MEF UNIVERSITY

LOCATION BASED RECOMMENDATION SYSTEM USING FLICKR GEOTAGGED PHOTOS

Capstone Project

Erbil Çakar

İSTANBUL, 2018

MEF UNIVERSITY

LOCATION BASED RECOMMENDATION SYSTEM USING FLICKR GEOTAGGED PHOTOS

Capstone Project

Erbil Çakar

Advisor: Assoc. Prof. Şuayb Arslan

İSTANBUL, 2018

MEF UNIVERSITY

Name of the project: Location Based Recommendation System Using Flickr Geotagged Photos

Name/Last Name of the Student: Erbil Çakar

Date of Thesis Defense: 09/09/2018

I hereby state that the graduation project prepared by Erbil Çakar has been completed under my supervision. I accept this work as a “Graduation Project”.

09/09/2018
Assoc. Prof. Şuayb Arslan

I hereby state that I have examined this graduation project by Erbil Çakar which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

09/09/2018

Director
of
Big Data Analytics Program

We hereby state that we have held the graduation examination of Erbil Çakar and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Assoc. Prof. Şuayb Arslan	………………………..
2. Prof. Dr. Özgür Özlük	………………………..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

Name	Date	Signature
Erbil Çakar	09/09/2018	

EXECUTIVE SUMMARY

LOCATION BASED RECOMMENDATION SYSTEM USING FLICKR GEOTAGGED PHOTOS

Erbil Çakar

Advisor: Assoc. Prof. Şuayb Arslan

SEPTEMBER, 2018, 27 pages

Flickr is a worldwide photo sharing application. In this project, photos with location information provided by Flickr will be used. By utilizing this location information, a recommendation system that recommends users the most suitable photo shooting locations based on their historical data will be designed. During this system design phase, different methodologies will be tried and a research will be conducted on the methods that provide the most appropriate solution to the problem.

Key Words: Exploratory data analysis, recommendation system, geotagged photos

ÖZET

FLICKR'IN COĞRAFİ ETİKETLENMİŞ FOTOĞRAFLARINI KULLANARAK KONUM BAZLI TAVSİYE SİSTEMİ

Erbil Çakar

Tez Danışmanı: Asst. Prof. Şuayb Arslan

EYLÜL, 2018, 27 sayfa

Flickr, dünya çapında bir fotoğraf paylaşım uygulamasıdır. Bu projede Flickr tarafından sağlanan konum bilgisine sahip fotoğraflar kullanılacaktır. Bu konum bilgisini kullanarak, kullanıcıların tarihsel verilerine dayanarak en uygun fotoğraf çekim yerlerini öneren bir tavsiye sistemi tasarlanacaktır. Bu sistem tasarım aşamasında, farklı metodolojiler denenecek ve soruna en uygun çözümü sağlayan yöntemi bulmak amacıyla araştırma yürütülecektir.

Anahtar Kelimeler: Keşifsel veri analizi, tavsiye sistemi, coğrafi etiketlenmiş fotoğraflar

TABLE OF CONTENTS

.

1. INTRODUCTION

Flickr is a worldwide photo sharing application. There are tens of billions of photos and 2 million groups on the Flickr system today (based on the extracted information at https://www.flickr.com/). With this application, people can share their photos, add location information and attach tags they like. Location information is obtained using the GPS module of the mobile device that shoots the photographs. In this project, only photos with location information is used. By using this location information, a recommendation system that recommends users the most suitable photo shooting locations based on their historical data is designed.

Recommendation systems are used in many areas today. The goal of these systems is predicting the points or preferences a user will give to an item. In general, recommendation systems are divided into three classes as methods [1]:

- Simple recommenders: The system does not recommend based on personal choices, it makes generalized recommendations to every user. One of the best-known examples of this method is the "IMDB Top 250" movie list (https://www.imdb.com/chart/top)
- Content-based recommenders: It is a method based on finding similarities between items. Similar items are recommended to the user. To find similarities between items, the contents (or metadata) of the items are utilized.
- Collaborative filtering engines: These systems use other users' data to determine user's preferences.

Collaborative Filtering method will be used in this project. This method can be divided into two sub-categories in itself:

- User-based filtering: These systems recommend items to users that have liked or purchased similar things. For example, users A and B are using the same items. If person A uses a new item that B does not use, this new item will be recommended to person B.
- Item-based filtering: These systems identify similar items based on how users have rated or used them in the past. Suppose that 5 people use X and Y items. The

system will determine X and Y as similar items. In this way, users who use X item will be recommended Y item by the system.

Collaborative Filtering methods use an item-item matrix in the background where similarities between all items are calculated. Different mathematical methods are used in these similarity calculations. The most common of these are the following methods:

- Cosine similarity: Assuming that A and B items are represented as a vector, the similarity is calculated by the cosine of the angle between these two vectors. The smaller the angle between the two vectors, the larger the cosine.
- Pearson similarity: Assuming again that A and B items are vectors, the Pearson coefficient between the two vectors represents the similarity.
- Jaccard similarity: Similarity is obtained by the number of users using items "A or B" dividing by the number of users using items "A and B".

1.1. A Brief Literature Survey

The location information provided by Flickr has been the subject of many different research areas and various studies have been conducted on this subject. Fors instance, more recently Kuo et al. (2018) studied different methodologies for finding places where people have shown interest using these geotagged photographs [2]. POI (Point of Interest) and ROI (Region of Interest) calculations are applied with clustering methods using location information. Gentile (2011) uses similarity prediction between locations using location and tag information [3]. Koochali et al. (2016) have done location and language matching using location and tag information [4]. In this work, Natural Language Processing (NLP) methods have been applied on tag information and attempted to determine the language it is written. The obtained information is studied by matching the geographical information of the photographs.

A lot of research has been done also on recommendation systems. Shani and Gunawardana (2010) worked on the evaluation and comparison of different recommendation systems [5]. There are also different researches on the collaborative filtering method that we consider implementing in this project. Researches that examine user-based, item-based and hybrid methods for collaborative filtering approaches have provided many different methodologies and algorithms in this regard [6, 7, 8, 9].

1.2. About the Data

Flickr provides application programming interface (API) to access to information (database) for developers and researchers. To access data, it is necessary to register and get "API Key" first. With this key, access to related web services is provided with the help of the source code. The list of web services, the parameters that they receive, and outputs are listed in the application page (https://www.flickr.com/services/api/). Photo search service is used mostly within the scope of the project. With this service, the output is returned in XML or JSON format. Each web service call returns a maximum of 250 photo information. It is possible to get all the photos that match the parameters we have provided by invoking the web service in a sequence of calls by changing page parameter. In this way, millions of photographs information can be accessed depending on the required criterion.

2. PROJECT DEFINITION

Although the location information from Flickr was used in other research subjects, I am unable to find a study that investigates the location recommendation methods using the available data. Location recommendation systems use various methods. Bao and Zheng (2017) have studied the features of various methods (such as content-based filtering, link analysis, or collaborative filtering) [10]. In this project, different recommendation models will be created by applying some of these methods to the Flickr location data. Also, these models will be compared with each other (by efficiency, accuracy, process speed etc.). Sufficient number of photos will be extracted from Flickr to create models (e.g. 10.000 records). What is important here is the parameter that will be used to extract the data. For example, it can all be pictures belonging to a specific location (for example, photos which are shot in Turkey), or it could be the photos belonging to a specific group of users. In this way, data can be prepared in accordance with recommendation system.

3. METHODOLOGY

The project is developed in a Python environment. Relevant web service calls and output in JSON format can be parsed within this environment. In order not to re-fetch the data every time with the web service, large datasets are created simultaneously and they are written to an output file. These large data sets were created by repeatedly calling the web service with different location parameters. Two datasets were created in this way. The first one contains a total of 381,500 photographs. When we group this photographs on a user-place basis, we get a dataset with a total of 1,065 records. The second dataset was obtained by giving location information of different countries in the world to obtain different user-place information. Although we had fewer photos in total than the first one, we had a lot more records in our user-place based group. A total of 189,844 photographs were used in this second dataset, and when they are grouped by user-place, we had 10,340 records in total. These two datasets will be used when examining the performance of the recommendation algorithms. In this way, we can observe the behavior of the algorithms in small and large datasets.

3.1. Preparing Data

The data frames provided by the pandas library are used during data acquisition phase and user-place based grouping of this data. To implement the recommendation methods, a library named “surprise” was used [11]. This library contains multiclass classification algorithms suitable for recommendation systems. Some of these algorithms are: kNN, SVD, NMF, Slope One and Co-Clustering.

First, to adapt the location information to the recommendation system in the project, we made the location behave like an item. In this recommendation system, it is necessary to find a method that allows users to rate the location information. To resolve this, the following method is applied: The number of photos a user takes at a given location can be measured as the rating(score) that the user assigns to that specific location. Because people take more photos and tend to share them in their favorite places. However, a rating may be between 1 and 5 and for this reason, it was assumed that users who share more than 5 photos in a location, the maximum value of the rating is set to 5 for this user and location pair. Since most photo counts in user-place groups are already under 5, applying this method also helps us eliminate outlier data. Below you can see the photo count distribution in the user-place groups before setting the maximum value to 5:

Table 1. Descriptive statistics of photo counts before setting maximum value to 5

photo count statistics by user-place	
count	10.340
mean	18,360155
std	123,369473
min	1
25%	1
50%	2
75%	7
max	6.070

Below you can see the photo count distribution in the user-place groups after setting the maximum value to 5:

Table 2. Descriptive statistics of photo counts after setting maximum value to 5

photo count statistics by user-place	
count	10.340
mean	2,671277
std	1,729275
min	1
25%	1
50%	2
75%	5
max	5

This method is applied to both datasets. First dataset has 1,065 (1K) rows of size and other dataset has 10,340 (10K) rows of size.

3.2. Recommendation System

The following multiclass classification algorithms provided by the “surprise” library have been used for designing the Recommendation system:

- Basic kNN: k-Nearest-Neighbors algorithm for classification [12]
- kNN with means: kNN algorithm considering the mean ratings of each user.
- kNN with Z-Score: kNN algorithm considering the z-score normalization of each user.

- kNN with baseline: kNN algorithm considering a baseline rating.
- SVD: Singular Value Decomposition algorithm, popularized by Simon Funk during the Netflix Prize [13].
- SVD++: An extension of SVD taking into account implicit ratings.
- NMF: Non-negative Matrix Factorization
- Slope One: A simple but accurate collaborative filtering algorithm [14].
- Co-Clustering: An algorithm based on co-clustering [15].

First, we trained the 1K dataset by applying 5-fold cross-validation to these algorithms. The "surprise" library gives us RMSE (Root Mean Squared Error), MAE (Mean Absolute Error) and data training time results for every 5 cross-validation. A comparison was made by taking the average of these 5 results.

The same 5-fold cross-validation process also applied to the 10K dataset, and the performance comparison between the algorithms was reconstructed over this larger data.

These algorithms have been tested on a computer with the following features:

- Intel ® Core™ i-7 2670QM CPU @ 2.20GHz
- 8GB RAM
- 64bit OS

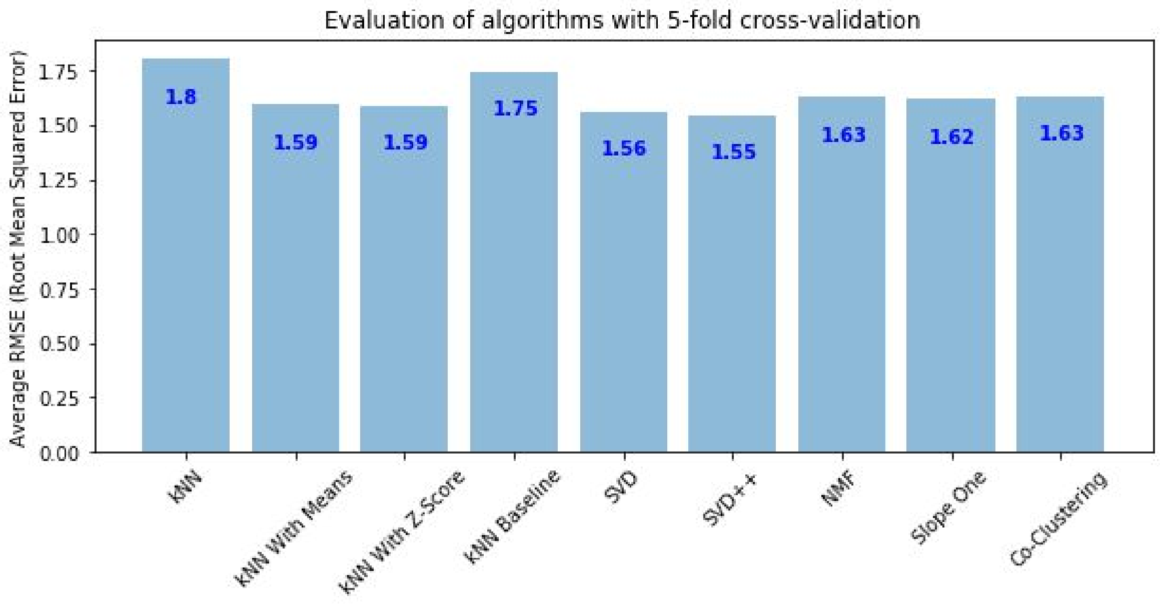
4. RESULTS

Since we have two datasets with different sizes, we are testing the algorithms separately for both.

4.1. Results for 1K dataset

Average RMSE scores with 1K dataset is shown below:

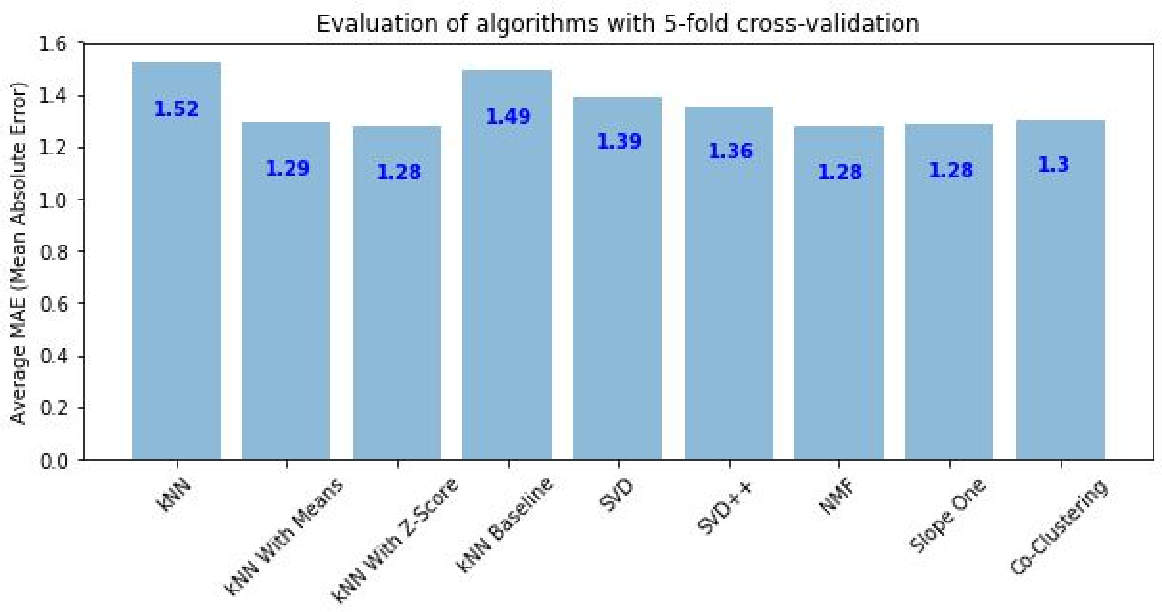
Figure 1. Average RMSE scores of algorithms with 1K dataset



As you can see, kNN algorithms with means and with Z-Score perform better than basic kNN and kNN with baseline. SVD algorithms also achieve better RMSE scores. Others have achieved average scores.

Now getting average MAE scores with 1K dataset:

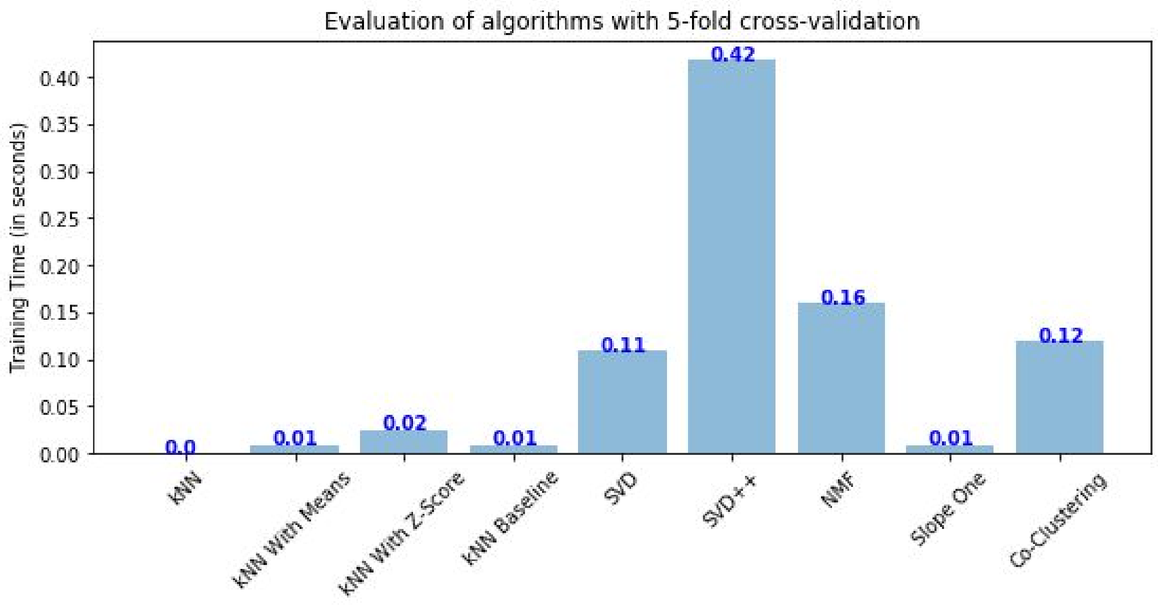
Figure 2. Average MAE scores of algorithms with 1K dataset



kNN with means, kNN with Z-Score, NMF, Slope One and Co-Clustering algorithms achieved the lowest MAE values. Instead of the good RMSE performance, SVD algorithms have higher MAE values than others. RMSE gives a relatively high weight to large errors since the errors are squared before they are averaged. As the RMSE scores of the SVD algorithms are better, we get a result that the records with high error are less in the model. With a higher MAE value, we can say that the model generally has lower performance. As a result, SVD algorithms can be successful in handling records with high error. Basic kNN and kNN with baseline algorithms have again the worst MAE values.

Now let's look at training times of algorithms with 1K dataset:

Figure 3. Training times of algorithms with 1K dataset

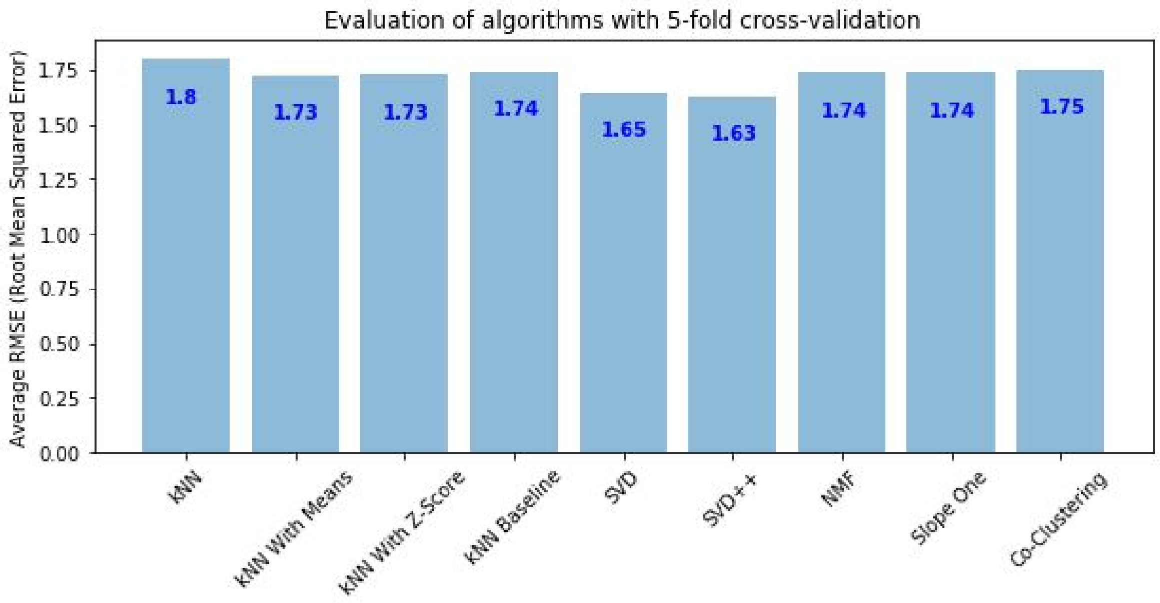


SVD, SVD++, NMF and Co-Clustering algorithms have higher training times. The others perform quite fast in the 1K dataset. Training time information can produce more meaningful results with 10K data set.

4.2. Results for 10K dataset

Average RMSE scores with 10K dataset is shown below:

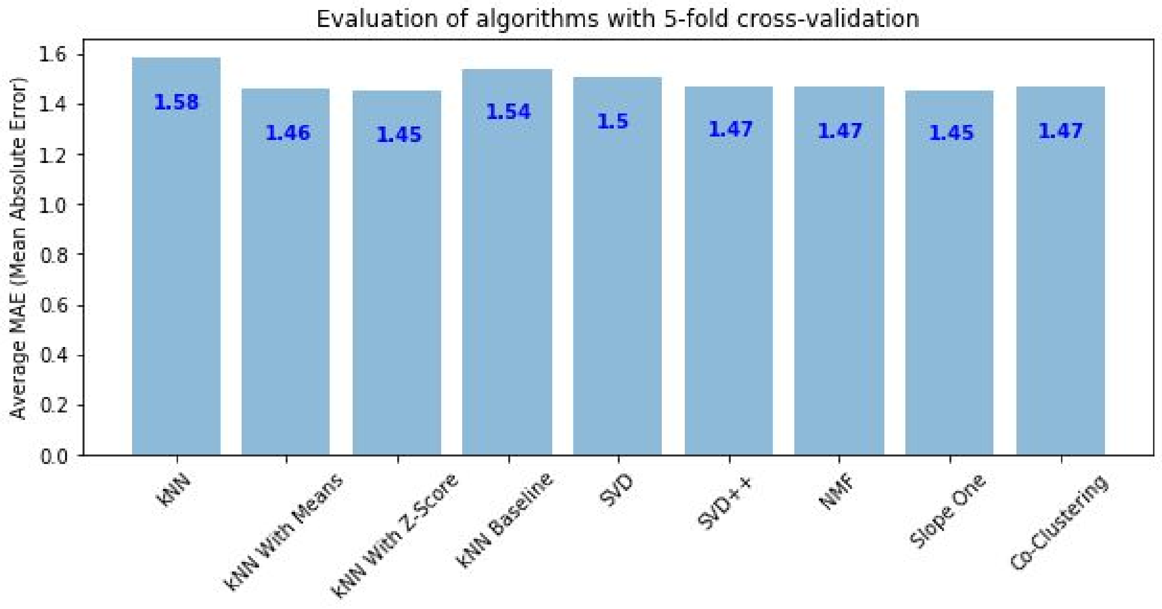
Figure 4. Average RMSE scores of algorithms with 10K dataset



As you can see, when we increase the data size, SVD algorithms achieve better RMSE scores than others. Basic kNN algorithm has the worst RMSE value.

Let’s look at average MAE scores with 10K dataset:

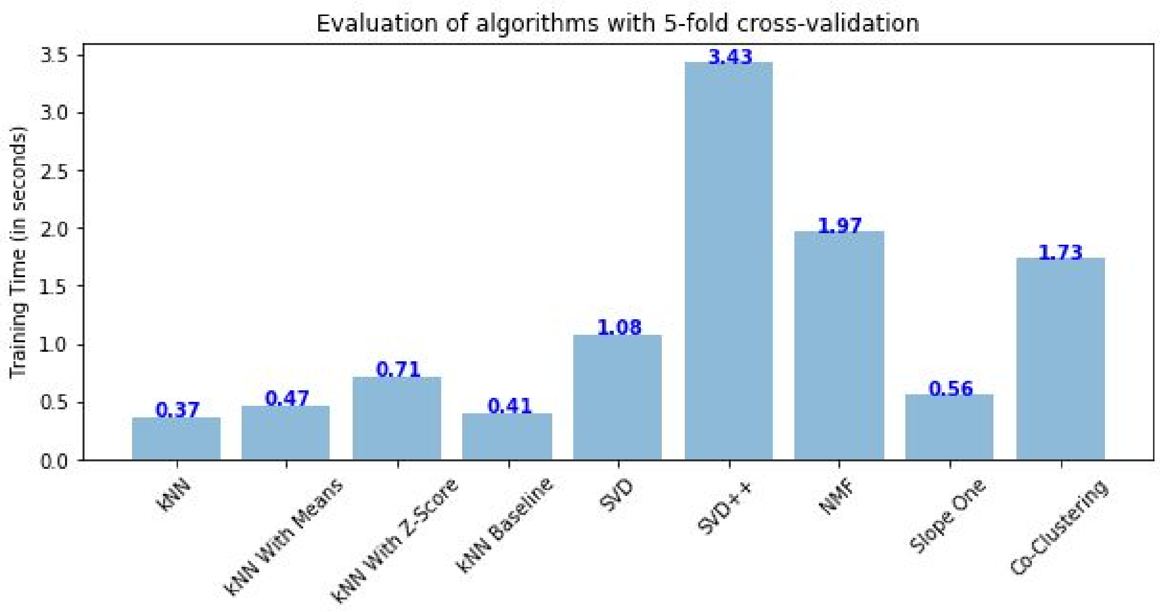
Figure 5. Average MAE scores of algorithms with 10K dataset



Except basic kNN and kNN with baseline algorithms, all others have close MAE values. These two algorithms showed the lowest performance for the MAE value in the 1K dataset.

And finally, let's look at the training times in 10K dataset:

Figure 6. Training times of algorithms with 10K dataset



SVD++ seems to be the algorithm that has the most training time. NMF and Co-Clustering algorithms also have model training durations close to 2 seconds. Other algorithms seem to be faster.

4.3. Evaluation of Results

Let's evaluate the performance of the algorithms one by one:

- Basic kNN: Although it has the lowest accuracy scores, we have seen it is the fastest algorithm.
- kNN with means and kNN with Z-Score: They performed above average as both RMSE and MAE scores. Also, working faster than most algorithms has made them the most suitable algorithms.
- kNN with baseline: Score is better than basic kNN but worse than other kNN algorithms. However, since it is close to the basic kNN as the working time, it may be a good choice if a fast algorithm is preferred.
- SVD: While SVD achieves a better score than other algorithms, the execution time is long.
- SVD++: SVD++ gives the best results in all algorithms. However, the model training time is the slowest.

- NMF: The NMF algorithm has an average score but is relatively slow as far as the execution time is concerned.
- Slope One: This algorithm has above average RMSE and MAE scores. Also, it is working faster than most algorithms. These results make it one of the most suitable algorithms together with the “kNN with means” and “kNN with Z-Score” algorithms.
- Co-Clustering: Co-Clustering algorithm has an average score. However, the training time is below the average.

4.4. Example of Location Recommendation

So far, we have examined the performance of the recommendation system with different algorithms. Now let's test the recommendation system on sample users using one of these algorithms.

Our 1K dataset had photographs of the city of Istanbul, Turkey. I choose a sample user from this dataset, from now on I will refer to this user as "user1" in order not to write the user name explicitly.

There are 5 pictures of this user in the dataset. The distributions of these photographs according to place names are as follows:

Table 3. Place distribution of photos for “user1” in 1K dataset

Place name	Photo count
Osmaniye, Istanbul, Turkey	1
Cankurtaran, Istanbul, Turkey	4

We’ll use the "kNN with Means" algorithm because it shows sufficiently good overall performance. After the 1K dataset is modeled, a matrix is created for each user-item match. If we take the recommendations of "user1" from this matrix, the three highest rated places are as follows:

Table 4. Top scored place recommendations for “user1”

Place name	Score
Esenkent/Bahcesehir, Istanbul, Turkey	5
Kilicali Pasa, Istanbul, Turkey	4.611
Rumeli Hisar, Istanbul, Turkey	4.583

According to the model based on the user's previous photographing locations, these are the most suitable locations for the user.

Let’s take 2 more sample users from same dataset and test the recommendation system with these users again.

The places and photographs taken by "user2" are as follows:

Table 5. Place distribution of photos for “user2” in 1K dataset

Place name	Photo count
Hocapasa, Istanbul, Turkey	3
Hobyar, Istanbul, Turkey	1
Cankurtaran, Istanbul, Turkey	5
Binbirdirek, Istanbul, Turkey	5
Muratpasa, Istanbul, Turkey	1
Sultanahmet, Istanbul, Turkey	5
Molla Fenari, Istanbul, Turkey	2
Alemdar, Istanbul, Turkey	5

We take the recommendations of "user2" from same user-item matrix, the three highest rated places are as follows:

Table 6. Top scored place recommendations for “user2”

Place name	Score
Suadiye, Istanbul, Turkey	5
Hacihesna Hatun, Istanbul, Turkey	5
Hamidiye, Istanbul, Turkey	5

Our last sample user “user3” has the following place-photo count distribution:

Table 7. Place distribution of photos for “user3” in 1K dataset

Place name	Photo count
Cankurtaran, Istanbul, Turkey	2
Kabatas, Istanbul, Turkey	1
Kemalpasa/Eminonu, Istanbul, Turkey	2

The three highest rated places that recommended for “user3” are as follows:

Table 8. Top scored place recommendations for “user3”

Place name	Score
Maltepe, Istanbul, Turkey	4.833
Esenkent/Bahcesehir, Istanbul, Turkey	4.704
Dizdariye, Istanbul, Turkey	4.198

If we look at the results of these 3 users' recommendations, we see that the recommendations with highest ratings are made for "user2". The reason is that we have more photos and location data of "user2". As conclusion we can say that, how much data a user has in the system, so accurate recommendations will make for that user.

5. REFERENCES

[1] Rounak Banik (2018, January 16). Recommender Systems in Python: Beginner Tutorial. Retrieved from https://www.datacamp.com/community/tutorials/recommender-systems-python

[2] Chiao-Ling Kuo, Ta-Chien Chan, I-Chun Fan and Alexander Zipf (2018). Efficient Method for POI/ROI Discovery Using Flickr Geotagged Photos

[3] Leonardo Gentile (2011). Using Flickr Geotags To Find Similar Tourism Destinations

[4] Alireza Koochali, Sebastian Kalkowski, Andreas Dengel, Damian Borth, Christian Schulze (2016). Which Languages do People Speak on Flickr? A Language and Geo-Location Study of the YFCC100m Dataset

[5] Guy Shani and Asela Gunawardana (2010). Evaluating Recommendation Systems

[6] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl (2001). Item-Based Collaborative Filtering Recommendation Algorithms

[7] Xiaoyuan Su and Taghi M. Khoshgoftaar (2009). A Survey of Collaborative Filtering Techniques

[8] Jun Wang, Arjen P. de Vries, Marcel J.T. Reinders (2006). Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion

[9] Robin Burke (2002). Hybrid Recommender Systems: Survey and Experiments

[10] Jie Bao, Yu Zheng (2017). Location-Based Recommendation Systems

[11] Hug, Nicolas (2017). Surprise, a Python library for recommender systems http://surpriselib.com

[12] Gongde Guo, Hui Wang, David Bell, Yaxin Bi, and Kieran Greer (2003). KNN Model-Based Approach in Classification

[13] Arkadiusz Paterek (2007). Improving regularized singular value decomposition for collaborative filtering

[14] Daniel Lemire, Anna Maclachlan (2008). Slope One Predictors for Online Rating-Based Collaborative Filtering

[15] Thomas George, Srujana Merugu (2005). A Scalable Collaborative Filtering Framework based on Co-clustering

6. APPENDIX

```
# importing packages
import flickrapi
import pandas as pd
from surprise import Dataset, Reader, evaluate, KNNBasic
from surprise import SVD, SVDpp, KNNWithMeans, KNNWithZScore, KNNBaseline
from surprise import NMF, SlopeOne, CoClustering
from surprise.model_selection import cross_validate, GridSearchCV
from collections import defaultdict
import numpy as np
import matplotlib.pyplot as plt

# api_key and api_secret are my register ID's to use Flick API
api_key = 'ebbbddbba58016fc80154cbb2317df9f'
api_secret = '59dc843971f5b784'

# calling Flickr API service
# inputs: register ID's and return format as json
flickr = flickrapi.FlickrAPI(api_key, api_secret, format='parsed-json')

# required extra information for flickr.photos.search method
extras='owner_name, geo'

# bbox parameter includes boundaries of latitude and longitude
# geographic location of Istanbul
b_box = '28.5,40.82,29.5,41.29'
# geographic locations of other cities from the world
# -75,39,-71,41
# 1,48,3,49
# 5,50,9,55
# 10,56,12,60
# search for photos located roughly in Istanbul
# per_page indicates the number of photos per page (per query)
geo_istanbul = flickr.photos.search(bbox=b_box,per_page=250,page=1, extras=extras)

# the result of query resulted 1400+ pages of photos
# there are 250 photos per page (this is maximum limit Flickr API provides)
total_pages = geo_istanbul['photos']['pages']
print(total_pages)

# list object for storing results
owner,ownername,place_id = [],[],[]

# for loop to get photo information with querying every page sequentially
for x in range(1, total_pages+1):
    print('Number of pages left: ',total_pages+1-x)
    geo_istanbul = flickr.photos.search(bbox=b_box,per_page=250, page = x, extras = extras)
    photo = geo_istanbul['photos']['photo']
    for result in photo:
        owner.append(result['owner'])
        ownername.append(result['ownername'])
        place_id.append(result['place_id'])

# merging list objects into a Pandas Dataframe
df = pd.DataFrame([owner,ownername,place_id]).T

# setting columns names of dataframe
df.columns = ['owner', 'ownername', 'place_id']
# creating copy of dataframe
```

```
df_grouped = df.copy()
# adding a new column to new dataframe
df_grouped['photo_count'] = 0

# grouping owner,ownername,place_id to get photo counts by user-place
df_grouped = df_grouped.groupby(['owner','ownername','place_id'], as_index=False).count()

# since rating input should be between 1 and 5, we set photo count = 5
# where photo count > 5
df_grouped.loc[ df_grouped.photo_count > 5, 'photo_count' ] = 5



# Codes for recommendation system. We use “surprise” package
reader = Reader(line_format='user item rating', rating_scale=(1, 5))
data = Dataset.load_from_df(df_grouped.loc[:,['ownername','place_id','photo_count']],reader)
trainingSet = data.build_full_trainset()

algo_svd = SVD()
algo_knn = KNNBasic()
algo_svdpp = SVDpp()
algo_knnWithMeans = KNNWithMeans()
algo_knnWithZScore = KNNWithZScore()
algo_knnBaseline = KNNBaseline()
algo_NMF = NMF()
algo_SlopeOne = SlopeOne()
algo_CoClustering = CoClustering()

# Run 5-fold cross-validation and print results
result_svd = cross_validate(algo_svd, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
result_knn = cross_validate(algo_knn, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```

```
result_svdpp = cross_validate(algo_svdpp, data, measures=['RMSE', 'MAE'], cv=5,
verbose=True)
result_knnWithMeans = cross_validate(algo_knnWithMeans, data, measures=['RMSE',
'MAE'], cv=5, verbose=True)
result_knnWithZScore = cross_validate(algo_knnWithZScore, data, measures=['RMSE',
'MAE'], cv=5, verbose=True)
result_knnBaseline = cross_validate(algo_knnBaseline, data, measures=['RMSE', 'MAE'],
cv=5, verbose=True)
result_NMF = cross_validate(algo_NMF, data, measures=['RMSE', 'MAE'], cv=5,
verbose=True)
result_SlopeOne = cross_validate(algo_SlopeOne, data, measures=['RMSE', 'MAE'], cv=5,
verbose=True)
result_CoClustering = cross_validate(algo_CoClustering, data, measures=['RMSE', 'MAE'],
cv=5, verbose=True)

# Means of cross-validation results for RMSE value
mean_svd = np.mean(result_svd['test_rmse'])
mean_knn = np.mean(result_knn['test_rmse'])
mean_svdpp = np.mean(result_svdpp['test_rmse'])
mean_knnWithMean = np.mean(result_knnWithMeans['test_rmse'])
mean_knnWithZScore = np.mean(result_knnWithZScore['test_rmse'])
mean_knnBaseline = np.mean(result_knnBaseline['test_rmse'])
mean_NMF = np.mean(result_NMF['test_rmse'])
mean_SlopeOne = np.mean(result_SlopeOne['test_rmse'])
mean_CoClustering = np.mean(result_CoClustering['test_rmse'])

# Codes for plotting mean values of algorithms
algorithms = ('kNN','kNN With Means','kNN With Z-Score','kNN
Baseline','SVD','SVD++','NMF','Slope One','Co-Clustering')
y_pos = np.arange(len(algorithms))
```

```
performance =
[mean_knn,mean_knnWithMean,mean_knnWithZScore,mean_knnBaseline,mean_svd,mean_
svdpp,mean_NMF,mean_SlopeOne,mean_CoClustering]

fig, ax = plt.subplots(figsize=(10,4))
ax.bar(y_pos, performance, align='center', alpha=0.5)
ax.set_xticks(y_pos)
ax.set_xticklabels(algorithms, rotation=45)
for i, v in enumerate(performance):
   ax.text(i-.2,v-0.2, str(round(v,2)), color='blue', fontweight='bold')
plt.ylabel('Average RMSE (Root Mean Squared Error)')
plt.title('Evaluation of algorithms with 5-fold cross-validation')
plt.show()

# Means of cross-validation results for MAE value
mean_svd = np.mean(result_svd['test_mae'])
mean_knn = np.mean(result_knn['test_mae'])
mean_svdpp = np.mean(result_svdpp['test_mae'])
mean_knnWithMean = np.mean(result_knnWithMeans['test_mae'])
mean_knnWithZScore = np.mean(result_knnWithZScore['test_mae'])
mean_knnBaseline = np.mean(result_knnBaseline['test_mae'])
mean_NMF = np.mean(result_NMF['test_mae'])
mean_SlopeOne = np.mean(result_SlopeOne['test_mae'])
mean_CoClustering = np.mean(result_CoClustering['test_mae'])

# Codes for plotting mean values of algorithms
algorithms = ('kNN','kNN With Means','kNN With Z-Score','kNN
Baseline','SVD','SVD++','NMF','Slope One','Co-Clustering')
y_pos = np.arange(len(algorithms))
```

```
performance =
[mean_knn,mean_knnWithMean,mean_knnWithZScore,mean_knnBaseline,mean_svd,mean_
svdpp,mean_NMF,mean_SlopeOne,mean_CoClustering]

fig, ax = plt.subplots(figsize=(10,4))
ax.bar(y_pos, performance, align='center', alpha=0.5)
ax.set_xticks(y_pos)
ax.set_xticklabels(algorithms, rotation=45)
for i, v in enumerate(performance):
   ax.text(i-.2,v-0.2, str(round(v,2)), color='blue', fontweight='bold')
plt.ylabel('Average MAE (Mean Absolute Error)')
plt.title('Evaluation of algorithms with 5-fold cross-validation')
plt.show()

# Means of cross-validation results for FIT TIME value
mean_svd = np.mean(result_svd['fit_time'])
mean_knn = np.mean(result_knn['fit_time'])
mean_svdpp = np.mean(result_svdpp['fit_time'])
mean_knnWithMean = np.mean(result_knnWithMeans['fit_time'])
mean_knnWithZScore = np.mean(result_knnWithZScore['fit_time'])
mean_knnBaseline = np.mean(result_knnBaseline['fit_time'])
mean_NMF = np.mean(result_NMF['fit_time'])
mean_SlopeOne = np.mean(result_SlopeOne['fit_time'])
mean_CoClustering = np.mean(result_CoClustering['fit_time'])

# Codes for plotting mean values of algorithms
algorithms = ('kNN','kNN With Means','kNN With Z-Score','kNN
Baseline','SVD','SVD++','NMF','Slope One','Co-Clustering')
y_pos = np.arange(len(algorithms))
```

```
performance =
[mean_knn,mean_knnWithMean,mean_knnWithZScore,mean_knnBaseline,mean_svd,mean_
svdpp,mean_NMF,mean_SlopeOne,mean_CoClustering]

fig, ax = plt.subplots(figsize=(10,4))
ax.bar(y_pos, performance, align='center', alpha=0.5)
ax.set_xticks(y_pos)
ax.set_xticklabels(algorithms, rotation=45)
for i, v in enumerate(performance):
   ax.text(i-0.2,v, str(round(v,2)), color='blue', fontweight='bold')
plt.ylabel('Training Time (in seconds)')
plt.title('Evaluation of algorithms with 5-fold cross-validation')
plt.show()

# Codes for recommendation example
# filtering data for a user name "yrmeydanci"
df_filtered = df_grouped.loc[df_grouped['ownername'].isin(['yrmeydanci'])]

# printing place names for user name "yrmeydanci"
for i in df_filtered['place_id']:
   place_info = flickr.places.getInfo(place_id = i)
   print(place_info['place']['name'])

reader = Reader()
data = Dataset.load_from_df(df_grouped.loc[:,['ownername','place_id','photo_count']],reader)
trainingSet = data.build_full_trainset()

# using kNN with means algorithm
knn = KNNWithMeans()
```

```
# fitting train data
knn.fit(trainingSet)
testSet = trainingSet.build_anti_testset()
predictions = knn.test(testSet)



# function to get top 3 recommended places
def get_top3_recommendations(predictions, topN = 3):
   top_recs = defaultdict(list)
   for uid, iid, true_r, est, _ in predictions:
      top_recs[uid].append((iid, est))
   for uid, user_ratings in top_recs.items():
      user_ratings.sort(key = lambda x: x[1], reverse = True)
      top_recs[uid] = user_ratings[:topN]
   return top_recs

# calling function, output is a list of users and 3 place_id's per row
top3_recommendations = get_top3_recommendations(predictions)

# printing recommended place names for user name "yrmeydanci"
for i in top3_recommendations.get('yrmeydanci'):
   place_info = flickr.places.getInfo(place_id = i[0])
   print(place_info['place']['name'])
   print(i[1])
```