Civilingenjör i datateknik, 300 credits

Optimize Ranking System With Machine Learning

Anton Gustafsson, Fredrik Mattsson

Computer science, 15 credits

Halmstad 2018-06-03
Abstract

This thesis investigates how recommendation systems have been used and can be used with the help of different machine learning algorithms. Algorithms used and presented are decision tree, random forest and singular-value decomposition (SVD). Together with Tingstad, we have tried to implement the SVD function on their recommendation engine in order to enhance the recommendation given. A trivial presentation on how the algorithms work. General information about machine learning and how we tried to implement it with Tingstad’s data. Implementations with Netflix’s and Movielens open-source dataset was done, estimated with RMSE and MAE.

I denna uppsats tittar vi på hur rekommendationssystem kan och har använts med hjälp av olika maskininlärnings metoder. Metoder som tas upp i denna rapport är decision tree, random forest och SVD. Tillsammans med Tingstad har vi försökt att implementera en SVD-metod på Tingstads rekommendationssystem för att förbättra deras rekommendationer. En grundlig överblick i hur de olika metoderna fungerar. Generell information om vad maskininlärning är och hur vi försökte implementera det med Tingstads data. Implementerar gjordes med Netflixs och Movielens open-sorce dataset, estimerad prestanda gjord med RMSE och MAE.
Acknowledgement

This thesis is the result of a bachelor degree project done during the spring term 2018 with Tingstad. This is a bachelor thesis in our education civil-engineer in computer science.

We would like to thank Carl-Johan at Tingstad and 3Bits for the support during this thesis. We would like to thank our supervisor Sepideh Pashami at Halmstad University for all the support and guidance she has provided throughout the whole project.

Anton Gustafsson & Fredrik Mattsson
Gothenburg 2018
# Contents

1 Introduction 1
   1.1 Problem statement 1
   1.2 Purpose 2
   1.3 Restrictions 2
   1.4 Tingstad and 3Bits 2

2 Background 3
   2.1 Recommendation system 3
      2.1.1 Content-based filtering 3
      2.1.2 Collaborative filtering 3
   2.2 Machine learning 4
      2.2.1 Supervised learning 4
      2.2.2 Unsupervised learning 4
   2.3 Related work 5
   2.4 Integrity and security 6

3 Theory 7
   3.1 Algorithms 7
      3.1.1 Decision tree 7
      3.1.2 Random forest 9
      3.1.3 Singular-value decomposition 10

4 Methods 13
   4.1 Time schedule and budget 13
   4.2 Platform settings 13
      4.2.1 Python 13
   4.3 Packages and Modules 14
      4.3.1 Scikit-learn 14
   4.4 Preparation 14
   4.5 Implementation 15
   4.6 Evaluation 15
      4.6.1 Statistical accuracy 16
      4.6.2 Decision support accuracy 16

5 Result 19
   5.1 Collection of data 19
   5.2 Analyze of the algorithms 19
      5.2.1 Decision tree 19
      5.2.2 Random forest 23
      5.2.3 SVD 24

6 Discussion 27

7 Conclusion 29

References 31

A Appendix 36
1 Introduction

During the past years machine learning applications has grown exponentially [1]. This thesis includes cooperation with Tingstad a B2B e-commerce and 3Bits a consultancy firm. Tingstad is aware of the importance to stay updated with the newest technology. They want to explore what the latest knowledge in algorithm development with machine learning could offer in actual, real and measurable results. It has been known for Tingstad that an optimized site search is important for great success [2]. Tingstad wants to see what an optimized site search could do in terms of increasing sale and the number of new customer registrations.

Why machine learning is of great value for online services can be found by looking into the giant movie service called Netflix. For Netflix a good recommendation engine has been crucial to their success. Every movie in the start screen that is being shown are personalized for each customer. Netflix has tons of data available for their customers. Having such vast amount of data causes customers to get a choice paralysis that in the end makes customer to watch nothing. By using machine learning algorithm they have increased engagement significantly, saving their company an annual 1 billion dollars [3].

From this you can tell a good recommendation system can have a great impact on a company’s total revenue. In today’s society, recommendation systems with machine learning are getting more relevant for each day. There is tons of data to be analyzed and that amount of data is impossible for a human to analyze by hand and knowing what is useful information. That is where machine learning excels. Finding useful relation in the information and provide better results than code written by a person.

1.1 Problem statement

3Bits is a consulting company who is responsible for Tingstad’s Webpage. 3Bits already use "elastic search" as there search algorithm, the algorithm gives points to item’s based on how close you are to the searched item. But often customers buys an item that is ranked with lower points and therefore have a higher index. It goes without saying, this is not optimal.

The question is could machine learning increase sale and customer registration? Are there any relations between customers and the product they buy? These are the question we hope finding an answer to with our project. If machine learning could find a relation we could give extra points to that item when the customer is searching on it. This would optimize the search algorithm and a product that has greater potential of being sold would be higher on the list with a lower index.
1.2 Purpose
Optimize the ranking system of the search results after each search, using machine learning, in order to give the customer more precise products that is of the customers interest, resulting into an increase in Tingstad’s sale and customer registration.

1.3 Restrictions
Restrictions in this thesis is to narrow down alternatives of choice of algorithm. Our knowledge of machine learning from the start was low and choosing the more simpler algorithms is required. A bigger restriction is access to data, how much data that are useful and accessible. The complexity in the program that is being developed can not be to high because of the time giving during this thesis is not enough for a development of a high complex program.

1.4 Tingstad and 3Bits
Tingstad is a B2B e-commerce company. They provide supplies and products to support big and small businesses in all industries. Tingstad offers around 30 000 products to their customers [4]. Tingstad is a family business and is currently in its second and third generation. Tingstad is one of the dominating companies that supplies within the nordic countries with an annual turnover of 1.8 billion SEK. The last decade Tingstad has been digitalizing their company and it has paid off [2].

3Bits is a consultancy who provides B2C and B2B e-commerce solutions as it would be their own for different businesses. They are working together with Lindex, Ragn-sells, Scandinavian Photo and Tingstad. They are one of the fastest growing e-commerce consultancies in Sweden [2].
2 Background

The following section contains information about different types of recommendation systems that exist and some information about machine learning algorithms that is to be investigated.

2.1 Recommendation system

Recommendation system can be a tool for personalizing content for each individual. When individuals are presented with many choices and can therefore get overwhelmed, machine learning can help avoiding the situation. Recommendation systems are commonly used for the purpose to increase sales. They can be divided into two parts, content-based filtering and collaborative filtering.

For using a recommendation system, it is crucial to have enough data for the system to work with. It applies to both content-based and collaborative filtering. If there are not enough describing information about the products or customer it will be hard to find the relations.

2.1.1 Content-based filtering

Content-based filtering systems handles data from the item’s attributes to make a recommendation [5]. For example, when we want to recommend a product for a customer, we can look at earlier ordered, recently searched or recently clicked product and compare it with other similar products in the catalog. With this comparison we can find item’s that would be of the customers interest based on the looked item. This system often has a disadvantage in finding advanced patterns [5], as finding relevant products that not always has a relation with earlier purchased items. Because content-based filtering only recommends similar products customer showed interest in, meaning that a customer do not get new products recommended and therefore only buy a small amount of the total products offered. For example if a person always buys cutlery there is no reason for the system to recommend a chair.

One way to overcome content-based problems with not enough provided information, is applying associative retrieval framework and related spreading activation algorithms to find transitive associations between the customers, by looking into the past transaction history, feedback and customers profile information. Another way to handle the sparsity, is by using a dimensionality reduction technique, Singular-value decomposition are to be discussed more closely later in this thesis [6].

2.1.2 Collaborative filtering

Collaborative systems often perform better than content-based, only if provided with enough information. You can divide collaborative filtering into user-based and item-based. User-based system usually take into account other customers bought/rated products [5]. For example, we have a restaurant A that is much the same as restaurant B. Restaurant B is in a group of other restaurants that is much alike itself and they buy mostly the same products. With this information we could give restaurant A recommendations based on what restaurant B and its
group buys. This gives us opportunities to give recommendation to new customers that we yet have no information about what type of item’s they want, as moreover recommend old customers new item’s based on their customer group’s bought item’s.

Item-based works in a similar way as user-based. Instead it searches for similar items but not the same way as content-based filtering. What differs between them are that content-based only looks on the attributes, item-based collaborative finds similar items ratings and puts an averaged rating on the item based on similar items rating of the same user [7]. Item-based collaborative is one of the most powerful techniques if provided with enough information [5].

Collaborative filtering systems that works with small amount of information about customers and rated items has often bad performance. If the new customer A does not buy the similar products or is similar to already existing customers, it is going to be hard to find recommendations that is relevant for customer A.

2.2 Machine learning

Machine learning are used for analyzing big set of data and to find relations and patterns in the dataset, for the use of processing results that is dependent on data that is and can be changing over time. In order for the algorithm to give as good results as possible it needs to learn how to handle the data. Machine learning algorithms have two ways of being taught how to learn and process data. These ways are called supervised and unsupervised learning.

2.2.1 Supervised learning

Supervised learning require labeled items. For example, you have a system deciding what type of fruit you are holding in front of a camera. To train this type of algorithms to distinguish how an apple look like, you hold the apple in front the camera and tell the algorithm this is an apple. Next time this apple or another apple is shown, the system should detect and mark the object as an apple.

2.2.2 Unsupervised learning

Unlike supervised learning unsupervised don’t require a human labeling. Unsupervised look for patterns and structure in data. For example, this algorithm that told us what fruit we held in front of a camera. Instead of telling the algorithm this is an apple we instead shows an apple and then a banana and we keep switching between these two fruit until the algorithm can distinguish that these two are completely different fruits.
2.3 Related work

Using machine learning for recommendation systems in e-commerce has grown massively over the past couple of years. Due to this, more companies want to implement machine learning on their own e-commerce, which is both easy and effective. The most common machine learning algorithms for recommendation systems is decision tree, it is known for its relative simplicity. Specific description about the method is found in 3.1.1.

In Hyunwoo Hwangbo et al. [8] and Menghan Yan et al. [9] articles talks about how they developed collaborative filtering models in the fashion respectively video areas. The fashion model is based on purchase and clicking data same type of data as Tingstad is using. Hwangbo et al. recommendation system is special developed for fashion, it takes into count season changes, product information and the decrease over time of the product. After an A/B testing of the recommending system show that the proposed system perform better results than the old one. Yan et al video recommending system takes a look how a SVD method can help the system and take away the “noise”. Linden et al. [10] industrial report talks about how item to item based algorithm is useful for their website. Benefits with the item to item collaborative filtering compared to traditional collaborative filtering it is good with limited user data and also react quick to changes in the user data. Jicheng Li and Xinyue Huang combined the SVD and K means functions to make better recommendations [11].

Recommendation systems is being used on a daily basis, it is often taken for granted and makes the user to expect the system to give them recommendations. One of the most known system is Netflix, who uses it to give their viewers a recommendation of movies they should see based on previously watched content and what resembling users seen. According to an article in a journal from ACM written by Carlos Gomez-Uribe and Neil Hunt [12], Netflix saves $1 billion each year on using recommendation system. In 2006 Netflix started a competition where the goal was to increase the accuracy with 10%. First 3 years later a winner was decided, team “BellKor’s Pragmatic Chaos”. They was awarded with the grand prize of 1 million dollars. The dataset is open-sourced and contains more than 100 million ratings from over 480 thousand customers on more than 17 thousand movies [14].

Youtube uses recommendation system to easily and quickly recommend videos to their users. Another popular system is Spotify’s discover weakly playlist, it is also built on recommendation system. Thanks to the system, Spotify increased their users from 75 million to 100 million in 2017 stated from a blog post [13]. Other large companies who uses machine learning for recommendation systems are Google, LinkedIn, Amazon and Twitter mentioned in two articles [15, 16].

According to Ivens Portugal et. al. article [16] are they most common algorithms for recommendation systems Bayesian, Decision tree, Matrix factorization-based (including SVD), Neighbor-based, Neural Network and Rule Learning. Spotify are using matrix factorization-based algorithm that is called SVD from the Sci-kit learn [17]. The winner of the Netflix prize BellKor’s Pragmatic Chaos used a SVD++ function in their algorithm [18].
2.4 Integrity and security

User integrity involves keeping a person’s personal information secret. A law, General data protection regulation called GDPR, was established to strengthen the laws about handling of personal records. Most recommendation systems use a lot of personal data to achieve the best recommendations. Collection and usage of such information require good protection to keep user integrity safe. Data integrity means assuring that the data structure and validness are intact.

Netflix prize data set that was released 2006 declared a winner 2009. They protected their users integrity by giving fake id’s, although the given time of a rating and the rating was still correct. Shortly after the contest ended, two researchers from University of Texas, Arvind Narayanan and Vitaly Shmatikov [19] announced they have been able to identify users from Netflix prize data set. They used the rated data from IMDB and Netflix data set, together with the time a rating was given, to identify customers from Netflix.

If a company has a working recommendation system which improve sales, others might wanna interrupt or in any way manipulate the system by destroying data integrity. Recommendation systems are vulnerable to an attack called shilling attack also called profile injection attack [20]. Shilling attacks are used to manipulate recommendation rankings. It is done by creating fake accounts. With enough fake users a recommendation system will be affected. How the system will behave depends on the goal with the fake users. Some goals could be to make the system do faulty recommendations or make it recommend desired products from the attackers point of view.
3 Theory

This section contains different algorithms that is commonly used for recommendation systems.

3.1 Algorithms

Following algorithms are all supervised learning methods and has simple complexity, meaning that they are easy to understand and explain. They are commonly used in our area of research.

3.1.1 Decision tree

A decision tree classify instances by looking for the best feature that divides data in the best way. Each node in the tree represents a feature in an instance that is to be classified. Each branch in the tree are values that the nodes can assume [21]. In other words there is a top node in the tree called root which has the feature that best divides the data. For example what is the color of the object. Depending on the color it will take a branch down to next node where a similar question will be asked and so forth. When there is no more branch to take you have ended up in a leaf that contains the final value.

![Decision tree](image)

Figure 1: Decision tree
Decision tree algorithms are often used because of its low complexity and it is easy to work with when the size of the tree is not too large. Thus make it a good algorithm for us to use. Although it is not an advanced algorithm it could still be of great value in optimizing this ranking system. A problem do follow when using decision tree. When training the decision tree, there is always a risk to overfit the set of training data when you want to achieve maximum accuracy [22].

Overfit happens when you use to much data to train the tree. A way to avoid this is by pre-prune, one pre-prune method is to stop the tree before it reaches its full size. It can also be avoided by using post-prune methods. An example of a post-prune method is to remove and assign nodes the most common class of the training instances that are sorted to it [21].

There are many methods that can be used in the sense of constructing decision trees. The most common used methods for constructing decision trees are linear splits at each internal node. The typical method selects a hyperplane or multiple hyperplanes at each internal node. Samples are assigned to branches that represents different regions of the feature space that is bounded by hyperplanes. There is a difference in methods in number and orientations of the hyperplanes and how they are chosen. It is possible to categorize these methods by the types of splits they produce that are determined by the number and orientation of our hyperplanes [22]. (See above figure)

- axis-parallel linear splits: A threshold is chosen on the values at a particular feature dimension, and branches get assigned samples to whether the corresponding feature value exceed the threshold that was set. It is possible to choose multiple threshold for assignment to multiple branches. There is a second generalization method that uses Boolean combinations of the comparisons against thresholds on multiple features. This results in tree that has a deep depth but still extremely fast execution [22].
• oblique linear splits: Splits samples are assigned to the branches according to
which side of a hyperplane or region bounded by multiple hyperplanes they
fall in, only exception is that the splits doesn't need to be parallel to any axis
of the feature space. A generalization is to use hyperplanes in a transformed
space, where each feature dimension can be an arbitrary function of selected
input feature. The trees decision regions can be finely tailored to the class
distribution, and the trees can be small. Its execution are dependent on the
complexity of the hyperplanes or the transformation functions [22].

• piecewise linear splits: Branches represent a Voroni tesselation (seen in above
figure, picture c) of the feature space. Assignment for samples are based
on nearest-neighbor matching to chosen anchor points. Anchor points are
selected among training samples, class centroids or derived cluster centers.
These trees can have a large amount of branches and have a shallow depth
[22].

Each type of category has different algorithms to achieve each splitting category.
for axis-parallel we have Sethi and Sarvarayudu's mutual information, the Gini in-
dex proposed by Breiman, Quinlan's information gain ratio or Mingers' G statistics.
For oblique we have Tomek links, simulated annealing or perceptron training. For
Piecewise can be obtained through supervised and unsupervised clustering [22].

3.1.2 Random forest

Random forest is an extension to decision tree where the randomness is built on
a hybrid of Bootstrapping and random subspace method. In random forest each
tree is grown on subset of features and not the entire dataset. This subsets are
randomly made for the purpose to make a wide diversity between each tree. How
large the subset should be varies [23].

Going through the random forest is done by going through each tree to a leaf
with the recommendation. Then collecting the recommendation from all trees and
taking out the recommendation that the majority is choosing as the result. The
accuracy from random forest depends on the strength of each tree and measure of
the dependence between the trees [24].

The advantages with random forest compare to decision tree is that you don’t
need to do any pruning. Another advantage is the overfitting, As said earlier overfit
happens when the tree becomes to big and this is not a problem in random forest
because its built on the subsets to prevent to big trees. Random forest is primary
used due to not overfitting and its speed [24].
3.1.3 Singular-value decomposition

Singular-value decomposition or commonly used word SVD is a matrix factorization technique often used to produce low-rank approximations. SVD is good at filtering out “noise” in the customer-product relationship. Noise disturbs the chance of finding the suitable products. As many algorithms there comes a price to pay. Computing the SVD algorithm is time consuming and makes it less relevant to use if time is crucial, though its cost in computing, it delivers fast online performance that only require a few simple arithmetic operations for each recommendation [25].

![Matrix Factorization Diagram]

Figure 3: Matrix factorization, where diagonal matrix s is excluded for easier understanding

As we can see on the picture above we have a matrix over users and item’s the users have rated. When applying SVD it gives us three matrices that we could do the dot product between and get a value that tell us how close related each user are to each other depending on their preferences. This means that we could find similarities between customers and give these customers suggestions based on one and another bought or browsed item.

If given a matrix A with the dimensions of m x n, with rank r, the singular value decomposition SVD(A), is defined as

\[
SVD(A) = U \times S \times V^T
\]  

(1)

The matrices U, S and V have the dimensions of \(m \times m\), \(m \times n\) and \(n \times n\), respectively. Matrix S is a diagonal matrix that only has r nonzero entries, which makes the effective dimensions of the three matrices \(m \times r\), \(r \times r\) and \(r \times n\), respectively. U and V are orthogonal matrices and S is a diagonal matrix which is called singular matrix. Values of the diagonal entries in the matrix S has the properties \(s_1 > 0\) and \(s_1 \geq s_2 \geq \ldots \geq s_r\). In the matrices U and V the first row of columns represents the orthogonal eigenvectors that is associated with the r nonzero eigenvalues of \(AA^T\) and \(A^T A\), respectively. U and V are called the left and the right singular vectors, respectively [25].
To perform a low-rank approximation of the matrix $A$ with the SVD method, it is required to retain only $k << r$ singular values by discarding entries and it is possible. The reduction process is done accordingly, first retain the first $k$ singular values from $S$, we call it $S_k$, and this is also done for the matrices $U$ and $V$ creating $U_k$ and $V_k$. To produce $U_k$ and $V_k$, is achieved by removing $(r - k)$ columns from $U_k$ and $(r - k)$ rows from $V_k$. When multiplying these three matrices it gives us $A_k$, and its formula looks as follows $A_k = U_k \ast S_k \ast V_k^T$. The new matrix $A_k$, is a rank-$k$ matrix that is the closest approximation of the original matrix $A$ [25].

The new matrix $A_k$ minimizes the Frobenius norm over all the rank-$k$ matrices. Researchers has pointed out that low-rank approximation on original space is better than the original space only because it filters out the “noise” in the customer-product relationship [25].
4 Methods

This section explain the implementation details of the algorithm

4.1 Time schedule and budget

Time given for this thesis was around 4 months. Budget included only personal investments, agreement on investments from the companies for extra software or resources needed, was not stated. The resources paid personal would be train tickets to Gothenburg. Only already existing tools would be provided from the companies side. The intended use of time for the development of project can be found in appendix in figure 17. First part of the thesis was to research about machine learning. Next part would be to gather data and prepare it for the algorithm. Once the data was prepared the program was to be developed and an A/B test during the last weeks of this thesis.

4.2 Platform settings

There is two programming languages taking in count in this thesis, R and Python, both contributes with good tools for machine learning algorithms. Both languages have well documented helpful packages for machine learning. R is a language developed for statistical computing and graphics [26]. Python was chosen because of its easy to learn syntax. Software for development of the code was chosen from either a simple editor where only code can be written, such program as notepad and anaconda or an IDE such as Microsoft’s Visual Studio code or Eclipse. In this case Visual Studio code or more known as VS code was chosen because of its many easy to import functions assisting you in your code writing and its debugger. In addition to choice of language and software, there was given by the company, a remote access to their network and computer for retrieving data outside the company.

4.2.1 Python

Python is an interpreted, interactive, object-oriented programming language. Python integrates modules, exceptions, dynamic typing, high level dynamic data types and classes. Python is a combination of remarkable power with a clear and simple syntax. There are many interfaces to many system calls and libraries, as well to windows systems and it can be extended in C or C++. Python can be used as an extension language for applications that need a programmable interface. Python can be runned on linux, mac and windows [28].

Pythons Package Index (PyPI) hosts thousands of third party modules for Python. In their own standard libraries and community-contributed modules. Some of them are web and Internet Development, Database Access, Desktop GUIs and Software and Game Development [29].
4.3 Packages and Modules

In Python as mentioned earlier, there are many packages containing algorithms. The libraries, packages and modules used are brought up below.

- Numpy, using its mathematic functions for calculating mean values.
- Pandas, using its read functions to fetch data from files, its functions for creating dataframes and handling of the dataframe.
- Scikit-learn, using its random forest and tree module and from the surprise package inside scikit-learn, reader, dataset, SVD and evaluate.
- Json, using the reader function for reading in data from a json file.
- Matplotlib.pyplot, using its graph functions for visualizing data in different graphs and using graphviz for visualizing the decision tree.

4.3.1 Scikit-learn

Scikit-learn or sometimes called sklearn, is a package available in python. Sklearn are open-source and commercial usable. Sklearn contains machine learning tools such as classification, regression, clustering, dimensionality reduction, model selection and preprocessing. It is using other packages Numpy, Scipy and matplotlib. Sklearn offers a wide range of the most popular machine learning algorithms, for example decision tree, random forest, SVD and SVM [30]. Several companies uses Sklearn. One of the biggest and most recognized user of Sklearn is Spotify. They use it for music recommendations [17].

4.4 Preparation

As a preparation for the development of the algorithm that is later to be implemented in Tingstad’s homepage, once data was received from the company, it was required to get knowledge of how to use machine learning algorithms in the chosen code language python. Therefore a tutorial was followed to gather knowledge about python's syntax, what packages exist, how to use the packages and functions to structure data to desired structure [31].

Material used both from the tutorial and for understanding the tutorial can be found on GitHub [32]. The tutorial bring up how to train neural networks algorithm and decision tree in python, using movielens open dataset. Scikits decision tree uses a method called CART, that uses the best split 3.1.1 that gives the best results. The tutorials material has been used to train a random forest, not included in the tutorial. The knowledge aquired from this tutorial was used when preparing data from Tingstad and using it for training the algorithm. Dataset from Netflix has also been used to train a SVD function [27]. Netflix’s data comes from the Netflix competition and is an open-source dataset. Netflix’s data contains data from 100 millions ratings and the ratings range between 1-5.
4.5 Implementation

One of the algorithms that is being used in this study is SVD. SVD is going to be integrated with the total program that is running on Tingstad’s web page. The program can be divided into 2 steps. In general these steps works as follow. A flowchart can be found in appendix 16.

- **Step 1, running:** The program receives input-data from the company in a file stating which customer searched on what product and the result of the search in form of a list. The program reads the file and put it in a variable. Data is then processed by the SVD and the result is a list of item’s, with a given rating each, that range between 4 values, where the top item’s correspond to the products the customer most likely would buy. With the results from the SVD algorithm the program runs a function that rearrange the search result list in a way that makes the customer buy more products. The programs result is sent back to Tingstad’s web page where it later is shown to the customer.

- **Step 2, training:** During the night, the program does a train phase. It reads data from two files, one CSV file has customers and their purchases and a JSON file has customers behavior data. It merges these lists that are extracted from the files and from that constructs a train dataset and a test dataset. With the datasets it retrain itself and get the newest update about what customers bought recently and how the customers behave.

By recommending the most relevant product for a customer after their search leads to better customer experience and a bigger chance of more revenue. If the customer searches for spoon and irrelevant spoons are shown as the first top 20 item’s and all relevant spoons shown 2 pages away, item’s that deep in most likely would not be browsed and therefore not purchased. This algorithms purpose is to bring relevant item’s to the top.

How to evaluate the actual performance of the algorithm an AB testing is to be done. AB testing will be performed by having 50% of the customers using the new algorithm, while the other half use the present one. The customer do not know which one they are using. To get a good and fair result from the comparison of the different algorithms, shall the comparsion at least be performed during a month. After the test a result can be seen where it states which algorithm that attracted the most customer registrations and gave Tingstad the highest income during the test period.

4.6 Evaluation

The evaluation of a recommendation system is done by splitting the data in two parts, first part is for training the algorithm and the second one to test how good it’s preforming. The performance is measured in two different accuracy, Statistical and decision support. Accuracy is a fraction of the correct recommendations. Evaluation could be calculated in some different ways, for example "precision", "recall", "F-measure", "MAE" and "RMSE". Depending on the used algorithm, different evaluation is more suitable.
4.6.1 Statistical accuracy

SVD is usually evaluated with statistical accuracy metrics, MAE and RMSE. Both metrics measure the deviation of the predicted value to actual value. Mean Absolute Error (MAE) is computed as

\[
MAE = \frac{\sum_{i=1}^{N} |x_i - y_i|}{N}
\]

(2)

Where \(x_i\) is the predicted value and \(y_i\) is the actual value. \(N\) is total number of values. Root Mean Square Error (RMSE) is given by

\[
RMSE = \sqrt{\frac{\sum_{i=1}^{N} (x_i - y_i)^2}{N}}
\]

(3)

The lower MAE and RMSE, the more accurately is the predictions. The bigger differences between these metrics is that RMSE put more weight on large errors and deviation [33].

4.6.2 Decision support accuracy

For decision tree and random forest is common to evaluate with accuracy, precision, recall and F-measure. They are computed as

\[
Accuracy = \frac{TP + TN}{TP + TN + FP + FN}
\]

(4)

\[
Precision = \frac{TP}{TP + FP}
\]

(5)

\[
Recall = \frac{TP}{TP + FN}
\]

(6)

\[
F\text{-measure} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}
\]

(7)

- TP stands for true positive and are the recommended item’s that was relevant.
- TN is for true negative and are item’s that are correctly not recommended.
- FP, often called false alarm, false positive are recommended item’s that are not relevant.
- FN, false negative is the part that is not recommended but are relevant for the user.
Figure 4 illustrate a recommendation. For example we have a picture of dogs and cats where there are 11 dogs(green) and 10 cats(red). If we ask the computer how many dogs there are in the picture and the computer identifies 6 out of 11. Out of 6 identified dogs, 2 of them where cats(false positives).

Precision(equation 5) tells the fraction of the recommended item’s that was relevant for the user. In the example this would be 4/6, four is correctly out of the six selected. But it does not tell the how many relevant item’s that we missed to recommend. Recall(equation 6) is a fraction of relevant item’s that are successfully retrieved of the item’s that should have been retrieved. The recall in the example would be 4/11 because it found four dogs out of eleven. F-measure(equation 7) is useful to use due to that it is combining precision and recall. F-measure puts the same weight on both precision and recall [34]. The Accuracy of the example is 12/21, twelve animals where correctly identified out of twenty one.
5 Result

In this section results from supervised machine learning algorithms are presented.

5.1 Collection of data

During the process of this project one type of data was collected, the collected data was information about the orders to Tingstad. The collected data includes costumerID and item. The collection of data was collected from Tingstad’s database and include data from 2017-2018.

Other used dataset was Movielens and Netflix. These datasets was used to test decision tree, random forest and SVD. Movielens dataset includes 100 thousand ratings and more attributes. Netflix’s dataset includes 100 million ratings that 480 thousand users gave to nearly 18 thousand movies.

Figure 5: Shows what information Movielens and Netflixs data contains

Costumers behavior data was never retrieved. 3Bits was responsible to create a function that would collect this data but was not successful to develop a fully functional program for retrieving behavior data.

5.2 Analyze of the algorithms

The algorithms evaluated was decision tree, random forest and SVD. Decision tree and random forest was evaluated by looking at decision support accuracy, RMSE and MAE. SVD evaluated by RMSE and MAE.

5.2.1 Decision tree

Decision tree was created on data from Movielens to predict the probability of a movie being bought. In the dataset from Movielens included attributes as genre, year, time-stamp and ratings. With no limitation on the depth of the tree, was the depth 27. A buy probability was created according to

\[
\text{buy probability} = 1 - \text{price} \times 0.1
\]

where price is a generated number between 0 and 10

\[
\text{movie.data['price']} = \text{np.round}((1 - \text{normalised.rating}) \times (1 - \text{normalised.age}) \times 10).
\]

Buy probability can be used as a validation to see if decision tree estimate cheap movies to be more likely purchased. Events was generated and for simplicity 1 thousand costumers was created that viewed 20 movies each. Different distribution could be found in figure 6.
Figure 6: Different distribution of data from Movielens

Figure 7 shows the amount of positive and negative data related to the price. Positive data in this dataset are where customers bought the movie and negative are movies that the customer did not buy. The price axis is normalized, at 0 is the average price and to the left the are cheaper and more expensive on the right side. From the graph we can see that cheap movies are more bought than expensive movies.

Figure 7: Distribution of positive and negative data related to the price
In figure 8 is the first levels of the created decision tree. First value in the nodes is the if-statement generated by the tree that it is using to navigate through the tree down to the bottom for a final result. Gini impurity is a splitting criterion ranging from 0 to 0.5. Gini is preferred to be as close to 0.5 as possible, this means that the classes has been as perfectly mixed. It is good because that means that each item in the class will have better relation to each item in the same class, and when new item’s are being put into the class it will have a better relation. In other words better recommendations. Samples is a variable stating how many sample that exist in each node. Value states how much information is gained from each input vector X and Y when combined is the total samples existing in the node. Vector X contains all the describing information about each movie and vector Y contains the outcome for each movie stating if the customer bought or not.

From the top 3 layers of the tree there are nodes with great gini. Having a bad gini in a node at a lower layer in the tree do not affect the end result. If the node with a bad gini would be a leaf the result would be affected immensly. Only looking on gini to find quality of the tree is not enough. If the gini is good but has huge amount of item’s, means there are to many item’s to recommend.

Figure 8: Decision tree, made to recommend movies from Movielens
Figure 9: probability of a movie will be bought from Movielens dataset

Figure 9 shows how likely a movie is bought related to its price and the average rating. In the graph each dot represent a movie and the rank is the probability that a movie will be bought. If the price is lower there are a bigger probability that the movie will be bought. From the graph with the ratings and probability is hard to see any pattern. What we can see in the figure is that low rated movies still have a high probability of being bought. Most likely because they are cheap and therefore higher buy probability. It seems the tree estimates price to have a big impact on being bought or not. We can also see that many movies with a high rating has a low chance of purchase. That most likely means that the price was to high for the customer to wanna buy that movie.

Figure 10: Accuracy of the decision tree from Movielens dataset

The accuracy of the decision tree is 70% for the training set. When the system get unknown data it preformed with an accuracy of 62%. RMSE and MAE for the decision tree has been calculated to 0.548 and 0.614 with a depth of 27. When limiting the trees depth to 10, a slightly change was given. RSME and MAE was 0.569 and 0.593.
5.2.2 Random forest

The training and test set for random forest done with Movielens. Parameters used are default settings where settings not in default is n_jobs = 4 and randomstate = 0. Max depth reached a depth of 39. Both figure 12 and figure 13 are similar to the decision trees result. This probably because the dataset is small and makes it less likely to be overfit. Random forest is better than decision tree when they are a big dataset, therefore is this result expected. The randomness of the subsets made the accuracy 0.6% worse than decision tree. RMSE and MAE calculated to 0.550 and 0.621.

```python
from sklearn.ensemble import RandomForestClassifier

RandomForestClassifier(n_estimators=10, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False, class_weight=None)
```

Figure 11: Default settings for random forest

![Random Forest Default Settings](image1)

Figure 12: Probability of a movie will be bought estimated by random forest

![Random Forest Probability](image2)
With different number of trees and max depth we received following results. With a number of estimators to be 60 with depth of 5 gave RMSE and MAE of 0.584 and 0.595, with same estimators and depth 25 RMSE and MAE calculated to 0.548 and 0.614. When limiting the trees estimators to 3 and a depth of 36 RMSE and MAE calculated to 0.560 and 0.620. We can see that number of estimators and depth has slightly impact the RMSE and MAE. Default value had good performance but with number of estimators to be 60 with max depth of 25 we received a small improvement.

5.2.3 SVD

A SVD function was developed. The process of developing a SVD function started with Netflix’s dataset. Netflix’s dataset is a big open-source dataset with over 100 millions of rows. We trimmed the developed SVD function on Netflix dataset to work with Tingstad’s dataset. This was done due to that Tingstad’s data was not available from the start. When substituting a dataset in SVD not much are required to be changed for the algorithm to work.
Figure 14 shows the distribution of a quarter of the total Netflix’s dataset. There are more higher ratings most likely since the viewer do not put effort in a movie that the viewer disliked.

On Netflix’s data estimated the accuracy by using cross-validation with 3 folds, a mean value of MAE was estimated to 0.7642 and RMSE to 0.9669. Before running the SVD function some data pruning was done where the movies with a low number of ratings and user with low number of rating was removed. The pruning was made by summarizing a movie’s total ratings from users and summarizing each customers ratings, a quantile of the lowest 20% was removed. In numbers means that if a movie had less than 3088 ratings, the movie was removed or if a user had given less than 5 ratings, the user was removed, making the total size go down from 1 million to 500 thousand rows. When limiting the data for the SVD function to 10 thousand rows the RMSE increased to 0.9976.

When running the SVD function on Tingstad’s dataset of 1 million rows a RMSE of 1.1586 was estimated. A limited amount of data of 25 thousand rows to SVD function did not inflict a higher RMSE score that was expected, the result was 1.159. There was never time to test this SVD function in real practice with AB-testing. In theory and with a RMSE of 1.159 is this function able to recommend item’s that are in the costumers interest. Due to never get costumer behavior data was it replaced with estimated data from a simulation.

Figure 15: SVD prediction of estimated rating a user would rate from Netflix dataset
6 Discussion

We are pleased with the given result due to having big problems with communication and availability of the data. The functions are able to recommend item's in form of a list. An RMSE of 1.159 was decided to be inside the boundaries of a good RMSE. The SVD trained on Netflixs data preformed with a RMSE of 0.967 compared to Netflix's own function that estimated 0.953 before the competition started 2006 [14]. It is difficult to compare deisision trees RMSE because the range of the estimated outcome has a small range between only 0-1 while SVD on Tingstad has 4 values and Netflix 5 values.

In J. Li and X. Huang article a SVD function is used after a K-mean clustering on Movielens dataset and estimated an RMSE score between 0.8 to 0.86 depending on the amount of neighbors it has [11]. This is a score close to our algorithm tested on Netflix dataset where we estimated 0.96. The difference in score between the algorithms depends substantially on being based on different dataset. In This comparison both dataset use ratings in a range of 1-5. X. Guan et al. used both Netflix and Movielens dataset and got RMSE value of 0.9306 respectively 0.9709 [35]. Our algorithm as mentioned earlier preformed 0.96 on the entire netflix dataset moreover X. Guan et al. used a subset of the dataset when estimated 0.93.

If the gathering of behavior data was available, the results would be more realistic. Simulation of the behavior data is far from perfect. It is simulated in a way that is not close to how reality looks like. Each action that a customer have done, is equal distributed. Meaning that each of the 4 actions has equal amount of data. In reality it is more likely that one of the actions contains more data than the other. For example the action that stands for bought would be less common than the action for clicking on the item.

Stijn Geuens, Kristof Coussement and Koen W. De Bock wrote an article [36] where they propose a framework for companies to help them choose collaborative filtering algorithms. In that article they worked on a synthetic data set that was simulated. They simulated the synthetic data set by trying to mimic already existing open source real life data set. Example on data set is Movielens and Netflix. According to article, higher sparsity levels brought more realism to the data set. To vary the input they used three purchase distributions, exponential, linear and a uniform distribution. Exponential was the most realistic. If to compare our simulation with theirs, our is far less advanced. We took a shortcut with the hope that the real behavior data was to be available before the end of this thesis.

There was never time to implement our function on Tingstad’s website meaning that there is no confirmation if our function would have increased the sales and costumer registrations. Therefor we did not complete the entire purpose of this thesis. We are able to take in a list of item’s and estimate the action each item a customer would interact with. Because it is an estimated score and the program never got online on Tingstad’s website, there is no guarantee that it actually provides any positive results.
7 Conclusion

We did not deliver a program that with machine learning could rearrange a list of item's in a way that would improve sales or registrations. We managed to do a supervised machine learning algorithm that was not fully automated and it was not integrated with Tingstad’s website. Better communication would have improved the time it took to specify the task. Getting earlier access to data, having a working behavior data gathering or already gathered behavior data would have improved the final result. Since this is the first time Tingstad started a bachelor thesis and our previous knowledge about machine learning was low have impacted the result.

Way of improving the algorithm is to take into account how many times an item has been bought earlier of the same costumer. Another interesting thing to investigate is how a decision tree performed better or worse than the SVD. It would be interesting to see if more parameters as customer profile information could improve the recommendations or if a more advance cluster method before using SVD or decision tree could bring better results. Tingstad had no interest in doing any kind of clustering in this thesis and was therefor never done.

Methods that have been used in this thesis is decision tree, random forest and SVD. The used data are from Movielens and Netflix’s open-source dataset and also data from Tingstad. Decision tree and random forest based on Movielens data. SVD was tested with Netflix’s and Tingstad’s data with good result. The SVD function estimated a RMSE of 0.967 on Netflix’s data respectively 1.159 with Tingstad’s data.

Our conclusion is, new knowledge has been received about machine learning. How to use machine learning and develop code around it in the given language Python. Knowledge about recommendation systems and their big role in everyday life. Experience in how it is to co-work with companies. Aware of problems that easily can occur when working with companies, and the importance in keeping a robust continuously communication.
References


Figure 16: Shows flowchart over how the program works.
Figure 17: Shows Gantt scheme over the project