

# EARIN project Final report

Krzysztof Rudnicki

Jakub Kliszko

June 12, 2023

## 1 Introduction

The goal of our project was to create a model for anime recommender. After entering anime name from the database model should output recommended anime's.

## 2 Used data and algorithms

### 2.1 Data

We used different data-set from originally specified in the project description. We decided to use Anime Recommendation Database from Kaggle: [LINK](#). Main reasons why we decided to use this database was that it was bigger than original one, was more recent, it was described as being 100% usable by Kaggle and still had decent amount of code examples.

We are mostly interested in `rating_complete.csv` file which contains information about anime ratings from users who completed the anime.

### 2.2 Algorithms

We decided to use collaborative filtering to develop our model. It makes personalized recommendations based on preferences of similar users.

We represent anime data-set as embedding vector.

We use K-nearest neighbors model and decided to test it out with different

metrics, neighbors and algorithms

### **2.2.1 Algorithms**

We decided to test our model with 2 algorithms:

1. Brute
2. Auto

Ball Tree and KD Tree do not work on sparse input (as is the case with our input) so we decided to omit them

### **2.2.2 Neighbor number**

We decided to test our model with 5 different neighbor amount:

1. 5 - Popular starting point for small-medium data-sets
2. square root of available data - Usually helps to balance between under-fitting and over-fitting
3. half of available data - Usually useful for checking overall trend than specific nuances
4. logarithm of available data - Used for very large data-sets
5. n-1 neighbors - Usually leads to over generalization as we use all instances except one for prediction

### **2.2.3 Metrics**

For brute algorithm we tested it with all possible metrics:

1. Cosine - Measures cosine of angle between vectors
2. Euclidean - Measures distance in straight line between two points
3. Manhattan - Measures sum of absolute paths between two coordinates

## **3 Intermediate results**

### **3.1 Results**

For intermediate solution we have implemented reading data from csv files, preprocessing them with optional showing of some of the information about the data and used model/learner for implementing neighbour searches

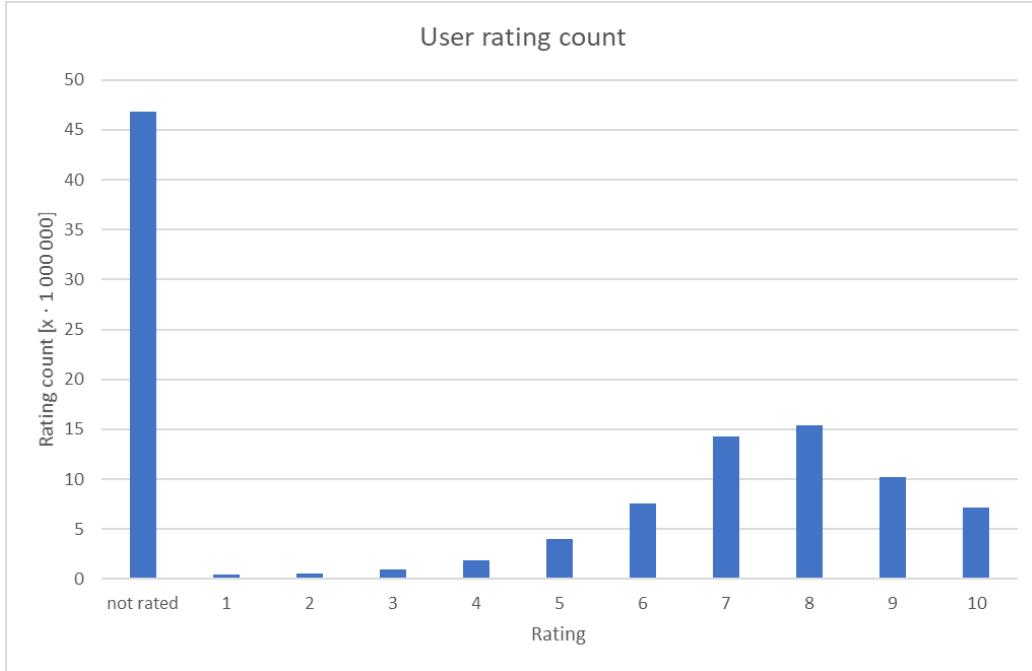
We implemented full manual mode for our program and added some parameters that will be later used for auto mode like choosing type of metric, algorithm, number of neighbors and seed

### **3.2 Insights**

We find out that the solution we wanted to initially base on: Kaggle code with tensorflow was too complicated and required too much computational time, so we changed it to a simpler, faster one that was still enough to provide the solution

We also found out when investigating the ratings that the rating is skewed towards higher values like 7, 8, 9 so the average rating is well above 5

Figure 1: User rating count



## 4 Using program

### 4.1 Arguments

There are 13 parameters that can be modified which either influence how the program behaves or how the model behaves, too see how they can be modified user needs to run

```
python main.py -h
```

Command

```
options:
-h, --help            show this help message and exit
--data_limit DATA_LIMIT, -dl DATA_LIMIT
                        Specify data limit,
                        Recommended at least 500k,
                        set to -1 for no limit
--seed SEED, -s SEED  Specify seed
```



- Database = database
- Metric = cosine
- Algorithm = brute
- Anime = RANDOM (program will randomly choose anime for which there should be recommendation)
- Neighbors = 5
- User Threshold = 500
- Anime Threshold = 200
- Recommendation Amount = 5
- Auto mode = False

#### 4.1.2 Reproducing

In order to reproduce test results user should use:

```
python main.py -au True -dl 600000
```

Command which will run auto mode with 600 thousand entries

## 5 Final experimental results

### 5.1 Experiments

All of our experiments were done on data limited to **600 thousand entries**, rest of parameters were default

We checked for 2 things in our experiments, precision of our algorithm and distances between recommended anime and input anime for different metrics and number of neighbors

Precision was calculated based on the rating of anime which was recommended, if the anime recommended had rating higher or equal to 8 then the recommendation was "good", so true positive is any anime which rating is above or equal to 8

Unfortunately we did not manage to calculate recall and F1 score (since it requires recall to be calculated)

## 5.2 Results

### 5.2.1 Precision

Figure 2: Precision from different metrics and auto algorithm

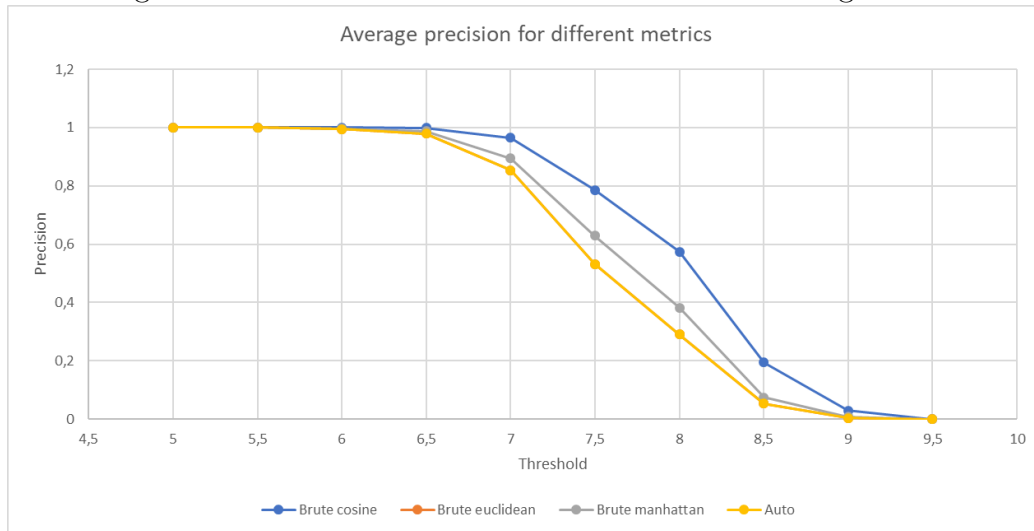
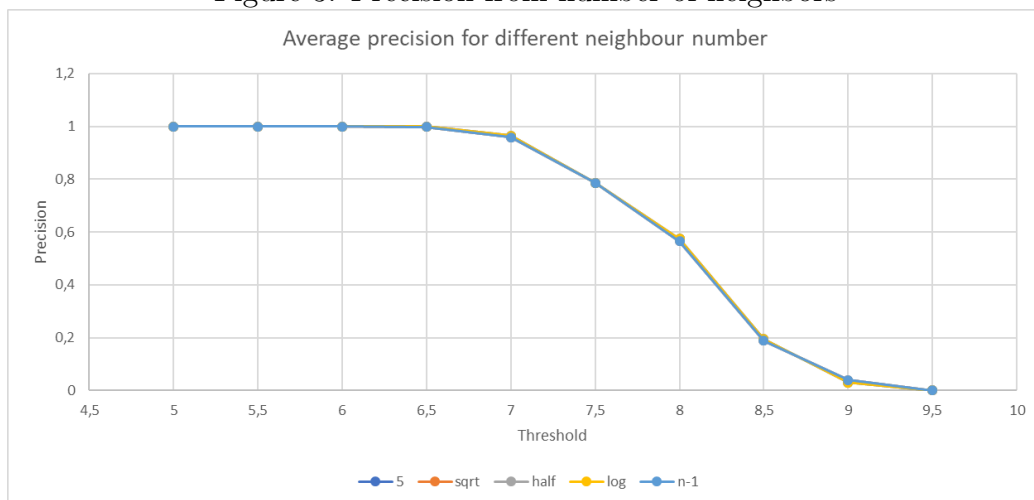


Figure 3: Precision from number of neighbors



## 5.2.2 Distance

Figure 4: Distance for Manhattan

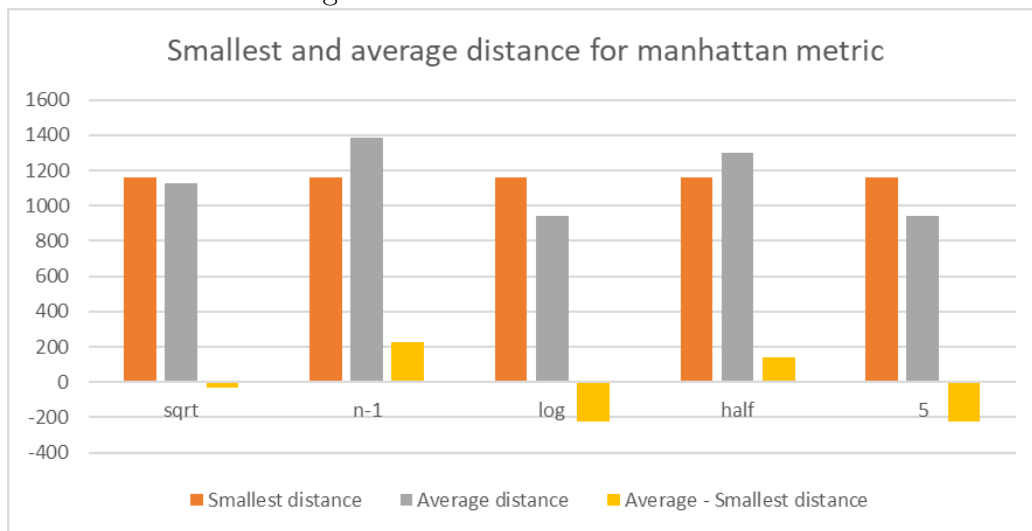


Figure 5: Distance for Euclidean

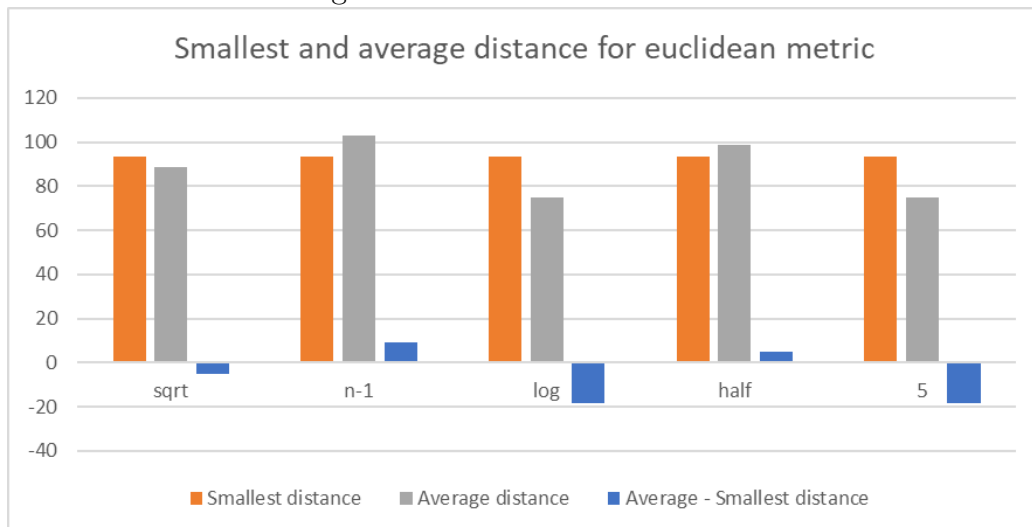




Figure 6: Distance for Cosine

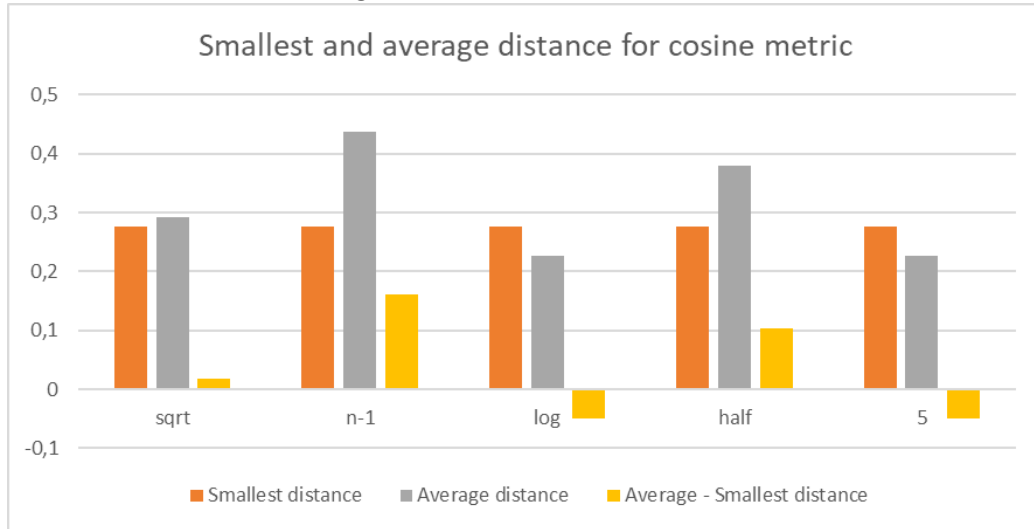
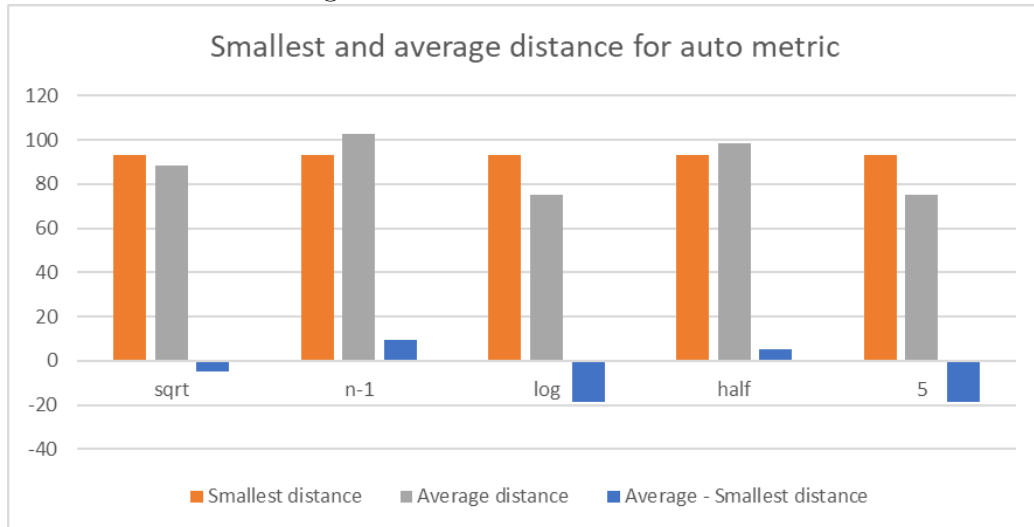


Figure 7: Distance for Auto mode



## 5.3 Discussion

### 5.3.1 Precision

As expected number of neighbors does not have influence on precision, metric used on different hand does, as we can see cosine metric achieves the best precision compared to cosine and euclidean metric.

This means that the anime that is recommended from a model that metric usually has higher rating than anime's that use euclidean or Manhattan metric.

Auto mode seems to use euclidean metric as the results between those two are exactly the same

### 5.3.2 Distance

Smallest distance was always the same within the same metric no matter the number of neighbors used

The only thing that changed was average distance and consequently difference between average distance and smallest distance

In general smaller number of neighbors used meant smaller average distance, notice how for neighbors equal to 5 and logarithm of number of entries average distance in all charts is roughly the same and below average distance for number of neighbours equal to square root of entries, half of entries or all entries but one

We think that it means that simply by taking more data into consideration when recommending anime we increase number of outliers that are radically "far" from input anime

For some reason sometimes the average distance was actually smaller than smallest distance, we did manage to understand why within time limit of this project

## 6 Challenges

### 6.1 Challenges themselves

**Precision** The biggest challenge was implementing the algorithm that checks precision of our model, we had to reject recall and F1 score since defining

what sort of base should they use to measure correctness of solution was not a trivial task

**Data Size** Another challenge was with the size of database itself which impacted speed of running new changes

**Visualisation** It was not easy to visualize the effects of experiments themselves, most program output is simple text and there is not much continues data that naturally aligns itself with some graphs

## 6.2 Tackling challenges

For precision we just settled on the idea of good recommendation to be any recommendation that recommends anime with rating above or equal 8

For Data size we introduced data limit argument which made it very easy and fast to introduce new changes

For visualization we agreed on going through the precision of algorithm based on different thresholds and showing data for distances in the histogram chart

## 7 Conclusions

**Best parameters** For our precision metrics we lean towards brute algorithm with cosine metric used, probably with logarithmic or 5 number of neighbors

Cosine metric provides us with the best precision and small number of neighbors offers smaller average distance, it also behaves as expected since the average distance is smaller than the distance between input anime and recommended anime

### 7.1 Solution satisfaction

Our solution works, it manages to go through entire data-set and return some recommendation

Those recommendation based on manual inspections are usually OK, model often behaves rationally for example by recommending sequel of anime when given anime's first part

We also manage to make basic precision algorithm which considering how

hard it is to define what a "correct" recommendation is, should be considered a success

Overall we are content with the result given limited time, knowledge and resources on our disposal

## **7.2 Potential improvements**

We did not manage to introduce more parameters when embedding data into vectors, like anime popularity or how controversial it is, this would probably make the model at least more interesting if not directly better