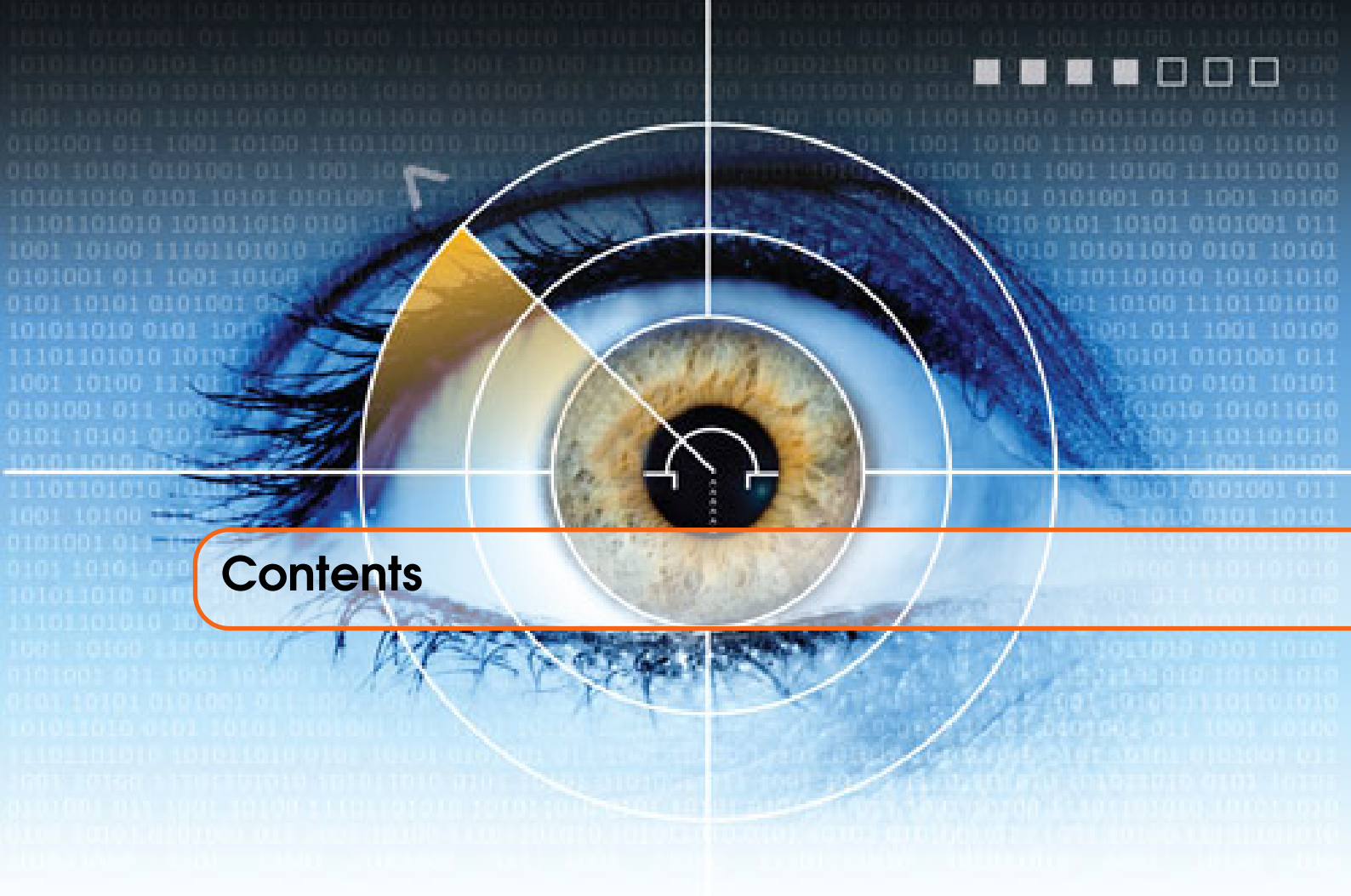
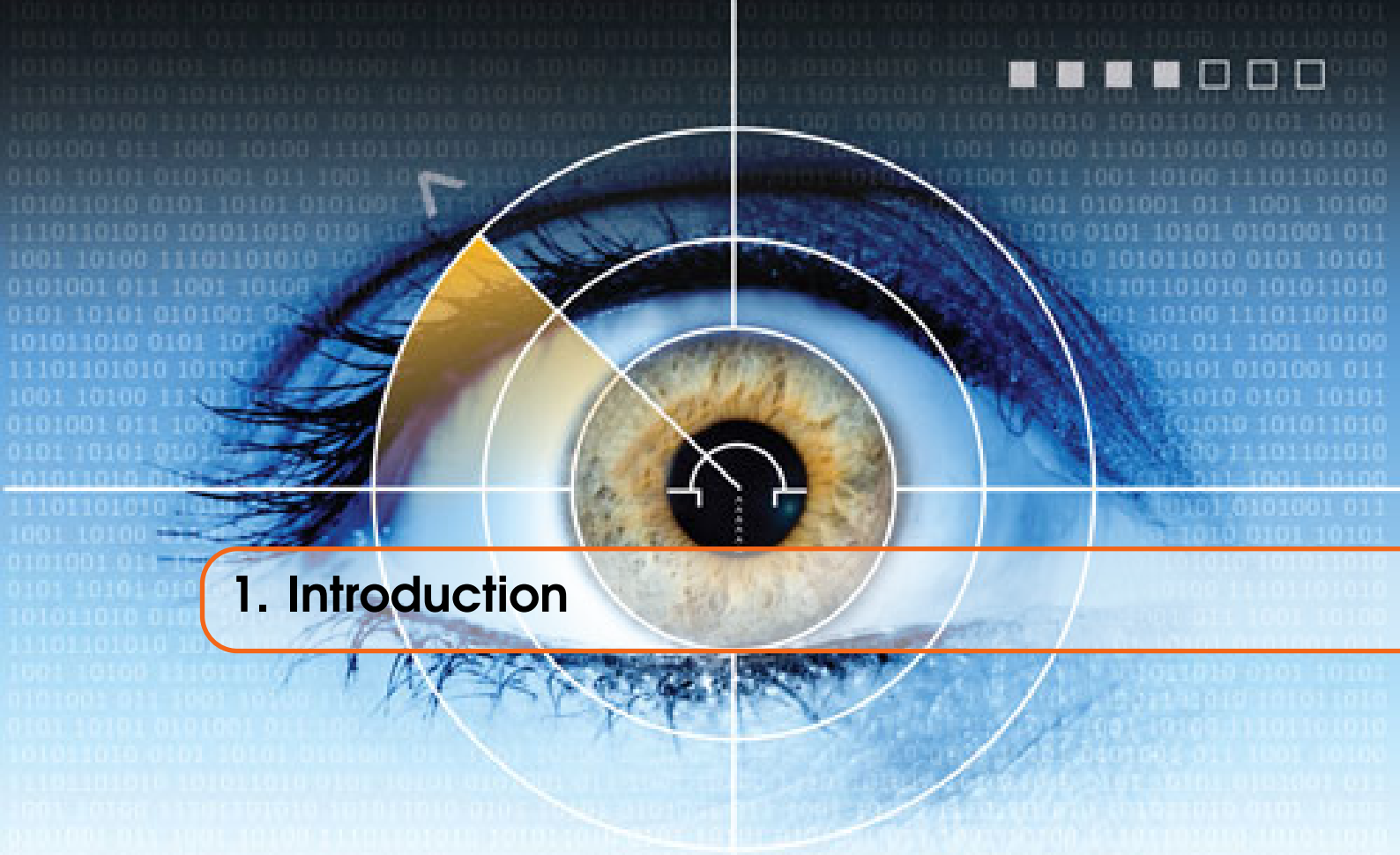
Click Through Rate Prediction

INFO3406: Assignment 2

Ian Cleasby SID: 430595639, Costas Korai SID: 440241607



Contents



1. Introduction

This report outlines the details of our teams attempts to undergo the Kaggle Display Advertising Challenge. In our attempt, despite facing technical difficulties, we have come up with a solution to classify the data.

1.1 Aim

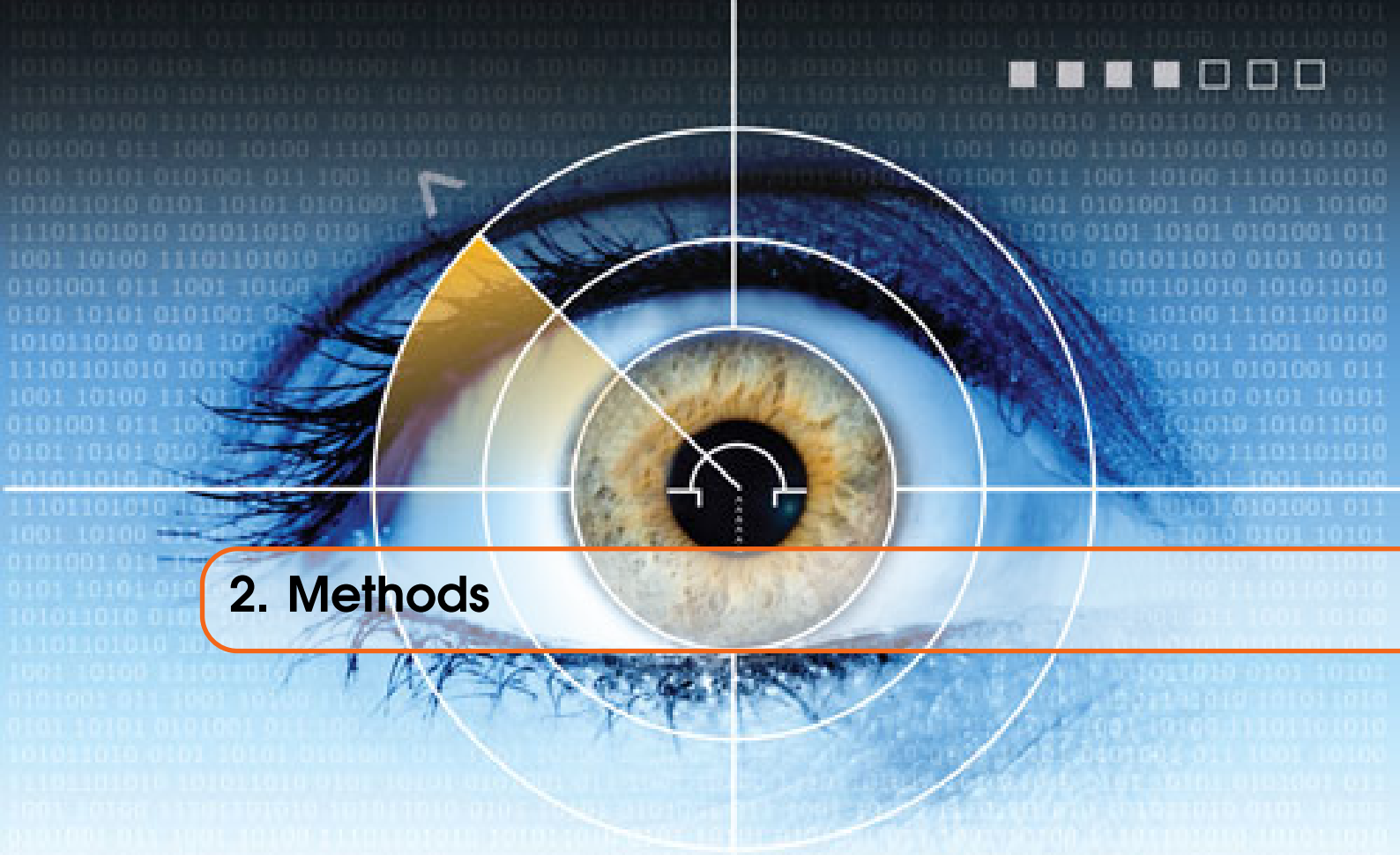
The aim of this assignment is to implement a click through rate prediction algorithm to run on the CriteoLab's Kaggle Display Advertising Challenge dataset[1]. The goal of this algorithm being to correctly predict for a given user if an advertisement when presented to him will be clicked on. Our algorithm will implement a locality-sensitive hashing(LSH) based k-nearest-neighbour (k-nn) classifier using map-reduce techniques. The objective of our algorithm will be to predict for a given user and the page he is currently visiting, whether he will click on a given advertisement.

1.2 Why is this study (the problem) important

As seen throughout this unit of study and during our first assignment the problem of accurately classifying information whether it be images or pure statistical data is an extremely important area of study in data analytics. In this assignment the problem we are given is a prime example of where this study can be applied to benefit real world tasks.

The original task our assignment was based on was CriteoLabs' Display Advertising Challenge, a programming challenged issued by Criteo a personalized retargeting company that works alongside internet retailers to create personalized online display advertisement. Criteo sponsored this challenge in order to push for the development of better optimized more accurate prediction algorithms for their specific data sets, which they could then implement in their own business improving the quality of their service.

Looking at Criteo as an example we can look at the different groups affected by this problem, Criteo itself whose continued survival and growth depends on it's constant improvement in this sector, programmers who can use such challenges to sponsor themselves while furthering this area of study and end users of online retail stores who will have the content provided to them change depending on the different algorithms used.
With this large range of groups affected by this problem it is plane to see that this area of study has far reaching effects in the real world and so must be considered important.Equally important though is the effect it has on data analytics overall, with the constant need for improvement this problem has essentially become fuel for data analytics giving both reason and resources for its continued study and improvement.



2. Methods

2.1 Similarity Metrics

For this assignment we where required to use two main similarity metrics in order to implement an LSH based K Nearest Neighbour, these metrics would be needed in order for us to map the initial data to sparse vectors and then to give us a metric for distance calculation between specific points.

2.1.1 Distance Calculating Equations

The two similarity metrics we are using are the cosine similarity and the Euclidean distance. The cosine of two vectors can be derived by the Euclidean dot product of two vectors. The equation is:

$$a.b = ||a||||b||\cos(\theta)$$

The similarity of two vectors of attributes is given by a dot product and magnitude

$$Similarity = cos(\theta) = \frac{A.b}{||A||||B||} = \frac{\sum_{i=1}^{n}(A_i B_i)}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

The euclidean distance is calculated as the square root of the sum of the distance between point p and q squared. The equation is:

$$d(p,q) = d(q,p) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Other distance calculating equations we considered were:

- Hamming
- Jaccard
- Taxicab

2.1.2 Locality-sensitive hashing

A locality sensitive hash is a hash function that takes some data point and hashes it into a number, with the added condition that two points that are close by some metric of interest in the original space have hashes that are close to the same value.
Within this assignment a goal was the implementation of Locality Sensitive Hashing. Locality-sensitive hashing (LSH) reduces the dimensionality of high-dimensional data. LSH hashes input items so that similar items map to the same "buckets" with high probability (the number of buckets being much smaller than the universe of possible input items).LSH differs from conventional and cryptographic hash functions because it aims to maximize the probability of a "collision" for similar items. Locality-sensitive hashing has much in common with data clustering and nearest neighbor search.

2.1.3 K-Nearest-Neighbour

In this unit we were introduced to the Nearest-Neighbour Classifier a method of inferring the class of an object based on its distance from the remaining objects within its data set. The basic idea of this method is that similar objects have similar properties in space, in our advertisement-user combinations would produce similar feature values within their given range. Looking closer at this method there are three requirements needed before it can be implemented the set of stored records, a distance metric to compute the distance between records and the number of nearest neighbours to retrieve which will have a different optimal value between varying implementations and data sets. The set of stored records we will use is the Kaggle Display Advertising Challenge dataset, in order to calculate distances between points in our data we will use the euclidean distance calculation method, our value for K will be decided based on results we get from tests.

2.1.4 Tested Hyperparrameter Results

Here we have the results of our testing, for this assignment all benchmarks were run within the spark virtual machine run with varying settings of processor counts.

Virtual box at 1 core Approximate total runtime of 38 hours

K value	20
Accuracy %	75

Virtualbox 2 cores run time of 8 hours for both K values

K value	10	20
Accuracy %	76	74.6

Virtualbox 2 cores run time of 16 hours

K value	5
Accuracy %	74

Virtualbox set to 4 cores on hyperthreaded machine runtime of 33 hours

K value	5
Accuracy %	76

Virtualbox 3 cores runtime of 4 hours

K value	20
Accuracy %	75

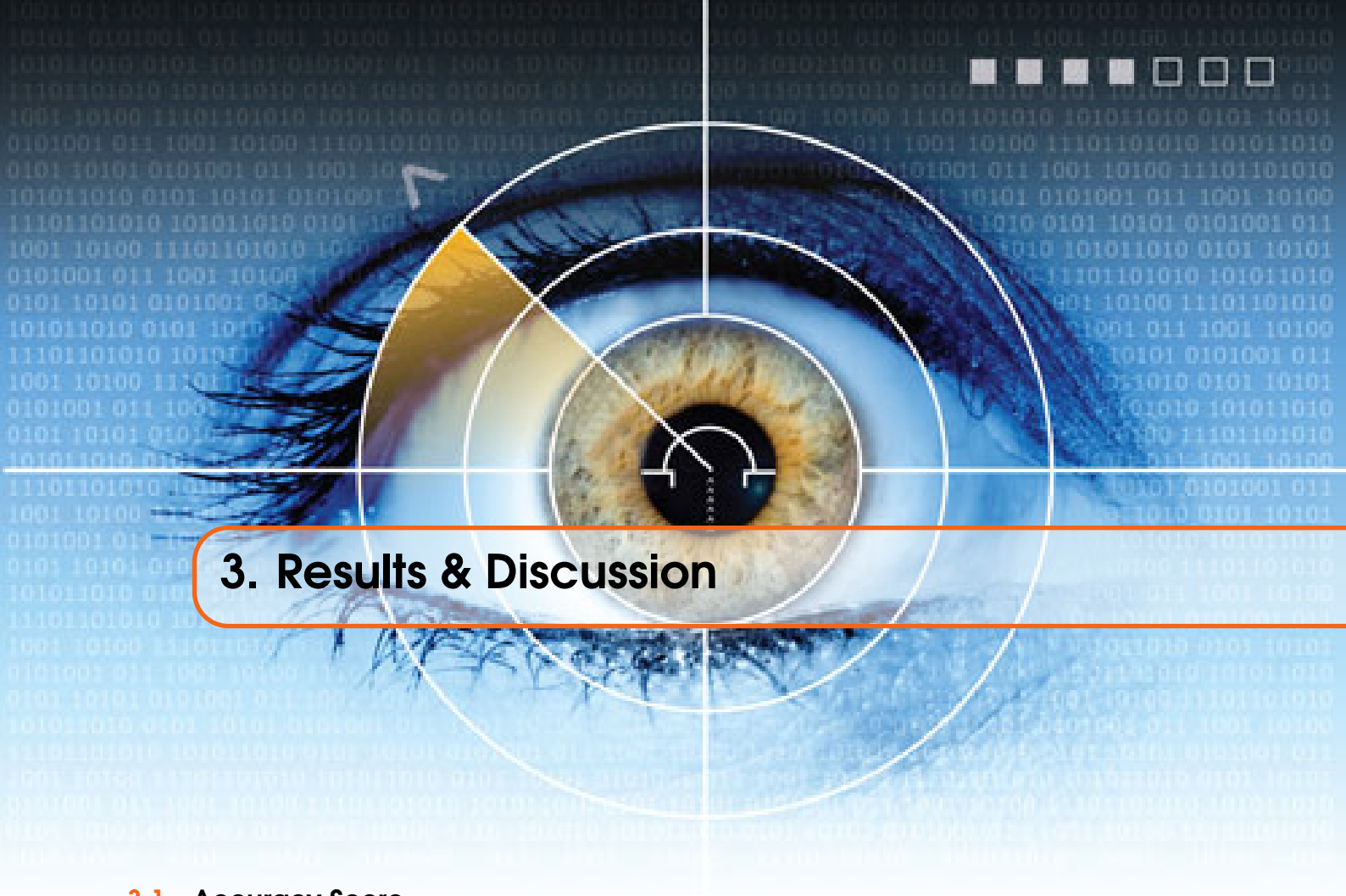
Accuracy at subset of test data size 1000:

2 core virtualbox with euclidean distance calculation

K value	1	3	5	7	10	11	15	20
Accuracy %	78	79	77	65	71	79	76	75

Run times of the subset of test data size 1000:

K value	1	3	5	7	10	11	15	20
Runtime	49mins	49min	46min	49min	50min	50min	50min	55min



3. Results & Discussion

3.1 Accuracy Score

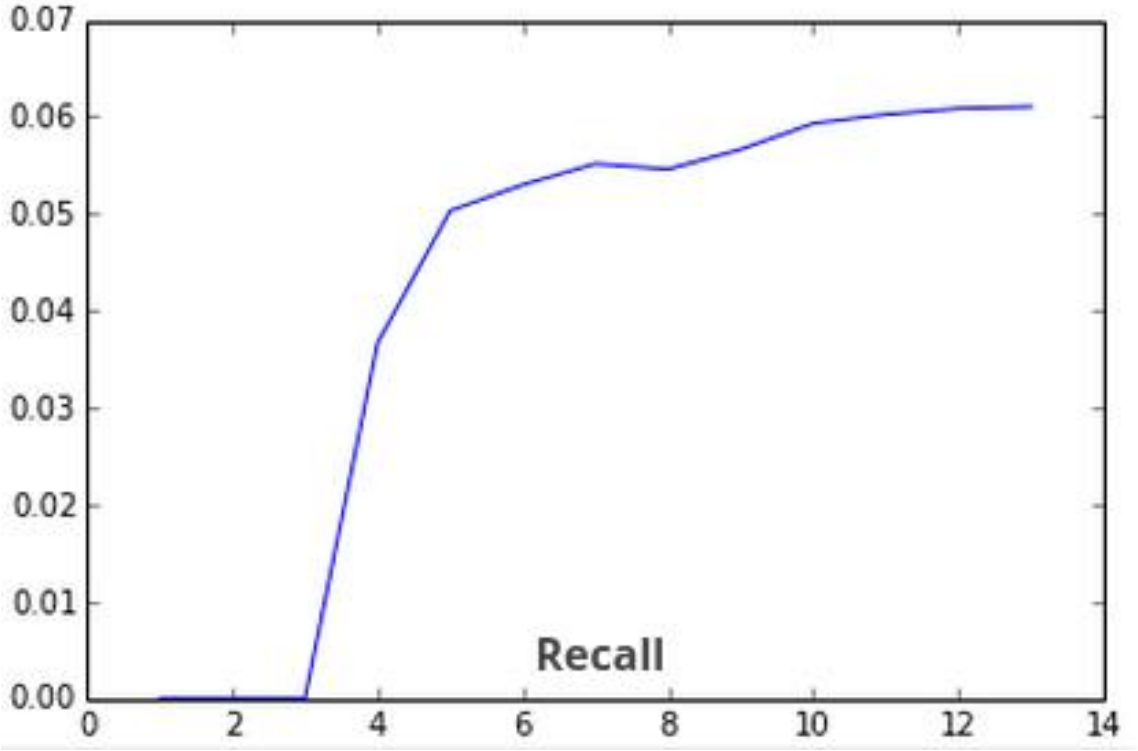
In this assignment we needed to implement the K-nn classifier, from our prior experience with it in assignment one we initially did not have high expectations for the maximum accuracy we could accomplish with it as visible in our results above, this was not the case. In assignment one our maximum accuracy was around 38% after testing and tuning had been done, while for this assignment we achieved 75% on the first run as such we looked to ensured that there was no factor skewing our results no such errors were found.

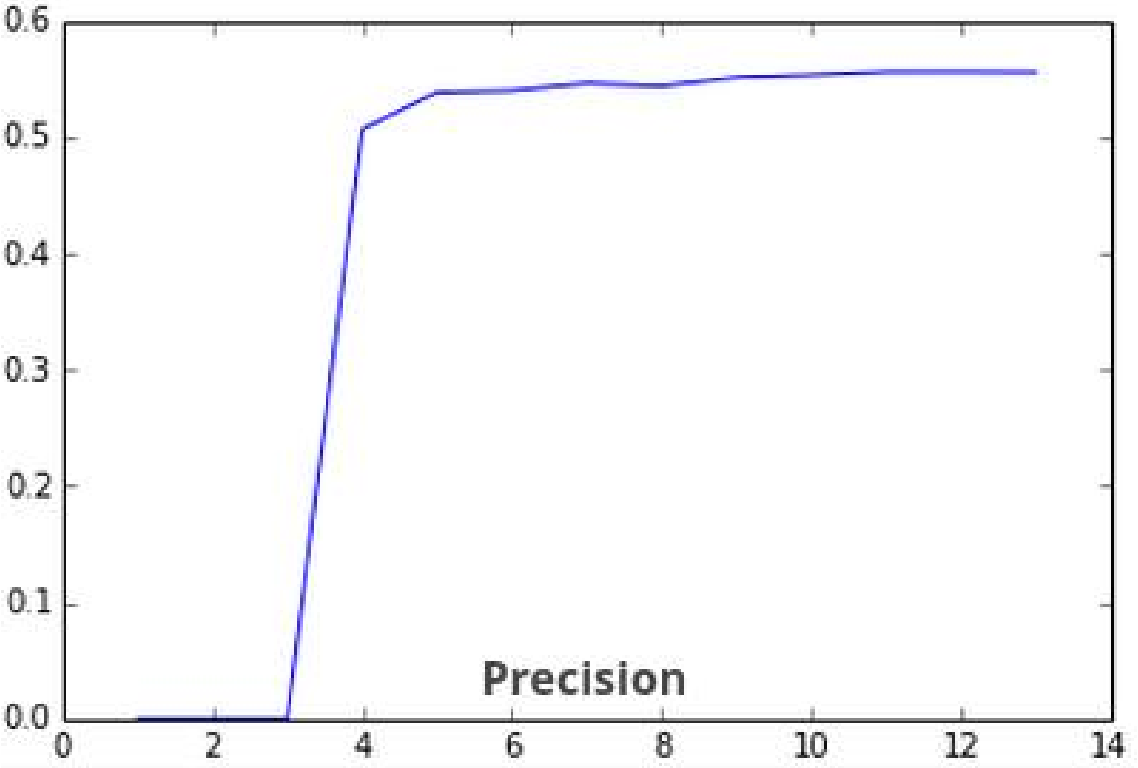
Looking away from the code and back to the data it its understandable why this high accuracy had occurred, similar to how the increase in classes between cifar-10 and cifar-100 in assignment one caused a decrease in the accuracy of K-nn in this case the small number of classes (only yes and no) and the large number of comparable features has allowed for a higher accuracy k-nn run.

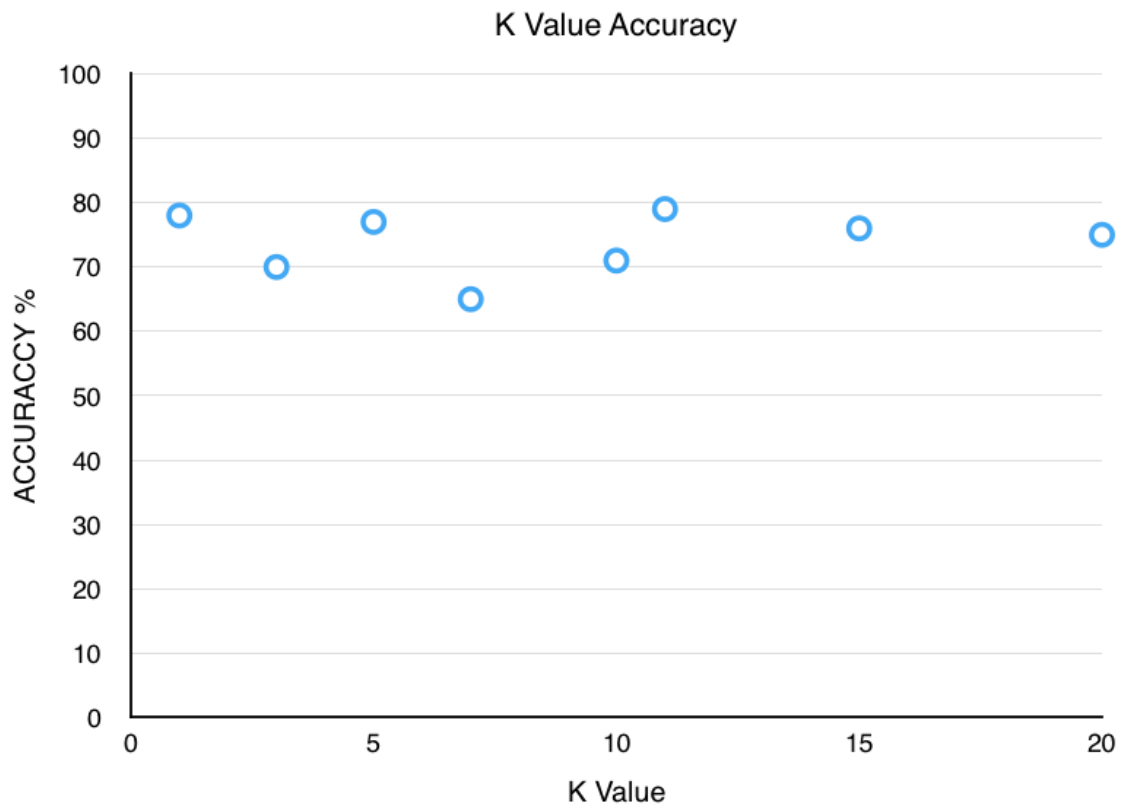
Hyperparameters while having less of a noticeable effect on our results still served their purpose of allowing us to fine tune our algorithm in order to increase our overall accuracy. Attempting to find the most suitable value for k by bench-marking different values of it we quickly realized running the full data set would delay us excessively. In order to allow us to choose a non-random value we decided to test a subset of one tenth of the original data as a scaled down representation of the whole and proceeded to test possible k values from 1 to 20.

In our results we found that our variation of K resulted in accuracy levels between 62 and 79 percent, surprisingly lower value K results yielded the most accurate predictions, these were not taken as our final K value, the reason for this being us wanting to have an algorithm less affected by noise. Knowing that with K-nn the larger the value of K the less it is affected by noise we decided to use K=11 as our final value.

Looking at the recall and precision graphs we see that overall we have a recall peeking at 0.06(where max is 1) and a precision of 0.54(max is 1), optimally we would want a higher value in both.





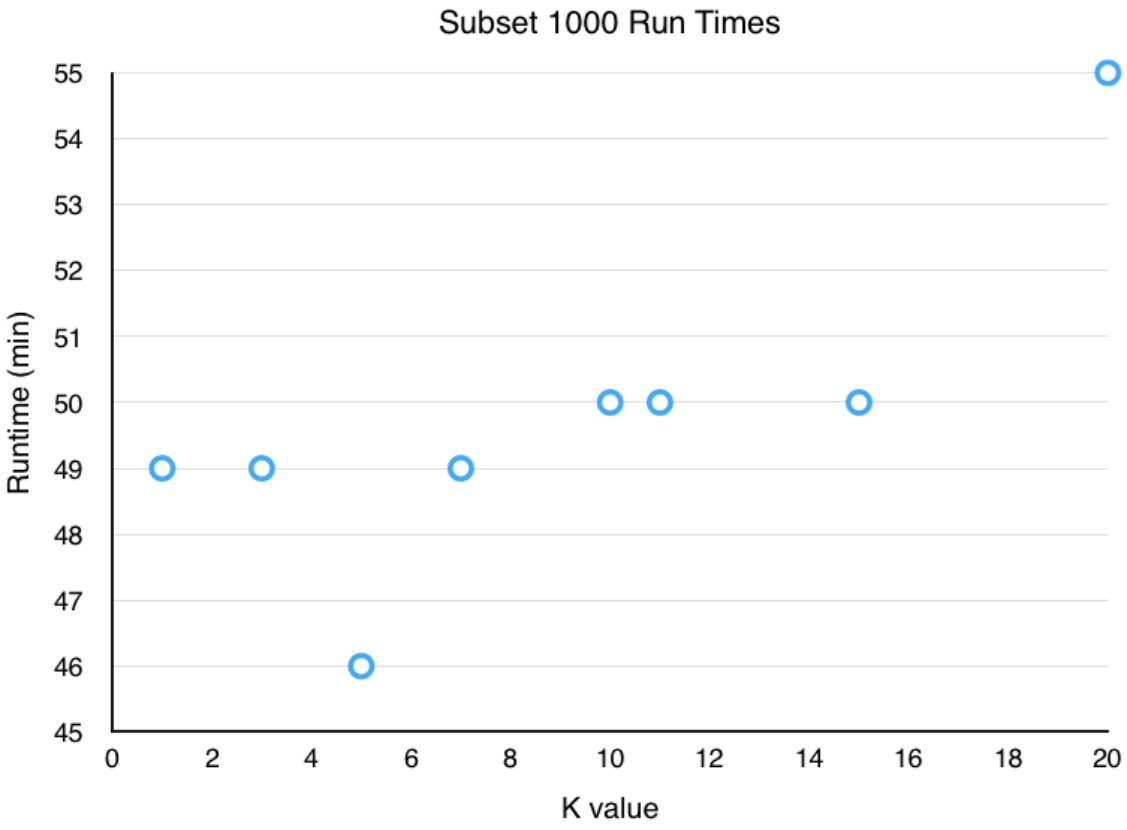


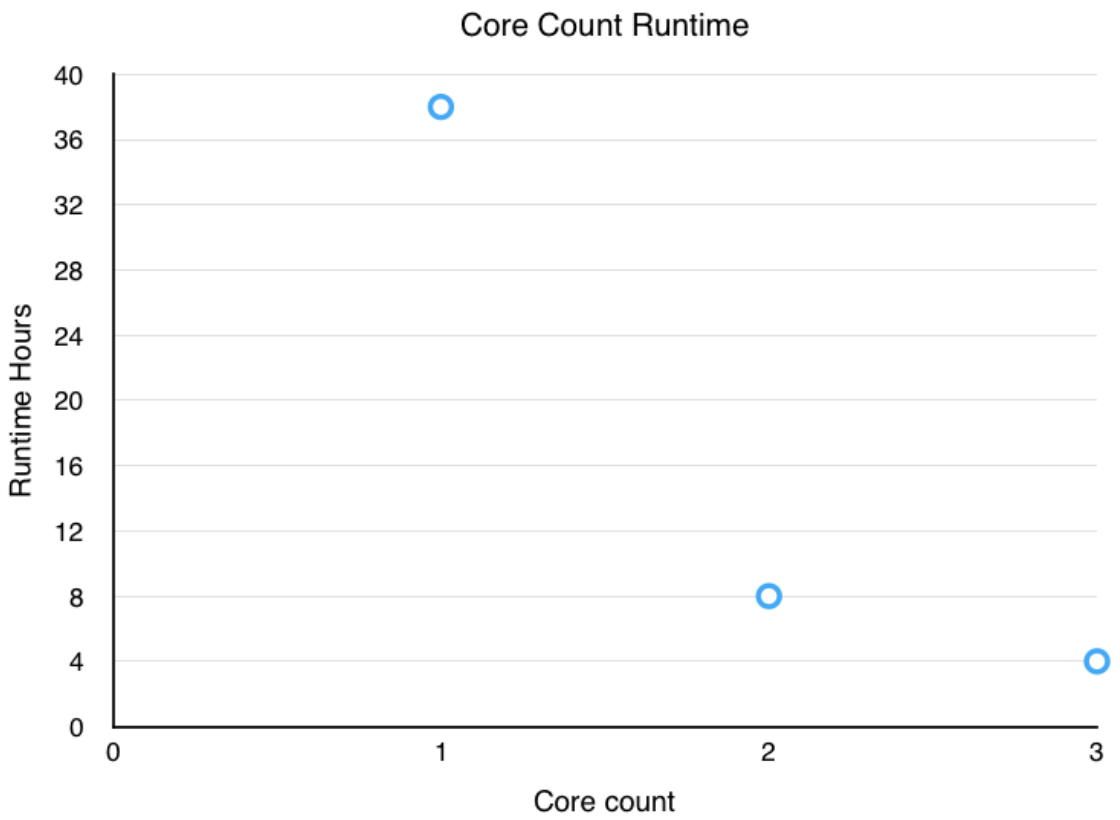
3.2 Speed

During our work on this assignment a constant issue we faced was speed, with k-nn needing to compare each test item to each training item calculations quickly build up and complete classification takes time. In order to improve our runtime our code was designed with parallel running in mind, such that when testing we could alter the amount of processing cores in our virtual machine so that data could be split between them and all together processed at a much faster rate.

Testing our code at multiple core counts we found a large variation in the our run times, as visible in our graph(core count runtime) there was a direct correlation between additional cores and reductions in run time, as was expected. A small issue we faced during our bench marking in this assignment was our limited access to higher core count processors. The majority of our testing machines were duel core and duel core hyper threaded processors as such initially we intended to test from 1-4 core run times, the problem we encountered was our virtual testing environment could not take advantage of our hyper threaded(virtual) cores limiting us to a lower total count. We also found that attempting to runt at the full hyperthreaded core count caused a large slow down pushing our runtime to the levels of a single core, this was considered an anomaly and kept out of our graphs but recorded in our overall results.

Looking at our graph(subset 1000 Run times) it is clear to us that for the values of K we have tested (between 1 and 20) for the most part there is not a large variation in run time other then at k=3 and k=20 most of our results are in a minute or two of each other. Based on our results we could take k=3 to be the best option speed wise although overall we decided that k=11 would be a better option. We believe 11 is the better option as although 3 would give a smaller run time its result falls out of the overall trend of our data an as such we are considering it an anomaly. When compared against the rest of the results again their are faster runs 11 was chosen over them as the difference between them was extremely small and as seen when looking at our accuracy 11 was also the best choice there.

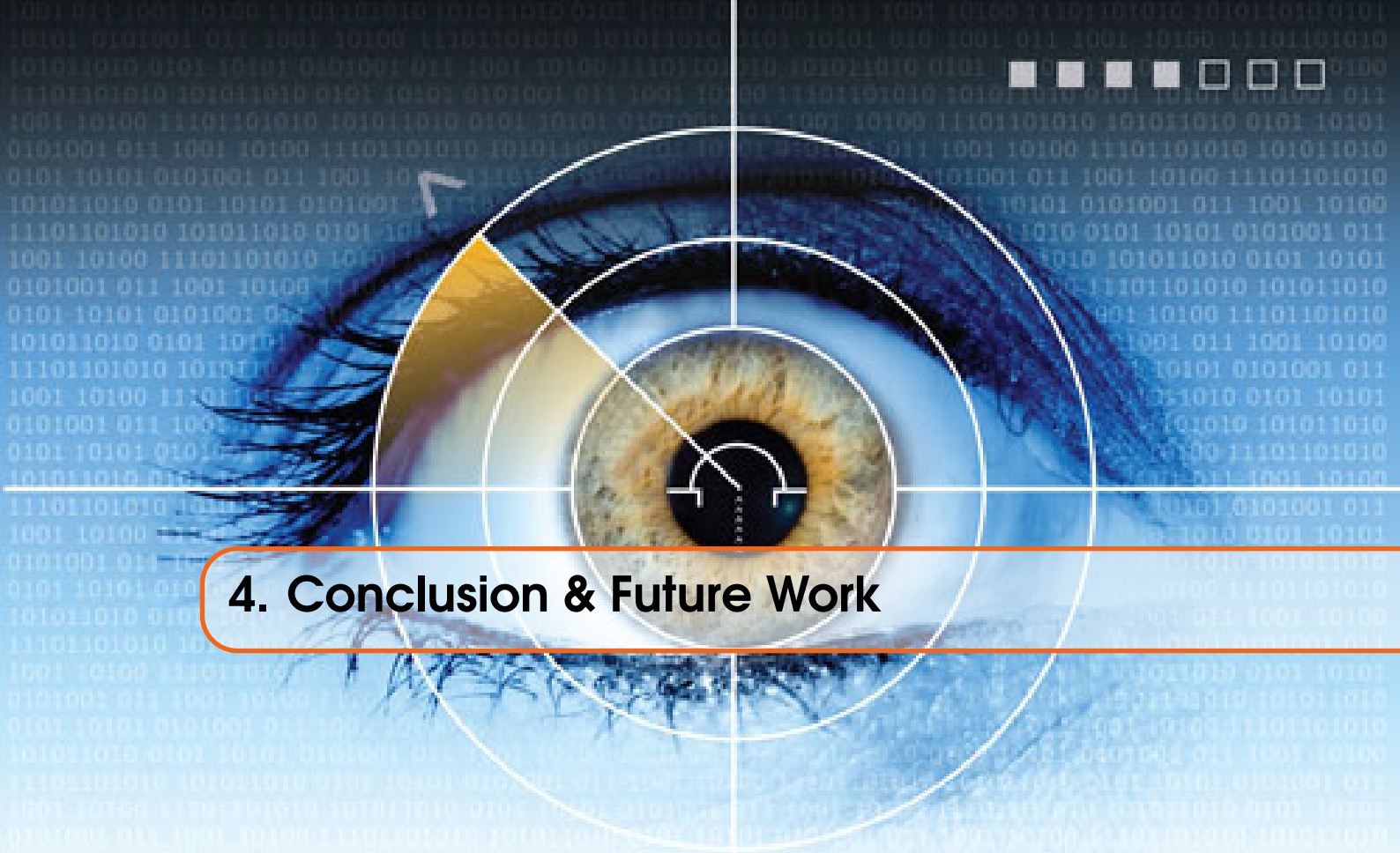




3.3 Personal Reflection

Overall, the assignment grew our perspective of classification showing us a very different task accomplished with similar means to image classification. We developed a better understanding on where data analytics is needed the tasks that data analyzers can accomplish. When working on this assignment, we revised classifiers, produced data allowing us to see how applying an optimization affects results, we developed new programming skills in the process of building the classifier and looked into performance improvements via hyperparameters, map reductions, parallel programming and hardware alterations.

Upon completion of this assignment it was evident that evaluating datasets is not always simplistic given that the classifier has to deal with many challenges in order to provide a very accurate score (80+%), although we also found that depending on the data accuracy can vary greatly.



4. Conclusion & Future Work

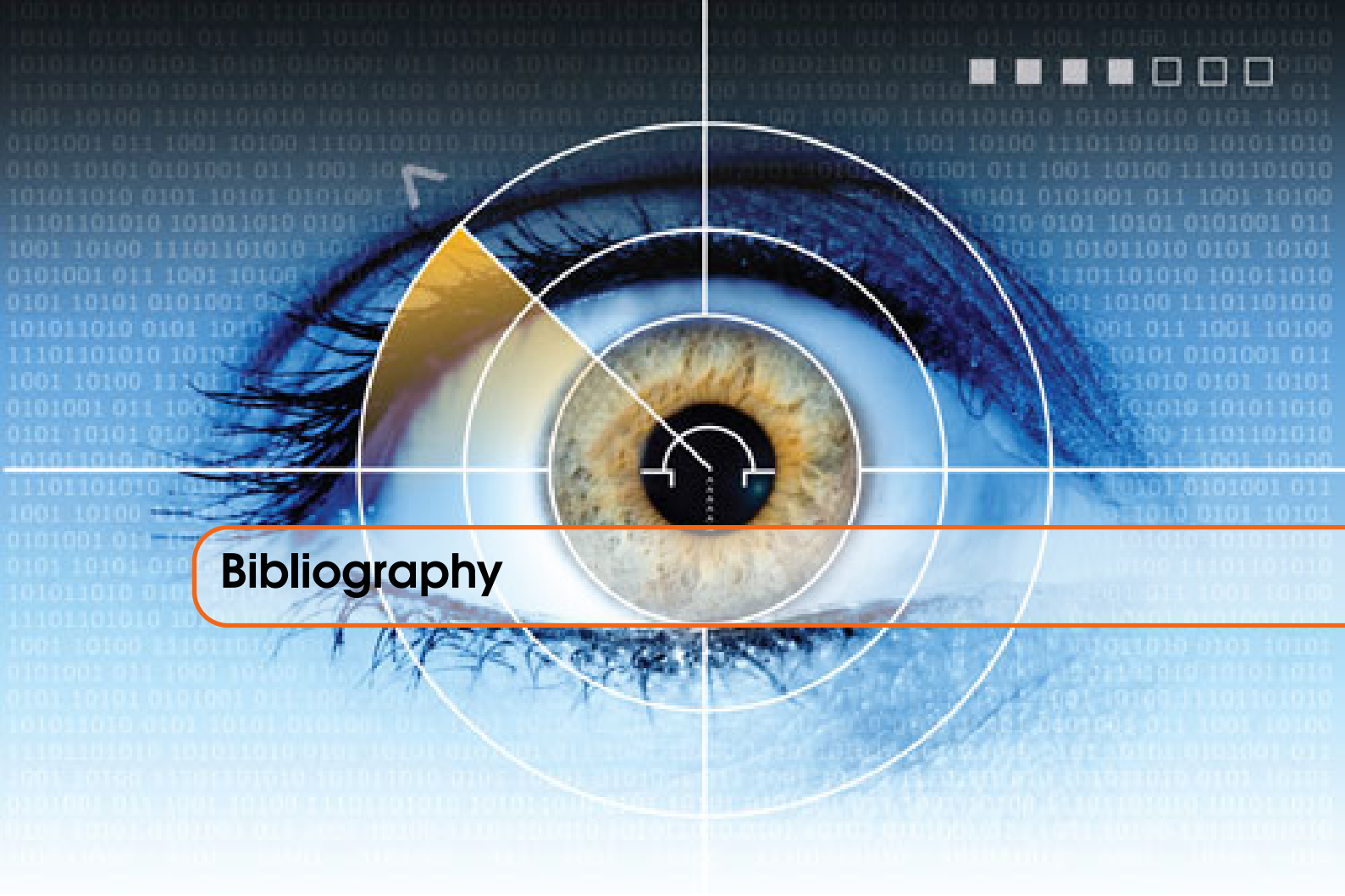
4.1 Conclusions

Throughout working on this assignment we have come to multiple conclusions about data analysis and classification. A trivial conclusion we came to quickly was the importance of optimization, studying computer science aiming for optimized code becomes second nature so when looking at our actual run times on what is a proportionally small data set when compared to real world data, there is no question in its significance.

A less obvious conclusion we came to was that different data will produce starkly different results from the same classification method. This realization comes after comparing our accuracy from assignment 2 to our assignment 1 results, in assignment one we struggled to increase our accuracy and never got passed 40% we had until this assignment believed this may have been due to the limitation of K-nn and would be similar on other data. Upon successfully running our prediction algorithm on this assignment's data set we found this was not the case and wanted to find a reason for this. In the end we came to understand that since our k-nn was very similar the results where due to the different(less) classes we needed to successfully predict between and the larger number of attributes we were comparing.

4.2 Future Work

Upon completion of this assignment what we found most interesting was the data sets affect on the accuracy when compared to our previous work, as such we have come to believe this may be an interesting future area of study. Specifically we would like to investigate how we may be able to optimize classification by altering the data it is performed on by say for example adding further attributes or removing ones that we may believe are decreasing similarity between items.



Bibliography

Articles

[1]"Display Advertising Challenge | Kaggle", Kaggle.com, 2016. [Online]. Available: https://www.kaggle.com/c/criteo-display-ad-challenge. [Accessed: 28- Oct- 2016].