**Instacart Market Basket Analysis**





**Moran Almagor & Shira Frank**

**Table of Contents**

1. **Introduction……………………………………………………………………………3**
2. **Methodology……………………………………………………………………………4**

2.1 Data preparation………………………………………………………………………………..4

2.2 Exploratory data analysis………………………………………..……5

2.3 Clear outcome variable definition………………………………..…..7

2.4 Variable engineering………………………………..…………………8

2.5 Outliers determination and treatment………………………………..9

2.6 Missing values ………………………………………….……..………10

# **Models………………………………………………………………..11**

3.1 Variable selection:………………………………………………......…11

3.2 Data preparation - Train/Dev/Test...………………………………....11

3.3 Model Selection ……………………………………..……………...…14

3.4 Model evaluation- Metrics selection………………………………….15

3.5 Model fine-tuning – Hyperparameters...……………………………..16

3.6 Testing the selected models………………………………………….16

# **4.Conclusion…………………………………………………………………...16**

# **1.Introduction**

Whether we shop from meticulously planned grocery lists or let whimsy guide our grazing, our unique food rituals define who we are. Instacart, a grocery ordering and delivery app, aims to make it easy to fill our refrigerator and pantry with our personal favorites and staples when we need them.

Retail is one of the domains that collects huge amount of transaction data everyday. They need to understand their customer’s purchasing patterns, find insights and behaviors in order to take better business decisions[1], in order to know which orders to make, discounts, sales, and predict what will be the amount of revenues in the near future. Those are valuable also to many E-Commerce websites, such as book sales on amazon.com, DVD rental service on netflix.com etc.  
The prediction based on personalized algorithm[2]. These algorithms model consumer shopping behavior and are used to automatically identify items that are new to the individual consumer, but are likely of interests to them.

The main goal in our project, is to predict a list of suggestion products to each user according to the predictions of the models that we will use.

The prediction[3] will be according to various parameters that are included in the Instacart data. Examples of the parameters are day of the week and time the orders were placed, in what order items were placed in the shopping cart, and which virtual departments and categories the purchased products belong to.

Previous works had been done by using models like "basket-sensitive random walks"[3], "SVD approximations[4]", "Support Vector Machine", "Logistic Regression". The AUC is roughly around 0.64 – 0.68.   
Those are the most common models to predict customer’s purchasing patterns, because they are supervised machine learning algorithm which can be used for both classification or regression problems.

## **2. Methodology**

**2.1 Data preparation**

The source of the data had been taken from a competition that had been held on august, 2017 in kaggle website. The dataset of "Instacart" contains a sample of over 3 million grocery orders from more than 200,000 "Instacart" users, 134 categories, 22 departments.  
The dataset is anonymized and does not contain any customer information, such as: geographic area of the users, real ID number, and the exact dates of purchases due to user privacy.

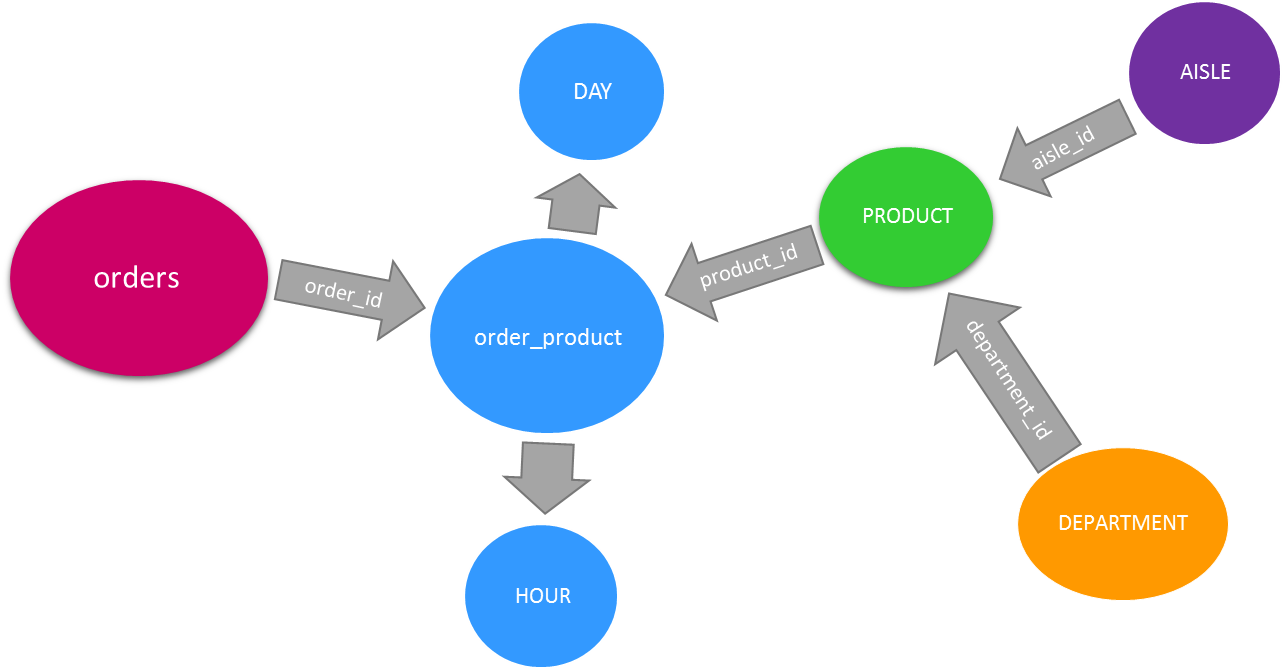
The data is available in 7 csv files :  
orders, departments, aisles ( categories ), products, order\_products\_\_train, order\_products\_\_prior, sample\_submission.

We will focused on 5 tables, that contains the relevant information; the first file contains products information like product id, product name, aisle id (where the product is placed in the store) and the department id. The second file contains orders and it links each order id with the customer who ordered the same. It also contains the day of the week, hour of day and sequence in which the order was placed. The third file contains the products purchased in each order with the sequence in which the product was added to the cart and whether the customer reordered it or not. The fourth and fifth files are the metadata for aisles and departments respectively, containing unique ids and names for each. For each user, there are multiple orders and each order has multiple products[1].  
We crossed the information from the different tables to one big table that contains the most important columns, "flat file" – user\_id, order\_id, order\_number, order\_dow (day of week), order\_hour\_of\_day, day\_since\_prior\_order, reordered, department\_id.

Due to the big size of the dataset that we got, above 30 millions of records, we couldn't succeed to upload all the data into python from SQL server, since of reasons of limited memory, we decided to cut the dataset.

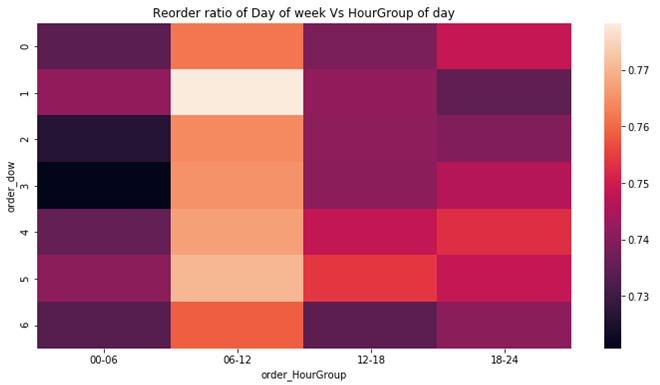
We decided to cut the data by classifying the top 30 products that were sold. Any customers who did not buy from this products list had been deleted, thus we were left with approximately 4.3 million records, instead of 30 million, and the amount of memory required for the project has been decreased significantly, which allows convenient and fast work.

# Entity Relationship Diagram (ERD):

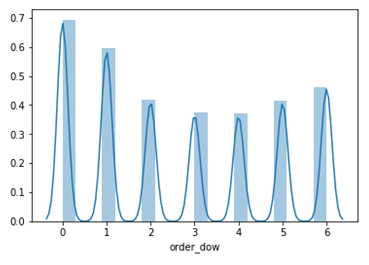


**2.2 Exploratory data analysis**

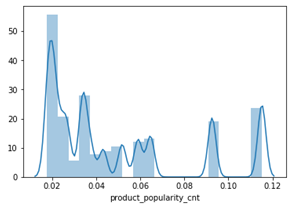
In order to understand the data, we used basic statistics to show, and analysis the different variables. We also demonstrate the data by graphs, which will help us, understand the type of distributions of numeric data and for categorical data.



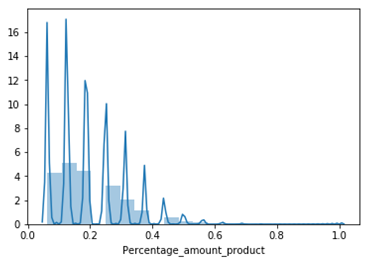
At the beginning, we were cruise to know when the most of the orders had beed done. Due to, we use the "sns.heatmap":   
from this plot, we can figure out, that the most popularly hours are from the morning, until noon (0-12).In those hours, and days, are most of the people ordering food.



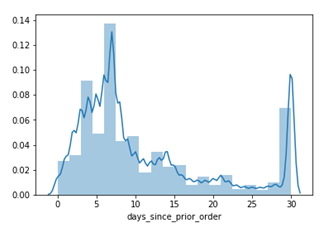
By using the command "sns.distplot (df.order\_dow…)", we could also learn about the most popularly day of orders, it's day "0", meaning to sunday.



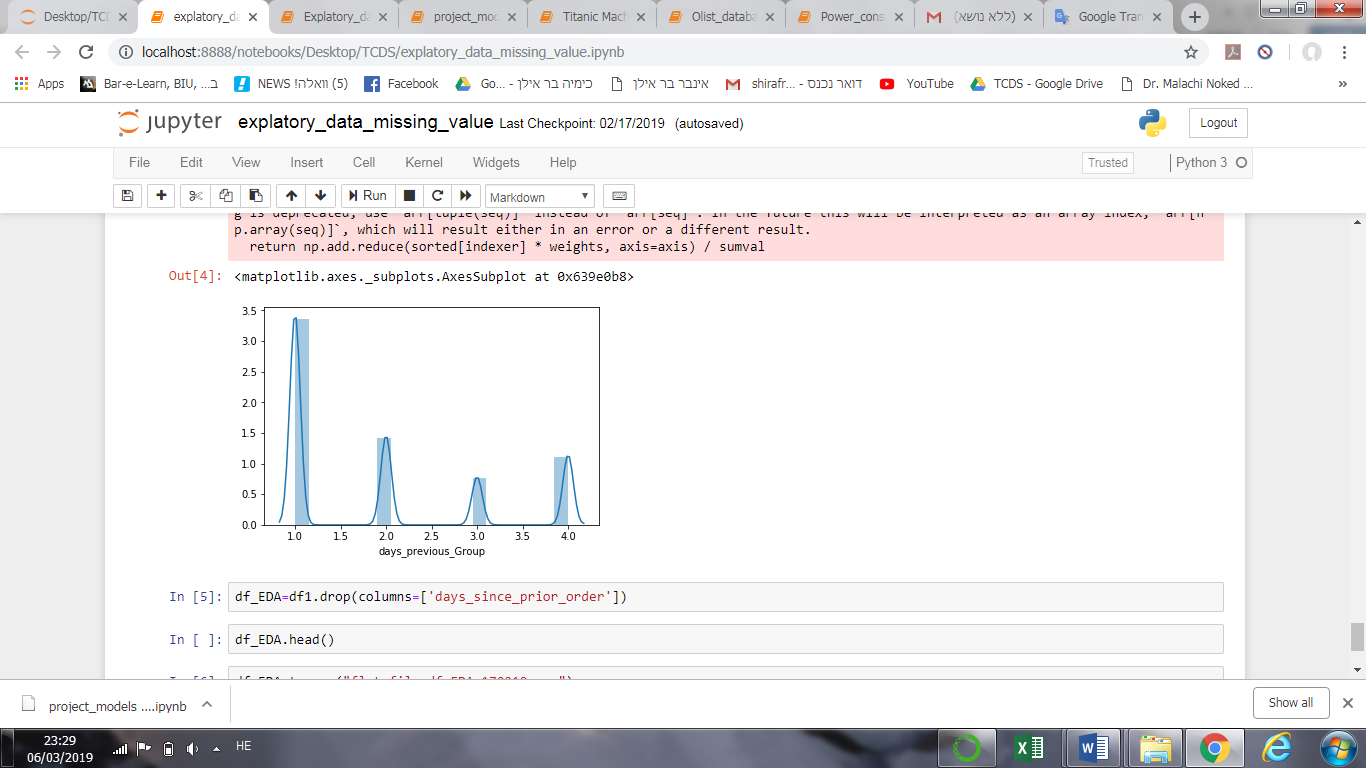
Since we wanted to enrich our data, as will be mention below, we added new column "product\_popularity\_cnt", which describes in percentages the popularity of product\_id, how many customers orders those products.



Other column that we added was "Percentage\_amount\_product", which describe in percentages the amount of products that user\_id purchase per order divided to the maximum products purchased per order (i.e 16 products).   
We could consume, that fewer customers orders big amount of products at each order.



This graph presents, the days since prior order.  
We can figure out that most of the orders were been done about a 7 days after the previous purchase.

We divided the days from the last purchase into groups in a new column, for arrange and enrich the data. We noted that most of the orders were made in the first week since the last order (1 = first week, etc) per user\_id.

**2.3 Clear outcome variable definition**

The outcome will be suggested products for each user according to the recommendations of a predictive model. The amount of the suggested products will be determined by analyzing and processing the dataset. It's will be between 1-5 products per user, according to their purchases.

**2.4 Variable engineering**

In order to enrich our data, we added six more columns as follows:

|  |  |  |
| --- | --- | --- |
| Serial number | Column name | Description |
| 1 | product\_popularity\_cnt | Describes in percentages the popularity of product\_id . |
| 2 | department\_popularity\_cnt | Describes in percentages the popularity of department\_id.. |
| 3 | IsOrganic | Describes if the product is organic or not, by boolean variable (no = 0/ yes = 1). |
| 4 | order\_HourGroup | Describes the HourGroup which time the order had been made, after dividing the hours into 3 categories according to the number of orders made per hour. |
| 5 | Percentage\_amount\_product | Describe in percentages the amount of products that user\_id purchased per order divided to the maximum products purchased per order (i.e 16 products) |
| 6 | days\_previous\_Group | Describe the column "days\_since\_prior\_order", by using the function "HGroups\_days", which represent how much time passed from the last order, by dividing the time that pass between one order to another per user\_id into 4 categories (1 = first week, etc.). |

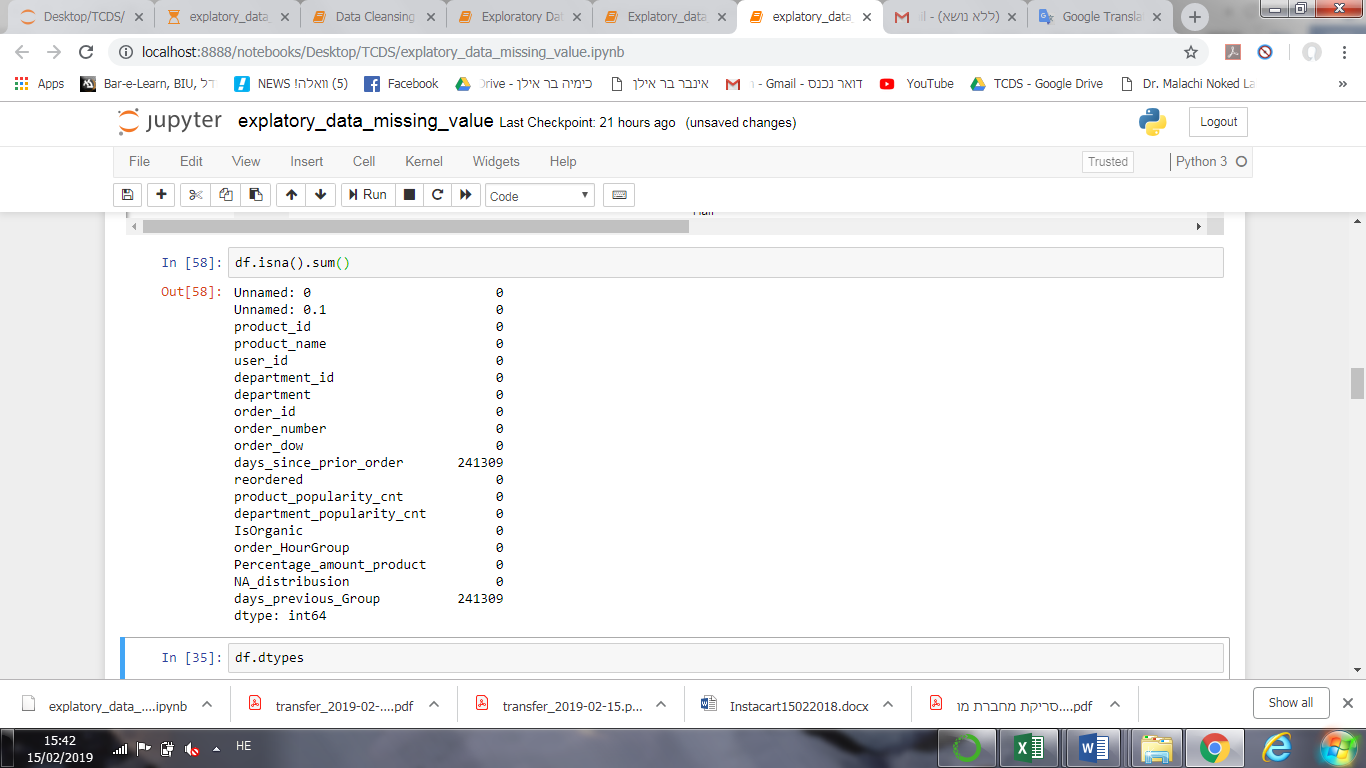
1. **product\_popularity\_cnt :** we count by the function "df.groupby", all of the product which customers bought per product\_id, and then those were divided by the total amount of the products, that people bought.
2. **department\_popularity\_cnt** : we count by the function "df.groupby", all of the product which customers bought per department\_id, and then those were divided by the total amount of the products, that people bought in all of those departments.
3. **IsOrganic** : we looked for the string "Organic" in column product\_name, and if this word was written, the number "1", represent it in the column IsOrganic.
4. **order\_HourGroup** :this columndescribes the HourGroup which the order had been made ,after dividing the hours into 3 categories according to the number of orders made per hour, by using function " hourgroup", that scanning all the data, and divided it, by definition.
5. **Percentage\_amount\_product** : we count all the amount of products that user\_id purchased per order, divided to the maximum products purchased per order. This column give "grades" per user\_id, and let us know if those use\_id customers who buy frequently, and in big quantity.
6. **days\_previous\_Group** : this column, Describe the days\_since\_prior\_order, by using the function "HGroups\_days", which represent how much time passed from the last order, by dividing the time that pass between one order to another per user\_id into 4 categories (1 = first week, etc.).

**2.5 Outliers determination and treatment**

In order to identify outliers we used unsupervised cluster analysis - DBSCAN – Density-based spatial clustering of applications with noise, is a data clustering algorithm, that determine the minimum number of points in a cluster (numPts) and the radius of our neighborhoods around a data point p (eps), and search for a random point having at least numPts inside a radius eps. If found, define as a core point. Search for new random points not in a cluster having at least numPts inside a radius eps. When found all the core points expand to the border points. Points that are not included, defined as outliers. In this project, It's calculates the outliersof selected variables (i.e 'order\_dow', 'days\_since\_prior\_order', 'product\_popularity\_cnt' and 'Percentage\_amount\_product') with extreme values and without them in order to measure their degree of influence on the dataset.

Unfortunately due to our big dataset we didn't succeed to run the dbscan but we verified using sql commends that their aren't any outliers in our data.

**2.6 Missing values**

In order to identify the missing values, we use the command ".isna().sum()", and we noted that all the missing values, are in the column 'days\_since\_prior\_order', and therefore also in the column 'days\_previous\_Group', because it is derived from the same original column that contains the missing values, as mentioned above.  
It can be concluded that for those customers, who have not previously purchased in 'Instacart', there are missing values in the columns 'days\_since\_prior\_order' and 'days\_previous\_Group'. Because the value "0"' did not appear in those column, instead it is wrote "NaN"/"None". The meaning are that those missing values are for customers that this is their first order.

Thus, we wanted to know the distributions of those purchases, for customers who bought frequently, and for those who bought at the first time.   
For this goal, we divided the dataset into two groups. The first contains all the customers who bought frequently (they have values in the columns 'days\_since\_prior\_order' and 'days\_previous\_Group'), and the second contains all the customers who bought at the first time.

We divided the dataset by adding new columns " NA\_distribusion", that contains "0" for missing values in the column 'days\_since\_prior\_order', and "1" for the others. According to this, we could divided the dataset to two groups.  
The distribution's plots show almost the same values of amount of purchases, and very similar means. The mean's gap between the two groups is 3.2%, therefore, we decided to delete the rows that contains the missing values, because it will not affect the results, on the calculations.

After deleting the chosen rows, we delete the columns " NA\_distribusion", because it's not relevant anymore.

# **3.Models:**

# **3.1 Variable selection:**

Our table contains 38 columns; 7 columns describes important various features of the data, and all the other columns are the 30 products that we will predict by using different models. Therefore, we will consider all the columns in our models, and won't do variable selection.

# **3.2 Data preparation - Train/Dev/Test:**

For division data, we split the data to train (80%), test (10%) and dev (10%). The columns - 'reordered', 'product\_popularity\_cnt', 'department\_popularity\_cnt', 'IsOrganic', 'order\_HourGroup', 'days\_previous\_Group', 'Percentage\_amount\_product', will be "X" part in the models, and all the 30 products will be consider as "y".

In order to checking the distributions of the partitions, and that there

are no significant differences between them, we will use "Table 1".

Hence, we added new columns "division\_data", which include the group

it belongs (train, dev, test).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **s.n** | **Variables** | **Categories** | **Population** | **Category\_test** | **Category\_dev** | **Category\_train** | **p\_value** |
| 0 | Individuals | n | 4025124 | 402512 | 402512 | 3220100 |  |
| 1 | reordered | 1 | 821,525.00 ( 20.40%) | 81,799.00 ( 20.30%) | 82,036.00 ( 20.40%) | 657,690.00 ( 20.40%) |  |
| 2 | reordered | 0 | 3,203,599.00 ( 79.60%) | 320,713.00 ( 79.70%) | 320,476.00 ( 79.60%) | 2,562,410.00 ( 79.60%) | 0.281 |
| 3 | product\_popularity\_cnt | Mean (SD) | 0.00 ( 0.00) | 0.00 ( 0.00) | 0.00 ( 0.00) | 0.00 ( 0.00) | 0.413 |
| 4 | product\_popularity\_cnt | Median (IQR) | 0.00 ( 0.00- 0.10) | 0.00 ( 0.00- 0.10) | 0.00 ( 0.00- 0.10) | 0.00 ( 0.00- 0.10) |  |
| 6 | department\_popularity\_cnt | 0.92943426 | 74,812.00 ( 1.90%) | 7,591.00 ( 1.90%) | 7,395.00 ( 1.80%) | 59,826.00 ( 1.90%) |  |
| 7 | department\_popularity\_cnt | 0.051991675 | 210,496.00 ( 5.20%) | 20,994.00 ( 5.20%) | 21,220.00 ( 5.30%) | 168,282.00 ( 5.20%) | 0.373 |
| 8 | department\_popularity\_cnt | 0.018574064 | 3,739,816.00 ( 92.90%) | 373,927.00 ( 92.90%) | 373,897.00 ( 92.90%) | 2,991,992.00 ( 92.90%) |  |
| 9 | IsOrganic | TRUE | 1,292,140.00 ( 32.10%) | 129,789.00 ( 32.20%) | 129,389.00 ( 32.10%) | 1,032,962.00 ( 32.10%) |  |
| 10 | IsOrganic | FALSE | 2,732,984.00 ( 67.90%) | 272,723.00 ( 67.80%) | 273,123.00 ( 67.90%) | 2,187,138.00 ( 67.90%) | 0.085 |
| 11 | order\_HourGroup | דצמ-18 | 69,806.00 ( 1.70%) | 6,967.00 ( 1.70%) | 6,985.00 ( 1.70%) | 55,854.00 ( 1.70%) |  |
| 12 | order\_HourGroup | 06-דצמ | 1,368,673.00 ( 34.00%) | 136,657.00 ( 34.00%) | 137,393.00 ( 34.10%) | 1,094,623.00 ( 34.00%) | 0.368 |
| 13 | order\_HourGroup | 18-24 | 1,880,181.00 ( 46.70%) | 188,154.00 ( 46.70%) | 187,987.00 ( 46.70%) | 1,504,040.00 ( 46.70%) |  |
| 14 | order\_HourGroup | 00-06 | 706,464.00 ( 17.60%) | 70,734.00 ( 17.60%) | 70,147.00 ( 17.40%) | 565,583.00 ( 17.60%) |  |
| 15 | days\_previous\_Group | 1 | 2,031,423.00 ( 50.50%) | 202,995.00 ( 50.40%) | 203,043.00 ( 50.40%) | 1,625,385.00 ( 50.50%) |  |
| 16 | days\_previous\_Group | 2 | 2,031,423.00 ( 50.50%) | 202,995.00 ( 50.40%) | 203,043.00 ( 50.40%) | 1,625,385.00 ( 50.50%) | 0.857 |
| 17 | days\_previous\_Group | 4 | 859,418.00 ( 21.40%) | 86,158.00 ( 21.40%) | 86,005.00 ( 21.40%) | 687,255.00 ( 21.30%) |  |
| 18 | days\_previous\_Group | 3 | 461,621.00 ( 11.50%) | 45,999.00 ( 11.40%) | 46,043.00 ( 11.40%) | 369,579.00 ( 11.50%) |  |
| 19 | Percentage\_amount\_product | Mean (SD) | 0.20 ( 0.10) | 0.20 ( 0.10) | 0.20 ( 0.10) | 0.20 ( 0.10) | 0.707 |
| 20 | Percentage\_amount\_product | Median (IQR) | 0.20 ( 0.10- 0.20) | 0.20 ( 0.10- 0.20) | 0.20 ( 0.10- 0.20) | 0.20 ( 0.10- 0.20) |  |
| 22 | 4605 | 0 | 3,954,352.00 ( 98.20%) | 395,411.00 ( 98.20%) | 395,452.00 ( 98.20%) | 3,163,489.00 ( 98.20%) |  |
| 23 | 4605 | 1 | 70,772.00 ( 1.80%) | 7,101.00 ( 1.80%) | 7,060.00 ( 1.80%) | 56,611.00 ( 1.80%) | 0.939 |
| 24 | 4920 | 0 | 3,943,134.00 ( 98.00%) | 394,193.00 ( 97.90%) | 394,237.00 ( 97.90%) | 3,154,704.00 ( 98.00%) |  |
| 25 | 4920 | 1 | 81,990.00 ( 2.00%) | 8,319.00 ( 2.10%) | 8,275.00 ( 2.10%) | 65,396.00 ( 2.00%) | 0.211 |
| 26 | 5876 | 0 | 3,939,206.00 ( 97.90%) | 393,852.00 ( 97.80%) | 393,958.00 ( 97.90%) | 3,151,396.00 ( 97.90%) |  |
| 27 | 5876 | 1 | 85,918.00 ( 2.10%) | 8,660.00 ( 2.20%) | 8,554.00 ( 2.10%) | 68,704.00 ( 2.10%) | 0.692 |
| 28 | 8277 | 0 | 3,942,158.00 ( 97.90%) | 394,202.00 ( 97.90%) | 394,241.00 ( 97.90%) | 3,153,715.00 ( 97.90%) |  |
| 29 | 8277 | 1 | 82,966.00 ( 2.10%) | 8,310.00 ( 2.10%) | 8,271.00 ( 2.10%) | 66,385.00 ( 2.10%) | 0.949 |
| 30 | 13176 | 0 | 3,649,352.00 ( 90.70%) | 364,717.00 ( 90.60%) | 365,137.00 ( 90.70%) | 2,919,498.00 ( 90.70%) |  |
| 31 | 13176 | 1 | 375,772.00 ( 9.30%) | 37,795.00 ( 9.40%) | 37,375.00 ( 9.30%) | 300,602.00 ( 9.30%) | 0.273 |
| 32 | 16797 | 0 | 3,884,339.00 ( 96.50%) | 388,356.00 ( 96.50%) | 388,299.00 ( 96.50%) | 3,107,684.00 ( 96.50%) |  |
| 33 | 16797 | 1 | 140,785.00 ( 3.50%) | 14,156.00 ( 3.50%) | 14,213.00 ( 3.50%) | 112,416.00 ( 3.50%) | 0.335 |
| 34 | 19057 | 0 | 3,950,565.00 ( 98.10%) | 395,153.00 ( 98.20%) | 395,116.00 ( 98.20%) | 3,160,296.00 ( 98.10%) |  |
| 35 | 19057 | 1 | 74,559.00 ( 1.90%) | 7,359.00 ( 1.80%) | 7,396.00 ( 1.80%) | 59,804.00 ( 1.90%) | 0.334 |
| 36 | 21137 | 0 | 3,766,011.00 ( 93.60%) | 376,801.00 ( 93.60%) | 376,551.00 ( 93.60%) | 3,012,659.00 ( 93.60%) |  |
| 37 | 21137 | 1 | 259,113.00 ( 6.40%) | 25,711.00 ( 6.40%) | 25,961.00 ( 6.40%) | 207,441.00 ( 6.40%) | 0.392 |
| 38 | 21616 | 0 | 3,953,998.00 ( 98.20%) | 395,386.00 ( 98.20%) | 395,443.00 ( 98.20%) | 3,163,169.00 ( 98.20%) |  |
| 39 | 21616 | 1 | 71,126.00 ( 1.80%) | 7,126.00 ( 1.80%) | 7,069.00 ( 1.80%) | 56,931.00 ( 1.80%) | 0.855 |
| 40 | 21903 | 0 | 3,788,367.00 ( 94.10%) | 379,066.00 ( 94.20%) | 378,758.00 ( 94.10%) | 3,030,543.00 ( 94.10%) |  |
| 41 | 21903 | 1 | 236,757.00 ( 5.90%) | 23,446.00 ( 5.80%) | 23,754.00 ( 5.90%) | 189,557.00 ( 5.90%) | 0.25 |
| 42 | 22935 | 0 | 3,913,235.00 ( 97.20%) | 391,469.00 ( 97.30%) | 391,196.00 ( 97.20%) | 3,130,570.00 ( 97.20%) |  |
| 43 | 22935 | 1 | 111,889.00 ( 2.80%) | 11,043.00 ( 2.70%) | 11,316.00 ( 2.80%) | 89,530.00 ( 2.80%) | 0.179 |
| 44 | 24852 | 0 | 3,563,367.00 ( 88.50%) | 356,376.00 ( 88.50%) | 356,463.00 ( 88.60%) | 2,850,528.00 ( 88.50%) |  |
| 45 | 24852 | 1 | 461,757.00 ( 11.50%) | 46,136.00 ( 11.50%) | 46,049.00 ( 11.40%) | 369,572.00 ( 11.50%) | 0.773 |
| 46 | 24964 | 0 | 3,917,801.00 ( 97.30%) | 391,681.00 ( 97.30%) | 391,897.00 ( 97.40%) | 3,134,223.00 ( 97.30%) |  |
| 47 | 24964 | 1 | 107,323.00 ( 2.70%) | 10,831.00 ( 2.70%) | 10,615.00 ( 2.60%) | 85,877.00 ( 2.70%) | 0.324 |
| 48 | 26209 | 0 | 3,887,062.00 ( 96.60%) | 388,527.00 ( 96.50%) | 388,793.00 ( 96.60%) | 3,109,742.00 ( 96.60%) |  |
| 49 | 26209 | 1 | 138,062.00 ( 3.40%) | 13,985.00 ( 3.50%) | 13,719.00 ( 3.40%) | 110,358.00 ( 3.40%) | 0.218 |
| 50 | 27845 | 0 | 3,889,534.00 ( 96.60%) | 389,140.00 ( 96.70%) | 388,908.00 ( 96.60%) | 3,111,486.00 ( 96.60%) |  |
| 51 | 27845 | 1 | 135,590.00 ( 3.40%) | 13,372.00 ( 3.30%) | 13,604.00 ( 3.40%) | 108,614.00 ( 3.40%) | 0.221 |
| 52 | 27966 | 0 | 3,888,900.00 ( 96.60%) | 388,829.00 ( 96.60%) | 388,896.00 ( 96.60%) | 3,111,175.00 ( 96.60%) |  |
| 53 | 27966 | 1 | 136,224.00 ( 3.40%) | 13,683.00 ( 3.40%) | 13,616.00 ( 3.40%) | 108,925.00 ( 3.40%) | 0.856 |
| 54 | 28204 | 0 | 3,937,665.00 ( 97.80%) | 393,738.00 ( 97.80%) | 393,756.00 ( 97.80%) | 3,150,171.00 ( 97.80%) |  |
| 55 | 28204 | 1 | 87,459.00 ( 2.20%) | 8,774.00 ( 2.20%) | 8,756.00 ( 2.20%) | 69,929.00 ( 2.20%) | 0.939 |
| 56 | 30391 | 0 | 3,944,440.00 ( 98.00%) | 394,549.00 ( 98.00%) | 394,464.00 ( 98.00%) | 3,155,427.00 ( 98.00%) |  |
| 57 | 30391 | 1 | 80,684.00 ( 2.00%) | 7,963.00 ( 2.00%) | 8,048.00 ( 2.00%) | 64,673.00 ( 2.00%) | 0.426 |
| 58 | 37646 | 0 | 3,952,960.00 ( 98.20%) | 395,174.00 ( 98.20%) | 395,302.00 ( 98.20%) | 3,162,484.00 ( 98.20%) |  |
| 59 | 37646 | 1 | 72,164.00 ( 1.80%) | 7,338.00 ( 1.80%) | 7,210.00 ( 1.80%) | 57,616.00 ( 1.80%) | 0.312 |
| 60 | 39275 | 0 | 3,927,200.00 ( 97.60%) | 392,798.00 ( 97.60%) | 392,649.00 ( 97.50%) | 3,141,753.00 ( 97.60%) |  |
| 61 | 39275 | 1 | 97,924.00 ( 2.40%) | 9,714.00 ( 2.40%) | 9,863.00 ( 2.50%) | 78,347.00 ( 2.40%) | 0.558 |
| 62 | 40706 | 0 | 3,942,495.00 ( 97.90%) | 394,242.00 ( 97.90%) | 394,264.00 ( 98.00%) | 3,153,989.00 ( 97.90%) |  |
| 63 | 40706 | 1 | 82,629.00 ( 2.10%) | 8,270.00 ( 2.10%) | 8,248.00 ( 2.00%) | 66,111.00 ( 2.10%) | 0.983 |
| 64 | 42265 | 0 | 3,948,612.00 ( 98.10%) | 394,962.00 ( 98.10%) | 394,802.00 ( 98.10%) | 3,158,848.00 ( 98.10%) |  |
| 65 | 42265 | 1 | 76,512.00 ( 1.90%) | 7,550.00 ( 1.90%) | 7,710.00 ( 1.90%) | 61,252.00 ( 1.90%) | 0.396 |
| 66 | 44632 | 0 | 3,950,312.00 ( 98.10%) | 394,921.00 ( 98.10%) | 395,117.00 ( 98.20%) | 3,160,274.00 ( 98.10%) |  |
| 67 | 44632 | 1 | 74,812.00 ( 1.90%) | 7,591.00 ( 1.90%) | 7,395.00 ( 1.80%) | 59,826.00 ( 1.90%) | 0.264 |
| 68 | 45007 | 0 | 3,920,463.00 ( 97.40%) | 392,039.00 ( 97.40%) | 392,010.00 ( 97.40%) | 3,136,414.00 ( 97.40%) |  |
| 69 | 45007 | 1 | 104,661.00 ( 2.60%) | 10,473.00 ( 2.60%) | 10,502.00 ( 2.60%) | 83,686.00 ( 2.60%) | 0.926 |
| 70 | 45066 | 0 | 3,946,644.00 ( 98.10%) | 394,714.00 ( 98.10%) | 394,606.00 ( 98.00%) | 3,157,324.00 ( 98.10%) |  |
| 71 | 45066 | 1 | 78,480.00 ( 1.90%) | 7,798.00 ( 1.90%) | 7,906.00 ( 2.00%) | 62,776.00 ( 1.90%) | 0.683 |
| 72 | 47209 | 0 | 3,815,353.00 ( 94.80%) | 381,653.00 ( 94.80%) | 381,646.00 ( 94.80%) | 3,052,054.00 ( 94.80%) |  |
| 73 | 47209 | 1 | 209,771.00 ( 5.20%) | 20,859.00 ( 5.20%) | 20,866.00 ( 5.20%) | 168,046.00 ( 5.20%) | 0.438 |
| 74 | 47626 | 0 | 3,872,693.00 ( 96.20%) | 387,121.00 ( 96.20%) | 387,163.00 ( 96.20%) | 3,098,409.00 ( 96.20%) |  |
| 75 | 47626 | 1 | 152,431.00 ( 3.80%) | 15,391.00 ( 3.80%) | 15,349.00 ( 3.80%) | 121,691.00 ( 3.80%) | 0.246 |
| 76 | 47766 | 0 | 3,856,087.00 ( 95.80%) | 385,688.00 ( 95.80%) | 385,739.00 ( 95.80%) | 3,084,660.00 ( 95.80%) |  |
| 77 | 47766 | 1 | 169,037.00 ( 4.20%) | 16,824.00 ( 4.20%) | 16,773.00 ( 4.20%) | 135,440.00 ( 4.20%) | 0.409 |
| 78 | 49235 | 0 | 3,950,218.00 ( 98.10%) | 394,890.00 ( 98.10%) | 394,896.00 ( 98.10%) | 3,160,432.00 ( 98.10%) |  |
| 79 | 49235 | 1 | 74,906.00 ( 1.90%) | 7,622.00 ( 1.90%) | 7,616.00 ( 1.90%) | 59,668.00 ( 1.90%) | 0.061 |
| 80 | 49683 | 0 | 3,932,073.00 ( 97.70%) | 393,200.00 ( 97.70%) | 393,089.00 ( 97.70%) | 3,145,784.00 ( 97.70%) |  |
| 81 | 49683 | 1 | 93,051.00 ( 2.30%) | 9,312.00 ( 2.30%) | 9,423.00 ( 2.30%) | 74,316.00 ( 2.30%) | 0.417 |

This table demonstrate if there is correlation between variables. One of   
 the most important conclusion from this table that there are no variables

with a p-value lower than 0.05. From P value, we can figure out the

relationship between the various variables. It is necessary, that there

are no connections between the columns, and the division done

randomly.

We separated the data to 6 parts: X\_train, Y\_train, X\_dev, Y\_dev,   
 X\_test, Y\_test. In order to insert the data into models, we transformed   
 the column 'order\_HourGroup', by using the function   
 "one\_hot\_encoding" from rows to separate columns and change the

columns type from string to integer.

# **3.3 Model Selection:**

Since our prediction, it is suggested products from 30 known products, for each user according to the recommendations of a predictive model, our Y\_prediction combine from 30 columns.   
The outcome from the models it's array, 30 columns for each product, and rows as the number of the customers. This kind of outcome classify our data as multiclass, and there are just few models, that supports this type of data, such as:

sklearn.linear\_model.RidgeClassifierCV, sklearn.tree.DecisionTreeClassifier,  
sklearn.tree.ExtraTreeClassifier,  
sklearn.ensemble.ExtraTreesClassifier,  
sklearn.ensemble.RandomForestClassifier,  
sklearn.neighbors.KNeighborsClassifier,  
sklearn.neighbors.RadiusNeighborsClassifier.

In our project, we decided to focus on 7 of the models that mentioned above.

# **3.4 Model evaluation- Metrics selection:**

In order to test the different types of models, the input information for the trains was the "X\_dev" on our 7 selected models. We measured their accuracy by unique function in order to figure out, which predict it the most precise outcome, which product will purchase by the specific user\_id. The results from the models:

|  |  |
| --- | --- |
| ***model*** | ***accuracy*** |
| Logistic Regression | 0.4404812775768176 |
| Ridge - Logistic Regression | 0.24100151051397226 |
| Decision Tree Classifier | 1.0 |
| Extra Tree Classifier | 0.9995205111897285 |
| Ensemble-Extra Trees Classifier | 0.9995801367412649 |
| Ensemble-Random Forest Classifier | 0.9995801367412649 |
| Neighbors KNeighbors Classifier | 0.9965019676432006 |

From this table we can conclude which models are the best to predict the correct outcome for this problem.

The most suitable models are –" Decision Tree Classifier ","Extra Tree Classifier", " Ensemble-Extra Trees Classifier", "Ensemble-Random Forest Classifier" and "Neighbors KNeighbors Classifier".  
Their accuracy is the highest.

Due to our type of data – multiclass, the column 'accuracy' combines all the calculations of the accuracy and the error.

# **3.5 Model fine-tuning - hyperparameters:**

Since in part of our models we reached to a high level of accuracy (99-100%) there is no fundamental need to improve to model.

# **3.6 Testing the selected models:**

On the selected models as mentioned above in **paragraph 3.4**, the input information was "X\_test" and we measured the accuracy in order to figure out which models predict the most precise outcome.

|  |  |
| --- | --- |
| ***model*** | ***accuracy*** |
| Decision Tree Classifier | 1.0 |
| Extra Tree Classifier | 0.9999279524585603 |
| Ensemble-Extra Trees Classifier | 0.9999006240807727 |
| Ensemble-Random Forest Classifier | 0.9999006240807727 |
| Neighbors KNeighbors Classifier | 0.9994236196684819 |

As we can see, the chosen models that predict the most accurate   
 results it is "Decision Tree Classifier".

# **Conclusion:**

We find that the most accurate model is "Decision Tree Classifier".

We analyzed our data and used multiclass models in order to create suggested products for each user according to the recommendations of a predictive model.

Our unique food rituals define who we are. Our main goal is aiming to make it easy to fill your refrigerator and pantry with your personal favorites and staples when you need them.

We can assume from our research that a man is a man of habits and pattern, which can be, predict from advance including his grocery list and purchase time therefore, there is no need to invest in exposing the customer to new products that are not in his comfort zone.

References:

[1] R. Priya, “RETAIL DATA ANALYTICS USING GRAPH,” 2018.

[2] B. Dias, “Grocery Shopping Recommendations Based on Basket-Sensitive Random Walk,” no. January, 2009.

[3] A. Flores-lopez, S. Perry, and P. Bhargava, “What ’ s for Dinner ? Recommendations in Online Grocery Shopping,” vol. xx, no. xx, pp. 1–6, 2017.

[4] N. Sano, N. Machino, K. Yada, and T. Suzuki, “Recommendation system for grocery store considering data sparsity,” *Procedia Comput. Sci.*, vol. 60, no. 1, pp. 1406–1413, 2015.