**MSc in Data Science and Management – Spring Semester 2022   
LUISS Guido Carli University - Rome, Italy**

**Machine Learning Course**

**Algorithmic Approach for NTT DATA Business Challenge**

**TEAM 4**

**Matteo Gioia - Vincenzo Junior Striano - Fabiana Caccavale - Marco Amadori - Claudia Imperatore**

**Table of Contents**

* Introduction
* Task 1- Customer Segmentation
* Task 2 – Recommendation Systems
  + Task 2.1 – Popularity-based recommendation system
  + Task 2.2 – Collaborative Filtering
  + Task 2.3 – Market Basket Analysis
* Task 3 – Delivery Time Analysis

**Introduction**

In March 2022, NTT DATA challenged the class with a business game to solve using Machine Learning. We’re going to dive in our models and the reasons which have driven our algorithmic choices.

The client is the CMO of an e-commerce which aggregates smaller independent sellers operating in Spain. We acted like a consulting team for NTT to build machine learning models to increase the performances and profits, leveraging three key activities:

* Customers’ clustering: segmenting customers based on similar features to create ad hoc marketing strategies.
* Recommendation System: algorithms to predict which are the products each user can be interested in purchasing.
* Delivery Time Analysis: to predict an accurate date of delivery for customers and getting insights on the drivers which influence the shipping time.

**Task 1 – Customer Segmentation**

In the present work, clustering means aggregating customers based on their similarity in features. For each cluster, we created a tailored marketing strategy. We tried multiple approaches:

* Traditional RFM (Recency-Frequency-Money):
  + Based on quantiles and RFM Score
* Machine Learning RFM:
  + Exploiting Unsupervised Machine Learning algorithm like K-Means clustering, euclidean distance will be used to choose the right number of clusters.
* K-Means Clustering over a set of features + RFM

The recency about a customer tells how much time has passed since the last purchase, the frequency tells how many times the customers has bought from the commerce in a certain amount of time, and the money are the revenues from the customer.

Traditional RFM split the cloud of customers in a predefined number of clusters assigning them a predefined label. RFM K-Means clustering using the elbow method allows to set the number of clusters according to the specific dataset we’re dealing with.

In “pure” K-Means clustering we selected some behavioral, purchasing features of the customers + RFM.

The limit we found out with RFM is that the Frequency component is one-inflated, positively skewed, which may result in misleading results.

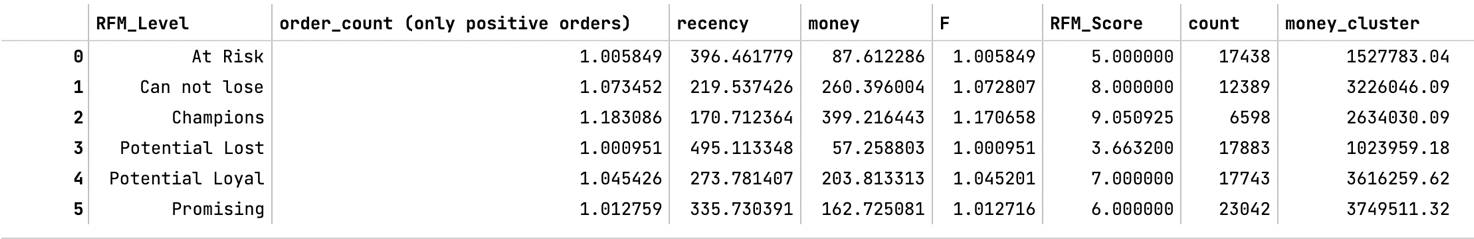
The “persona” per each cluster resulting from the traditional RFM shows better separations between others. For example, the *Champions* persona shows approx. 2 orders, closer recency and the highest amount of money spent: 6598 people and 2 million of euros in sales.

Immagine che contiene tavolo

Descrizione generata automaticamenteK-Means RFM let us perceive different sides of the medal: Persona 0 is a currently active person but spends few money with a low order count. Same for all the other clusters but 3. Persona 5 is a shopper but has a far recency.

Immagine che contiene testo

Descrizione generata automaticamenteThe last technique this is useful for understanding additional data on the customers’ behavior. For instance, it is possible to see that there are people that spends a lot by using installments and those that also use installments by spending a less amount of money.

In conclusion, we are going use the traditional RFM for the reasons stated above.

**Task 2 - Recommendation system**

The aim of the recommendation system is to increase sales by recommending to the customers products they might be interested in. For this task we designed three recommendations systems which acts throughout the whole time spent by the customer on the website:

* Popularity-based recommendation system (for those customers with no purchasing history) - first contact
* Collaborative filtering: recommendation based upon similarities between customers – stay on the website
* Affinity Analysis and Market Basket Analysis – during the check out

The models were built upon the dataset *Recommendation*, an excel custom-built dataset which contains the customer unique ids, the products the customers bought, the review score of the products and the category to which the products belong to.

**Task 2.1 - Popularity-based recommendation system**

Recommendation systems based on popularity are an easy way to target new customers with no purchasing history and solve the so called "cold-start" problem.

In this system, the popularity of a given product, is the sum of the review count + average rating + n. of units sold. Clearly, they do not consider the single preferences of the users but can be useful to attract newcomers’ curiosity, eventually leading them to explore the e-commerce catalogue. This feature could be implemented on the e-commerce home webpage, by displaying the most popular products.

Immagine che contiene tavolo

Descrizione generata automaticamenteFor the task, a dataframe is created to displays products as indexes and number of products sold, review count and the average rating in columns. Since those features have very different standard deviation, we scaled them using the MinMaxScaler by scikit-learn. Otherwise, the purchase count would have biased our results. We order the dataframe by the **score**, which is the sum of those feature. Based on their position in the dataframe, we calculate the **rank** which is inversely proportional to the score: the product with the highest score has rank 0, for example. The lower the rank, the higher the chance to be displayed to new customers.

**Task 2.2 - Collaborative filtering**

For the collaborative filtering task, the dataset is subset in a smaller matrix storing the values only of those who reviewed at least 2 products and of those products receiving at least 2 reviews. That is, to reduce sparsity while having more consistent results and lowering computational time.

The Surprise library is used for the products recommendation. It computes the similarity among users based on 5 supervised algorithms: SVD (Singular Value Decomposition), SVD++ (used by Netflix), NMF (Non-Negative Matrix Factorization), and 2 K-Nearest Neighbors based algorithms. It tries to infer the score for a product a customer would give, considered its similarity to a customer already reviewed that product.

The best parameters, for all the algorithms, were found thanks to GridSearch Cross Validation. Unfortunately, no faster Random Search Cross Validation was available. They minimize the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) as accuracy performances. At the end of the process the scores of the models were compared according to Immagine che contiene testo

Descrizione generata automaticamentetheir RMSE, MSE, Fit Time and Test Time. The best one was chosen upon these parameters.

The tables and the graphs show that the best model is the SVPpp, considering the trade-off between the RMSE, MAE, the computational power, and time required.

**Task 2.3 - Market Basket Analysis**

We went further into the analysis by applying the *Market Basket Analysis*. This analysis computes how many times an item is bought together with another one in a shopping cart (Basket). Once all the "association rules" of the past baskets are known, whenever an item is put in the shopping cart by the customer, we can suggest which other product to buy together based on past associations. Unfortunately, only 3% of past orders involves two items in the same basket: the recommendation system will be poor, but that's an interesting starting point.

The FPgrowth algorithm was chosen because it is faster than the much common *Apriori* that it is used in this kind of analysis. To be faster, the FP algorithm changed the organization of the data into a tree rather than sets. This tree data structure allows for faster scanning, and this is where the algorithm saves time.

To calculate associations, the algorithm needs a dataframe which displays the orders on the rows, the unique items in the columns and the cells are 0/1 whether the order contains the corresponding item. The *Antecedent* is the item bought first and Y or *Consequent* is the item most likely to be bought.

The algorithm retrieves the following parameters per each association:

* Support: it signifies the popularity of the item, if an item is less frequently bought then it will be ignored in the association.
* Confidence: it describes the likelihood of purchasing Y when X is bought. Sounds more like a conditional probability but it fails to check the popularity of Y. To overcome that, there is the *lift*.
* Lift: it combines both confidence and support. A lift greater than 1 suggests that the presence of the antecedent increases the chances that the consequent will occur in each transaction. Lift below 1 indicates that purchasing the antecedent reduces the chances of purchasing the consequent in the same transaction.

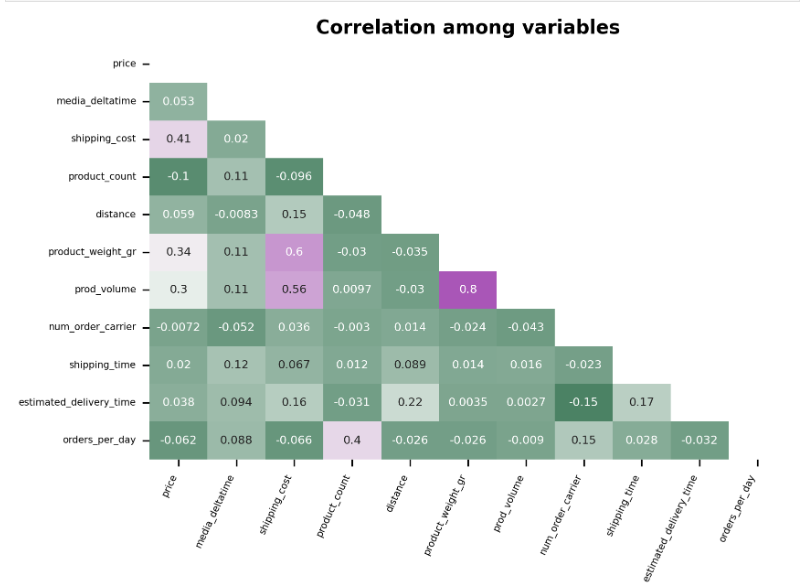
Immagine che contiene tavolo

Descrizione generata automaticamenteTo increase the sales, we are going to suggest those items that combined have the highest chance to be bought. Therefore, a lift parameter higher than one will drive the selections of those products.

**Task 3 - Delivery Time Analysis**

We realized 2 shipping time models with different target actors:

* Customers, with the aim to to predict an accurate date of delivery.
* Management to get insights on the drivers which influence the shipping time and support the process of decision making.

First, the correlation matrix was realized to get a sense of the relations among features.

* The Random Forest allows for a better interpretability of the model: this is useful for the management to understand and improve those drivers that influence the shipping time.
* on the other hand, our Artificial Neural Network can hugely be improved to a better experience for the customers, even sacrificing interpretability, which is not required: customers only want to know when they will receive the package, and nothing else.

To build the models, a baseline was chosen to compare the results. NTT provided historical predictions of delivery times vs. actual ones. In our case, the mean absolute error is 14.78 and the root mean squared error is 16.48.

The variables chosen are *product\_weight\_gr*, *media\_deltatime*, *shipping\_costs*, *distance*, *orders\_per\_day* while the independent is *shipping time*. Other than the self-explaining variables, the *media\_deltatime* is the historical average time the seller took to deliver a package, and *orders\_per\_day* is the number of orders the sellers must deal with during the same day.

We built six regression model algorithms:

* Poisson Regression, since *shipping time* is distributed as a Poisson and a count variable.
* K-nearest Neighbors, that predicts the y value based on the average of the known K nearest neighbors.
* Support Vector Regression, where we used a Radial Basis Kernel to accommodate the variables.
* Random Forest Regression that builds multiples decision trees to perform the output, combining higher accuracy and explainability of single decision trees, while lowering the overfitting.
* Gradient Boosting Regression, that regressor in a forward stage-wise fashion. In each stage a regression tree is fit on the negative gradient of the given loss function.
* Artificial Neural Network, which are usually characterized by a higher accuracy and less interpretability.

After having compared them, is possible to see that the best performing algorithm basing on the considered metrics is the Random Forest, with a mean absolute error = 1.60 and a root mean squared error = 1.89. Therefore, the predictive performance compared to our baseline (which has a MAE = 14.77 and a RMSE = 16.48) was greatly improved.