Rating Prediction for Google Local Reviews

Raghasuma Aishwarya Putchakayala
MS CS, UC San Diego
rputchak@ucsd.edu

Shreyas Anantha Ramaprasad
MS CS, UC San Diego
shananth@ucsd.edu

Sourabh Raja Murali
MS CS, UC San Diego
srajamurali@ucsd.edu

Sumanth Ramachandra Hegde
MS CS, UC San Diego
s1hegde@ucsd.edu

Abstract

Rating prediction is an important area of research in building recommender systems. In this project, we consider rating prediction on the Google Local Reviews dataset. With only a few interactions per user, the task of building an accurate model that can generalize becomes challenging. We first explore and analyse the dataset to extract meaningful features. We then present our approach, a factorization machine utilizing location-based attributes for personalization. We further present a comprehensive evaluation of different interaction-based and feature-based models. Finally, we demonstrate the potential of our approach outside the low data regime by evaluating model performance on a subset of the dataset with more interactions per user.

Keywords: Exploratory data analysis, Recommender systems, Latent factor models, Factorization machine

1 Introduction

Recommender systems have grown to be a fundamental part of our everyday lives, influencing everything we buy online, the information we consume everyday and much more. Broadly speaking, these systems process, filter and display the information available in a way that suits the individual user and his preferences. Ratings are one of the most popular types of interaction data and thus rating prediction is a fundamental task in building recommender systems.

In this project we have built a recommender system for rating prediction on Google Local Reviews. The dataset consists of user ratings for businesses all around the world, with additional details on user background, business operating duration, location, etc. The dataset is challenging, with with very few interactions per user. In Section 3, we first present our analysis of the dataset where we try to understand variations in different attributes and gain preliminary insights on what meaningful features we can incorporate. We focus on building a factorization machine which tries to model the user and item along with feature information. We explore incorporating place category, place popularity and the location of the business. We show that simple linear latent factor models perform best and factorization machines with higher latent dimensions perform much worse (Section 4). We also consider a subset of the dataset with more interactions per user. We demonstrate that model performance significantly improves over simple baselines in this regime.

2 Related Work

Recommender systems, broadly speaking, fall into two categories: feature-based methods, typically using some heuristics on what data is important, and model-based methods, which try to model user preferences and item characteristics directly from the interaction data. The simplest kind of recommendation system is a memory based approach which simply looks at the history of users interaction to recommend similar items[5]. These use various heuristics to measure similarity between items and users. Factorization machine [7] is a model-based approach to recommendation, and extends simple matrix factorization based models to incorporate user and item features. Another popular model-based approach is Factored Item Similarity Model (FISM) [2], which avoids explicitly modelling users and relies only on item representations. FISM is more suitable for scenarios where each user's interactions are few in number. Current state of the art methods are typically based on deep learning [4], where a neural network is used to model user behaviour. The Google Local reviews dataset was introduced by [1]. The authors considered the problem of sequential recommendation, where they predict a user's future interactions based on the user's current or historical interactions. One of the models that were developed, termed TransFM, made use of squared Euclidean distance as it's inner product. It used translation and metric-based approaches with Factorization Machines. The geographical coordinates from the Google local dataset were used in the evaluation of the TransFM model in a geographical setting. In our project, we consider the related task of rating prediction for a user and an unseen item (in this case, a business). Two related datasets are the Amazon dataset[6] and the Foursquare dataset[3]. The Amazon dataset contains a large corpus of product ratings, reviews, timestamps and related metadata, collected from Amazon.com from May 1996 to July 2014. The Foursquare dataset consists of interaction data related to user check-ins at different venues from December 2011 to April 2012, originally taken from *Foursquare.com.* [6] considered the task of recommending complementary items to a user's previous clothing purchases, and [3] built a recommender system to

predict user ratings using location information. In this work, we consider the Google Local reviews dataset since it consists of data from a large variety of businesses ("items") and has significant sparsity (few interactions per user) which make it challenging.

3 Exploratory Data Analysis and Feature Selection

In this section, we introduce the dataset and discuss the cleaning methodology. Also, we elaborate on useful statistics and trends that we observed that assisted in feature selection.

3.1 Dataset Overview

The Google local reviews dataset consists of user reviews of businesses and the associated user and business metadata. The data is contained in three files - User Data(4,567,431 entries), Places Data (3,116,785 entries) and Review Data (11,453,845 entries).

The following fields are present:

- User Data -
  - *userName* - Name of the user(String)
  - *jobs* - A list describing the job history of the user including organization, start date and end date
  - *currentPlace* - The user's current location (includes city, state and GPS co-ordinates)
  - *previousPlaces*: A list of the user's past locations (includes city, state and GPS co-ordinate)
  - *education* - A list describing the educational history of the user including institute, start date and end date
  - *gPlusUserId* - A string which is unique to each user
- Places Data -
  - *name* - Name of the business
  - *price* - One of '', None, $, $$, $$$
  - *address* - As a list of strings
  - *hours* - As a list of Weekday and Timing
  - *phone* - String denoting the phone number
  - *closed* - String indicating whether the place is closed
  - *gPlusPlaceID* - A string which is unique to each place
  - *gps* - A latitude and longitude tuple
- Review Data
  - *rating* - A 0 to 5 rating for the business by a user
  - *reviewerName* - String indicating the user's name
  - *reviewText* - String with the user's review of the business
  - *categories* - A list of strings specifying the category of the business
  - *gPlusPlaceId* - ID of the business
  - *unixReviewTime* - The unix timestamp of the review
  - *reviewTime* - The date in MMM DD YYYY (MMM is first three letters of the month)
  - *gPlusUserId* - This is the user ID of the user who gave the rating

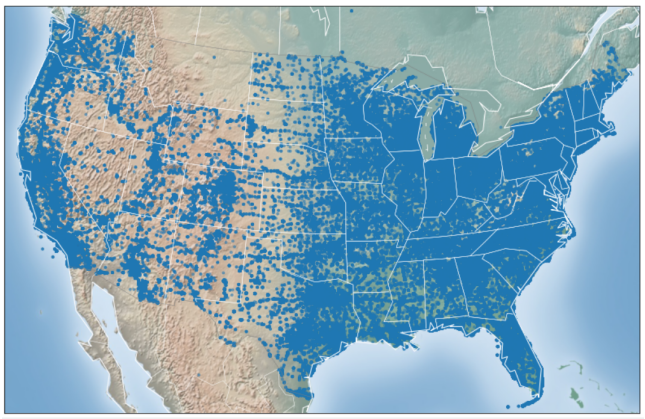

Figure 1. Distribution of reviews across USA

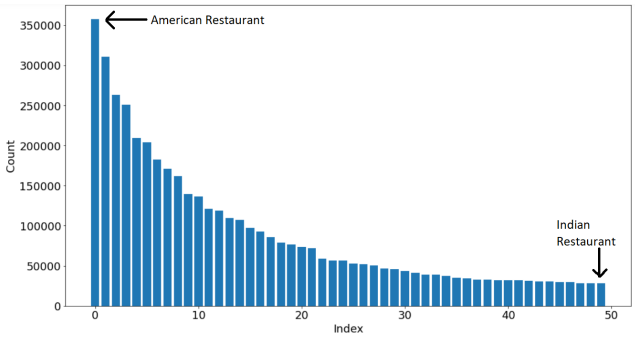

Figure 2. Number of Reviews for the Top 50 Categories

3.2 Data Cleaning

The original dataset contains data about businesses and reviews from all around the world. Since processing this huge amount of data will be computationally expensive (considering the limited hardware we are working with), we filtered it to include only people, places and reviews from only the USA. Figure 1 shows the distribution of user reviews across USA.

In addition, the following filtering was done:

- Businesses which have 'None' or '' for the price field were removed
- There are 7959 categories of businesses. We filtered out the top 50 categories and their distribution is show in Figure 2. As you can see, the dataset is dominated by reviews from a few categories (mostly different types of restaurants)
- The GPS co-ordinates of several businesses had decimal points in the incorrect position and these were fixed

After cleaning the data, we are left with 749,000 entries for the review data, which we then split to Train, Validation and Test sets. Further for the training set we filtered out multiple users who only had a single review. Our final dataset consists

of 532,880 samples, with the train, validation and test sets in the ratio 70%/20%/10%.

3.3 Data Visualization

We have performed a comprehensive analysis on several attributes of the dataset and we present our findings in this subsection.

We examined how the user rating varies with the price of a business and found that people tend to give a slightly higher rating for places that are relatively inexpensive. This is intuitive because people in general tend to have greater expectations from expensive places, and more often that not they are not met. In Figure 3, we can see that a higher fraction of $ places receive a rating of 5 than places with $$$ price.

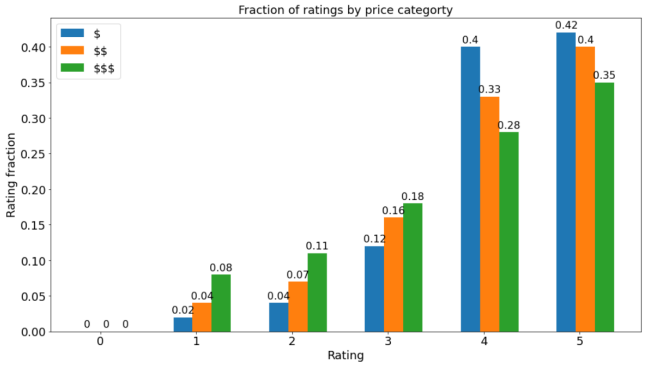

Figure 3. Rating fractions per price category

Next, we analyzed the variation of number of ratings and average ratings versus time of the day & day of the week. We observed that the number of reviews are 12% higher on Saturday and Sunday. This can be attributed to the fact that people prefer visiting places during the weekends. Also, we saw there is a small correlation between time of the day and the average rating - people tend to give a higher rating during the middle of the day (around 2 pm as seen in 4). Figure 5 is a plot of the average rating versus the day of the week, and there isn't any significant relationship between them.

We also studied the impact of the geographical location of a business on the number of reviews and the rating. In Figure 1, it is evident that the number of reviews are concentrated in densely populated areas. Also, the location seems to impact the average rating of a place as seen in Figure 6. There are clusters with similar average ratings which might give an insight about the general rating tendency of the people in an area.

Next, we analyze the relationship between the number of reviews that a place has got versus the rating. Figure 7 shows that as the number of ratings increase, the average rating increases implying that popular places are rated higher. A similar graph was plotted between number of ratings per

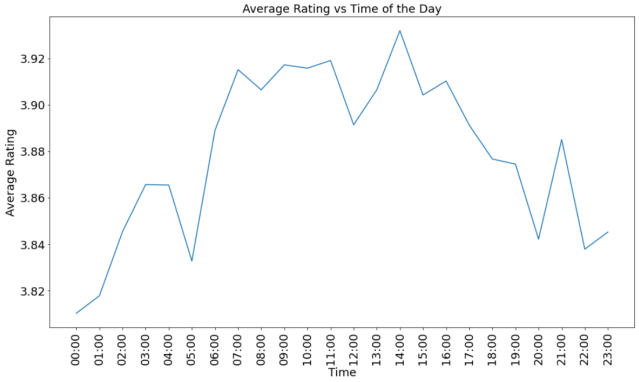

Figure 4. Average rating vs Time of the day

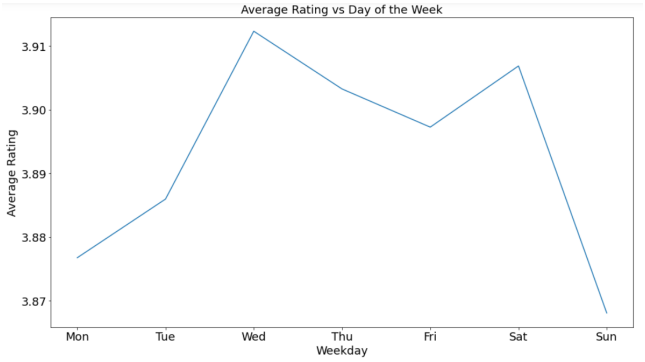

Figure 5. Average rating vs day of the week

user and the average rating, and there was no noteworthy trend.

In addition to the features discussed previously, there is one feature - the review text that we are not using. Something like the length of the review or its sentiment may be highly predictive of the rating, but using this is not appropriate for our task. The rationale behind this is simple: since we are trying to predict how a business would appeal to the

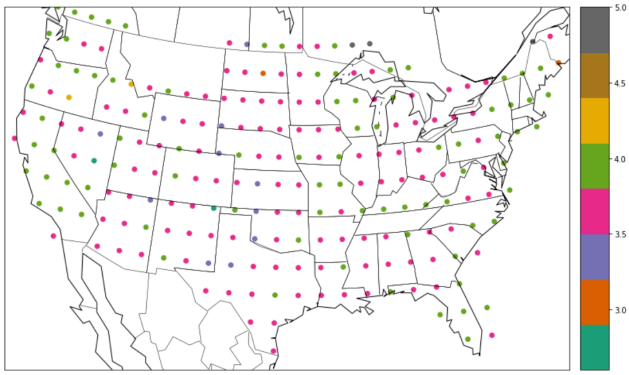

Figure 6. Average rating across USA

Figure 7. Number of reviews per place vs Average rating

preferences of a user, there is no interaction data (like review text) available to us. Thus, since the motivation behind predicting the rating is that one can suggest places that a user has not visited, the review text is not suitable for our prediction task.

3.4 Feature Selection

In this subsection we would be discussing the features that were tried for the models that were experimented with.

- Grid Popularity - We considered the minimum and maximum range of latitude and longitude values of USA and divided the area into grids of size 1 degree by 1 degree identified by a co-ordinate corner of the grid. We looked at number of places corresponding to which we have reviews in each grid
- State Popularity - Since we have the address information corresponding to each place, we captured the number of place reviews corresponding to each US state
- Place Popularity - Number of users who have reviewed/rated a particular place
- User Activity - Number of places reviewed by a particular user
- Category of the place - Since the number of categories that a place belong to is high we use a one hot encoding of the category that a particular place belongs to
- Price Category of the place - This particular information is part of the Places Data where and we used the ordinal encoding of the Price Category where "$" corresponds to 1, "$$" to 2 and so on
- Place Co-ordinates - The geographical coordinates of the place.

Since we experimented with both feature based models and latent factor models, we have used user and item interaction data along with some of the above mentioned features in our latent factor models.

4 Model Selection and Evaluation

In this section, we explain the rating prediction models that we experimented with and their results. We start with a trivial model that always predicts the average rating, and this serves as our first baseline. Then, we explore linear regression and decision trees which are feature based models. We perform ablation tests to figure out the relative importance of the features we talked about in Section 3. Lastly, we used simple linear latent factor models, factorization machines, and Factored Item Similarity Model(FISM) to model the interaction between the user and the business. Finally, we summarize our findings across all the models following which we discuss the results of experiments with a filtered and dense version of the dataset. The evaluation metric chosen is the Mean Squared Error, where e_t is the error at index t.

$$Mean\ squared\ error(MSE) = \frac{1}{n}\sum_{t=1}^{n} e_t^2$$

Also, we did consider using classifier models, but they cluster the data into discrete categories. This does not align with our objective of providing ratings that can be used by a downstream recommender system, and so we don't comment much about classifier models in this report.

4.1 Global Average Baseline

First, we look at the performance of a simple baseline: always predicting the average value of all the ratings in the training set. This gave us a validation MSE of 1.467.

4.2 Linear Regression

Linear Regression finds an underlying linear relationship between input feature vectors and output labels. Our motivation for performing linear regression was to use the simplest model to validate our claims made during feature visualization, and also to figure out the relative importance of the features. Linear Regression takes the following form for this specific problem:

$$\begin{aligned} Rating = {} & \theta_0 + \theta_1 * (Place\_Popularity) \\ & + \theta_2 * (User\_Popularity) + \theta_3 * (Grid\_Popularity) \\ & + \vec{\theta}_4 * (\overrightarrow{State}) + \vec{\theta}_5 * \overrightarrow{(Place\_Category)} \end{aligned}$$

Here, $\vec{\theta}_4$ and $\vec{\theta}_5$ are vectors because we used an encoding scheme for *State* and the *Place_Category*. We experimented by using a 49 dimensional one hot encoding for the state, and also by using the state's mean coordinates. We found that the one-hot encoding performed better and this is expected as the ordering of place co-ordinates does not correspond naturally to an ordering of place quality. For the *Place_Category*, we went ahead with a 49 dimensional one-hot encoding since we filtered by the top 50 categories.

To understand the relative importance of different features, we performed ablation tests on the 5 features above, and we found that the *Place_Popularity* and the *Place_Category* are the two most important features - this holds good irrespective of the encoding scheme used for the *State*. We also experimented with feature transformations (squaring/ cubing/ log) and found that the model worked better without any of these transforms

After tuning the hyperparamteres, we got a training MSE of 1.263 and a validation MSE of 1.405. This served as our Baseline 2 which we attempt to beat in the subsequent subsections.

4.3 Decision Tree (XGBoostRegressor)

In order to improve our MSE, we looked at a feature based model that can better fit our data, and the ensemble decision tree based XGBoostRegressor was the perfect choice. The same set of features as that of Linear Regression were used, and we got an MSE of 1.120 on the training data and 1.408 on the validation data after hyperparamter(depth, regularizers) tuning.

Another incentive for using XGBoostRegressor is its *feature_importances_* attribute. Using this, we got deeper insights on the relative importance of the features. We saw that in certain states(like FL and SC), the location tends to be more important when compared with other states. Also, certain categories of places tend to receive a higher rating - for example, fast food restaurants in general are rated higher than dessert shops. Once again, we performed ablation tests and we got results similar to the linear regression ablation experiments.

This model's performance was comparable to linear regression on the validation set, but it overfit the training set more than the linear model. The lack of improvement can be attributed to the fact that decision tree based models can often overfit on the training data. Further, both the models do not capture interactions between users and items, and hence we move onto to latent factor models in the next subsection.

4.4 Latent Factor Models

Traditional regression approaches fail to capture the dynamics of interaction data. We have used historical interactions along with place features in order to capture the complex semantics that underlie users' behavior and preferences. In traditional regressors we know exactly what a feature corresponds to whereas here the hidden dimensions explain the variation in people's opinions, without necessarily knowing exactly what the dimensions correspond to.

4.4.1 Linear Latent Factor Models(Surprise and FastFM).

We started by implementing a traditional latent factor model where we modelled the user and item interactions in terms of latent user and item factors as shown in Equation 1

$$r(u,i) = \alpha + \beta_u + \beta_i + \gamma_u \cdot \gamma_i \tag{1}$$

where α, β_u, β_i are the bias terms, and γ_u and γ_i are K dimensional vectors describing the user and item respectively. This model was fit using the Surprise package and we experimented with the number of latent factors and the regularization parameters. Due to the sparse nature of the user interactions, the model tended to overfit the data. After careful tuning of the regularization parameter, we obtained a training MSE 1.055, validation MSE of 1.426 and a test MSE of 1.439. Also, we varied the number of latent dimensions in the 0 to 5 range, and we observed that the best results are from 0 or 1 latent dimensions. A latent factor model based solely on the user-item interactions relies heavily on the number of interactions for each user/item. In the Google local review dataset, most users have made under 5 reviews, and this explains why the model does not perform better.

Next, we tried to implement the factorization machine model of FastFM which implements :

$$f(x) = w_0 + \sum_{i=1}^{F} w_i x_i + \sum_{i=1}^{F} \sum_{j=i+1}^{F} \langle \gamma_i, \gamma_j \rangle x_i x_j \tag{2}$$

where the first two terms are the offset and bias terms, and the last summation is the feature interaction terms.

As a first step we considered feature vector consisting of one-hot encoded user u, one-hot encoded place i of u as a baseline for the latent factor models. The regularization value used was 20, with $K = 0$. We obtained a train MSE of 1.026 and validation MSE of 1.413 which conforms to what we obtained using the Surprise package.

4.4.2 FastFM with features.

Through our EDA we discovered that popularity of state and place capture valuable information. We also felt the location of the place itself would also be important to look at. Hence we decided to use these features alongside in our FastFM model.

FastFM model with place co-ordinates resulted in a train MSE of 1.025 and validation MSE of 1.411.

FastFM model with state and place popularity gave a better performance compared to other models where we obtained a train accuracy of 0.727 and validation accuracy of 1.378. Even here, $K > 0$ had worse performance on the validation set.

4.4.3 FISM with FastFM.

Factored Item Similarity Model (FISM) model based approach learns item representations but abstain from using user representations. The rating given by a user u for item i is given by [2]:

$$f(u,i) = \alpha + \beta_u + \beta_i + \frac{1}{|I_u \backslash i|} \sum_{j \in I_u \backslash i} \gamma_j^{'} \cdot \gamma_i \tag{3}$$

We implemented FISM using a factorization machine in FastFM. Each feature vector x consisted of only the "query" and "target" representations of places. The regularization

used was 20, and we found no improvement with $K > 0$. We obtained a train MSE of 1.120 and validation MSE of 1.412.

4.5 Summary of Results

In this section, we discussed the performance of several models for the rating prediction task. Table 1 effectively summarizes our observations. It is seen that, that a factorization machine using the metadata as features performs the best on the validation set. This model has a test MSE of 1.388 which is in the same range as the validation MSE.

Table 1. Table showing the train and validation MSEs of various models

Model	**Train MSE**	**Validation MSE**
Trivial Mean Predictor (Baseline 1)	-	1.467
Linear Regression (Baseline 2)	1.263	1.405
Decision Tree (XGBoostRegressor)	1.120	1.408
Latent Factor Model using Surprise	1.055	1.426
Factorization Machine with only User and Place ID	1.026	1.413
Factorization Machine with features	**0.727**	**1.378**
FISM	1.120	1.412

4.6 Performance on filtered dense dataset

In order to better understand the power of factorization machines, we performed an experiment wherein we filtered the dataset even more to include only the reviews of users who have given at least ten reviews, thus reducing the sparsity of the dataset. This left the dataset with 142,227 reviews which we again split into train/validation/test sets in the same ratio as before.

The trivial sample mean model gave an MSE of 0.976 on the test set, and this formed our baseline. The simple latent factor model using surprise got slightly better MSE of 0.875 on the training data, 0.950 on the validation set, and 0.941 on the test set.

As expected the FastFM model built on this sampled dataset performed much better as mostly active users were considered. We obtained a train MSE of 0.824 and validation MSE of 0.889. The MSE on the test set was 0.877. Clearly, this demonstrates that factorization machine models perform better when there are more user interactions rather than on new or inactive users.

5 Conclusion

In this work we studied numerous ways of predicting user ratings on the Google Local reviews dataset. The challenging nature of the dataset made it incredibly difficult for both feature based and interaction based methods to perform considerably better than the simple baseline. While we also explored deep factorization machines as an alternative to avoid explicit feature selection, this did not perform better than our linear latent factor models. A numerical feature corresponding to the price attribute (over simple categorical encoding) worked better, along with popularity of the business in terms of previous interaction history. Further modeling second order interactions in the factorization machine led to overfitting, due to the sparse nature of the dataset. We believe even the implementation used in the FastFM library also contributed significantly to the patterns we observed. The improvement in performance on the subset of the Google local reviews dataset illustrates the power of our approach.

References

[1] Ruining He, Wang-Cheng Kang, and Julian McAuley. 2017. Translation-based recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*. 161–169.

[2] Santosh Kabbur, Xia Ning, and George Karypis. 2013. FISM: Factored Item Similarity Models for Top-N Recommender Systems. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Chicago, Illinois, USA) *(KDD '13)*. Association for Computing Machinery, New York, NY, USA, 659–667. https://doi.org/10.1145/2487575.2487589

[3] Justin J. Levandoski, Mohamed Sarwat, Ahmed Eldawy, and Mohamed F. Mokbel. 2012. LARS: A location-aware recommender system. *Proceedings - International Conference on Data Engineering* (30 July 2012), 450–461. https://doi.org/10.1109/ICDE.2012.54 IEEE 28th International Conference on Data Engineering, ICDE 2012 ; Conference date: 01-04-2012 Through 05-04-2012.

[4] Yifei Ma, Balakrishnan Narayanaswamy, Haibin Lin, and Hao Ding. 2020. Temporal-Contextual Recommendation in Real-Time. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2291–2299.

[5] Julian McAuley. in press. *Personalized Machine Learning*. Cambridge University Press.

[6] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 43–52.

[7] Steffen Rendle. 2010. Factorization Machines. In *2010 IEEE International Conference on Data Mining*. 995–1000. https://doi.org/10.1109/ICDM.2010.127