieval & inverted index. However this implementation is inefficient for large datasets. Construction of a nearest neighbor graph is an important task in machine learning applications. However, it is computationally expensive, especially in constructing such graphs for huge amount of datasets and when the data is of high dimensional. Python’s open source machine learning library Scikit-learn uses k-d trees for implementation of knn algorithm. Here we proposed an **Inverted index** which is a counter for curse of dimensionality in k-d trees from data structures.

2. INTRODUCTION

KD-Tree is a data structure useful when organizing data by several criteria all at once.

KD-Tree data structure is much like the regular binary search tree that you are familiar with. Except each node would typically hold not just 1 value, but a list of values. When traversing a KD-Tree, we cycle through the index to the values list, based on the depth of a particular node.

The **"K"**in the name **"KD-Tree"**refers to the number of properties that each data point has. In our example, since it's a 2 dimensional plane, we have exactly 2 properties – the X coordinate and the Y coordinate. So the root node starts off arbitrarily picking one of those. Let's suppose it's the X coordinate. Then its children, the second level, alternate to using the Y coordinate. The grandchildren nodes cycle back to using the X coordinate, and so on as shown in the below two figure.

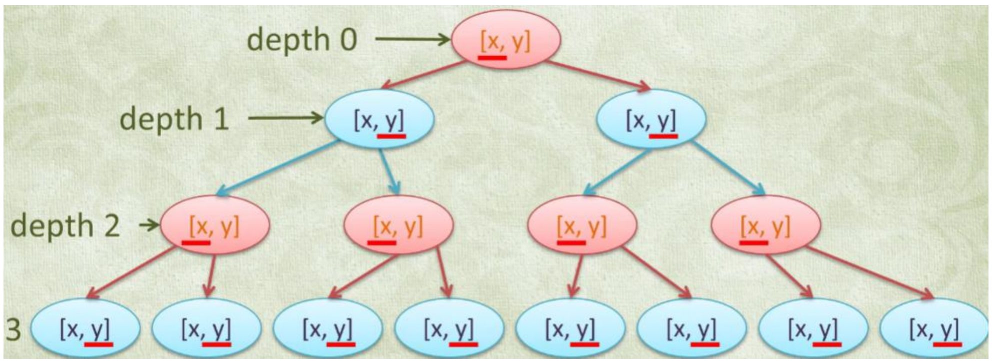
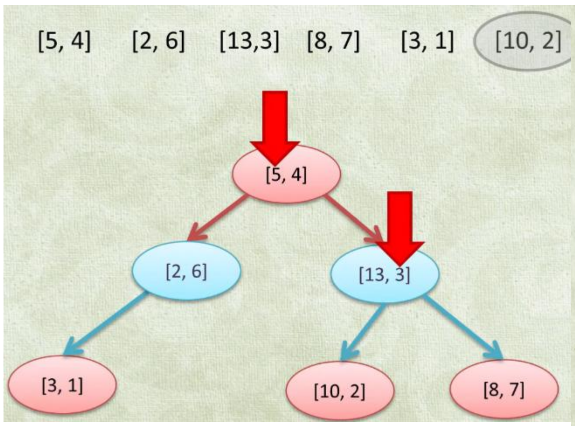


Fig: 2.1 KD-Tree Structure

When inserting a new node, just like in the typical binary search tree, we compare the node's value and go left if it's smaller, right if it's bigger. Consider the example where we need to insert set of values [5,4] [2,6] [13,3] [8,7] [3,1] [10,2] and the KD-Tree would be as shown in the below figure.

Visually you can think of the root node dividing the XY plane into left and right sides, since we decided to start off using the X coordinate. This is shown in the below two figures where the nodes are visually plotted on the graph consisting of X coordinate and Y coordinate and are plotted accordingly as shown in the below two figures.



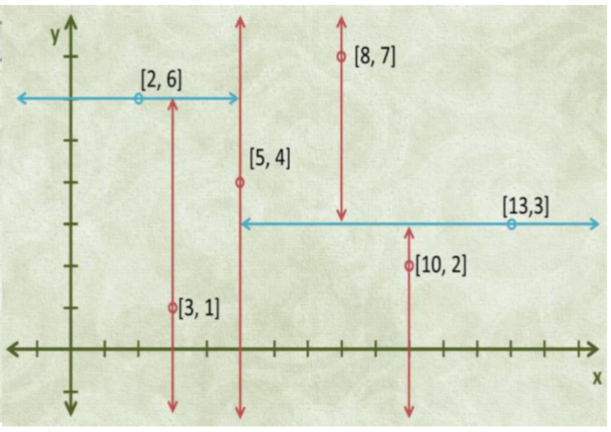


Fig: 2.2 KD-Tree for given points **Fig: 2.3 Data points visualization**

Suppose our target point is [9,4] and our goal is to find its nearest neighbor i.e. **KNN**then

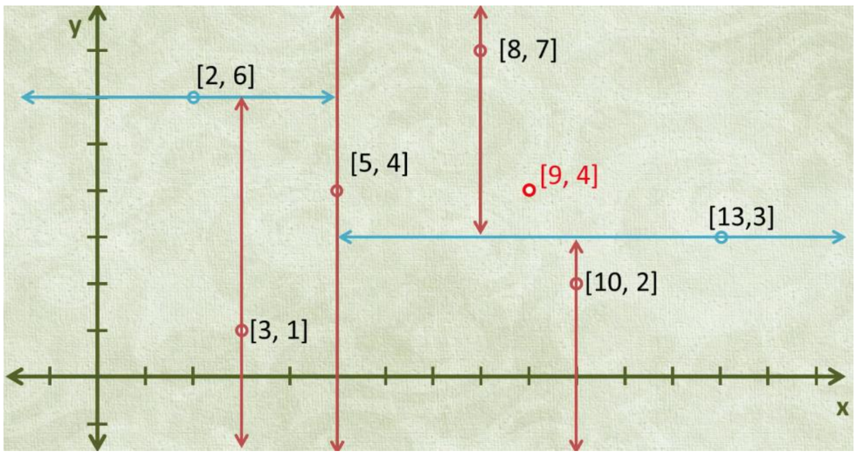


Fig: 2.4 Target point finding in 2 dimensional space

we walk down the tree and keep calculating the shortest distance to the target point from each of the point and we group the target point into any one of data set points.

KD-Trees fails to handle high dimensional data, i.e. with n number of data points and n dimensional space it can’t give accurate results and may miss nearest neighbors. So to overcome such situations we are discussing a new algorithm inverted index which do not miss any nearest neighbor and to increase KNN efficiency.

3. LITERATURE SURVEY

This work is based on paper [1], which gives an overview about the construction of a nearest neighbor which is computationally expensive and especially when the data is of high dimensionality. Where parallel implementation of nearest neighbor graph construction using kd-trees and ball trees, with parallelism provided by OpenMP and the Galois framework.

Paper [2] provides an information about Best Bin Search or BBF Search is an approximate algorithm which finds the nearest neighbor for a large fraction of queries and a very close neighbor in the remaining cases. This is also involved in a fully developed recognition system, which is able to detect complex objects in real, cluttered scenes in just a few seconds.

From paper [3] multidimensional binary search trees are developed (kd-tree) as a data structure for storage of information to be retrieved by associative searches. In this paper kd-trees are defined and examples are given.

In paper [4] an algorithm and data structures are presented for searching a file containing N record, each described by k real valued keys, for the m closest matches or nearest neighbors to a given query record. The computation required to organize the file is proportional to kN log N.

Paper [5] describes about the Bag Of Visual Words (BoVW) features that quantize and count local gradient distributions in image similar to counting words in texts have proven to be powerful image representations. Naive Bayes, Logistic Regression, k-nearest neighbors, Random Forests, AdaBoost and linear Support Vector Machines (SVM) as well as generalized Gaussian kernel SVMs are applied in BoVW for image classifications.

Paper [6] describes the design and implementation of a tool called ParaMeter that produces parallelism profiles for irregular programs.

In addition paper [7] finds similar patches in images, by treating each image patch as a point in a high dimensional space, we can use nearest neighbors algorithm to compute the exact same results in a fraction of the time.

Paper [8] is about non-approximate acceleration of high-dimensional nonparametric operations such as k nearest neighbor classifiers. This paper concentrates on pure k-NN classification.They introduce new ball-tree algorithms that on real-world data sets give accelerations from 2-fold to 100-fold compared against highly optimized traditional ball-tree-based k-NN . These results include data sets with up to 106 dimensions and 105 records, and demonstrate non-trivial speed-ups while giving exact answers.

Paper [9] is about metric data structures in high-dimensional or non-Euclidean space that permit cached sufficient statistics accelerations of learning algorithms. They used the anchors hierarchy-a fast data structure and algorithm for localizing data based only on a triangle-inequality-obeying distance metric and show how this, in its own right, gives a fast and effective clustering of data. But more importantly this paper show how it can produce a well-balanced structure similar to a Ball-Tree.

Paper [10] focusses on a system which will take any given dataset and desired degree of precision and use these to automatically determine the best algorithm and parameter values. It also describes a new algorithm that applies priority search on hierarchical k-means trees, which have been found to provide the best known performance on many datasets. After testing a range of alternatives, they found that multiple randomized k-d trees provide the best performance for other datasets.

Paper [11] argue that existing DSLs can be implemented on top of a general-purpose infrastructure that (i) supports very fine-grain tasks, (ii) implements autonomous, speculative execution of these tasks, and (iii) allows application-specific control of task scheduling policies. To support this claim, this paper describe such an implementation called the Galois system.

Paper [12] describes local spatial error dependence, one can construct sparse spatial weight matrices. As an illustration of the power of such sparse structures, they computed a simultaneous autoregression using 20 640 observations in under 19 min despite needing to compute a 20 640 by 20640 determinant 10 times.

Paper [13] provides an information about Scikit-learn which is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. Emphasis is put on ease of use, performance, documentation, and API consistency.

Paper [14]this paper, argue that the dependence graph is not a suitable abstraction for algorithms in new application areas like machine learning and network analysis in which the key data structures are “irregular” data structures like graphs, trees, and sets. To address the need for better abstractions, this paper introduced a datacentric formulation of algorithms called the operator formulation in which an algorithm is expressed in terms of its action on data structures.

Paper [15] In this work they efficiently distribute the computation of knn graphs for clusters of processors with message passing. Extensions to our distributed framework include the computation of graphs based on other proximity queries, such as approximate knn or range queries. These experiments show nearly linear speedup with over 100 processors and indicate that similar speedup can be obtained with several hundred processors.

Paper [16] focusses on improving the KD-tree for a specific usage: indexing a large number of SIFT and other types of image descriptors. It have extended priority search, to priority search among multiple trees. By creating multiple KD-trees from the same data set and simultaneously searching among these trees, it have improved the KD-tree's search performance significantly.

Paper [17] This note presents a simplification and generalization of an algorithm for searching k-dimensional trees for nearest neighbors reported by Friedman et al. I-3]. If the distance between records is measured using Lz, the Euclidean norm, the data structure used by the algorithm to determine the bounds of the search space can be simplified to a single number. When a k-dimensional tree is built, this plane can be found from the principal eigenvector of the covariance matrix of the records to be partitioned. These techniques and others yield variants of k-dimensional trees customized for specific applications.

Paper [18] Here, Dynalign, a program for predicting secondary structures common to two RNA sequences on the basis of minimizing folding free energy change, is utilized as a computational ncRNA detection tool. The Dynalign-computed optimal total free energy change, which scores the structural alignment and the free energy change of folding into a common structure for two RNA sequences, is shown to be an effective measure for distinguishing ncRNA from randomized sequences.

In addition paper [19]offer a quantitative roadmap for improving the performance of all frameworks ((GraphLab, CombBLAS, Giraph, SociaLite and Galois among others) and bridging the "ninja gap". We first present hand-optimized baselines that get performance close to hardware limits and higher than any published performance figure for these graph algorithms. It characterize the performance of both this native implementation as well as popular graph frameworks on a variety of algorithms.

4. METHODOLOGY

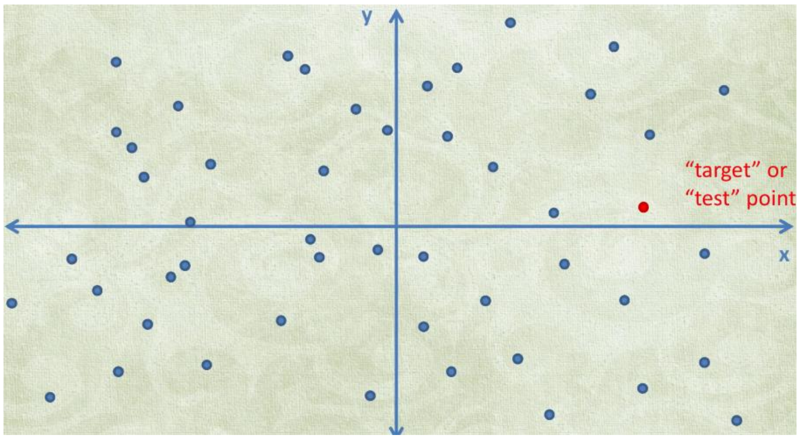


Fig: 4.1. Two Dimensional Space with a Target Point

When we look at this picture the target point will be testing point and if we want to find the k nearest neighbor we don’t need to compare distance between each points because just by seeing the picture we know what are the nearest neighbors. So we can instantly pick out nearest neighbors from a two or three dimensional space.

This is what we see, but the machine on the other hand sees the same data set like this

- Training set:
  {(1,9),(2,3),(4,1),(3,7),(5,4),(6,8),(7,2),(8,8),(7,9),(9,6))
- Testing instances:
  (7,4)

Our goal is to find three nearest neighbors from the given training set. So how do we do that? Once we look at the problem this way like a machine does it’s not nearly as obvious what the three nearest neighbors are. So in general if we are in a low dimensional space i.e. if we are in 2D there are some algorithms for doing this task efficiently, but if we are in n-dimensional space there is no trick or method that allows us to select the three nearest neighbors exactly.

So what we have to do is we have to take the entire data set and one by one compute the distance between each data points in the data set and just enumerate all the distances and that is expensive because for a data set of size **n** it will require n computations of distance and each distance computation is going to take **d** operations and is many dimensions. So your complexities **nd** and that is the testing time complexity not training and has zero training complexity but it’s very expensive a testing point especially have a lots of data and you want to have lots of data because lots of data allows you to build accurate classifiers. So how do we make it fast?

So there basically two things we can do, the complexities nd, so we can either try to make d smaller that's called dimensionality reduction, i.e. we are trying to reduce the number of attributes and here we are basically limited to very simple methods so we can use feature selection and throw away attributes that don't look promising and any thing more complicated run a factor analysis to reduce the dimensionality factor analysis itself is typically cubic in d $O(d^3)$.

One more promising thing is to reduce n, n is the number of examples that we have. So the basic idea is we approximately pick m out of n instances and then we do the comparision within that smaller set and then you pick the nearest neighbors out of that subset so our complexity is going to be md so m is as big as we are going to pick it.

How we pick the smaller number of items? It depends on what kind of data that we are dealing with if you are dealing with low dimensional data and your data is real valued so you are in low D space then we use KD-trees, it only works for the low d and it only works for the real valued data. If we're in high dimensional space and our data is sparse and discrete then and in such case we use inverted lists which is totally a different technique and then there was generic technique called fingerprint technique. KD-trees and fingerprinting are approximate, what's meant by that is they can miss nearest neighbors.

Inverted Lists is an exact technique you're always guaranteed to get your nearest neighbors.

These all the above points are demonstrated in the below points:

Algorithm:

- Training: O(1), but testing: O(nd)
- Reduce d: dimensionality reduction
    - Simple feature selection, other methods $O(d^3)$
- Reduce n: don't compare to all training examples
    - idea: quickly identify m<<n potential near neighbors
        - compare only to those, pick k nearest neighbors -> O(md) time
    - **K-D trees**: low dimensional, real valued data
        - O(d log n), only works when d<<n, inexact: may miss neighbors
    - **Fingerprinting**: high-d, sparse or dense

        - O(dn’) n’<<n… bits in fingerprinting inexact: may miss neighbors
    - **Inverted lists**: high dimensional, discrete data
        - O(d’n’) where d’<<d, n’<<n, only for sparse data (eg. Text), exact: do not miss any neighbor.

INVERTED INDEX / LISTS:

An inverted index [21] is an index data structure storing a mapping from content, such as words or numbers, to its locations in a document or a set of documents. In simple words, it is a hash map like data structure that directs you from a word to a document or a web page.

There are two types of inverted indexes: A **record-level inverted index** contains a list of references to documents for each word. A **word-level inverted index** additionally contains the positions of each word within a document. The latter form offers more functionality, but needs more processing power and space to be created.

Suppose we want to search the texts “hello everyone, ” “this article is based on inverted index, ” “which is hash map like data structure”. If we index by (text, word within the text), the index with location in text is:

hello	(1, 1)
everyone	(1, 2)
this	(2, 1)
article	(2, 2)
is	(2, 3); (3, 2)
based	(2, 4)
on	(2, 5)
inverted	(2, 6)
index	(2, 7)
which	(3, 1)
hashmap	(3, 3)
like	(3, 4)
data	(3, 5)
structure	(3, 6)

The word “hello” is in document 1 (“hello everyone”) starting at word 1, so has an entry (1, 1) and word “is” is in document 2 and 3 at ‘3rd’ and ‘2nd’ positions respectively (here position is based on word). The index may have weights, frequencies, or other indicators.

Steps to build an inverted index:

- **Fetch the Document**
  Removing of Stop Words: Stop words are most occurring and useless words in document like “I”, “the”, “we”, “is”, “an”.
- **Stemming of Root Word**
  Whenever I want to search for “cat”, I want to see a document that has information about it. But the word present in the document is called “cats” or “catty” instead of “cat”. To relate the both words, I’ll chop some part of each and every word I read so that I could get the “root word”. There are standard tools for performing this like “Porter’s Stemmer”.
- **Record Document IDs**
  If word is already present add reference of document to index else create new entry. Add additional information like frequency of word, location of word etc.

Example:

Words	Document
ant	doc1
demo	doc2
world	doc1, doc2

Advantage of Inverted Index are:

- Inverted index is to allow fast full text searches, at a cost of increased processing when a document is added to the database.
- It is easy to develop.
- It is the most popular data structure used in document retrieval systems, used on a large scale for example in search engines.

Inverted Index also has disadvantage:

- Large storage overhead and high maintenance costs on update, delete and insert.

5. LIMITATIONS AND DISCUSSIONS

Construction of a nearest neighbor graph is often a necessary step in many machine learning applications. However, constructing such a graph is computationally expensive, especially when the data is high dimensional. Python's opensource machine learning library Scikit-learn uses k-d trees and ball trees to implement nearest neighbor graph construction. However, this implementation is inefficient for large datasets.

k-d trees can be used only for the 2-dimensional and 3-dimensional data and so on up to some limited dimensions. As we go on increasing the datasets and the dimensions of the plane then it will be more difficult to analyze the data and retrieve the information from the available datasets. So, in such cases k-d trees fails to cope up with the high dimensional datasets as we discussed earlier. So here in such cases comes the aid of **inverted indexing**, where they can be able to handle high dimensional data in an efficient manner resulting in accurate information retrieval from the given datasets which are of high dimensional.

6. FUTURE SCOPE AND CONCLUSION

This paper mainly focusses on the implementation of the K Nearest Neighbor algorithm which is widely used in many machine learning applications using. The only disadvantage using k-d tree in KNN is k-d trees fails to deal with high dimensional data.

So to overcome such problems with trees, we discussed about inverted index algorithm for the parallel implementation of the K Nearest Neighbor Algorithm which helps in getting nearest neighbors from high dimensional data. To increase the efficiency and to reduce the time involved in solving the problems related to the high dimensional data sets more efficient algorithms are to be implemented to get exact and accurate results.

7. REFERENCES

[1] Nazneen Rajani, Kate McArdle, Inderjit S. Dhillon, “ Parallel k Nearest Neighbor Graph Construction Using Tree-Based Data Structures”, IEEE, 2015.

[2] J. S. Beis and D. G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In Computer Vision and Pattern Recognition, 1997. Proceedings, 1997 IEEE Computer Society Conference on, pages 1000–1006. IEEE, 1997.

[3] J. L. Bentley. Multidimensional binary search trees used for associative searching. Communications of the ACM, 18(9):509–517, 1975.

[4] J. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. Transactions on Mathematical Software, 3(3):209–226, 1977.

[5] C. Hentschel and H. Sack. Does one size really fit all? evaluating classifiers in bag-of-visual-words classification. In i-KNOW Proc. 14th Intl. Conf. Knowledge Technologies and Data-driven Business. ACM, 2014.

[6] M. Kulkarni, M. Burtscher, R. Inkulu, K. Pingali, and C. Cas¸caval. How much parallelism is there in irregular applications? In ACM sigplan notices, volume 44, pages 3–14. ACM, 2009.

[7] N. Kumar, L. Zhang, and S. Nayar. What is a good nearest neighbors algorithm for finding similar patches in images? In Computer Vision–ECCV 2008, pages 364–378. Springer, 2008.

[8] T. Liu, A. W. Moore, and A. Gray. New algorithms for efficient high-dimensional nonparametric classification. The Journal of Machine Learning Research, 7:1135–1158, 2006.

[9] A. W. Moore. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence, pages 397–405. Morgan Kaufmann Publishers Inc., 2000.

[10] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In VISAPP (1), pages 331–340, 2009.

[11] D. Nguyen, A. Lenharth, and K. Pingali. A lightweight infrastructure for graph analytics. In Proc. 24th Symposium on Operating Systems Principles, pages 456–471. ACM, 2013.

[12] Pace and Barry. Sparse spatial autoregressions. Statistics and Probability Letters, 1997.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830, 2011.

[14] K. Pingali, D. Nguyen, M. Kulkarni, M. Burtscher, M. A. Hassaan, R. Kaleem, T.-H. Lee, A. Lenharth, R. Manevich, M. M´endez-Lojo, et al. The tao of parallelism in algorithms. ACM Sigplan Notices, 46(6):12–25, 2011.

[15] E. Plaku and L. E. Kavraki. Distributed computation of the k nn graph for large high-dimensional point sets. Journal of parallel and distributed computing, 67(3):346–359, 2007.

[16] C. Silpa-Anan and R. Hartley. Optimised kd-trees for fast image descriptor matching. In Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, pages 1–8. IEEE, 2008.

[17] R. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. Algorithmica, 6(4):579–589, 1991.

[18] A. Uzilov, J. Keegan, and D. Mathews. Detection of non-coding rnas on the basis of predicted secondary structure formation free energy change. BMC Bioinformatics, 7, 2006.

[19] N. Satish, N. Sundaram, M. A. Patwary, J. Seo, J. Park, M. A. Hassaan, S. Sengupta, Z. Yin, and P. Dubey. Navigating the maze of graph analytics frameworks using massive graph datasets. In Proc. 2014 ACM SIGMOD Int'l. Conf. on Management of Data, 2014.

[20] K. Bache and M. Lichman. UCI machine learning repository, 2013.

[21] https://www.geeksforgeeks.org/inverted-index/

[31]Akbarzhon Madaminov, “Recommendation Systems”, Engpaper Journal

[32]Aathi oli.S , “REVIEW PAPER ON PHISHING ATTACKS”, Engpaper Journal

[33]Rania Fernando, “IoT based – Street Light Controlling System”, Engpaper Journal

[34]K. SAI BHARGAV, V. RAJENDRA, “Study on Data Structures for Machine Learning”, Engpaper Journal

[35]Brundha P, Guruprasad K N, Amith V Hiremath,Sirisha R, Chandrakanth G Pujari , “Face Detection Based Smart Attendance System Using Haar Cascade Algorithm”, Engpaper Journal

[36]Afsana Nadaf , “RFID BASED LIBRARY MANAGEMENT SYSTEM”, Engpaper Journal

[37]Mr. Vedant Thube, Neha Thakur, Mr. Siddhesh Balsaraf,Ms. Priyanka Hanchate, Dr. S. D. Sawarkar , “Accident Prevention using Eye Drowsiness & Yawning Detection”, Engpaper Journal

[38]Abhishek A Hishobkar, Rutuja Gaonkar, Jagdish Chintamani , “DIGITAL DIARY”, Engpaper Journal

[39]Pooman Suryavanshi, Aryan Ghadge, Manali Kharat , “TAXI SERVICE for VISUALLY IMPAIRED”, Engpaper Journal

[40]Mr. Pankaj yadav, Shila Jawale, Mr. Ashutosh Mahadik, Ms. Neha Nivalkar, Dr. S. D. Sawarkar , “NEWS ARTICLES CLASSIFICATION”, Engpaper Journal

[41]Rahul Chavan, Manvee Bhoir, Gaurav Sapkale, Anita Mhatre, “Smart Tourist Guide System”, Engpaper Journal

[42]Rutik Desai, Akash Jadhav,Suraj Sawant ,Neha Thakur , “Accident Detection Using ML and AI Techniques”, Engpaper Journal

[43]Anagha Vishe,Akash Shirsath, Sayali Gujar, Neha Thakur , “Student Attendance System using Face Recognition”, Engpaper Journal

[44]Ms.Sayali Patekar, Shila jawale, Ms.Pranali Kurhade, Mr.Shubham Khamkar , “Smart Classroom Application”, Engpaper Journal

[45]DOSHI SAKSHI, DEVYANI CHAUDHARI, POOJA GAIKWAD, RUTUJA CHABUKSWAR,MRS. SUJATA KOLHE, “TOURISM SIMPLIFIED THROUGH VOICE”, Engpaper Journal

[46]Afreen Fathima,Samreen Jameel, Pathan Ahmed khan , “ACCIDENT DETECTION AND ALERTING SYSTEM”, Engpaper Journal

[47]Suman Zareen, Tuba Masood, Pathan Ahmed khan, “E-Commerce Web Application with Augmented Reality”, Engpaper Journal

[48]Lok Shan CHAN, “Selection of Waterfall and Agile Methodologies in Software Testing”, Engpaper Journal

[49]Barve Rutu, “CLOUD COMPUTING SYSTEM FOR GAMING”, Engpaper Journal

[50]Harshvardhan Singh, “Machine Learning: Fake News Blocking”, Engpaper Journal

[51]M.Al Batahari, “SERVERS ROOM MONITORING SYSTEM USING IOT”, Engpaper Journal

[52]AYUSHI ANKITA RAKSHIT, “VIRTUAL MASTER USING PYTHON”, Engpaper Journal

[53]Baldeep Kaur, “REAL TIME SLEEP DROWSINESS DETECTION USING FACE RECOGNITION”, Engpaper Journal

[54]Suchitav Khadanga, “Two Stage CMOS Operational Amplifier From Specification to Design”, Engpaper Journal

[55]nidhi sharma, “Introduction to Remote Sensing”, Engpaper Journal

[56]Rohith N Reddy, “COVID-19 Detection using SVM Classifier”, Engpaper Journal

[57]Swapnil Kole, “COVID-19 Database on Consortium Blockchain”, Engpaper Journal

[58]TejalLengare, PallaviSonawane, PrachiGunjal, ShubhamDhire, Prof.Shaikh.J.N , “Accident Detection & Avoidance System in Vehicles”, Engpaper Journal

[59]Abhishek Pawshekar, Deepti More, Akash Khade, Pratiksha Wagh, Ganesh Ubale, “Augmented Reality: to converting and placing object into 3D model”, Engpaper Journal

[61]Prof.Ubale.G.S, Pranjal Adhav,Pooja Gaikwad, Sushama Nadavade ,Pooja Kale , “Iot based Bridge Monitoring System”, Engpaper Journal

[62]Divya Deewan, Priyanka Maheshwari, Sanjay Jain, “A REVIEW OF BATTERY-SUPERCAPACITOR HYBRID ENERGY STORAGE SYSTEM SCHEMES FOR POWER SYSTEM APPLICATION”, Engpaper Journal

[63]Prof.Ansari.M.B, Pranjal Adhav,Pooja Gaikwad,Sushama Nadavade,Pooja Kale, “Survey on MyHelper IOT based Bridge Monitoring System”, Engpaper Journal

[64]Shreyas.S.J, Saddam hussain, Chaithra E, “COMPARATIVE STUDY ON SEISMIC RESPONSE OF MASONRY INFILLED RC FRAME BUILDINGS AND MIVAN BUILDINGS WITH DIFFERENT PERCENTAGE OF WALL OPENINGS”, Engpaper Journal

[65]Yusuf Ali Hassan, “Somali Power-Grid Significant Challenges”, Engpaper Journal

[66]Ahmed N. Elhefnawy, “Refractive IR Objective Optical Design Operating in LWIR band For Military Observation Applications”, Engpaper Journal

[67]S MANJULA, D SELVATHI and SUCHITAV KHADANGA, “Design of low-power CMOS transceiver front end for 2.4-GHz WPAN applications”, Engpaper Journal

[68]Suchitav Khadanga, “Fabrication of MEMS Pressure Sensor on thin film membrane”, Engpaper Journal

[69]Suchitav Khadanga and Dr. K.R.Suresh Nair, “An Introduction to Bluetooth”, Engpaper Journal

[70]Suchitav Khadanga and S. Ahmad, “DESIGN AND FABRICATION OF LOW COST MICROWAVE OSCILLATOR”, Engpaper Journal

[71]Ameen Ahmed, Noushad S, Suchitav Khadanga, K.R.Suresh Nair, P.K.Radhakrishnan, “DEVELOPMENT OF LOW PHASE NOISE SMALL FOOT PRINT SURFACE MOUNT VOLTAGE CONTROLLED OSCILLATOR”, Engpaper Journal

[72]Suchitav Khadanga , “Synchronous programmable divider design for PLL Using 0.18 um cmos technology”, Engpaper Journal

[73]Kavya.G.R, Shivaraju.G.D, Dr. T V Mallesh, S R Ramesh, “PROGRESSIVE COLLAPSE RESISTANCE OF FLAT SLAB BUILDING”, Engpaper Journal