# Data Mining Cup 2022

Course: Data Mining II

presented by Data Anonymous(Team 4)
Jiaxin Yuan, 1820345
Nikita Ivanov, 1576774
Patrick Knab, 1610046
Daniel Kettemann, 1754896
Yunjing Dai, 1820527
Mayank Agarwal,1725206
Xingyu Zhong, 1820539
Subash Poudel, 1717048
Oghenekeno Omogha,1725331
Weiqing Zhu, 1754783

submitted to the
Data and Web Science Group
Prof. Dr. Heiko Paulheim
University of Mannheim

May 2022

# Contents

# Chapter 1

# Data Mining Cup 2022 Report

## 1.1 Introduction

Customer retention and maximizing profits are core objectives of most companies as they aim to remain in business. Importantly, customers are the fuel of every business. Thus understanding customer behavior in regards to observing their buying pattern, favorite and associated products, and having a customer relationship is pivotal. Businesses have adopted different methods for sharing vital information about their products with their customers. One of these methods is a newsletter approach which can be used to engage customers and inform customers about new products. In this project, Pia and Philip, who runs an e-commerce business, have adopted the newsletter to communicate special offers, similar products from previous buys, and new customer additions. However, they realized a shortcoming where they frequently recommended the same products to their clients. To work around this shortcoming, they implemented a filter to exclude certain products from being recommended for a fixed number of days, which also did not eliminate the shortcoming. Thankfully to data mining, learning from data has made it possible to achieve a high standard to make data-centric decisions and help businesses reach their objectives. Using available historical data from Pia and Philip, we aim to implement a model to predict the week a returning customer would likely purchase a product they have frequently purchased. Our model will eliminate the use of a filter on excluding products for a fixed number of days by using the purchase and product history of a customer alongside product feature information for prediction. The prediction output will be a four weeks long period from 01-02-2021 to 28-02-2021, with each week bearing a ”replenishment” and ”no replenishment” label.

The rest of this work is the data exploration section, where we gain first-hand insights into the dataset. Following is the preprocessing step for transforming the raw data for our machine learning algorithms in the modeling section. We finally evaluate the results and performance of our model on different metrics like precision, recall, and F-1, which are commonly used metrics in data mining tasks [1].

## 1.2 Data Exploration

As a first step in our prediction task, we perform exploratory data analysis to visualize and discover insights from the outlook of our dataset. Firstly, we understand the features from the ”Items”, ”Orders” and ”Category” datasets. In the items dataset, we identify ”**itemID**” - *a unique identifier for every product*, ”**brand**” - *brand of the product* as the categorical features and 5 associated numerical product features which fall into a finite category. The category dataset also contains numerical features like ”**category**” - *category identifier for products* ranging from [0,...,4200] and ”**parent_category**” -*a broader (parent_category) associated with a category identifier*. In the orders dataset we have the **date** - *a record for the date of a transaction* which ranges from 2020-06-01 to 2021-01-31, a positive **order** integer for *number of orders of a product at a given date by a specific user*, **itemID** and **userID**, are *identifier for items sold* and *identifier for purchaser of the item* respectively. We explore cyclical information from this dataset, such as user orders, average order/reorder period, and brand loyalty distribution.

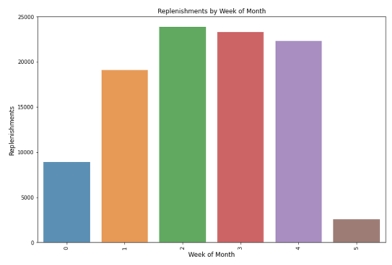


**Figure 1.1: Cyclical effect - Replenishment by week of month**

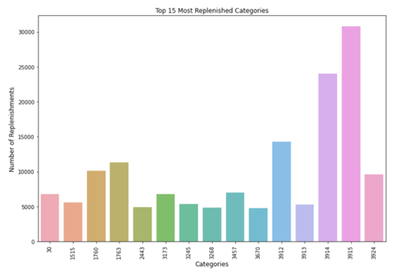


**Figure 1.2: Top categories reordered**

To understand the user buying behavior, we explore periods where users make the most purchases and the number of orders being made by users. In Figure 1.1 we observed that users make more orders in the second and third week of the month. To further understand how a user personalizes a product, we identify to which categories a product a user ordered belongs to in Figure 1.2. Since we want to predict users’ replenishment according to their preferences, we also look into brands users reordered the most. We observe that brands ”186” and ”1496” are reordered the

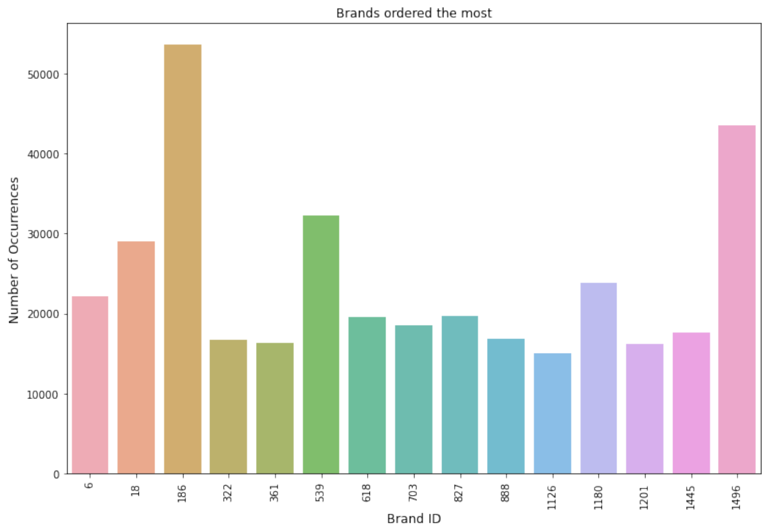


**Figure 1.3: Brand loyalty - Brands reordered the most**

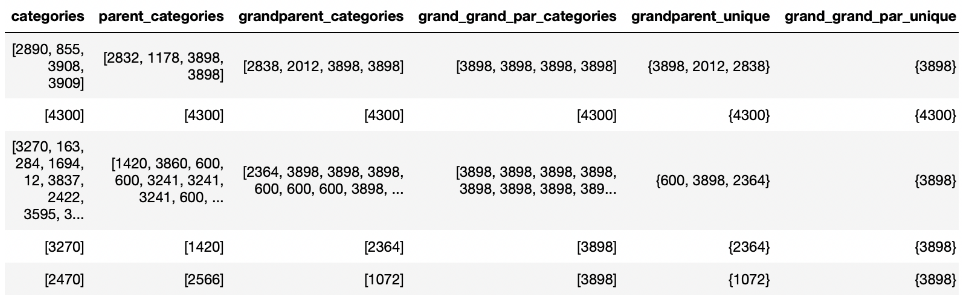
most, which is visualized in Figure 1.3. This could be an indication of brand loyalty, as we often see that as soon as a product satisfies the user's expectation, he/she is more likely to buy the same or similar product of the same brand. With this information from the data exploration, we begin with preprocessing our data, which is the topic of the next section.

## 1.3 Data Prepossessing

There are three distinct datasets: item dataset, order dataset, and category hierarchies dataset. Considering this as a time series problem, we split the ordered dataset into order-before-Jan, order-before-Dec, order-Jan, and order-Dec, the former 2 for training and the latter 2 for testing.

However, there are some issues with the given datasets. Firstly, there are some missing values within column categories in the item dataset. We decided to fill all missing values with the root category [3898] according to our research concerning the category hierarchy of the items. Moreover, some unsupported data types exist in our datasets, like a list of categories from the item dataset and date from the order dataset. The latter was easily solved by extracting latent information and generating more time-concerning features. However, the former is still not perfectly dealt with after we tried a lot.

The first approach we tried was sticking to the hierarchical structure of the categories by incorporating children's categories to their ancestors so that we could retain enough information at a high level in the meantime. Initially, we merged the item and order datasets based on a common feature by itemID. Then, we converted the list of categories into a single element, the first ancestor of all the items in the list. To perform this task, first, we have generated more new lists (parent categories, grandparent categories) by finding their parent category according to the category, each time keep only unique categories, then iterate until the list firstly reaches only one element. While performing this preprocessing step, we found some problems, such as information loss. The method would only work when the items of the list are correlated to each other, and the items should be at the same depth of the hierarchy tree. Otherwise, they will always reach the end node since they do not share a same parent. Figure 1.4 below shows the process of converting a list of elements into one element.



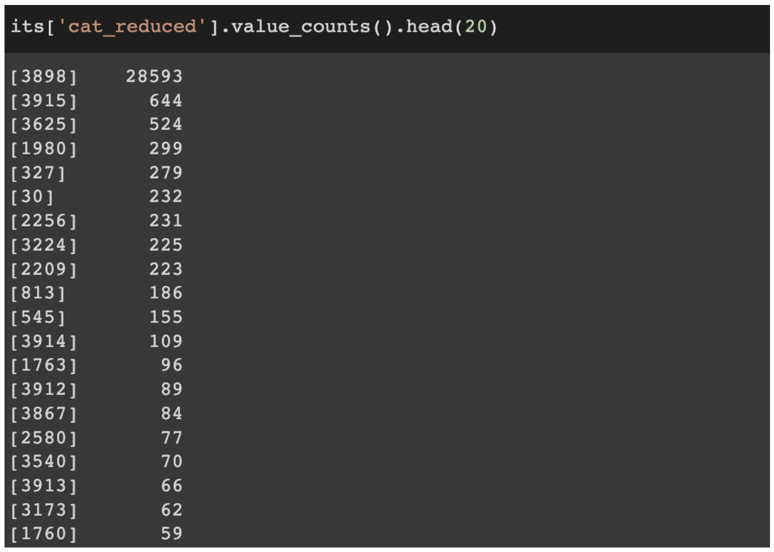
| categories | parent_categories | grandparent_categories | grand_grand_par_categories | grandparent_unique | grand_grand_par_unique |
|---|---|---|---|---|---|
| [2890, 855, 3908, 3909] | [2832, 1178, 3898, 3898] | [2838, 2012, 3898, 3898] | [3898, 3898, 3898, 3898] | {3898, 2012, 2838} | {3898} |
| [4300] | [4300] | [4300] | [4300] | {4300} | {4300} |
| [3270, 163, 284, 1694, 12, 3837, 2422, 3595, 3... | [1420, 3860, 600, 600, 3241, 3241, 3241, 600, ... | [2364, 3898, 3898, 3898, 600, 600, 600, 3898, ... | [3898, 3898, 3898, 3898, 3898, 3898, 3898, 389... | {600, 3898, 2364} | {3898} |
| [3270] | [1420] | [2364] | [3898] | {2364} | {3898} |
| [2470] | [2566] | [1072] | [3898] | {1072} | {3898} |

**Figure 1.4: newly generated columns**

The second method we tried is eliminating the hierarchy structure and just focusing on the most frequent category on the list. Specifically, we reduced the categories list by only retaining categories with which top 50 replenishment rate. We removed the category bought less for reduced lists with more than one element. However, for the lists with no top 50 replenishment rate, we covered them by root node 3898. Because all multiple categories have been transformed to one single element, information loss is inevitable. Moreover, the newly generated column has a strongly imbalanced distribution, with category 3898 taking up the most. The distribution is illustrated below.

### 1.3.1 Feature Engineering

Our dataset did not have complete information about the user activities and order of items based on time, so eight new features were being created, namely user_total_purchase_time, user_total_orders, item_total_purchase_time, item_total_orders,



```
its['cat_reduced'].value_counts().head(20)

[3898]    28593
[3915]      644
[3625]      524
[1980]      299
[327]       279
[30]        232
[2256]      231
[3224]      225
[2209]      223
[813]       186
[545]       155
[3914]      109
[1763]       96
[3912]       89
[3867]       84
[2580]       77
[3540]       70
[3913]       66
[3173]       62
[1760]       59
```

**Figure 1.5: new feature distribution of the 2nd method**

weekday,date_of_month, label, and regression label.

In the following, we will explain those new features in more detail.

1. User_total_purchase_time: We have aggregated the userId and calculated the average total purchase time within a month for each user.

2. User_total_orders: We have calculated each user's total purchase time within a month.

3. Item_total_purchase_time: Total purchase time within a month for each item.

4. Item_total_orders: Average total order of an item within a month.

5. Weekday: We have extracted the weekday from the date column.

6. date_of_month: The day of the month.

7. Regression label: We use the days' interval between the last buying and the current time to calculate the regression label.

8. Label: To generate label features, we used the submission dataset. We have calculated the label as follows: If the predicted date (last buying date and the interval) is not greater than seven, it has the label one. Between 7 to 14, it is the label two; between 14 to 21, it is three; between 21 to 28, four; and if

the value is less than zero and greater than 28, then the corresponding label is zero.

Figure 1.6 shows the new feature which we have previously described.

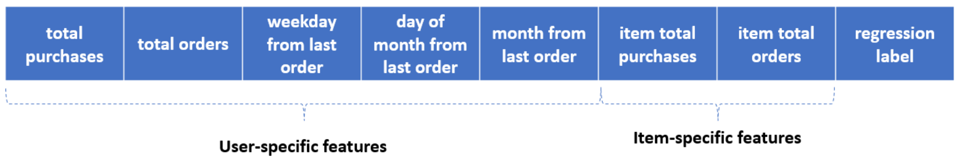


**Figure 1.6: new features**

## 1.4 Modeling and Results

### 1.4.1 Iteration #1

In the first iteration, we preprocessed the data by combining the data from the dataset from the orders, category, and item hierarchies. Our modeling approach was to get a baseline model using basic ML Algorithms such as Naive Bayes, KNN, and Decision tree. We used accuracy as the evaluation measure for the evaluation setup and performed cross-validation to test our models. We also tried under-/oversampling of the minority classes. Since we had an unbalanced dataset, we tried to use both undersampling and oversampling of the respective majority and minority classes to achieve better class balance. We also experimented with SMOTE sampling. However, it added more noisy data to the dataset, so we proceeded with the under-sampling approach. Moreover, our focus was on understanding the cyclical effects of the items and hierarchies in order to model the data. In this regard, we tried reducing dimensions from the dataset by reducing categories and item features and simultaneously generating new features.

#### Feature Engineering and selection #1

Throughout the feature engineering, we primarily used the original columns from the dataset, which provided us with better results as the dataset we were using was a merged dataset; hence, feature selection was based on the columns in the original dataset.

**Models applied #1**

We used KNN, Decision Tree, and Naive Bayes in the first iteration to get a baseline score after the hyperparameter tuning.

**Results #1**

The results of the models with the default parameters:

**Table 1.1: Iteration #1 results**

| **Dataset** | **Naive bayes** | **KNN** | **Decision Tree** |
|---|---|---|---|
| **orders before dec** | 0.356 | 0.30 | 0.31 |
| **orders before jan** | 0.312 | 0.295 | 0.310 |

The results of the models with performed hyperparameter tuning:

**Table 1.2: Iteration #1 results**

| **Dataset** | **Naive bayes** | **KNN** | **Decision Tree** |
|---|---|---|---|
| **orders before dec** | 0.356 | 0.33 | 0.35 |
| **orders before jan** | 0.312 | 0.330 | 0.353 |

In the first iteration, we saw that the decision tree was the best classifier with an accuracy of 0.35 with hyperparameter tuning on the cross-validated dataset.

### 1.4.2 Iteration #2

In the second week, we applied more preprocessing steps to the dataset and hence changed the modeling setup. We understood the cyclical effects as shown in the Figure 1.1. We added new features aimed at getting more user-focussed and item-specific features. In addition, we also used the regression label to perform a regression with these models, and their coefficients were used as a new feature.

**Feature Engineering and selection #2**

For the feature engineering phase, we selected all new features. We tried with the new preprocessed dataset to understand how these new features affect the models and whether we could get better results compared to the first iteration.

**Models applied #2**

We used the same three models as before, Naive Bayes, KNN, and Decision tree, and also played with an AutoML library, AutoKeras, to compare the results with these three existing models.

**Results #2**

The results of the models with the default parameters:

**Table 1.3: Iteration #2 results**

| Dataset | Naive bayes | KNN | Decision Tree |
|---|---|---|---|
| **orders before dec** | 0.378 | 0.318 | 0.273 |
| **orders before jan** | 0.329 | 0.287 | 0.262 |

The results of the models with performed hyperparameter tuning:

**Table 1.4: Iteration #2 results**

| Dataset | Naive bayes | KNN | Decision Tree |
|---|---|---|---|
| **orders before dec** | 0.378 | 0.367 | 0.389 |
| **orders before jan** | 0.329 | 0.320 | 0.344 |

Comparing with the 2nd iteration, as shown in the table 1.3 and 1.4 with new features, this iteration performs better in Naive Bayes and decision tree models. The model's results were better before December than in the January dataset. Moreover, we also used the AutoKeras Neural Network algorithm in AutoML library as a reference model to evaluate the performance of these basic ML models. The results from the AutoKeras model were based on Precision, Recall, F1 score, and accuracy.

Below are the results from the AutoKeras model for each target class:

| Class | Precision | Recall | F1- Score |
|---|---|---|---|
| **Class 0** | 0.98 | 0.84 | 0.90 |
| **Class 1** | 0.07 | 0.30 | 0.11 |
| **Class 2** | 0.09 | 0.31 | 0.14 |
| **Class 3** | 0.09 | 0.27 | 0.14 |
| **Class 4** | 0.11 | 0.22 | 0.13 |

**Table 1.5: Auto ML results**

### 1.4.3 Iteration #3

In the third and final iteration, we used an approach from ensemble learning, specifically stacking algorithms which have performed very well in the last two iterations and evaluating the result. In this iteration, we changed the approach for the generation of training labels which were part of a preprocessing issue in the first two iterations. We split the preprocessed data into a train and test set with a split size of 60: 40. Additionally, we split the train set into a new train and validation set with a ratio of 70:30.

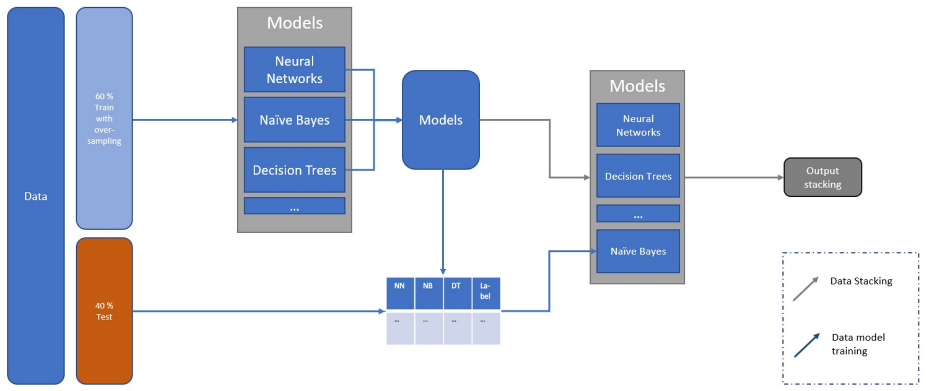


**Figure 1.7: Ensemble approach using Auto Keras**

#### Feature Engineering and selection #3

We used the same number of features as in the last iteration.

#### Models applied #3

We trained a neural network with AutoKeras, Naive Bayes, and a decision tree as base models for our stacking approach.

#### Results #3

The results of the models with the default parameters:

The results of the models with performed hyperparameter tuning:

As shown in table 1.7, the final iteration performs better on decision tree and KNN models. The model's results were better before January after using the evaluation function.

| **Dataset** | **Naive bayes** | **KNN** | **Decision Tree** |
|---|---|---|---|
| **orders before dec** | 0.358 | 0.33 | 0.353 |
| **orders before jan** | 0.312 | 0.457 | 0.405 |

**Table 1.6: Iteration #3 results using default parameters**

| **Dataset** | **Naive bayes** | **KNN** | **Decision Tree** |
|---|---|---|---|
| **orders before dec** | 0.358 | 0.38 | 0.419 |
| **orders before jan** | 0.312 | 0.54 | 0.517 |

**Table 1.7: Iteration #3 results using hyperparameters**

### 1.4.4 Model Evaluation and Results

Upon completing three iterations of classification using different preprocessed datasets, algorithms, and approaches, we saw that the unbalanced data could largely influence the prediction's performance. Our best model was the stacking approach, which we also used for our submission.

## 1.5 Conclusion

Overall, this project taught us a lot about using ML algorithms and preprocessing strategies. A few learning we take away from our project are the following:

- We focused too much on user-based features compared to items or category features.
- We lost a lot of information during the dimension reduction of categories.
- SMOTE with time-series data produced more noise and unusable data than without SMOTE.

**Link to Github Repository**

https://github.com/P4ddyki/Data-Mining2

# Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.