Machine Learning Engineer Nanodegree

Capstone Project

Dominik Lindenberger
November, 2018

I. Definition

Project Overview

The goal of this project is to build a recommender system for climbing routes within a given climbing area. The system will take a climber's preference into consideration.

My Capstone Project is located in the world of **rock climbing**.

> *Rock climbing is an activity in which participants climb up, down or across natural rock formations or artificial rock walls. The goal is to reach the summit of a formation or the endpoint of a pre-defined route without falling.* [1]

Here is a video of a rock climber on a famous route called Action Directe.

In recent years rock climbing has gained a lot of traction and is becoming more popular as an outdoor sports for a larger number of people. As such, there exists a potentially large audience who could be interested in this project.

Unlike the climber from the video above most people do rock climbing as a leisure activity. Therefore, time they can spend to go climbing is limited. As a hobby climber planning a climbing trip it is quite common to have a time window of round about 1-3 weeks at the climbing destination of choice. The majority of climbing areas boast a large number of routes to climb, far greater than anyone can achieve within a single trip. For example, the climbing area 'Frankenjura', one of Germany's largest climbing area, contains more than 10,000 climbs. It is therefore in the interest of the climber to know and attempt those climbs that are most enjoyable to her or him and just skip the rest.

Often times there is a common agreement on what are the 'best' routes in a certain area and they are common knowledge. Most guide books contain some sort of list of the "Top climbs" in that region.
However, climbers often differ in their personal preferences. Some prefer to climb on small crimpy holds, while others enjoy long dynamic movements on large holds.
Additionally, for some type of rock, the climbs deteriorate over time as heavy traffic 'polishes' holds with formerly good and important friction.

Therefore, climbers would benefit from a personal recommendation of the best climbs within a climbing area,

that match their climbing level as well as their personal preference.

Meanwhile there are many websites where climbers can track their climbs and ascents in a virtual logbook and also rate the quality of the route. Most famous websites of this kind are

- 8a.nu
- theCrag.com
- UKClimbing.com

With this project, I will explore the database of 8a.nu - published on Kaggle - to establish a personal recommendation system for climbers venturing into new climbing areas.

There has been quite a lot of academic research on the topic of recommender systems as well. Here is a short excerpt of publications:

- Su and Khoshgoftaar provide a good introduction into CF in their article A Survey of Collaborative Filtering Techniques
- In their 2018 paper Christakopoulou and Karypis provide an overview of Local Latent Space Models for Top-N Recommendation

As I am an avid climber myself, this project is something I take a strong personal interest in.

Problem Statement

The structure of the problem is that of a **Ranking problem** or more precise a **Collaborative Filtering problem**, where we try to suggest climbs based on a climber's similarity to other climbers.

We will try to measure similarity based on ratings given to a climbed route. Climbers who climbed the same routes and gave those routes similar ratings, might also agree in their opinion on routes they haven't climbed, yet.

So, we'll try to cluster climbers that climbed similar routes and that gave similar ratings to those routes using K-Means. This is an *Unsupervised Learning* technique.

Once we have identified clusters of climbers with similar taste, we'll determine a cluster's most recommended routes based on ratings and number of ratings. This is a **user based collaborative filtering** technique.

In order to give a climber a personal recommendation, we have to determine to which cluster he or she belongs and recommend the **top n routes** from the cluster's top recommended routes that he or she has not climbed yet.

Our anticipated solution will be:
For any given climber, we can provide a personal recommendation of *n* routes that he or she has not climbed before.

The following tasks should get us there:

1. Consolidation of climbing routes (major cleanup/prep work, because users make typos in route names

which result in duplicate routes in the data set)
2. Match climbers' recommendations with routes
3. Determine cluster size using *Silhouette Score*
4. Cluster climbers based on recommendations using *K-Means*.
5. Calculate *Weighted Rating* of every route for climber's cluster to receive an absolute ranking of all routes.
6. Recommend top *n* routes that climber has not climbed before.

Metrics

Basically, our problem is a ranking problem. We want to show the most attractive and suitable climbs to a user. However, we'll frame it as a rating prediction problem.
Using different models we will predict ratings of so far unrated routes. For this purpose we will use **RMSE** (Root Mean Squared Error) as our evaluation metric.

The project model as well as all of the benchmark models can be evaluated using RMSE, thus making this metric the favorable choice.

II. Analysis

Data Exploration

The data set for this project was downloaded from Kaggle at
8anu climbing logbook

Here is an overview of the data contained in the data set.

Column	Description	Datatype	Use
`crag_country`	3-letter country code of the country where the crag is located.	categorical	no
`crag_id`	Unique id for identifying each crag.	int	no
`crag`	Name of the crag	text	no
`sector_id`	Unique id for identifying each sector.	int	yes
`sector`	Name of the sector. A sector is a specific area within a crag.	text	yes
`route`	Name of the route the climber has climbed.	text	yes
`grade_id`	Unique id for identifying each climbing grade.	int	statistics
`grade`	Climbing grade given to that route as per the French grading system	categorical	statistics

`year`	Year the route was climbed	int	statistics
`date`	Date the route was climbed. The date format is number of seconds since 1970-01-01.	int	statistics
`method_id`	Unique id for identifying each type of ascent.	int	statistics
`method`	The type of ascent the climber made on that route.	categorical	statistics
`notes`	Additional information the climber provided for this climb, e.g. Soft graded, i.e. fairly easy climb for the given grade	categorical	statistics
`raw_notes`	Encoding of different notes and combination of notes	int	statistics
`rating`	Rating given to the climb by this climber. This is our target attribute.	int	target
`user_id`	Unique id for this climber.	int	yes
`user_country`	3-letter country code of the country where this climber is from.	categorical	statistics
`user_city`	City where this climber is from	text	statistics
`sex`	The climber's sex. 0 indicates male, 1 indicates female.	int	statistics
`height`	The climber's height in cm	int	statistics
`weight`	The climber's weight in kg	int	statistics
`birthdate`	The climber's date of birth	date	statistics
`started_climbing`	The year the climber started climbing.	int	statistics

In the table above the column `Use` indicated how we plan to use the given column.

- `target` - this is a target attribute.
- `yes` - this column will be used during exploration and analysis.
- `no` - not planned to use that column during analysis and exploration.
- `statistics` - this column will not be used for analysis but may be interesting later on to do user statistics etc.

Our dataset contains **114,587 logged climbs**. 60,303 out of these contain a rating. That is approx. 52.63%.

Missing values

Let us take a look if we have **missing values** or **zero values**.

Number of missing values per column:

```
id           0
sector_id    0
sector       0
route        0
rating       0
user_id      0
dtype: int64
```

This means there are no entries missing in the columns that we look at.

Number of zero values per column:

```
id               0
sector_id    12301
sector           0
route            0
rating       48291
user_id          0
dtype: int64
```

However, there a quite a significant amount of sector_IDs missing and also a lot of ratings.

Unique values

A naive calculation on the number of unique sectors and routes within these sectors. Submitted by a number of distinct users.

```
sector_id      350
route        12471
user_id       3292
dtype: int64
```

Why do we call the above estimate *naive*?

According to climb-europe.com *"there are approximately 1,000 crags spread out in a beautiful forest terrain"* (Note that *crags* in the above quote is the same as *sectors* in our dataset.) This seems fine since in our dataset there are 351 different sectors noted.

However, let's look closer here. What are the records where `sector_id` is 0?
We take a look at a number of samples.

	id	sector_id	sector	route	rating	user_id
22531	1366698	0		Gelegenheit macht Diebe	3	9660
72926	3458547	0	Gumpenwand	Zollfrei	0	16195
94967	4322459	0		Doppelpack	2	9493
101666	4487349	0	Hintere Reibertsbergwand	Schnecke	0	20095
75222	3511955	0		Glaspatronenmatch	3	9255
32469	1921878	0		Rayando el sol	0	11628
71909	3442794	0	Vogelherdwand	Rechts außen	0	10264
45119	2463848	0		Wo Ist Dein Paradies?	3	3166
54809	2869412	0		Ninja Turtle	0	18285
62462	3070359	0	Mosenberger Block	Görglesweg	0	37450

So, what we can see is, that `sector_id == 0` actually does not belong to a proper sector, but rather denotes that sector is not known in this case. Later we'll have to remove those entries.

In the article it continues to claim that *Frankenjura boasts in excess of 10,000 routes.* Now this is where we should get a little suspicious. In only 350 sectors our dataset apparently contains already more than 12,000 routes - which is well above the 10,000 mentioned in the article.

Lets dig deeper here. Maybe there are a number of duplicates in the registered routes. Let's see if an example can confirm this.

```
df[df['route'].str.startswith('knack')].drop_duplicates(subset=['route'])['route']
```

Output:

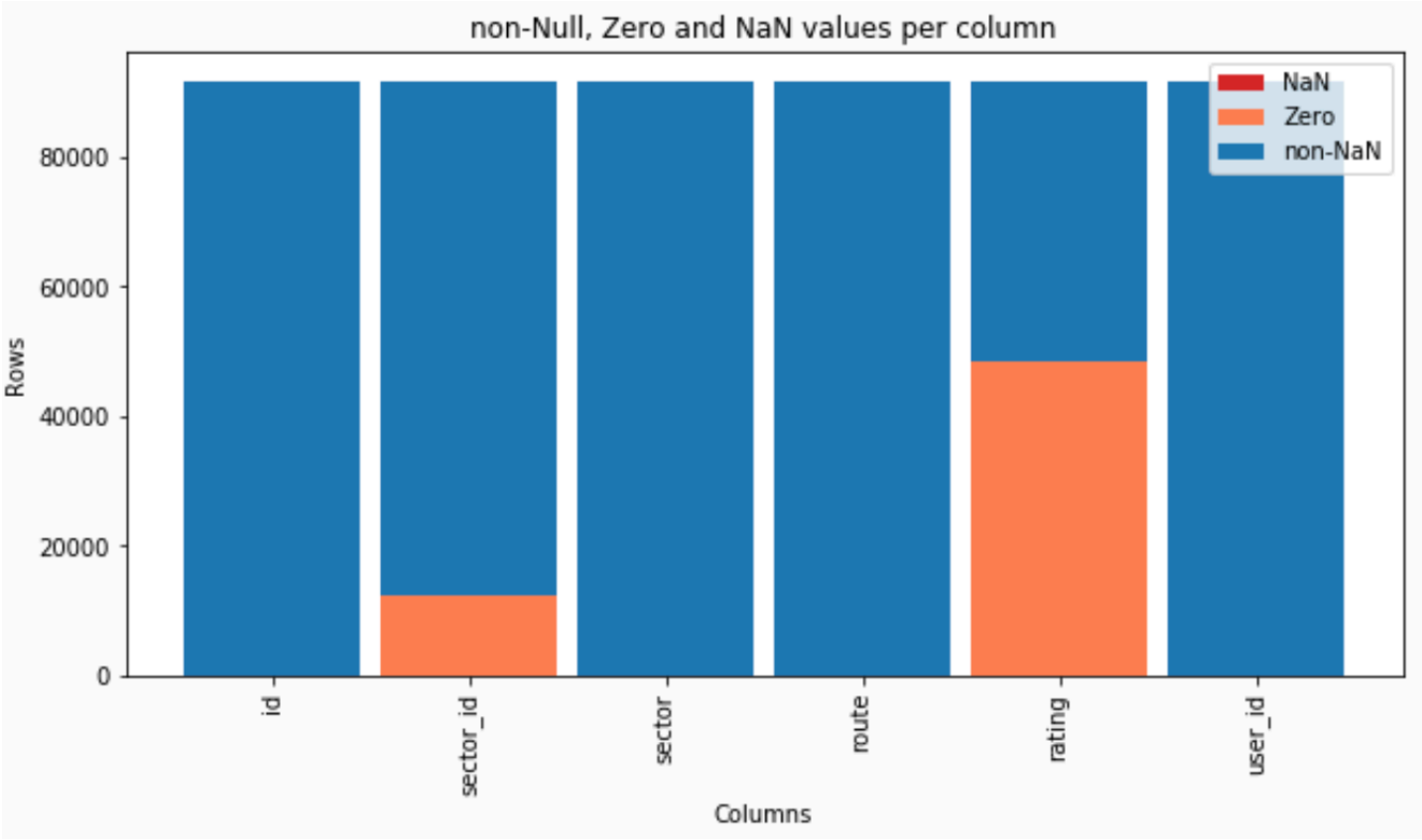
```
108084       knack  back
101251    knack und back
38966     knack and back
Name: route, dtype: object
```

We will have to consolidate those duplicates during Preprocessing.

Exploratory Visualization

Missing values

Lets visualize the **Missing** and **Zero values** per column.

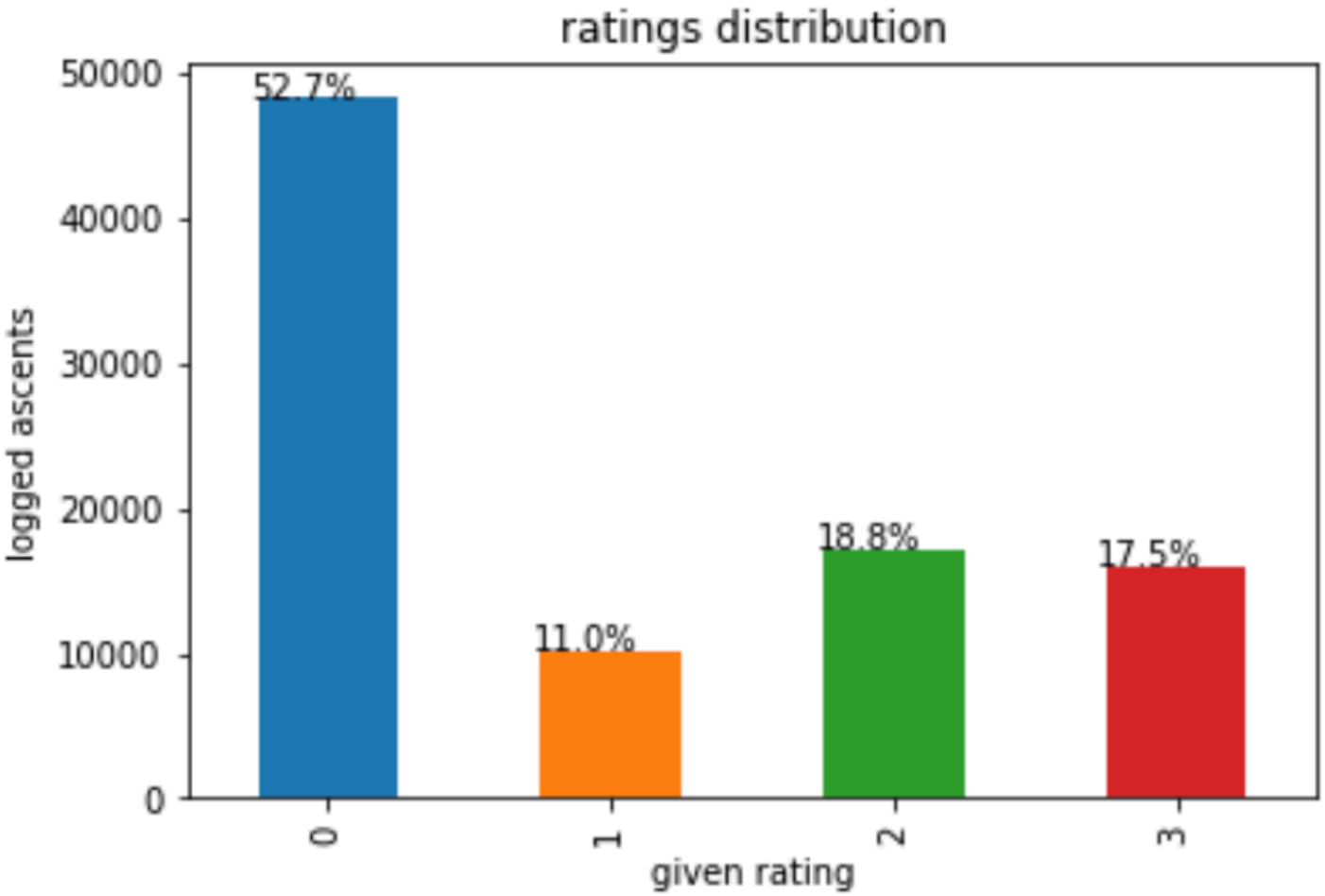


In our data set `0` is in indicator for missing values, except for `sex` where `0` indicates `male` (and `1` for `female`).

From the graph above we can conclude that we have missing data in `sector_id` and `rating`.

Since `sector_id` and `rating` are important for our analysis, we have to consider what to do about those missing values during Data Preparation later on.

Analysis of our target variable `rating`



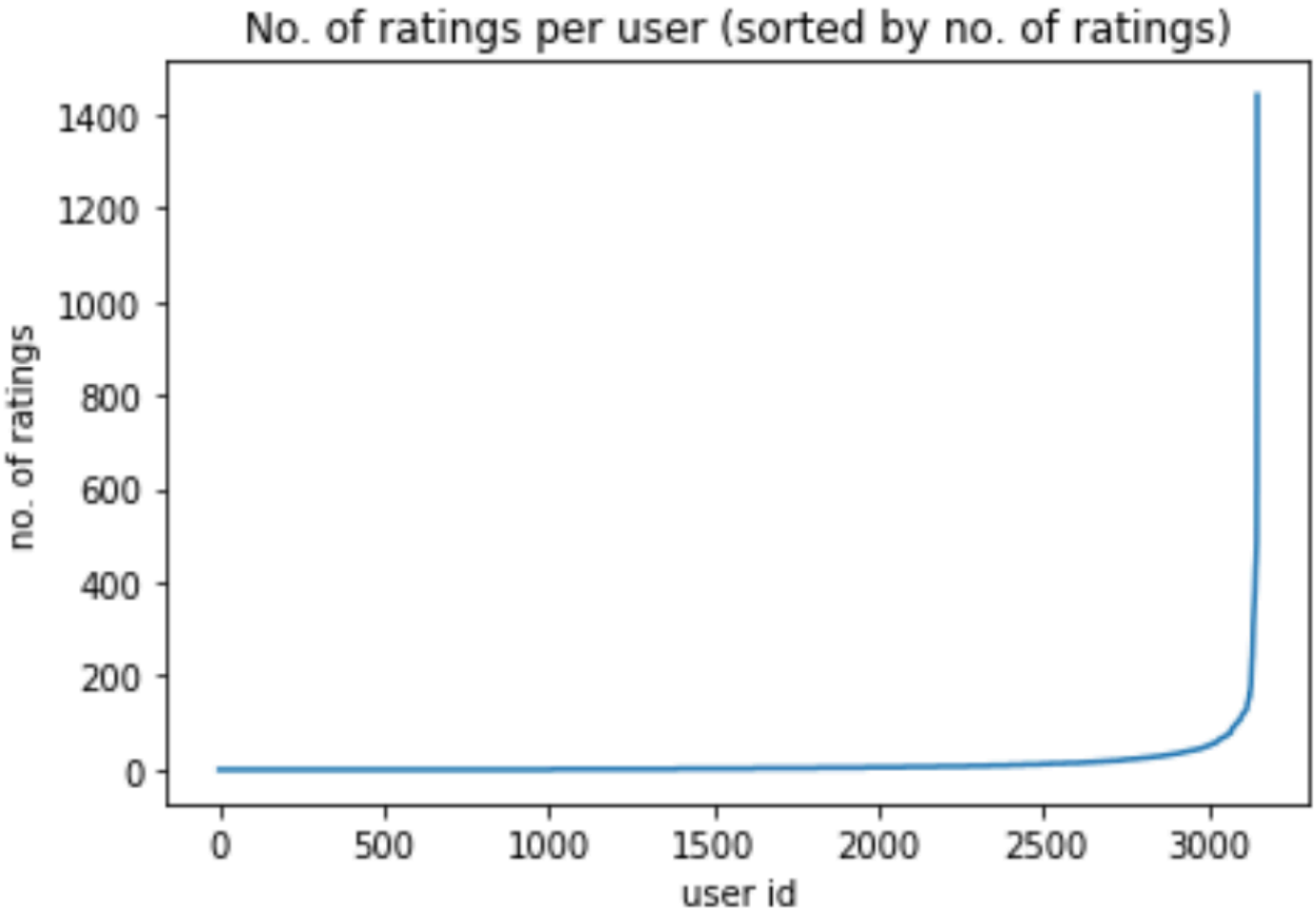
From the distribution of the target variable `rating`, we can see that there are four distinct values. With 52.7% of ratings equal to 0, only about half of the ascents have been rated.

In the climbing community it is commonplace to rate only good climbs by marking them with a star. Thus, the values `1`, `2` or `3` correspond to one, two or three stars. Where one star is a good route and three stars is an exceptionally good climb.

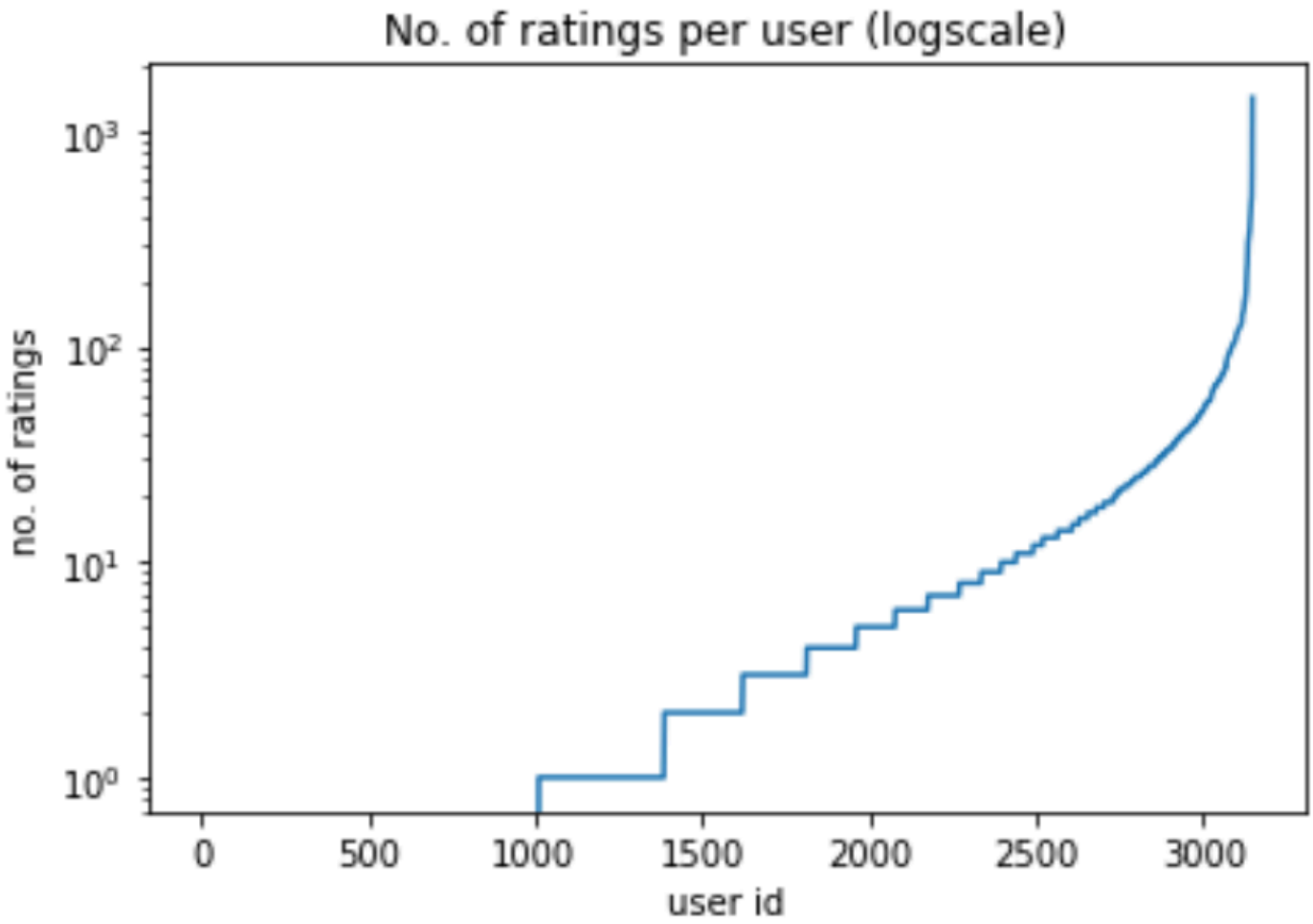
Hence the value `0`, or zero stars, denotes that a climb was either not rated or was not worth a star according to that user.

Approximately 2/3 of the users have rated at least one item, i.e. gave a rating of 1, 2 or 3.

Let's take a look at how many ratings users typically gave.



From the above plot we can see that there must be one or several outliers with a large number of ratings (around 1600). Let's apply the logarithm on the y-axis to get a better picture.



Now, this has more information. We can see that approx. 30% of users gave no rating at all. And more than 99% of users gave less than 200 ratings.

Let's look at the percentiles information in detail to confirm this.

```
count    3292.000000
mean       13.177400
std        49.680102
min         0.000000
25%         0.000000
33%         1.000000
50%         2.000000
75%         9.000000
90%        29.000000
95%        53.000000
99%       153.270000
max      1665.000000
Name: rating, dtype: float64
```

Algorithms and Techniques

Our problem is a *Collaborative Filtering* problem. As such, this problem lends itself to exploring data with **K-**

Means algorithm and try to find natural clusters of climbers. Our main parameter for K-Means is the number of clusters.

The *Silhouette Score* is a suitable way to find out the number of clusters we should aim for using K-Means. Through K-Means we hope to achieve a clustering of climbers based on their similarity. Climbers within the same cluster should then have a high probability of giving high ratings to the same routes.

Benchmark

As a benchmark we'll use a **random recommender**, i.e. for a given user, any route will be given a random rating from 1, 2 or 3 stars. This will be our quantitative measurement to compare against. Results for this benchmark (RMSE) will be obtained during implementation as data pre-processing is required before.
For a **qualitative benchmark**, we'll compare results to the recommendations by climbing magazine *klettern.de*.

III. Methodology

Data Preprocessing

As a first preprocessing step we'll remove all entries where `sector_id == 0`. We have established earlier, that here the actual sector is not known, thus the entry is not usable.

Now we'll have to tackle route name consolidation. As identified earlier during *Data Exploration* there are many duplicate routes. Since route names are per sector, we can reduce processing amount by consolidating route names within each sector.

The following steps will take us to a consolidated list of routes:

- Transform all route names to lowercase and remove special characters (`string_cleaning(df, columns)`)
- Find similar route names by applying *Jaro Winkler Distance* on the route names.
- Pick the most used route name as the route name of choice
- Do this for all sectors

Eventually, we only have to get unique values for all `sector_route` names to arrive at a list of unique routes. After the consolidation our new count for routes is **6754**, which is a lot less than what our naive count was and seems a lot more reasonable.

Luckily, getting a unique list of users is easy, by just applying `.drop_duplicates()` on `user_id`.

Finally, we'll also get a list of unique ratings, by excluding `rating == 0` values as this is the same as *not rated*.

Implementation

As outlined earlier, we want to apply K-Means to find clusters of similar users. The first step in that process is

pairing the list of routes with the list of user ratings using `pd.pivot_table(df_ratings, index='user_id', columns= 'sector_route', values='rating')`.

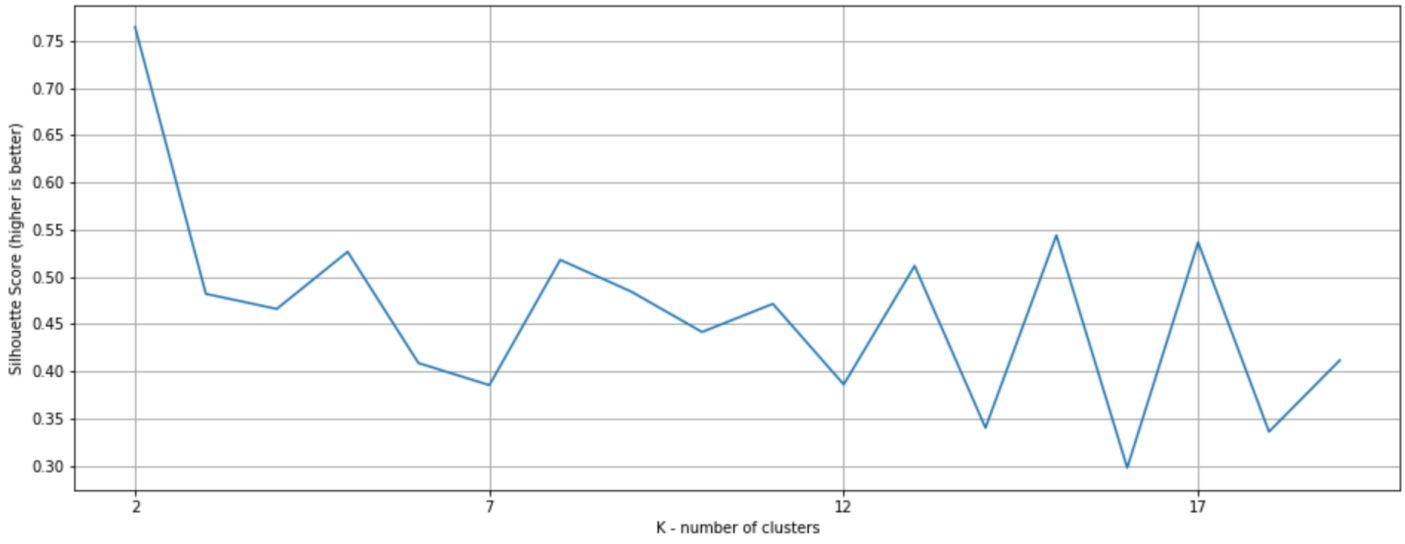
The results look like this:

sector_route user_id	AMPHITHEATER_angel dust	AMPHITHEATER_die dunkle seite	Affentheater_affenhitze	Affentheater_affent
15	NaN	NaN	NaN	N
18	NaN	NaN	NaN	N
32	NaN	NaN	NaN	N
38	NaN	NaN	NaN	N
79	NaN	NaN	NaN	N
88	NaN	NaN	NaN	N

The many `NaN` values are an indicator for how sparse this matrix is - which is typical for user rating matrices.

Before going into clustering, we have to transform our matrix into a real sparse matrix, using `to_coo()`.

Another useful precursor for clustering with K-Means is finding out how many clusters we should aim to find. The **Silhoutte score** will help us out here. The corresponding silhoutte score in our case looks as following:



According to the diagram the best number of clusters is 2. This is not really the answer we were hoping for. If there are only two clusters, then our assumption of grouping similar users into clusters seems not to hold as apparently there are not a lot of dividing characteristics among users. However, let's explore further before taking a decision.

Knowing the number of clusters, producing the clustering is a simple one liner

```
predictions = KMeans(n_clusters=2, algorithm='full').fit_predict(sparse_ratings)
```

As the result of K-Means' `fit_predict` is an `ndarray` we have to add this as a new column to our dataframe to make use of it.

```
clustered = pd.concat([user_route_ratings.reset_index(),
                       pd.DataFrame({'group': predictions})], axis=1)
```

Let's see how the actual numbers look like for number of users within each cluster.

```
1    2121
0      18
Name: group, dtype: int64
```

These numbers confirm our suspicion that the clustering has not brought the diversivication of users that we had hoped for.

Let's look at a visualization of the resulting clusters regardless.

```
cluster # 0
# of users in cluster: 18. # of users in plot: 18
```

The routes are sorted by most rated and best rated from left to right. Each square represents the rating of one user.

Prediction - First attempt

In order to get our **prediction** of recommended routes for a single user, we will do the following for each user:

1. Determine to which cluster a user belongs
2. Calculate average rating of each route within cluster and sort routes accordingly

3. Recommend the top n routes based on our calculated average that user has not climbed, yet.

1 - Cluster membership

The function `cluster_membership(clustered, user_id)` is a simple lookup in the `clustered` dataframe and yield the user's `cluster_id`.

2 - Average rating per route

To calculate a route's average rating within a cluster we'll use the following *Weighted Rating* formula:

$$\text{Weighted Rating (WR)} = \left(\frac{v}{v+m} \cdot R\right) + \left(\frac{m}{v+m} \cdot C\right)$$

where

- *v* is the number of ratings for that route
- *m* is the minimum ratings required to be listed in the chart
- *R* is the average rating of the route
- *C* is the mean vote across the whole report

This formula makes sure that the number of ratings is being considered and avoids the problem of a route with a single three star rating ending up at the top of our list.

3 - Top n routes per climber

The function `top_n_routes_cluster(cluster, n=0)` returns the top n routes of a given cluster sorted by the Weighted Rating formula described above.

The functiuon `top_n_routes_user(clustered, user_id, n)` uses the results of `top_n_routes_cluster(cluster, n=0)` to determine a user's top n routes, i.e. the routes a user has climbed already are excluded from this list.

Qualitative result check

For a qualitative result check, let's look at the top 20 recommended routes within a cluster and count how many of the routes show up in the Top 100 list of climbing magazine *klettern*. The list is available on the klettern.de website.

We retrieve the top 20 routes of cluster 0 using `top_n_routes_cluster(cluster, 20)`

And the result looks as following, sorted by weighted rating:

	sector_route	rating_average	rating_count	wr
3301	Rolandfels_die vollendung	2.869565	69.0	2.778928
2855	Neumühle_witchcraft	2.827160	81.0	2.752556
3214	Richard Wagner Fels_fight gravity	2.772455	167.0	2.737014
1408	Grüne Hölle_vgeln verboten	2.795181	83.0	2.725027
3448	Roter Fels_schaumschlger	2.803571	56.0	2.702773
2909	Obere Größweinsteiner Wände_sautanz	2.787879	66.0	2.702398
4728	Waldkopf_slimline	2.769231	78.0	2.697412
2243	Kuhkirchner Wand_primeur de luxe	2.870968	31.0	2.691729
2214	Krottenseer Turm_chasin the trane	2.718519	135.0	2.678164
3133	Püttlacher Wand_treibjagd	2.757576	66.0	2.675371
469	Bärenschlucht_rauchende bolts	2.843750	32.0	2.674436
450	Bärenschlucht_herkules	2.717742	124.0	2.674072
161	Ankatalwand_computerspiele	2.729167	96.0	2.672860
2017	Jungfernriss_sms	2.947368	19.0	2.665831
1859	Holzgauer Wand_nimue	2.718750	96.0	2.663245
2565	Marientaler Wände_stromlinie	2.739130	69.0	2.662045
1688	Herzwand_lwenherz	2.875000	24.0	2.655545
3217	Richard Wagner Fels_magnet	2.765957	47.0	2.654135
3108	Püttlacher Wand_das geschenk	2.800000	35.0	2.650638
2944	Obere Schlossbergwände_liebe ohne chance	2.663043	184.0	2.635299

We find 9 out of 20 recommended routes also in the Top 100 list of klettern.de. So, our recommendations seem to be going the right direction.

	Felsname	Routenname	Bewertung
1.	Wolfstein	Froschkönig	3-
2.	Hartelstein	Alte Talseite	3
3.	Förstelstein	Kleine Kante	3
4.	Breitenberg Südwand	Stier	3
5.	Weißenstein	Leon	3
6.	Türkenfels	Feierabend	3
7.	Bärnhofer Wand	Bärnhofer Kante	3+
8.	Lindenturm	Ostkante	4
9.	Leienfelser Pfeiler	Rotzlöffel	4
10.	Eibenwände	Hobbitkante	4+
11.	Jubiläumswand	Nico	4+
12.	Treunitzer Klettergarten	R5	4+
13.	Dreistaffelfels	Hungerrampe	4+
14.	Zehnerstein	Gerade Westwand	4+
15.	Hohe Reute	Willi Kapp Ged.-Weg	4+
16.	Vogler Ged.-Wand	Sellaerinnerungen	4+
17.	Langer Berg Wände	Westkante	4+
18.	Röthelfels	Zinnenwand	5-
19.	Haselstaudener Wände	Rechtsaußen	5-
20.	Weiße Wand	Daniel	5-
21.	Eibenwände	Schwarze Kante	5
22.	Napoleon	Westriss	5
23.	Jubiläumswand	Lurchi	5
24.	Hohe Reute	Südwand	5
25.	Hintere Stadelhofener Wände	Elliweg	5+
26.	Treunitzer Wand	Zeitelweg	5+
27.	Stierberger Gemsenwand	Weiße Verschneidung	5+
28.	Intensivstation	Direkter Wamperlweg	5+
29.	Rote Wand	Reinhard-Karl-Ged.-Weg	5+

	Felsname	Routenname	Bewertung
30.	Elfenwelt	Palast der Hohen	6-
31.	Haselstaudener Wände	Prinzessin	6-
32.	Dreistaffelfels	Bamberger Pfeiler	6-
33.	Glatzenstein	Purtscheller Ged. Weg	6-
34.	Freudenhaus	Venusfalle	6
35.	Hohe Reute	Zwischenprüfung	6
36.	Riffler	Vollrathriss	6
37.	Kleine Wacht	Vagabundenpfeiler	6
38.	Hartensteiner Wand	Die 101-Jährige	6
39.	Maximilianswand	Wand der Abendröte	6
40.	Hartelstein	Südpfeiler	6+
41.	Zehnerstein	Seifertriss	6+
42.	Schlosszwergwand	Alf	6+
43.	Treunitzer Klettergarten	Pfeilerweg (R7)	6+
44.	Bärnhofer Wand	Nürnberger Weg	6+
45.	Leupoldsteiner Wand	Vertigo	7-
46.	Roter Fels	Schaumschläger	7-
47.	Mittelbergwand	Dir. Kletterboumweg	7-
48.	Totensteinwände	Hohe Liebe	7-
49.	Gößweinsteiner Wände	Haringer Ged.-Weg	7-
50.	Rodenstein	Frankenschnellweg	7
51.	Röthelfels	Devil's Crack	7
52.	Ruine Leienfels	Leienfelser Pfeiler	7
53.	Bärnhofer Wand	Waldrausch	7
54.	Emporwand	Jericho	7
55.	Hintere Stadelhofener Wände	Strahlentod	7
56.	Schottersmühler Wand	Geier Sturzflug	7
57.	Ziegenfelder Wände	Basic Instinct	7
58.	Rote Wand	Knupper Ged.-Weg	7
59.	Reichelsmühler Wand	Aquaplaning	7+
60.	Kalte Wand	Schneekönigin	7+
61.	Dooser Wand	Siebter Sinn	7+
62.	Schöne Aussicht	Schöne Aussicht	8-
63.	Stadeltenne	Stadeltenne	8-
64.	Püttlacher Wand	Abseitsfalle	8-
65.	Bandstein	Herbstmanöver	8-
66.	Roter Fels	Hannig Mayer Ged.-Weg	8-

	Felsname	Routenname	Bewertung
67.	Ankatalwand	Computerspiele	8
68.	Kühllochfels	Reif für die Insel	8
69.	Heldwand	Heldbräu	8
70.	Trautner Ged.-Wand	Non Stop	8
71.	Weißenstein	Dampfhammer	8
72.	Rabenfels	Katalysator	8
73.	Rolandfels	Die Vollendung	8+
74.	Kühllochfels	Sommernachtstraum	9-
75.	Gößweinsteiner Wände	Sautanz	9-
76.	Püttlacher Wand	Treibjagd	9-
77.	Heldwand	Götz von B.	9-
78.	Stadeltenne	Maßarbeit	9-
79.	Marientaler Wände	Stromlinie	9
80.	Obere Schlossbergwand	Liebe ohne Chance	9
81.	Richard-Wagner-Fels	Magnet	9
82.	Krottenseer Turm	Chasin' the trane	9
83.	Grüne Hölle	Vögeln verboten	9+
84.	Neumühle	Witchcraft	10-
85.	Student	Simon	10-
86.	Solarium	Desperado	10-
87.	Bärenschluchtwände	Center Court	10-
88.	Holzgauer Wand	Nikita	10-
89.	Geheimbund	Mikrokosmos	10
90.	Zwergenschloss	Plan B	10
91.	Emporwand	Land of Confusion	10/10+
92.	Eldorado	Stonelove	10+
93.	Bärenschluchtwände	Drive by Shooting	10+/11-
94.	Krottenseer Turm	Wallstreet	11-
95.	Schneiderloch	Burn for You	11-
96.	Zwergenschloss	Powerplay	11-
97.	Schlaraffenland	Shangri-La	11-/11
98.	Waldkopf	Action Directe	11
99.	Schneiderloch	Corona	11/11+
100.	Planetarium	Supernova	11+

The suggested routes are all in the upper experienced to expert level. Which is no surprise, as we have seen earlier, that the majority of climbed routes is from that range.

We still need to tackle the problem of the few clusters. With only two clusters of users and more than 90% of users in one cluster, we can hardly provide personalised recommendations.

Alternative Implementation - SVD

Let's try a **Matrix Factorization based algorithm** for an alternative. The Surprise package implements **SVD**, an algorithm that became popular for winning the 1M$ Netflix prize.

We can use our cleaned ratings input also in combination with SVD. Training the SVD on our data is as simple as creating an algorithm object with `SVD()` and calling `cross_validate()` on our rating data.

```
svd = SVD()
_ = cross_validate(svd, data, measures=['RMSE'], cv=5, verbose=True)
```

The output is measured using the *Root Mean Squared Error*.

```
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.6527  0.6515  0.6457  0.6533  0.6519  0.6510  0.0027
Fit time          2.16    2.15    2.12    1.88    1.88    2.04    0.13
Test time         0.07    0.06    0.06    0.06    0.06    0.06    0.00
```

In our case we get a `RMSE = 0.6510` on a 5-fold cross validation run.

For example user `user_id = 5512` the top 10 recommended routes are

```
[('Bleisteine_offenbarung', 2.477913536224587),
 ('Neumühle_witchcraft', 2.4578274392648605),
 ('Weidlwanger Wand_the dance alone', 2.4490098743900375),
 ('Ankatalwand_computerspiele', 2.4468467744030757),
 ('Moritzer Turm_high gravity day', 2.431914905737715),
 ('Heldwand_gtz von b', 2.3965405132721855),
 ('Bärenschlucht_brentter', 2.3780055666037585),
 ('Student_simon', 2.3712190426118784),
 ('Kuhkirchner Wand_primeur de luxe', 2.3456612739659577),
 ('Roter Fels_schaumschlger', 2.3223257808653144)]
```

In our qualitative check against the klettern.de top list, we can find again 50% of the routes recommended by our personal recommender. This, confirms that also our new recommender suggests routes which are among the best in the area.

Refinement

In order to refine results of the SVD algorithm, we're executing a grid search on the parameters:

- `n_epochs` - The number of iteration of the SGD procedure.
- `lr_all` - The learning rate for all parameters.
- `reg_all` - The regularization term for all parameters.

The best results were achieved testing for RMSE by

```
0.6530371431723033
{'n_epochs': 15, 'lr_all': 0.007, 'reg_all': 0.2}
```

Re-running the model with updated parameters from grid search yields a slightly improved mean `RMSE = 0.6489` (previous result was `RMSE = 0.6510`.

```
Evaluating RMSE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.6480  0.6464  0.6568  0.6485  0.6450  0.6489  0.0041
Fit time          1.66    1.52    1.48    1.57    1.60    1.57    0.06
Test time         0.12    0.06    0.06    0.06    0.07    0.08    0.02
```

IV. Results

Model Evaluation and Validation

In our final solution we were able to give an individual recommendation of the top n routes for any individual user. This solves the problem as stated earlier.

Our final model uses `SVD` to produce recommendations based solely on user ratings. This approach was chosen as the previous attempt, using K-Means, failed to produce a significant cluster diversification (more than 99% of users were ended up in the same cluster). SVD has proven very useful in recommendation tasks, e.g. Netflix movie recommendation.

Assessing the robustness of our algorithm by comparing results of explicit vs implicit feedback.

The SVD algorithm considers only **explicit feedback**. I.e. if a user gives a rating of one star, then this is counted as a one star rating. However, the **implicit feedback** of a user is not considered. Implicit feedback means that a user, by chosing to rate a particular route he or she climbed (as opposed to not rating that route) already provides a positive indication for that route.

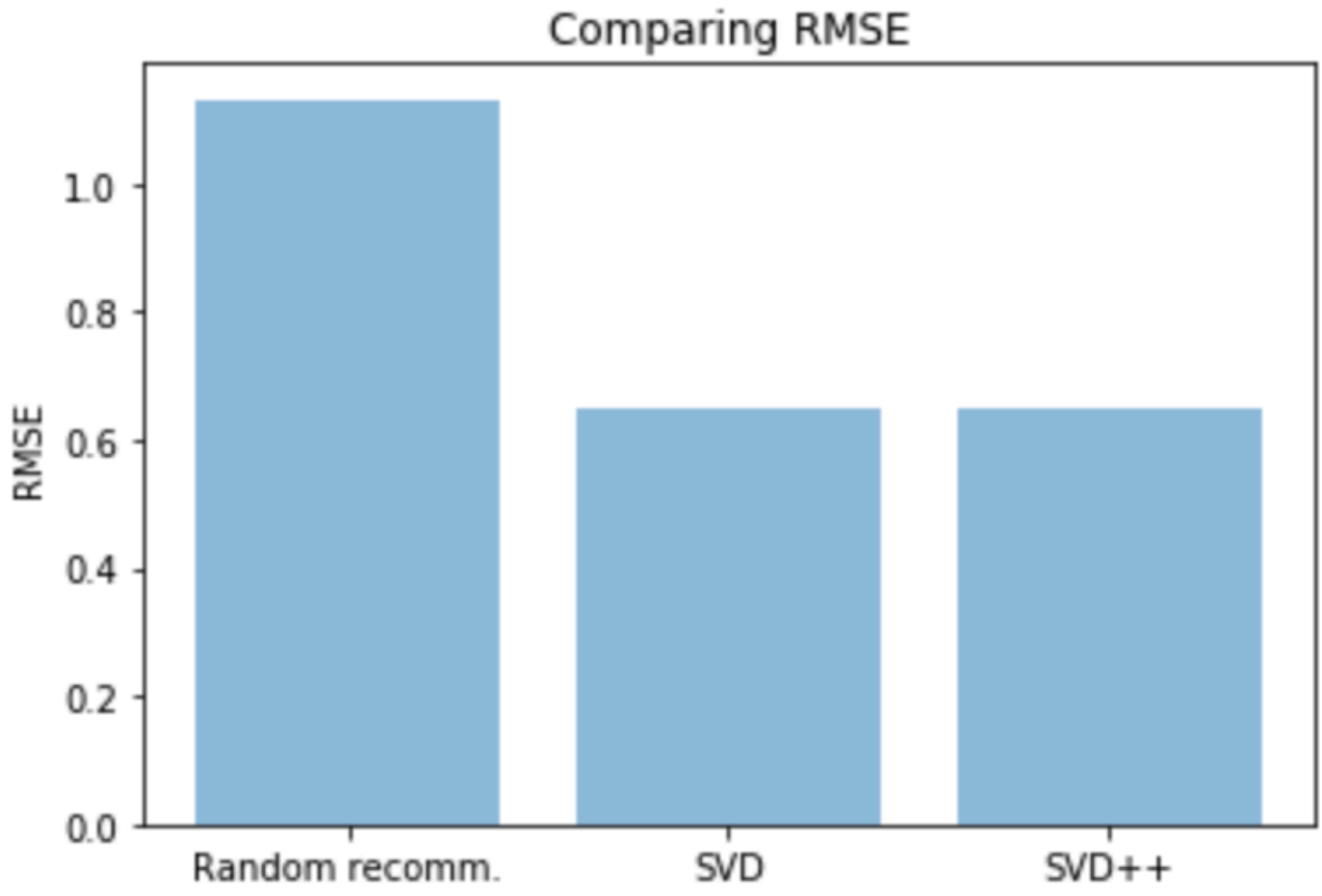
The **SVD++** algorithm can take that implicit feedback into account.

```
svd_pp = SVDpp(n_epochs=15, lr_all=0.007, reg_all=0.2)
_ = cross_validate(svd_pp, data, measures=['RMSE'], cv=5, verbose=True)
```

```
Evaluating RMSE of algorithm SVDpp on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    0.6486  0.6493  0.6507  0.6437  0.6451  0.6475  0.0026
Fit time          44.89   44.37   44.21   44.51   44.41   44.48   0.23
Test time         1.19    1.15    1.12    1.07    1.19    1.14    0.04
```

The following figure shows the similarity of RMSE of SVD vs SVD++. The random model is also part of the graph to put both algorithms into perspective.



Thus, we can see that there is no significant difference between implicit and explicit feedback.

Justification

The comparison of our solution to the initial benchmark:

Model	Random model	SVD	SVD tuned
RMSE	1.1321	0.6510	0.6489

We can see a significant improvement of *RMSE* from Random model over the SVD to the tuned SVD model (smaller RMSE is better), although the tuning did not gain much.

Thus, we have **solved the problem** of **personalised recommendations** for climbers.

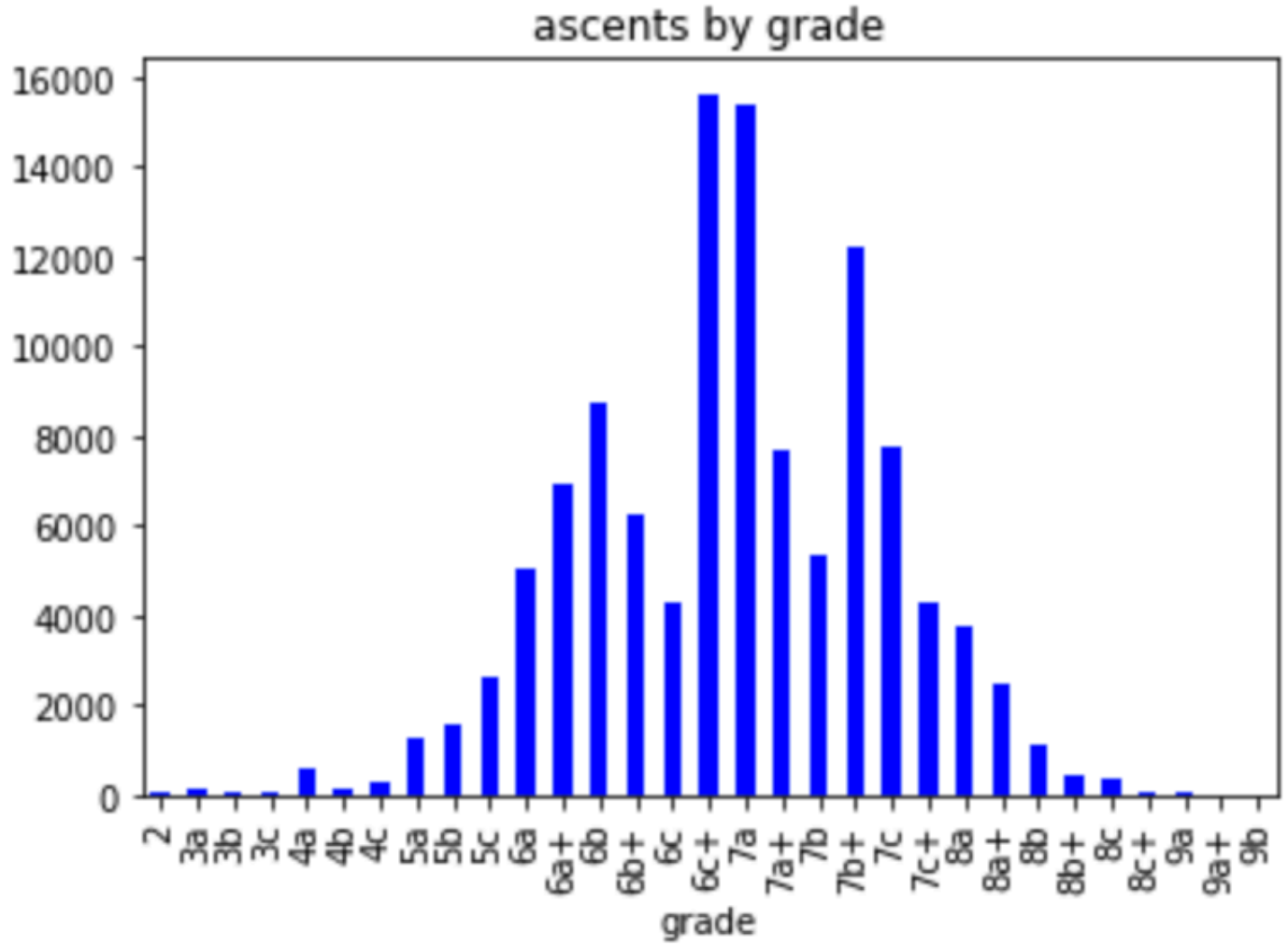
V. Conclusion

Free-Form Visualization

One interesting observation about the input data is the following.

Let's look at the distribution of routes by climbing grade (= difficulty level of the route) to understand whether our

climbers are mostly beginners, advanced or professionals.



Our distribution seems to have the general form of a bell shape, i.e. a Gaussian distribution. What seems odd though, are the two dents in the distribution at 6c and 7b.

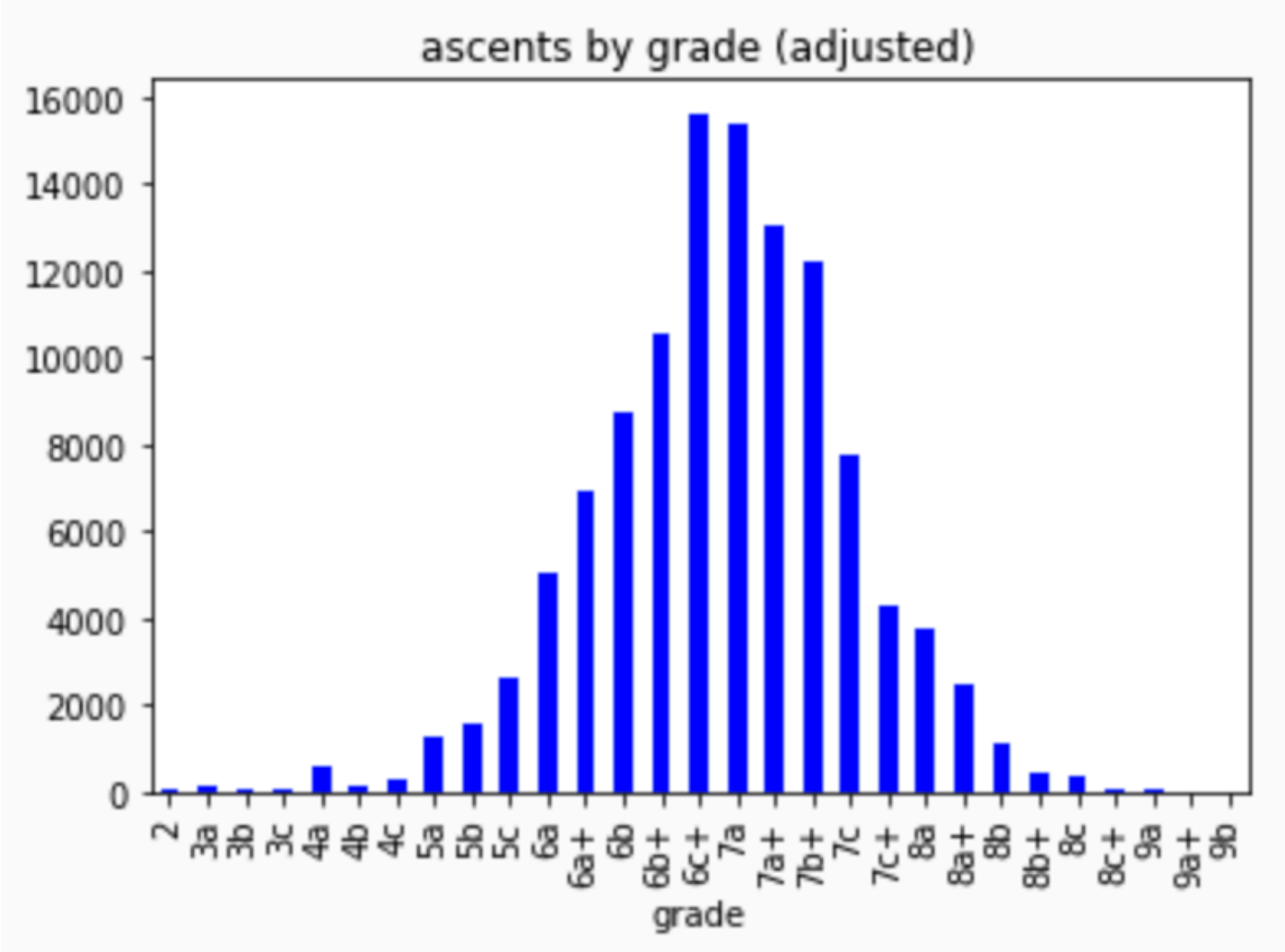
For this to understand we need to have some background knowledge on the different climbing grading systems used in different countries. The grading system used by 8a.nu, which is the source of our data set, is the *French* grading system. In Germany a different grading system is used called *UIAA*.

One peculiar thing about different grading systems is, that they do not follow the same step size. E.g. in French scale grading the step from one grade to the next higher grade could be smaller than the steps in between grades in UIAA. This can be seen when looking at a grade comparison chart as shown below.

YDS (USA)	British		French	UIAA	Saxon	Ewbank (AUS - NZL)	Ewbank South Africa	Nordic		Brazil
	Tech	Adj						Finnish	SWE/NOR	
3-4	1	M	1	I	I	1-2	1-2	1	1	I
5.0						3-4	3-4			I sup
5.1	2		2	II	II	5-6	5-6	2	2	II
5.2		D				7-8	7-8			II sup
5.3	3		3	III	III	8-9	8-9	3	3	
5.4		VD	4a	IV	IV	10-11	10-11	4	4	III
5.5	4a	S	4b	IV+	V	11-12	11-12	5−	5−	III sup
5.6	4b	HS	4c	V	VI	13	13	5	5	IV
5.7	4c	VS	5a	V+		14-15	14-15			
5.8		HVS	5b	VI-	VIIa	15-16	16	5+	5+	IV sup
5.9	5a		5c	VI	VIIb	17	17-18		6−	V
5.10a		E1	6a	VI+	VIIc	18	19	6−		VI
5.10b	5b		6a+	VII-		19	20		6	
5.10c		E2	6b	VII	VIIIa	20	21	6	6+	VI sup
5.10d	5c		6b+	VII+	VIIIb		22		7-	
5.11a		E3	6c 6c+		VIIIc	21		6+		7a
5.11b				VIII-		22	23		7	7b
5.11c	6a	E4			IXa	23	24	7−	7+	7c
5.11d			7a	VIII	IXb	24	25	7		
5.12a		E5	7a+	VIII+	IXc	25	26	7+	8−	8a
5.12b			7b			26	27	8−		8b
5.12c	6b	E6	7b+	IX−	Xa	27	28	8	8	8c
5.12d			7c	IX	Xb	28	29	8+		9a
5.13a		E7	7c+	IX+	Xc	29	30	9−	8+	9b
5.13b	6c		8a				31	9	9-	9c
5.13c		E8	8a+	X−	XIa	30	32	9+		10a
5.13d		E9	8b	X	XIb	31	33	10−	9	10b
5.14a	7a	E10	8b+	X+	XIc	32	34	10		10c
5.14b			8c			33	35	10+	9+	11a
5.14c	7b	E11	8c+	XI−		34	36	11−		11b
5.14d			9a	XI		35	37	11		11c
5.15a			9a+	XI+		36	38			12a
5.15b			9b	XI+/XII−		37	39			12b
5.15c			9b+	XII-		38	40			12c

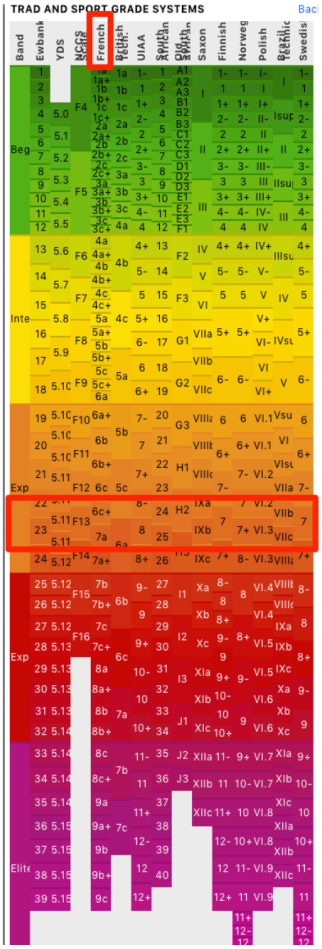
All the route in Germany are given grades following UIAA scale. If a climber wants to log a climb in Germany with a UIAA grade on 8a.nu, he has to convert the grade to French scale. As there are no real corresponding entries for 6c and 7b in the UIAA scale, this explains the dent in our distribution.

Therefore, joining the numbers from 6c and 6b+, as well as 7a+ and 7b seems reasonable.



And the result of the distribution of logged climbs is now as expected.

With the majority of climbs around 6c+, 7a level, what does this actually mean? Let's take a look at the assessment given by the website thecrag.com. And we can see that the majority of logged ascents are in the upper *Experienced* range.



Beginner	The level of difficulty for your first couple of days of climbing (seconding or top roping). Many people may achieve these grades on their first day of climbing. There are still some very scary and/or dangerous climbs at this level (eg Bard).
Intermediate	Typical grades for people with less than a years climbing. Note that most people cannot climb this level on their first couple of days of climbing. Statistically speaking, most outdoor climbing is done in this band. A lot of experienced climbers end up backing off to this level as they enjoy their climbing into old age.
Experienced	People can achieve these levels if they have been climbing fairly regularly for a couple of years. These grades are where the social climbers start becoming rare.
Expert	You really need to be training in a focused way to climb at this level. Not so many people reach this level.
Elite	You climb for a living, are sponsored and have a full time trainer and masseur looking after you. This is the best of the best. If you are normal, you would not have a hope of even dogging up one of these climbs.

Reflection

The entire process of finding a solution to the initially outlined problem started with extraction of data from the source data base. After an analysis of the data at hand to discover missing values, irregularities, etc. within the data, corresponding data preparation steps were decided.

The data preparation comprised dealing with missing data and consolidation (cleaning) of route names as there were many duplicates in the data. Cleaning up the route names took considerably longer than I had anticipated. It reminded me of the statement that 80% of time usually is spent on data cleaning, when data 'from the wild' is being used insteat of clean datasets.

Our initial idea of using K-Means to form clusters of similar users did not work out well. Using the Silhoutte Score we determined that the best number of clusters would be k=2. After running the clustering algorithm it became further clear that our data was not suitable for K-Means as more than 90% of users ended up in one of the two clusters. Hence, this was no basis for creating personalized recommendations.

To the rescue came SVD, an algorithm that has proven its usability already on the Netflix movie recommender challenge.

Using SVD we greatly improved over our benchmark, the random recommender model and thus solved our problem. We compared also what would happen if we considered implicit feedback as well and found that there was no real change in our recommendation output.

Improvement

A practical improvement to the current solution could be, that a user has the ability to restrict the recommended routes to different constraints, such as specific country, specific climbing area, specific crag. This should be rather easy to implement.

Potentially better predictions might be produced if the algorithm would consider additional user details (height, weight, years climbing experience, ...) instead of being based on only user ratings.

---

1. What is Rock Climbing? ↩