Slovak University of Technology in Bratislava

Faculty of informatics and information technologies

FIIT-5212-103121

Martin Schön

Sales demand forecast using machine learning

Bachelors's thesis

Supervisor: Igor Stupavský

May 2022

Slovak University of Technology in Bratislava

Faculty of informatics and information technologies
FIIT-5212-103121

Martin Schön

Sales demand forecast using machine learning

Bachelors's thesis

Degree course: Informatics
Field of study: 9.2.1 Informatics
Place of elaboration: Institute of Informatics, Information Systems and Software Engineering, FIIT STU Bratislava
Supervisor: Igor Stupavský
May 2022

Slovak University of Technology in Bratislava
Ústav informatiky, informačných systémov a softvérového inžinierstva

Faculty of Informatics and Information Technologies
Academic year: 2021/2022
Reg. No.: FIIT-5212-103121

BACHELOR THESIS TOPIC

Student:	**Martin Schön**
Student's ID:	103121
Study programme:	Informatics
Study field:	Computer Science
Thesis supervisor:	Ing. Igor Stupavský
Head of department:	doc. Ing. Valentino Vranić, PhD.

Topic: **Predikovanie dopytu tovaru s využitím strojového učenia**

Language of thesis: English

Specification of Assignment:

Za posledné roky zaznamenala oblasť e-commerce výrazný nárast. Čoraz viac zákazníkov volí možnosť kúpy tovaru cez internet a jeho dodanie k dverám obydlia v porovnaní s fyzickým kúpením si tovaru v predajni. S prílivom zákazníkov stúpa aj potreba predikovať ich správanie za účelom optimalizovania rozhodnutí a zvolením najlepšej stratégie. Jednou z kľúčových oblastí je predikcia dopytu zákazníkov po tovare. Úspešné predikovanie správania sa zákazníka v tejto oblasti prispieva k lepšiemu rozmiestneniu tovaru po skladoch, čoho dôsledkom môže byť lepšia dostupnosť a rýchlejšie dodanie tovaru. Analyzujte rôzne prístupy na základe strojového učenia na riešenie problému predikovania dopytu tovaru.
Navrhnite model s vhodne zvolenými vstupnými parametrami v predikcii dlhodobejšieho časového horizontu berúc do úvahy sezónny dopyt po produktoch.
Implementujte a experimentálne overte na vstupom časovom rade obsahujúcom aj historické dáta na vyhodnotenie úspešnosti predikcie.

Length of thesis: 40

Deadline for submission of Bachelor thesis:	16. 05. 2022
Approval of assignment of Bachelor thesis:	23. 11. 2021
Assignment of Bachelor thesis approved by:	doc. Ing. Valentino Vranić, PhD. – Study programme supervisor

Declaration

I, Martin Schön, hereby declare that this thesis is my original work created under the supervision of Igor Stupavský and Martin Brezáni. I have always provided the source of the quotations used in this work and I have made clear what was done by others and what I have created myself.

In Bratislava, 15.5.2022 .....

Martin Schön

Special thanks

My thanks go primarily to my supervisor, Igor Stupavský for his human approach, willingness, professional guidance, help beyond his duties, and for his valuable advice provided whenever it was needed.

Special thanks go to my family, fiancée, and loved ones for their support, help, and motivation for the completion of the bachelor's thesis.

Martin Schön

Anotácia

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ

Študijný program: Informatika

Autor: Martin Schön

Bakalárska práca: Predikovanie dopytu tovaru s využitím strojového učenia

Vedúci bakalárskej práce: Ing. Igor Stupavský

Konzultant: Bc.Martin Brezáni

Máj 2022

Náplňou tejto práce je predikcia dopytu založená na predpovedaní vývoja časových radov. V analytickej časti sa zaoberá rôznymi využívanými spôsobmi predikcií, a to najmä, hoci nie len, v oblasti e-commerce. Okrem toho prebieha analýza rôznych postupov spracovania dát – konkrétne typy zhlukovania, normalizácie a výpočtov vzdialeností medzi dátami. Dáta použité v tejto práci pochádzajú z medzinárodného internetového obchodu, ktorý sídli na Slovensku. Náplňou tejto práce je získať, pochopiť a upraviť dáta tak, aby boli vhodné na tvorbu modelov strojového učenia. To znamená, že je potrebné tieto informácie vyčistiť, rozdeliť do zhlukov a normalizovať. Po spracovaní dát je potrebné overenie rôznych typov strojového učenia, táto práca sa zaoberá porovnaním 3 rôznych tipov regresorov. Tieto modely je treba správne nastaviť, natrénovať a následne overiť ich efektivitu. Súčasťou takéhoto porovnania je aj vyhodnotenie s použitím vhodných štatistických metód. Výstupom bude porovnanie XGBoost, CatBoost a AdaBoost modelov strojového učenia s tým, že tieto modely budú pre účely porovnania riadne natrénované a teda, v prípade dobrých výsledkov aj pripravené na používanie pri predikcii v e-shope.

Annotation

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA

FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES

Degree course: Informatics

Author: Martin Schön

Bachelors's thesis: Sales demand forecast using machine learning

Supervisor: Ing. Igor Stupavský

Consultant: Bc.Martin Brezáni

May 2022

The purpose of this paper is the demand forecasting based on the prediction of time series. The analysis part describes different commonly used methods of forecast - and mainly, even though not only, in the e-commerce sphere. Apart from that, the analysis covers the data processing techniques - clustering types, normalisation, and distance computing. Data used in this work come from an international online store, based in Slovakia. This thesis purpose is to obtain, to understand, and to modify the data to be suitable for machine learning models. That shows the need for data cleaning, cluster dividing, and normalising. After the data processing, we need to evaluate different types of machine learning, since this paper compares 3 different regressor types. There is a need for setting up, training, and evaluating of these models. The part of such comparison is also an evaluation with the usage of suitable statistical methods. The final product will be the comparison of XGBoost, CatBoost, and AdaBoost regressors. Those models will be trained for the evaluation, so in case of successful results, they are ready to be used in e-shop forecasts.

Table of contents

List of Images

List of Tables

1 Introduction

1.1 Motivation

The number of people shopping online increases every year. The younger generation uses online shops daily. The ongoing Covid-19 pandemic situation also brings more people to e-Commerce [34]. Those stores then fight for the said customers and are trying to find ways to be a relevant and interesting option. Online shopping provides benefits for consumers as well - everything is available within several clicks, products and prices are easily comparable, and much more.

There are several factors that people take into account when shopping (online or not). One of them is, without doubt, the price. Very simply said: the higher the price, the lower the amount of product sold. Every company's goal is to maximise its profit, and every customer's goal is to find the best deal. The obvious thing is that companies do have a limited amount of goods - so they can not sell an infinite amount for a very low price. If they run out of product, the shop can lose a part of its customers to competitors. Because of that, companies need to set the price right, and that has proven to be a very difficult task.

When setting prices, one has to consider several different factors, because the price is not the only thing affecting sales. We can think about aspects such as season, month, country, special occasions, etc. This can be very difficult to account for a person making decisions, which can lead to not optimising the profit. Either the price would be too low and the shop would sell all products for a lower price than the customers were willing to pay or, in another scenario the prices could be set too high, which may lead to customers losing interest in buying products.

The shop we are cooperating with is currently setting prices mostly manually with the usage of historical data. This process is not very effective for several reasons - one of them is mentioned above. This process can also be time-consuming and expensive. It can be difficult for the person working in a price setting to predict seasons and find patterns in the data.

However, if we knew how many items are going to be sold for a given price in the future, we could benefit from it - for example, we could prepare for the increased demand of a given product by creating more of it, we could adjust warehouse layouts

for better access of workers or easily adjust the price tag, maximise our profit and satisfy the customers. That is where demand forecasting with machine learning comes into use. Demand forecasting consists of a predictive analysis that tries to predict customer behaviour and shopping habits. It is used by different e-shops and seems to be an effective way to help set prices [13] [33]. Forecasting demand can be done using market research, statistical methods, experimentation (promotions, holidays), and more.

1.2 Description of the problem

In our work, we will use demand forecasting with machine learning usage. Demand forecasting, at its basic level, does not necessarily require computers. The sellers were always trying to forecast the demand. However, now, in the age of machine learning, forecasting techniques have become more effective than ever.

Demand forecasting that uses machine learning methods analyses historical data and tries to estimate and predict the future behaviour of the customers. Of course, it will never be 100% accurate, since people's behaviour is often unpredictable. However, done right, it can be accurate enough to help us make profit via price modelling. There are long- and short-term options depending on the goal of the forecast. In this thesis, we will focus on long-term forecasting, as that is what suits our requirements best.

In our work, we will also use parameters to improve the efficiency of the prediction models. Some forecasting models do not use parameters and focus only on sales in time. While this approach might not always be wrong, the prediction can lack some important data - for example, seasonality. That is why we think it is important to predict using more information about customers than only sales.

Our goal is to design, create, and train models that will be capable of predicting the demand at a given price. For that, we need to create those models and train them using historical sales data and their parameters. We also need to make sure that these models are well trained and capable of being accurate enough. Another big challenge is to figure out which products are similar and if they are capable of using the same prediction model or to figure out whether there are such products. Those products will then be grouped into clusters and one data forecasting model

could serve different products. This should help with effectiveness, as creating, maintaining, and validating a predicting model for every single product in the e-shop could be time-consuming.

There are several machine learning methods used for demand forecasting. For example, autoregressive integrated moving average (ARIMA) models are used [11]. Another very popular option is to use regression-based forecasts. Style, which is gaining popularity is forecasting via neural networks. There is many options and its fully up to us to explore and fit models to the data.

Regardless of choice, good quality data is important for those models to work properly. We will need to prepare the data and choose appropriate parameters. We will obtain the required data from the e-shop thanks to the cooperation with the company SoftPoint. These data will then be used to train and test the models used in this work.

2 Analysis

2.1 Clustering

Object grouping is an important process in various areas including e-commerce, technology, and marketing. Clustering is a statistical method of dividing and grouping data into groups (clusters) based on similarities. It can be considered as one of the most important unsupervised learning problems [17]. Each product should always belong to precisely one cluster and those clusters should not overlap. Items in one cluster are "similar" to each other while being "different" from items in other clusters. What classifies as similarity should be explained for each data set separately, as each data are unique. The goal of clustering is to develop a tool that can classify objects into clusters [24]. This tool is called a classifier. A well-designed classifier should be able to predict the class of the new, unclassified object based on existing clusters.

Objects can be clustered using patterns. Those patterns take different aspects of the product into consideration and create clusters of similar products. Efficiency of the clustering algorithm can be evaluated by feeding it unknown objects and comparing the expected cluster with the one returned by the algorithm.

2.1.1 Types of clustering

There are different clustering approaches - they can be either hierarchical or partitional.

Partitional clustering is typically faster than hierarchical and it returns a predefined number of clusters - usually called k.

Hierarchical clustering finds smaller and smaller clusters using previously known clusters - it only requires a similarity measure without a predefined number of clusters. It returns a more meaningful division, so these algorithms are better for categorical data with a good similarity measure.
The hierarchical algorithm approach can be further divided into bottom-up (agglomerative) or top-down (divisive). On the other hand, partitional algorithms create all clusters at the same time. Partitional clustering includes distance-based, model-based, and density-based algorithms.

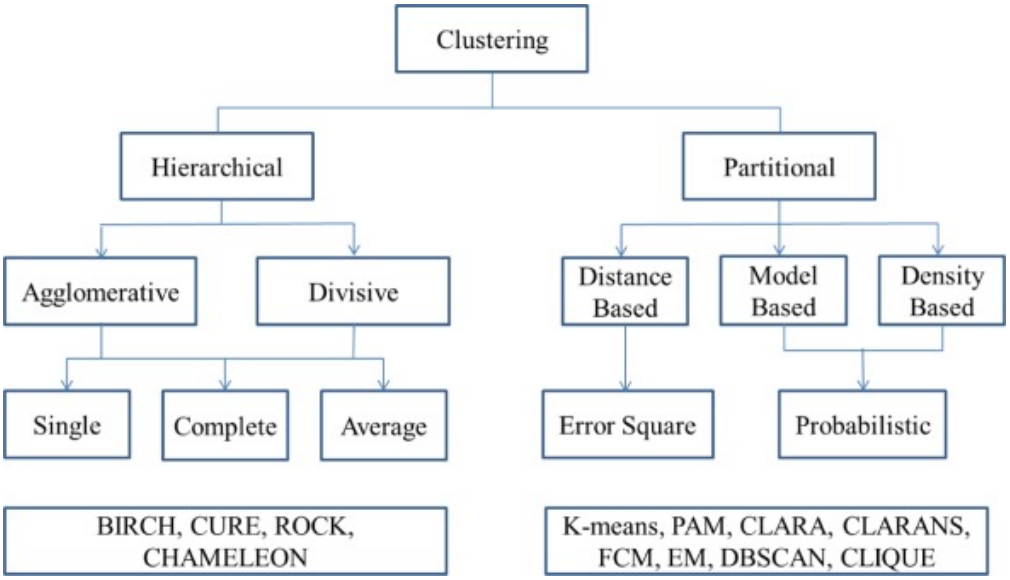

Fig. 1: *Types of clustering [24]*

2.1.2 K-means

The K-means algorithm is one of the best known clustering methods. It splits data into *k* cluster - with k being user-defined number. This algorithm is distance based and uses centroids to determine the cluster of a data point. The point is assigned to the cluster whose centroid is the closest to the given point. The centroid (or the geometric centre) is the mean of all data points in the given cluster [17].

This algorithm is initialised with k (randomised) centroids, which then determine clusters of points - that usually offers space for improvement, as clusters tend to be wild after the first iteration. It should be improved in the next steps of the algorithm: it computes new (more accurate) centroids of clusters, based on the average position of their points. Then the points are reassigned based on new centroids and these steps can be repeated until the clusters are perfect (or at least good enough).

2.1.3 DBSCAN

DBSCAN is another widely used algorithm: it is partitional and density based. It creates clusters from data points that are closely grouped, while points that lie alone in distant places, in low density regions, are marked as outliers [25]. This

method can create clusters of various sizes and shapes, which is surely an advantage. It is great for large data sets with outliers and noise, because (unlike the K-means) this method is not biased towards those outliers.

DBSCAN algorithm input contains the neighbourhood's size (**Eps**) and the minimum amount of points required to exist in the Eps neighbourhood (**MinPts**). It then starts at a random point that has not been visited yet. The algorithm determines whether its neighbourhood of size Eps contains MinPts: if there is minimal density around the point. If no, then the point is marked as an outlier. This point can potentially become part of a cluster based on Eps of another point, but for now, it is not a part of a cluster.

If the minimum required is met, a new cluster is created. This cluster contains a given point alongside all points in its Eps-neighbourhood. Each of those points (in the new cluster) is then also examined: DBSCAN checks for their Eps-neighbourhoods and determines if they meet the required density. If they do, all points in the Eps-neighbourhood of the given point are also added (and examined) into the cluster. Those points are called border points because they are part of the cluster even though they are not in the Eps-neighbourhood of the starting point. Once there are no more points to be added (the density of all points in the cluster were checked and no other point meets the criteria), the cluster is finalised and we move onto the next random unvisited point (as long as such point exists) [17].

2.1.4 Hierarchical

Hierarchical clustering is trying to build a hierarchy of clusters. Algorithms of this clustering create trees, where leaves are values from a data set. Those data are then clustered in clusters represented by internal nodes of the tree (each internal node creates a cluster of descendant leaves) [8]. Those algorithms do not necessarily require a predefined number of clusters, which can be advantageous.
As mentioned, there are two different ways to hierarchical clustering:

- Agglomerative: Bottom-up algorithms, where each part of a data set starts in its own cluster and those clusters are then merged into bigger and bigger clusters with increasing number of elements.
- Divisive: Top-down approach - an algorithm begins with one big cluster,

which is then divided into smaller and smaller clusters containing similar data points. This is a less used approach, because of its higher complexity when compared to agglomerative algorithms.

2.2 Computing distance

Since some of these algorithms are distance based, they need to compute the distance. A distance measure is an objective way to compute how far away a point in a data set is from another data point. There are various ways to compute a distance in machine learning - the most common being Euclidean, Minkowski, Manhattan, Mahalanobis, and Hamming [18]. We are going to look closer at the first two mentioned methods.

2.2.1 Euclidean distance

Euclidean distance is calculated between two points in Euclidean space. It is also known as Pythagorean distance because it can be calculated from Cartesian coordinates with the usage of the Pythagorean theorem. In **two-dimensional** space, let's have two points: p with coordinates (p_1, p_2) and q at (q_1, q_2) as coordinates. The formula for calculating the distance between those two points is then:

$$d(p, q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

There also exists a way to use Euclidean distance in a **higher-dimensional** space with a generalised formula. In n-dimensional space with two points p and q with p placed at $(p_1, p_2, \ldots, p_n)$ and q at $(q_1, q_2, \ldots, q_n)$, the distance between those two is [30]:

$$d(p, q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}.$$

If we are trying to find a specific distance in thousands of comparisons - for example, the closest points of two objects - and this calculation would be then performed millions of times, it is common to use the non-square-rooted version of the calculation and save a considerable amount of computing power by doing so. This version is also known as Euclidean squared distance. Relative values after the modification will remain the same and we are still able to find the closest/ furthest points without a problem. An example of the usage of Euclidean distance in machine learning is calculating the distance between two rows of data with numerical values. [16]

2.2.2 Minkowski distance

This distance is used in a normed vector space (vector space with the defined norm) and is considered a generalisation of Euclidean and Manhattan distances.

The Minkowski distance has it's order ***p*** (as an integer) between two **n**-dimensional points $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$. We can change the value of ***p*** to manipulate the formula and to calculate the distance between two points in different ways. The definition of this distance is given by equation [9]:

$$d(X, Y) = \left(\sum_{v=1}^{n} |x_v - y_v|^p \right)^{\frac{1}{p}}$$

If p $\geq$ 1, then the Minkowski distance is a metric (distance function), while otherwise it can not be, because it violates the triangle inequality.

When p = 2 the distance becomes equal to Euclidean distance, while p = 1 means it is the same as Manhattan distance. There also exists a variant of Minkowski when p = ∞ (with a limit) which is known as Chebyshev distance. [28].

2.2.3 Time series clustering

A series of time-ordered data points is called a time series. In other words, a time series features value change as a function of time. This type of data is commonly used in economics, science, engineering, marketing, and many more [1].

Time series clustering is a special type of clustering. While a time series is usually consisting of a large number of data points, in clustering, it can be viewed as one object. Such objects can later contain interesting patterns - either frequent or rare ones. This raises a number of challenges such as anomaly and intrusion detection, process control, and developing methods to recognise dynamic changes [35].

Done correctly, this can help detect those patterns across such data sets and group similar data sets into clusters. Those clusters should consider regular patterns as well as surprising jumps in time series, and use methods to detect them. This is a challenging task, because of real world problems - anomalies, discords, and novelty.

2.2.4 Dynamic Time Warping

Dynamic Time Warping (DTW) can be used on any time series data - more precisely on any linear sequence data. It is a recursive algorithm that measures the level of similarity between two sequences regardless of their speed and trend shifts (it can detect some trends happening at different times). It provides a better measurement of irregularities between two signals compared to other distance measures - thanks to the invariance to warping in the time axis [6]. In other words: it is a technique that wraps two linear sequences in a way, where similar events are at the minimal distance between them. A very famous usage of the DTW is in automatic speech recognition because there are many different speech speeds at which users can talk. [38]

The algorithm for computing dynamic time wrapping uses dynamic programming - the result is a cost matrix C, with the values calculated by using the formula:

$$C_{i,j} = ||x_i - y_j|| + minC_{i-1,j}, C_{i,j-1}, C_{i-1,j-1}$$

where $||x_i - y_j||$ is the Minkowski distance between two points with a given *p* -norm. This fills the matrix C and the best path from the cost is identified. The matrix also helps with time complexity (which is always a concern with recursive algorithms), because the algorithm does not have to recursively run several times for every single field of the matrix. That allows the time complexity of the DTW to be *O(nl)* with *n* and *l* being the length of the two input sequences.

2.3 Time Series Forecasting

As mentioned above, a time series is a set of values at a specific time. In our case, that is, how many products were sold and when they were sold. Time series data can (and usually do) show information about seasonality (the repeating pattern) and trend (the overall increase or decrease of the values over time).

What we would like to know, however, is what value will be reached at any given point in the future. For that, we are using a time series forecast - a machine learning technique that serves for the prediction of future values. It presents an essential field of machine learning, because predicting the future might be very beneficial. [5]

There are several ways of forecasting, with the most common being regression, ARIMA (autoregressive integrated moving average) models, and neural networks.

2.3.1 Regressions

Linear regression forecasting (LRF) is used for predicting the future values from the past. It is a powerful statistical tool that is very helpful when detecting trends in data. However, it is highly limited to linear relationships and struggles when facing outliers. It also assumes that the data are independent, which is not always the case, so this method might not be suitable for everyone.

Machine learning offers multiple approaches to regression, so these might be worth looking into. For example, XGBoost, which stands for Extreme gradient boosting, is a gradient-boosted decision tree machine learning library which can be used for regression problems. This supervised machine learning approach tries to find patterns in data and features and then predict the future based on those patterns in the data. It gained large popularity in the last few years, thanks to its good usability and wide variety.

2.3.2 LSTM

Neural networks are very effective because they use layers to make predictions more accurate. They are learning - for a specific purpose. The biggest disadvantages are the requirements for relatively high computing power and training difficulties.

Long Short-Term Memory (LSTM) network is a specific type of RNN (recurrent neural network) with wide usage in machine learning. It is a very powerful tool of deep learning because it contains feedback connections that allow LSTMs to remember and evaluate the past. LSTMs are useful for various tasks, for example, speech recognition, handwriting recognition, and acoustic modelling.

In the LSTMs, the units are used to achieve gradient flow in the network. An LSTM unit is composed of four parts:

- **Input gate** - This gate checks if the information from the current time step, flowing into the memory block of LSTM, is relevant enough.
- **Output gate** - Serves for controlling the flow of output values from the cell into the rest of the network.
- **Forget gate** - Decides what information from the past time steps should be used in the memory block and what should be forgotten.
- **Cell state** - Calculates the state of the current cell. That is achieved by multiplying the previous cell state through the forget gate. This is then (element-wise) added to the current state of the cell which creates a cell state. It is the representation of the memory in the LSTM network.

All the states defined above allow the LSTM network to compute a mapping from an input $x = x_1, x_2, \ldots, x_T$ to an output $y = (y_1, y_2, \ldots, y_T$ sequence. It does so using the following formulas iterative from t = 1 to T [23]:

$$i_t = \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \tag{1}$$

$$f_t = \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \tag{2}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_c) \tag{3}$$

$$o_t = \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_t + b_o) \tag{4}$$

$$m_t = o_t \odot h(c_t) \tag{5}$$

$$y_t = \phi(W_{ym}m_t + b_y) \tag{6}$$

Parameters: ***W*** - matrices weight between gates, ***b*** is for bias vectors, σ represents siqmoid function, ***i, o, f, c*** are gates (input, output, forget, cell state) which share size with ***m*** - the cell output activation vector. $\odot$ stands for element-wise vector's product with ***g, h*** as the cell input/ output activation function. Lastly ϕ represents the network output activation function.

LSTM also is not the final form, there are several different variations of LSTM networks such as Vanilla, CNN, Bidirectional, Deep, and many more. As we can see, those networks provide very useful help when forecasting data and are precise, thanks to their nature.

2.3.3 GRU

Gate recurrent unit (GRU) neural network is like a LSTM with a forget gate, but lacks an output gate, thus it contains fewer parameters compared to LSTM. Therefore, GRU networks are usually executed faster and use less memory than LSTM. These neural networks are achieving better performance on smaller data sets where this approach is comparable with or even outperforms LSTMs [10].

Gate recurrent unit neural networks incorporate two gate operating mechanisms known as the update and reset gate. The update gate decides how the information is passed along to the next state, while the reset gate is used to determine how much of the information from the past is needed to be neglected and whether that information is important or not.

GRU also has several different variations, which usually are reducing the total number of parameters - such as GRU1 (each gate is computed using only the previous hidden state and the bias), GRU2 (each gate is computed using only the previous hidden state) or GRU3 (each state is computed using only the bias).

2.3.4 ARIMA

The ARIMA (autoregressive integrated moving average) uses strict statistics, therefore it only needs prior data of a time series to make a prediction. It is great in predicting with minimum features and proves to be great for short-term predictions. Regarding a longer time forecast, ARIMA predicts poorly [37].

ARIMA is a generalisation of an ARMA (autoregressive moving average). It consists of three parts [11]:

- AR (autoregressive) - represents the part of a time series that can be explained as a linear combination of the past values. An ARIMA model is known as p, which is the number of lag observations in the model.
- I (integrated) - the difference of time series before applying AR/MA part of the model. In other words, it indicates if the data values were replaced with the difference between the actual values and the previous ones. This helps the model to get fitting data. It is represented as d which marks the degree of differencing (how many times the observations were different).
- MA (moving average) - the last part of ARIMA shows that the regression error can be explained as a linear combination of errors from the past (at various times in the past). Expressed as q which is known as the order of the moving average.

ARIMA models are generally denoted as ARIMA (p, d, q) where the parameters are non-negative integers. For seasonal models the denotation of ARIMA is then ARIMA (p, d, q)(P, D, Q) where *P, D, Q* refer to the seasonal part of this model (to autoregressive, differencing, and moving average terms). When an ARIMA model parameter is zero, the corresponding parts of ARIMA ("AR", "I", "MA") can be dropped. For example, ARIMA (1, 0, 1) can be rewritten as ARMA(1, 1) or ARIMA(0, 0, 2) equals to MA(2). Changing parameters in ARIMA allows it to work in various scenarios.

2.4 Data normalisation

Data normalisation is a data processing technique used for data manipulation to scale down (or up) variables to relative values so that their comparisons make more sense. Normalisation can be done on numeric values only, so it is required to think about this in other preprocessing parts as well - programmers are usually replacing string variables by mapping numbers to values (if possible).

Normalisation is an important part of data handling for machine learning. Machine learning algorithms look for trends in data and this could be significantly harder if the values of different variables are in completely different scales. What could also happen is that machine learning could value data points with bigger values over those with a smaller range - even though their importance can be the other way around (i.e., years with the range of 1990-2021 compared to months in the range 1-12. Machine learning could be biased towards years, even though months are more important when looking for a trend).

2.4.1 Min-Max

Min-Max is very simple and one of the most commonly used normalisation techniques. It re-scales the values of the feature to the range of <0,1>. For every column of the data, the minimum value is scaled to 0, the maximum value is transformed to 1, and every other data piece gets represented by a decimal in this range. This decimal is obtained from the following formula:

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

where *x'* contains normalised data and *x* is the range of original values, with *min(x), max(x)* as minimum and maximum values of the range.

The range of Min-Max normalisation, however, does not have to be <0,1>. The user can adjust this value and select different ranges, which leads to different results [22]. A different formula is used to calculate the scaled values of variables in the range given by the user [19]:

$$x' = \frac{x - min(x)}{max(x) - min(x)} * (z - y) + y$$

where new variables *z, y* define boundary <*y, z*>.

The problem with this type of normalisation is that it does not handle outliers very well, since an outlier would cause the remaining data to be squished while a significant part of the range could remain empty. One solution to this is to handle outliers before using Min-Max normalisation or to use another normalisation style.

2.4.2 Standardisation

This form of data operation is sometimes also known as Z-score normalisation. This normalisation tries to avoid outlier bias thanks to the usage of the standard deviation. The values are scaled so that the mean of the values of a given parameter is 0 and the parameter is unit-variance (the standard derivation and the variance tend towards 1). This type of normalisation works best with the Gaussian distribution in the data array, but it works even if the values are distributed in a different way (although it is not as effective) [19]. The formula for calculating z-score values for a given feature is:

$$x' = \frac{x - average(x)}{\sigma}$$

where *x'* is z-score normalised data, *x* is the element of the group, σ stands for the standard derivation of feature and $average(x)$ is calculated as mean.

2.4.3 Softmax

Softmax normalisation is based on the softmax function (also known as the normalised exponential function). It turns a vector of *n* values into a vector of *n* values that sum to precisely 1. The Softmax transforms the values of the variable to the range of <0,1> so that we can talk about those values as probabilities. This is most commonly used to transform the output of the neural network - it scales the output from real values to nicely scaled, easy to work with decimals. In other words, the softmax function converts the output of a neural network to a normalised probability distribution in multiple dimensions.

The formula for calculating softmax:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

with σ representing softmax, $\vec{z}$ as input vector, e^{z_i} stands for the standard exponential function for the input vector. *K* is the number of multi-class classifier's classes and e^{z_j} is the standard exponential function for the output vector. This function applies the standard exponential function to every element of the input vector z. The result is then normalised by dividing it by the sum of all results - so that the sum of the outputs is 1.

Softmax is a powerful approach to the normalisation of multi-class classification problems. That means it works best when predicting data with only one right answer - where the different outputs are mutually exclusive. That is because to increase the probability prediction of one class model, it has to decrease the probability of other outputs.

2.4.4 Cyclical encoding

Cyclical encoding is a great normalisation technique to use with cyclical data - like days, hours, minutes, seasons, etc. If we were using a different normalisation type for such data, we might confuse the model and there is a possibility of the model incorrectly interpreting those data. For example, if we were to MinMax normalise the day of the week, we would get a 0 value for Monday, with 1 being Sunday (and all other days would be valued in the interval (0, 1)). However, this is not accurate as Monday follows Sunday, rather than Sunday being completely far away from Monday.

That is when cyclical encoding can be implemented. It uses the periodicity of sine and cosine values over the <0, 2π> cycle, which generates values in the interval of <-1, 1>. For example - say, we want to encode the days of April (values from 1 to 30). The way the mapping is done is by encoding all day values as represented in MinMax scaling, with the minimum being 0 and the maximum is 2π (so after this step day 1 is represented by 0 and day 30 is converted to 2π).

After that, the cosine and sine representations are calculated from these values. The important thing is to use both functions - sine and cosine as well so that for each unique original number (for each day) we get a unique combination of two new values (cos and sin). If we only used one function, we would get incorrect data, since these functions return the same value for two different inputs (for example, sin(0)

= $\sin(\pi) = 0$). The combination of sine and cosine is advantageous, as it creates normalised cyclical values for the original cyclical data. The following image represents how beneficial this representation of the the values from an example are:

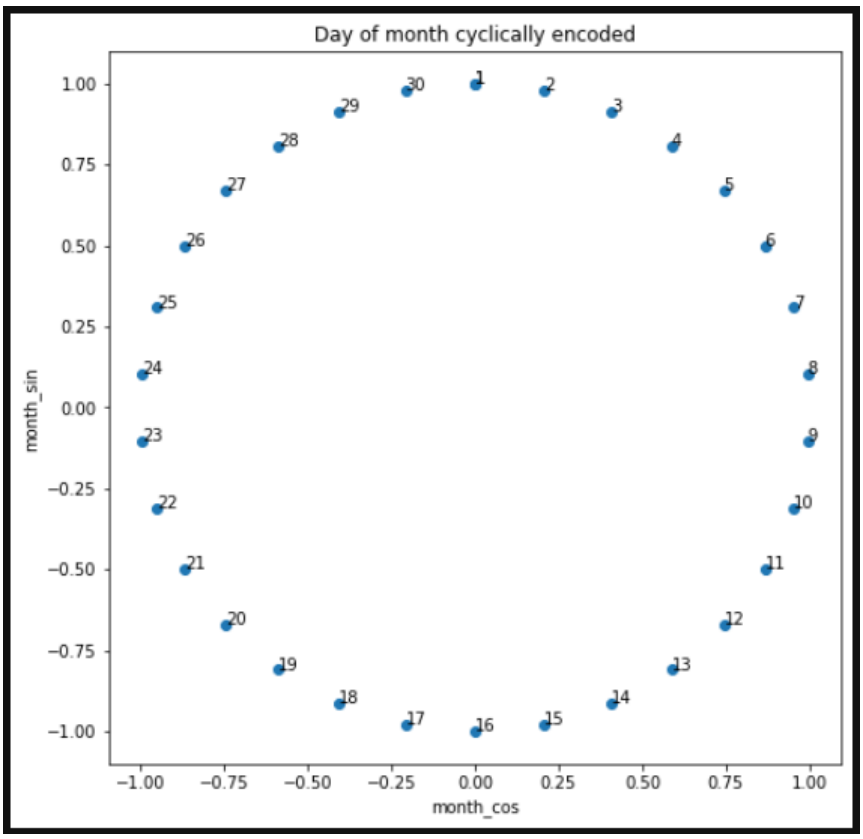

Fig. 2: *Sine and cosines of cyclically encoded month days*

This system is not perfect. It breaks one feature into two, which mathematically adds more weight to a given feature from the perspective of the algorithm. Moreover, some machine learning methods (e.g., regressors) are not able to process two features simultaneously, so they will treat each of them as if they were single features. However, there are no perfect solutions, we always have to choose trade-offs in given scenarios, this one might prove to be beneficial if used correctly.

2.5 Related studies

The field of forecasting demand is very rich - in the latest years, various studies have explored this topic and various ways to predict the user's behaviour. It is a very active and quickly evolving part of machine learning, therefore several interesting articles were written in the past few years.

2.5.1 Rue La La

A very similar task was explored and the solution was programmed by Harvard University [13]. The quest is to help an online retailer Rue La La which sells items in time-limited events, so finding trends might be even more challenging. In this case, there is an attempt for demand forecasting and using this knowledge to price items correctly - this is achieved by using two models - a prediction model and a price optimisation model with the usage of prediction data. An interesting approach is using regression trees for both models. The items are aggregated at the style level and all different sizes of one product are aggregated as well - because the price stays the same.

The input data are collected from the past - record of sales of styles sold in the past and estimates of the demand of the styles sold in the past. The provided data were from the time window of 2.5 years - each data point was a time-stamped sales record. Those data points contain a price, the quantity sold, the event start time, the event length, and the initial inventory of the item. Brand, size, colour, manufacturer's suggested retail price (MSRP), and hierarchy classification were also available for each item. From this, the features were chosen - the price of the style, the MSRP, and the relative price of competing styles (the competing style is the style from the same subclass and event).

To find the historical estimated demand, researchers grouped hourly sales of all items that did not sell out throughout the event duration. Since this creates almost 1000 demand curves, hierarchical clustering was used. To estimate the demand for sold-out items, the time when the item was sold out was identified and the demand curve to estimate the typical sales proportion within that amount of time was used.

Finally, the features and historical estimates were used to build a regression model for each department. The output was a model interpretable for managers and

merchants. Several different regression approaches were tested - with train and test split, using 5-fold-cross-validation on the training data if required. The results were compared using different techniques - median absolute error, mean absolute error, and both median and mean absolute percentage error. The regression trees came out on the top, so they were used for the demand prediction.

2.5.2 Demand forecast using LSTM

Another approach to demand prediction was brought by a paper written in 2019 [3]. In this case, the researchers worked on a forecasting framework of an online marketplace data set from Walmart.com - using the real-world data gathered by this online seller. An inspiring part of this solution is the usage of a natural grouping of products into the hierarchical structure - with the usage of the predefined subcategory/category/department/super-department distribution. What is also considered in prediction is the similarity of items - this allows us to assume that the demand for one product also affects the demand for another item (such as complimentary/substitute products). Even though using predefined groups can be beneficial, there are still better ways to group items - in this case, it is the usage of the K-means clustering on a set of time series with the usage of features as the sales quantile over total sales, the percentage of zero sales, the strength of a trend and several others. The other groups created in this work are determined by the height of the sales ranking and the zero sales density. This creates 3 groups - the most important for e-commerce (G1) with a high sales ranking and low zero sales density, the least important one (G2) with low sales and high zeroes, and the rest of the products (G3).

An interesting part of this work is also pre-processing. Missing values are replaced by the strategy called forward-filling, which consists of replacing the missing values with the most recent valid observation available. The sales values are also normalised with the usage of a mean-scale transformation, where the mean stands for the scaling factor of normalisation. Predictions are then done with the usage of a moving window which turns a time series into pairs of <*input*, *output*> where the input represents the number of days fed to the model before outputting m days of the predicted values.

Regarding data, the paper uses the past sales data of products in the database. In terms of other features used to train the model, the researchers used a window of holidays, seasons, day of the week, and subcategory types. The result coming from the model is an output window predicting sales in a given day. The used data set consists of 18254 different items from a single super-department, consisting of 16 different categories. The sales data are collected from 190 consecutive days in 2018, with the last 10 days serving as the test part of the data. 10 is also the number of days for the output of prediction (the output moving window size).

For the model itself, this paper provides a solution to the usage of LSTM neural networks with peephole connections, which uses LSTM where the input and forget gates incorporate the previous cell of LSTM. In this study, the LSTMs provided by the TensorFlow framework are used. As for the density of products, the best approach seems to be creating a separate LSTM model on each subgroup of products. The technique to measure errors is an mMAPE - the modified version of the mean absolute percentage error. For the hyperparameters of LSTM, the paper talks about the usage of *adam* and *COCOB*.

The last part of the prediction process is the post-processing layer, in this case, it consists of rescaling and de-normalising the predicted values generated by the LSTM. This process back-transforms the values to their original scale of sales.

2.5.3 ARIMA usage in demand forecasting

Yet another solution to forecast the demand in e-commerce is explored - with the usage of the autoregressive integrated moving average (ARIMA) model - by Jamal Fattah et al. [11]. In this case, the researchers have the need to predict the demand of products for a Moroccan food company. The prediction is divided into three parts: identification, estimation, and verification.

The ARIMA model used in this paper is the ARIMA model (1, 0, 1) with the constant value. In this scenario, it has proven to be the best configuration - which was measured by comparing the abilities of different ARIMA models. The best model is as simple as possible while it minimises certain criteria as variance, maximum likelihood, AIC, and SBC. It was then evaluated by comparing the experimental and the predicted sales in the same period. This model was trained with the historical

demand data in a 5-year interval - from January 2010 until December 2015. Then, with the usage of the ARIMA model (1, 0, 1), the IBM SPSS (Statistical Package for the Social Sciences) Forecasting is used to forecast the next 10 months. This prediction was considered good enough and it proves the model's usability for forecasting and modelling the future demand.

2.5.4 Gradient Boosting–Based Machine Learning methods in Real Estate Market Forecasting

The last paper mentioned in this section is from the field of the Russian real estate market. [12] In this paper, the researchers are looking at the growing market and are comparing several different gradient boosting methods for the prediction of the prices and sales of real estate.

The data necessary for such a paper were taken from Dom.MinGKH website - obtained parameters were commissioning year, the number of floors, playground nearby, energy efficiency class, type of the building, load-bearing walls, etc. Those data were not written on the website, but were not ideally formatted - thus data parsing was necessary. After obtaining and processing those data via script, the following part is the analysis of them. The mainly analysed parameter is the price and its influence on the market. One interesting calculated factor, for example, is the ability of a household to pay a mortgage over their property.

For the price forecasting itself, there are 4 Python regression models used and compared - LinearRegression, CatBoostRegressor, XGBoostRegressor, and AdaBoostRegressor. The methods used for the evaluation are R^2 (Coefficient of determination) and root-mean-square error (RMSE). The final results were extracted from the prediction in two big Russian cities - Chelyabinsk and Khanty-Mansiysk. Based on these metrics, CatBoostRegressors come on the top, with the best values in both cities. It is closely followed by XGBoost and AdaBoost regressors, with Linear Regression performing poorly, especially in Khanty-Mansiysk.

At last, the CatBoostRegressor model is used for feature ranking and selection. There are 44 features used at first, but ranking feature influence on the model has shown that the top feature (Total living space) accounts for over 50% of the influence over predictions. The suggested future work of this paper is exploring more machine

learning techniques and obtaining more data for the algorithms already tested.

2.6 Draft of solution

There is much more to explore in regard to the topic of forecasting the future. Even in recent years many papers have been written on the topic of various predictions - using ARIMA [15] [2] [20], prediction with the usage of LSTMs [4] [39] [7], various regressions [26] [14], or using other techniques [21] [29] [2]. Some papers comparing ARIMA to LSTM or to regressors were also published [27] [36] [31]. The topic is still being widely explored and quickly changing, since machine learning is a field hugely rising in popularity in the past few years.

Data selection and pre-processing are a big part of the solution. We are expecting the most important parameters to be the date (notably month and week of the year), price, season, and group. Groups of products can be determined by predefined categories of items or by using clustering on them (e.g. hierarchical). There are also other optional features to explore, such as the number of sales of similar items, item price changes, the occurrence of customer's wish lists, or watchdogs on items. Another part of preprocessing is the normalisation in which we will test the approaches described earlier in this work. We also need to take care of the missing and outlier values, apart from the price.

It seems that, for our task, there are various approaches to model selection possible. The first thing we have to do is to find the right data in the database, obtain them correctly, explore what features are available, and do an analysis of them. It is hard to guess which model path will be the best for us, we can decide once we have enough information about the available data.

Using the regressions will probably be a first choice, as there are different types, so we might choose from different variations. What is also an advantage is the benefit of grouping products into bigger categories without the need to train a model for each item separately. Usage of Neural Networks might be also interesting. Neural networks allow to achieve great results when predicting periodical events (such as seasonal product sales). Finally, the usage of the model depends on the data, so we will decide on this later in the process.

Of course, after the prediction is made, the evaluation is an important part to

find if our predictions are working well. From the published works we can see that to measure the error and to train the model as well as possible, the MAE (mean absolute error) or the MEDAE (median absolute error) are very effective. We might also consider trying their less common counterparts of MEDAPE (median absolute percentage error) and MAPE (mean absolute (percentage) error), but those seem to not be as effective in our scenario, as they are problematic in scenarios, where values are zeroes (which will most likely occur in our sales) [13] [3].

When it comes down to the implementation, we will be working with Python programming language, since it contains a wide variability of libraries, suitable and helpful for machine learning problems.

3 Problem description

3.1 Goal of problem solving

The goal of this work will be to obtain the correct data from the e-shop database, describe them, prepare them for machine learning algorithms, then implement a machine learning model over those data, and then evaluate the success rate of machine learning predictions made on these data. There is a need to understand and interpret the data correctly, as well as their appropriate preparation for ML algorithms. The motivation is to determine whether creating such a model is possible and worth investing time and money of the company.

3.2 Requirements specification

This part specifies the requirements our solution must meet. We talk about the technical part of the solution (and it will be mentioned later), as well as about the expectations of the results and programme parts.

3.2.1 Functional requirements

Finally, the system should be able to generate models and predictions, in conjunction with the model evaluation of its success rate. To accomplish that, we need to ensure that good quality data will be fitted into the model. The required parts of this process are the following (not in final order):

- Obtain data and describe the relationships between them.
- Explore the database and ensure that the data are valid.
- Describe the data, their features, and their meaning. Determine which features are valuable and which are not.
- Divide the data into logical groups based on their similarities.
- Prepare the data for machine learning by encoding and normalising the values.
- Select a machine learning model to train on the data.

- Select the right features and the right approach to prediction.
- Tune the hyperparameters of the selected prediction model to ensure the best results possible.
- Evaluate the success rate of the chosen approach.

The final user is going to be the company SoftPoint, with the coordination of Muziker e-Shop, which provided us with the data and which is the target for these predictions. The final approach should be applicable to any other sales forecasting problem, given the data are similar, however, the code and specifics of this solution require exact fitting to some unique data parts of Muziker's database.

3.2.2 Technical requirements

The whole solution will be provided in Jupyter notebook files (.ipynb), which are executable in Jupyter Lab or Jupyter Notebook. The programming language used in this solution is Python for code, and PostgreSQL is a database management system running on the database server. Regarding the data, the.csv files will be used for local storage. Parts of the data are under nondisclosure agreement, but enough data for the machine learning part of the solution will be accessible, so that the algorithms can be rerun if needed.

All tests and coding are done on a Lenovo laptop (specifications are OS: Ubuntu 21.04 LTS, RAM: 16GB, CPU: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, GPU: GeForce GTX 1050), so the time complexity is ever mentioned, it is based on this device.

Time optimisation is not expected as part of this work, because of the nature of this field. While, of course, the code is not supposed to be unnecessarily slow, machine learning techniques and model creation are expected to be very time-consuming, as they are, in general, expensive to compute. We are also working with a significant amount of data, which might also decrease the time efficiency.

3.3 Draft of the implementation

Solution of this problem should be available to the user in several independent Jupyter scripts. We decided to divide the main code into 4 big parts, with the

intention of each solution being a unit itself. It is expected that some overlaps will occur, but generally speaking, those segments should be well divided.

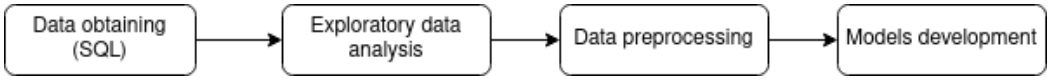

Fig. 3: *Main parts of the solution*

The first step is to obtain data from the company's database - with the usage of PostgreSQL script written and run via Python, with the usage of a remote connection to the database. The most challenging part of the solution is understanding the database and selecting the right data from it. Furthermore, analysis of the data is required, which should fulfil the goal of a deeper understanding of the data and should allow us to describe it. For that goal, we will use various graphs, number descriptions, and other data exploration approaches. After we understand the data, we can prepare them for machine learning with the usage of clustering, solving the problem of nulls and outliers, and normalisation techniques. Eventually, the development of the models can be explored. We will attempt to fit the data on a regressor, as it seems to be a very effective and powerful way for prediction. The final step in the development of the models will be to evaluate them with the usage of different evaluation metrics and concluding results.

4 Solution description

This chapter of the work will bring closer attention to the final version of the individual sections of the proposed solution. Firstly, we will look at the methods and results of those sections; more about the code and functions themselves will be in the implementation part of this chapter.

4.1 The database

The e-Shop with which we are working has a crucial part of its data stored in the PostgreSQL database. To access this database, the user needs the (secret) connection information, which was provided to us because we are collaborating with the company. We got a read-only access to the production database, which was sufficient. Understanding the contents stored in the data lake is a time-consuming process, since it contains multiple schemes, which add to more than 300 tables together with some tables containing thousands and thousands of rows over tens of columns. Fortunately enough, a significant part of this database was irrelevant for sales time series forecasts.

Finally, we found eight tables which proved to be useful for the task of finding sales of products over time. Their relations along with the parameters they contain are displayed in the image. There are four mandatory tables and four tables that are an advantageous addition because they may contain possible features, but are not necessary for the information about products sold or about the sales themselves. The four important tables are shop_orders, shop_baskets, shop_basket_items and catalog_products. The main table to store information about products is catalog_products. It contains one row for each product that is, or was available in the e-Shop, along with many additional details about it, for example, when the product was added to the database, what reviews it received from the customers, the foreign key to the table of categories, to the table of brands, and much more. We chose columns that are relevant for our task, columns showing the parameters of the product that seem interesting to look onto. However, not storing past data might not be the best approach, since we are losing information because their historical value is not saved in any other table.

Fig. 4: *Database scheme of used tables*

The table catalog_products is then referenced in the shop_basket_id table. There is information about concrete product sales in the product shop_basket_items - for example, how many pieces of a given product were purchased in a given order, or what the price was (as the price changes and is not written in catalog_products). Moreover, the important purpose of this table is to allow for the many-to-many data model between the tables catalog_products and shop_baskets, by storing the reference of both these tables for every item sale. With that, we can identify which products were sold in which baskets (and then we can reference the baskets with orders), so this table is crucial in finding the orders of products.

Finally, information about the order and purchase itself is stored in two tables - shop_basket_items and shop_orders. Not every basket in the table of shop_basket is an order, however, we are only choosing the baskets which are referenced in shop_orders, so in this case every basket represents an order - in one-to-one relation. Finally, there are two tables, which combined give us all information about the orders we could need. Crucially, what we are getting is the date of the order and whether it was cancelled or finalised. We also get some other interesting information about the orders, i.e., the total order price, country of order, the number of products in the basket, and others.

The four tables, with smaller impact, but interesting features, we have mentioned earlier are setting_currencies, catalog_category, catalog_brand and catalog_segment. Categories, brands, and segments are mapped onto the products - each product has its category, brand, and segment assigned, but each has their own table for easier management. Brands and segments are just standalone features of the products, but categories are stored in a tree, and some categories are linked to another. Lastly, the currency table is important since it contains information about the currency in which the order was made. It is not so crucial, although, because the exchange rate is stored in the order itself, so even without this table, it would be possible to calculate the unified price.

We have merged those tables in an SQL query, selected the correct data, downloaded them via the Python function, and stored them in a .csv (comma-separated values) file. This way, they are prepared for future use, without the need to approach the database every time we need those data.

4.2 Exploratory data analysis

After we have downloaded the database, our goal was to see what data we have. Exploratory data analysis (EDA) is an approach to analysing data, their values, connections, etc. It is often done with the usage of graphs, plots, and other visualisation techniques, with the goal of making the data easily readable and understandable for people working with them. Analysing and visualising data was always an important part of data collection, and EDA was very promoted by the statistician and mathematician John Tukey in the 1970s [32] to encourage data exploration and hypothesis testing. Today, data analysis is still an essential part of many fields, including data science and machine learning techniques.

Regarding the EDA in our solution, the first thing we want to determine is how much data we have. We will do that by analysing the .csv file loaded to the Pandas data frame (therefore, if we are talking about the .csv file or about the data frame, we are referencing the same data lake, which represents the data downloaded in the former section). There are approximately 3.6 million rows and 58 selected columns in the data frame. When talking about the data types of these columns, 3 of them contain dates, 28 columns have numeric values, 5 of them are booleans, and 22 columns contain other data types (most of them are strings). Our EDA will consist of looking at the data and cleaning some parts of it as well, if they prove to be unusable. The cleaner the data we end up with, the better the results of the machine learning algorithms.

The next thing we want to do is to explore and clean up the null values. While dealing with nulls is not necessarily part of the EDA, we want to explore the data we will use further, and if the log contains an unknown/ missing value (null), it will be unusable since we do not want it to affect EDA. One exception to this is the column canceled_date from the orders table - if a cancelled date exists, then the order was cancelled and we do not want to consider it in sales. Therefore, this part of EDA drops some rows from the data frame - approximately 25000 rows containing nulls and 100 000 cancelled orders. We will also drop the column of cancelled dates as it now contains only nulls.

The following parts that we want to explore are the different types in the database. First, we examine the types determined in the database. There are item types stored in the products table - and it divides the products into 9 different types. The most common product type is *standard*, with more than 85% products belonging to this type. Analysis of this parameter showed us that there is a need to drop more rows - specifically, we want to drop all the gift types because we are not selling those products, but rather giving them away - which does not show the real sale potential of item. There are more types of items with unalike names that are used for gift giving (each marks various gift reasons) - *gift*, *extra_gift*, *club*. Together with those, we might drop *gift_voucher_buy* and *gift_voucher_apply* because they only mark distinct payment methods. Lastly, we can unify *set* and *user_set*, which then leaves us with 3 item types: *standard* (now containing more than 98% of the data), *set and digital_licence*.

The basket types are only divided between 3 unique values: *standard*, *express_checkout* and *internal*. After the conversation we had with the e-Shop management team, we know they all are normal order types, with the type only influencing other database parts.

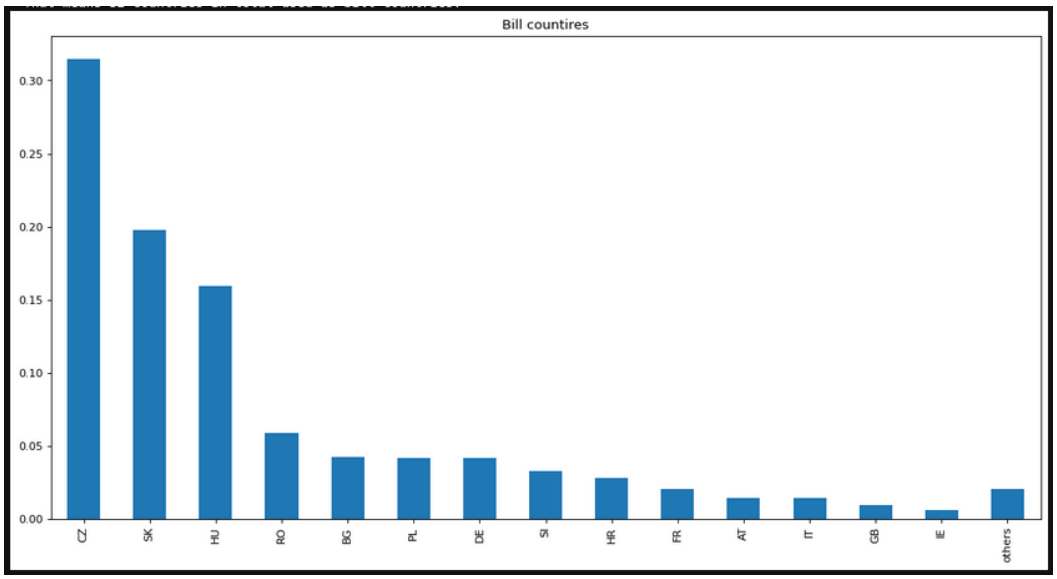

Fig. 5: *Bill countries distribution*

The e-shop with which we are working is available in many countries and sells products in many different currencies throughout Europe. Together, 10 different currencies are used in 32 countries. The billing countries graph shows that the largest market is in the Czech Republic, with more than 30% orders coming from this country. The most common currency used, as we found out, is the Euro followed by the Czech crown and the Hungarian Forint. We have to unify the prices in the data, as sale prices are written in the currency of the specific order. Fortunately, orders do have an exchange rate in euros as part of each log, so the task of unifying the values (in euros) is simpler. Unified values enable us to compare prices across the board without weird or mismatching values.

Continuing with the parameter exploration, we come to boolean values. There are five of them - is_in_stock, is_ended, is_new, is_boosted, is_fifo. These column names are understandable, and the latter three mentioned have minuscule variances (99.9% of cases where the value is false for each of the latter three columns). That leaves us with is_ended product and is_in_stock to explore. Both are useful for finding the current state of the product, however, since we do not have historical values, they are not expected to be useful features.

In total, around 150 thousand unique products were sold at least once in the history of our database. Their price varies greatly, starting at a few cents, with the most expensive product being sold for 22799 euros. According to this, the products have a different purchase price (the price the e-shop has to pay for the product). We can also see how many products are currently in stock: some of those values might even be negative, which happens if a customer makes an order while the store has 0 pieces in stock. Another interesting thing that we can see from an EDA of products is that more than 80 % of the purchases made are purchases of single products, so only a small number of customers buy more than 1 piece of the same item as once (as expected).

Review count and scores are another part of the product description analysed. The review score includes the reviews of four sub-parts - price, quality, properties, and overall - which are also accessible as separately stored values. Each of these review numbers is represented by a percentage. Although a large number of products

have not yet received reviews in the new e-Shop (25%), we fortunately have the old reviews scores from an old e-Shop. If the product has no reviews, it will be listed with 0% review score in the database, but apart from those, the number of review scores is high (over 80%): with the 100% score is the second most common after 0%.

In the following part, we look at the dates of the orders. We are using the doc_date column to determine the date when the order was made. The data available from this data set start in January 2019, and the last date we are using is April 24, 2022. We want to see how the values did over the year - which is possible with the usage of a rolling average over the week. This helps us to see the data more clearly and better understand the trends that might occur in the data. Since days of the week do not affect this type of graph, it is much better to decompose down to the trend, seasonal sales, and the residual part of the orders.

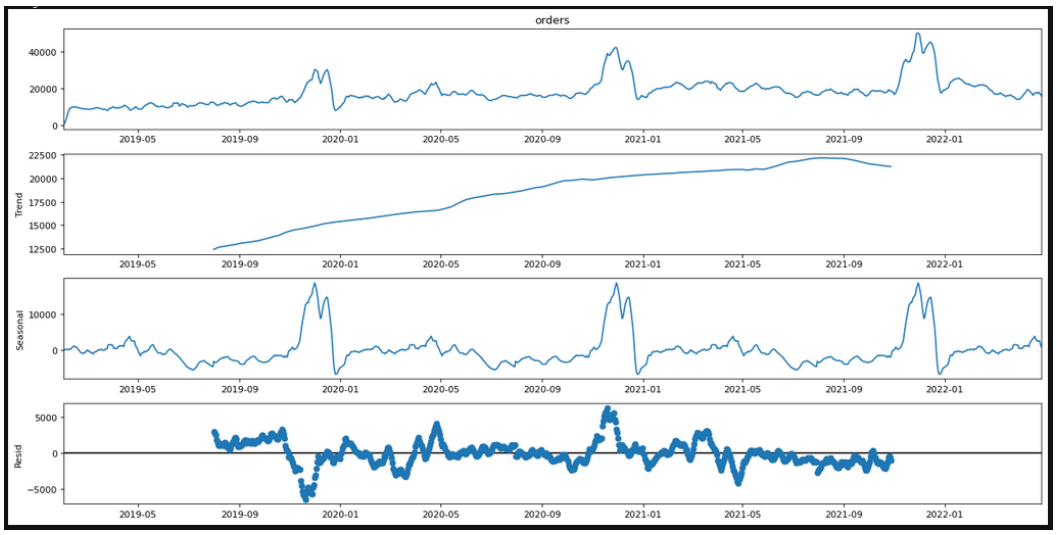

Fig. 6: *Seasonal sales decomposition of the orders*

This clearly shows that the strongest season of the year is around Christmas, with a smaller peak in summer. Residuals occur at any time of the year, but they are also interesting data points to see. The last part of decomposing the trend is also a very powerful feature of the data set; it is beneficial for the e-Shop that it seems to be raising over the years, even though we can see a small decrease at the end of the

trend line - it might be a warning sign to remember this factor, since it can affect the accuracy of our predictions because it is the first major decrease in trend line we can see.

The last part of the EDA is dedicated to categories, segments, and brands. The analysis of this part is not very complicated; the most complicated is the category analysis. The products are divided into 2643 different categories, with the largest (LP records) only having slightly more than 3% of the products assigned to it. Another interesting factor to note about the categories is to see how deep in the category tree they place. This number is in the range <3, 7>, with 5 being the most common depth among the choices. When talking about brands, we have over 5000 of them, and the distribution between them is the only new feature they provide. The same is true for segments, with the number of segments being much lower - 16, while the biggest one is music. We expect the segments to be more influential as they are splitting the data into larger groups.

At the end of the EDA, we are saving the data into a new updated .csv file, because even though making changes is not the primary goal of an EDA, we made some changes and dropped some rows as well as columns. Saving the data frame to the new file takes some memory space, but the advantage is accessing the changed values immediately without the need to run EDA every time we want to get those data.

4.3 Data pre-processing

The next part of the solution is to further pre-process the data. At first, we are going to change or represent the non-numeric columns to numbers (either integers or floats). This is essential, since machine learning algorithms are, in general, not very good at directly processing categorical data. This leads to the need to convert these data to a format that is understandable by machines. We start by taking care of the columns that contain dates. At this point, we have only two date columns - doc_date (which shows us the date of the order) and product_since (this serves the purpose of tracking when the product was added to the e-shop). Two differ-

ent approaches for this are used, as both date columns represent different things. The first approach is to divide the date (in the doc_date) into multiple columns - days, months, years, weeks, and weekdays of the date and store them in individual columns. The other solution, which is more suitable for the product_since, is to calculate the number of days that have passed since the date and instead store that value.

The great advantage we have is that every table contains the primary key of its id (as shown in 4). That means if we have any values that are unique for the id of the table (e.g., each name is mapped to each id in the products table), we can remove the value and let it be represented by the id only. That leaves us with the numeric representation of categorical values, which is data lake natural and easily reversible, by selecting the id from the given table directly from the database. By this technique, we get rid of all string values represented by id - different names, parameterized names, codes, etc. The remaining string columns are then encoded with the usage of Ordinal Encoder. An ordinal encoder from the Sklearn library takes the array of categorical data and converts them to ordinal integers. The great feature of this approach is the ability to easily encode all required columns with a convenient way to show which value is represented by which number. Lastly, we convert boolean (True / False) values to ones and zeroes.

Moving on to the pre-processing, we want to check the outlier values of the selected columns and delete any occurring outliers. If the following formula is true for a given x_i, then the value x_i is an outlier:

$$|x_i - \overline{x}| > 3 * \sigma$$

Where x_i is the parameter value compared, $\overline{x}$ is the mean of all values in a given set, and σ is the standard deviation of such a set. Using 3 times standard deviations from the mean is a good idea, because it cuts only outliers that are really extreme (in the normal distribution, 99.7% data fall within three standard deviations of the mean). In our scenario, this will drop around 5% of the data, as our distribution is not ideal, but it is a reasonable price to pay for cleaner and more useful data.

4.3.1 Clustering

Clustering is also an essential part of data pre-processing. We are clustering the products together, based on the parameters they share across all orders, without any duplicates. They are clustered based on 23 different features: for example, product purchase price, days in the shop, product brand, etc. In this way, each product comes to the clustering process exactly once. We are not including any information unique for the orders - it would be possible to add the number of total sales in history, but we found out that even without such features, clustering works well and creates logical groups of products.

We wanted to try hierarchical clustering on this data set, but unfortunately, it is not suitable for such a large amount of data, because of the need to create the distance matrix. We decided to use kMeans for the first layer of clustering (agglomerative clustering handles large data sets way better) in 101 clusters. After they are divided, we also use hierarchical clustering on those clusters, categorising the data into three hierarchical clusters each. Finally, we will have 303 clusters ready to be used. This number can be easily adjusted by changing the number of clusters, either kMeans or hierarchical, but this number of clusters seems satisfactory.

After this part of the pre-processing, we generate and save the data in the new file: *data_after_preprocessing.csv*. This file contains approximately 2.7 million rows and 44 columns. Up until this point, the data dumps were under an NDA (non-disclosure agreement) licence, so they are not freely available. However, this new file contains encoded data that are available for use. That is also the reason for the last part of pre-processing to be done in the last jupyter notebook, alongside with the modelling - because from this point the data are available for use and the last notebook is runnable by anyone interested (appendix for more: C.2).

4.3.2 Data preparation

The big decision made in the last stages of the pre-processing is to predict sales weekly, rather than daily. This is because for proper data prediction, we need to fill the holes in data sales, as in days / weeks when the product was not sold at all because of the nature of the data. If we were to fill the values for each day in the

time window, only 2.3 % of the values would be non-zero. Predicting every day would also be much more difficult for machines. Additionally, if we were to predict daily sales, we would have to account for differences in weekdays, so in the end the best solution would probably be to use the rolling average of the week, thus taking weekly sales into consideration as well.

Compared to that, predicting weekly sales seems to be adequate - even according to the sales team of the company. Due to the shop specifications, we do not need to know exactly how many products are going to be sold on every day, so it is not worth paying the large computational price for the daily predictions. Additionally, from the quick tests we did, it seemed like daily predictions would be less accurate than weekly ones - however, this theory was only tested (and proven) on a smaller data sample. The advantages of weekly predictions are too great to ignore.

With this in mind, we will prepare weekly sales for each product, ever since it joined the e-shop up to the most recent day of the data. The only problem is that some data are missing - for example, we do not know the price of the product in the weeks when the product was not sold. We will solve this by using the knnImputer, which finds the closest value from the data and fills the missing values accordingly. We will also add new features to the dataset, that is, the amount of products sold k weeks ago. K is the number in the interval from (1, 4), meaning that every weekly sale will also have the past four weeks of sales as a feature.

The last part of the pre-processing done is normalisation. We are using 3 different types of data encoding. We are using cyclical transformation for weeks and months of sales; as explained in the analysis, this approach is great for cyclical values such as these. Then we scale the year with the classical normalisation ($x_i = x_i/max(x)$). The remaining normalisation type, MinMax normalisation, is used on the other available features (20 features in total). Creating those weekly normalised data can be very time consuming if done for every instance of programme. That is why we added the option to store these data as separate .csv files - generated data of weekly sales of each cluster. It is highly recommended to use this option, as it is worth trade-off of memory for computing time. This means that the next time we work with a given cluster, we can only load csv instead of generating the data again. After we have prepared the data, we can move on to the machine learning itself.

4.4 Modelling

The first part of model creation is to choose the data features. At first, we are dropping the columns that we cannot influence for the future, as they would not be suitable features for prediction because they are unknown variables. For example, the total amount of products in the basket falls into this group, as it will be the result of the order, not the input variable. Another feature selection is performed using the variance threshold. This reduces columns with variance lower than 85%, but this threshold can be easily adjusted. 85% seems to be acceptable, therefore it does not exclude valid features, but also helps to reduce the useless ones.

Next, the cluster to make predictions on is selected, data of weekly sales are calculated, and normalisation is performed. The big problem when making this part of the predictions was that we normalised the data *after* the test-train split. This created problems in the modelling, as the train and test parts were normalised separately and the models were confused because of that: if we normalise with the MinMax, for example, the number 1 in the train split could represent the original value of 40, while in the test it would still be 1, even though the original value was 15.

Therefore, after fixing this problem, we normalise the data and only after that make the test and train split. The criterion for this split is the year of sales; 2022 is testing part of the data, which gives almost five months worth of predictions. This time is long enough to adapt the e-Shop towards the expectations, but is also a reasonable prediction time for the models to be accurate.

Now, after the preparation is done, it is time to make models. We have chosen to go and test regressions on this data set - specifically XGBoost regressor, CatBoost regressor, and AdaBoost regressor. As Google trend search shows that XGBoost is still the most popular among these regressors. CatBoost regressor, developed in 2017, is a relatively new open-source machine learning algorithm. As well as XGBoost, CatBoost uses decision tree technology and gradient boosting. AdaBoost, which stands for adaptive boosting, also brings its regressor. AdaBoost regressor is as well based on gradient boosting. Comparing those three similar, yet different regression approaches will be an interesting part of this work.

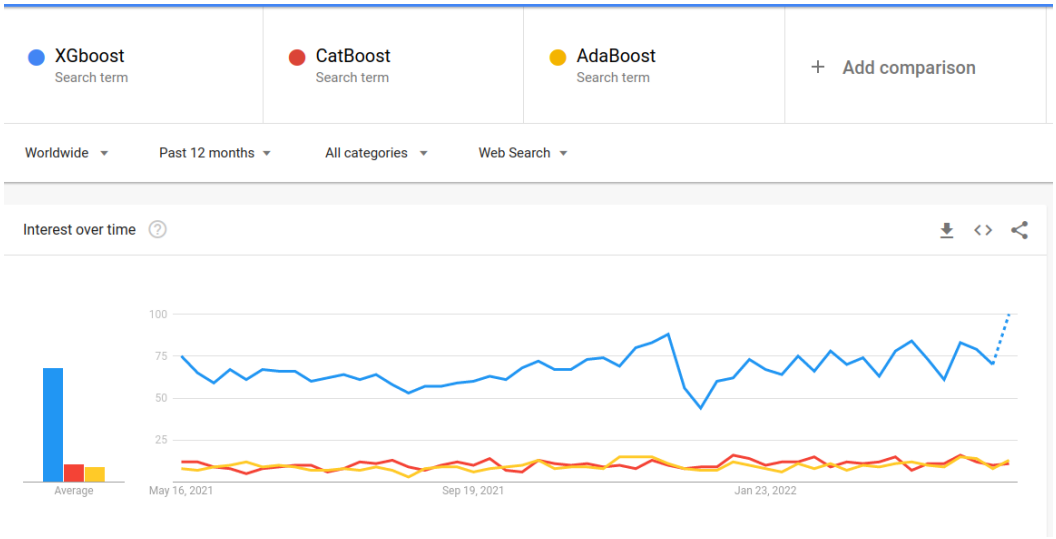

Fig. 7: *XGBoost vs CatBoost vs AdaBoost Google search trends.*

The creation and life cycle of each regressor is approximately the same. First, we found which hyperparameters are suitable for the model. Then we did hyperparameter tuning, using the grid or random search on the possible parameter values. GridSearch tests every hyperparameter configuration possible, but for more parameters (for example, in XGBoost Regressor), this process would take hundreds or thousands of hours. However, a random search is a great alternative. For the fraction of computing time and power, we are able to find a very good configuration of the model. We could also try hyperparameter tuning for each cluster separately, but as mentioned before - it is consuming a substantial amount of time and power, with a very small trade-off of minor improvements. The total overkill would be a full grid search for each cluster on its own - a quick estimation is that this approach would take about 138 years.

Therefore, in the end, the best choice is to find the best hyperparameters of the model on one cluster, with the usage of the RandomSearchCV (We are using GridSearchCV for Ada, since it has a smaller amount of hyperparameters) - we will tune every model on the same cluster, so that the results are comparable. After hyperparameter tuning, we have a model prepared for training and prediction. After making the predictions, their accuracy is also evaluated. That is the penultimate step of the programme. The last remaining part of the work is the results evaluation.

4.5 Implementation

Before advancing to the solution evaluation, we are going to talk about the most important and interesting parts of the code, as well as the technologies used to create this work. If the reader is not interested in this part of the work, they can move on to 4.6, which brings an evaluation of the solution.

The final code is separated into 4 Jupyter notebook (.ipynb) files. Those notebooks are named based on their function, but each name starts with the order of the notebook in their chain. The person with access to the database should be able to start running notebooks at the number 00, the original file data.csv (which is under NDA) allows the user to start at notebook 01. With a freely available data set, the only runnable notebook is 03, but it is also the most important notebook in the chain from the perspective of model creation and testing. Other notebooks are accessible without data, so the code, as well as the output of Jupyter notebook cells, is available to be explored.

4.5.1 Technologies used

Full list of used technologies, alongside with the installation materials can be found in appendix C.1.

Python - The programming language used to programme this solution was Python 3.8.10. This language is great for Data Science and Machine Learning tasks.

Pandas - Pandas is a library built on top of the Python programming language. We use Pandas DataFrame to store and manipulate the data in the entire solution. Pandas offers a wide scale of functions for easier work with the data.

Jupyter - Project Jupyter is a free open-source project that allows the creation of coding notebooks, in which the entire code of this paper was written.

Models - XGBoost and CatBoost are projects of their own, with their own products as well as Python libraries. AdaBoostRegressor is part of the sklearn library, which was also helpful in other ways.

4.5.2 Data obtaining

The data are generated from the PostgreSQL database, via a Python script using the psycopg2 library. It allows us to create the connection with the database, with

the useful part being the usage of environment variables stored in the .env file: it is important because we do not want to write database connection details into the Python notebook. The connection between the tables was already described in 4. To obtain all useful information, our query starts in the shop_orders table and joins other tables, which means that every row in the final data dump is based on order. However, we had to split the SQL query into several smaller queries (with the usage of LIMIT and OFFSET), as the database is not coping very well with such a large query. We have stored those smaller query parts into separate DataFrames, then merged them into one big DataFrame, and saved those data into a .csv file.

4.5.3 Clustering

In our work, clustering is considered a part of pre-processing, because it prepares the data into groups that will be predicted together. The data obtained are of too little density, so we cannot create a fully functional model for each product separately (and even if we could, it would likely be ineffective). That is where clustering shows to be beneficial, and it can help us to group the data much better together. We are using sklearn Kmeans clustering: which on its own is an unsupervised machine learning algorithm, so we are giving products their clusters by 'predicting' them.

```
from sklearn.cluster import KMeans
model = Kmeans(n_clusters=101)
model.fit(data)
data_clusters = model.predict(data)
```

After this part, we use scipy functions to calculate the distance matrix between individual products of the cluster with their features. The metric for distance is the Euclidean distance and divides the kmeans clusters into smaller parts. After this, we just store new values into the database (kMeans cluster and hierarchical cluster), so they are easily accessible.

4.5.4 Models creation

The last notebook (03.Models.ipynb) contains the best reusable code and for a very good reason. While the former notebooks and functions were usually only

required to be run once, the functions in this part of the code are meant to be used more than once - and they usually are, at least once for each used regressor.

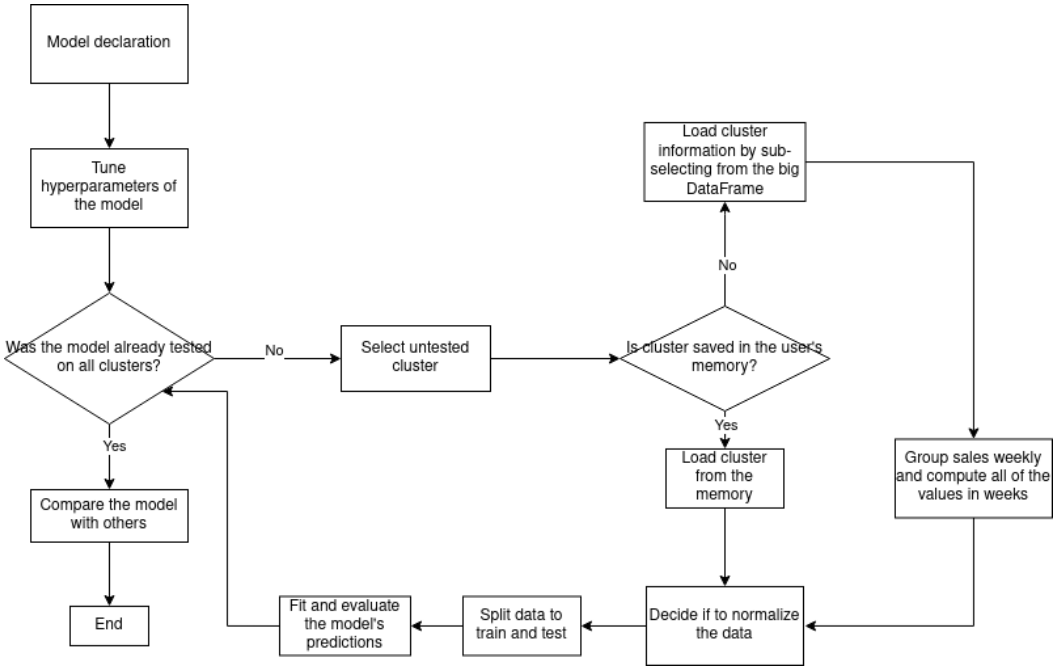

Fig. 8: *Model flowchart used in the modelling part*

In the above image, we can see the process through which a model goes when we are trying to evaluate it. The models themselves are taken from their libraries, and we are just adjusting their hyperparameters. The hyperparameter that seems to be essential and is used in all three model types (although under different names) is the learning rate. It determines the maximum step size of each iteration towards the best model - the model with the minimum of a loss function. Hand in hand with this parameter, we have the learning_rate of the model. There are many different hyperparameters to set up for each model (the full list of hyperparameters for each model is in their documentation: CatBoost, XGBoost and AdaBoost), and we are attempt to optimise the important ones in the hyperparameter tuning.

The main function of this code is *run_evaluate_model*. This function gets a model and cluster numbers (kmeans and hierarchical) as a parameter and calls the data preparation (*prepare_cluster_data*) function over the selected cluster, splits the data (function *test_train_yearly_split*), trains the model by fitting the data, calculates

predictions and evaluates the result (through function *evaluate_model*). The output of this function is then a Json file containing the values of the evaluation methods over the train and the test data split. The output of the evaluation is one JavaScript Object Notation (JSON) file for each regressor type (so 3 in total) in the following format:

```
{"kmeans-1_hierarchical-1": {
      "train": {"medae" : 0.002, "mae" : 0.1374, ...},
      "test": {"medae": 0.006, "mae" : 0.1532, ...}
  },
  "kmeans-1_hierarchical-2": {
     "train" : {...},
     "test" : {...}
  },
  ...
}
```

Files like this contain evaluation numbers for all evaluation metrics used for this task, and the metrics are further explained below. The important thing is that we can always easily access those numbers and we can see which model is best performing in various metrics, or for different clusters.

Table 1: Feature importance

Feature name	%
sold_2_weeks_ago	15.05
reviews_average_score	8.56
reviews_count	8.48
reviews_average_score_properties	6.12
product_id	5.36
sold_4_weeks_ago	5.21
days_in_shop	4.98
sold_1_weeks_ago	4.77
segment_id	4.73
reviews_average_score_price	4.17

We are using 25 features (listed in C.3) and the importance of the top 10 is shown in the table 1. The important part of the model creation is to understand which columns are affecting the model the most. For this reason, we are providing a feature evaluation which consists of calculating the feature importance. To evaluate the importance, we are training 10 models for each regressor type and then we calculate the mean of the features importance across the board. The code can be easily adjusted, so that we could calculate the feature importance based on one model for more accurate feature ranking, but right now we just want to know how impactful the features look overall.

4.6 Evaluation of solution

For the final evaluation, we have chosen to look at the predictions made by the models and to compare the results in several metrics. All of these are calculated with the usage of functions implemented in sklearn, which take real numbers, predict numbers and compare them with their given metrics. Our goal is to determine what accuracy of the predictions we have reached with the usage of the regressors and then compare those regressors.

4.6.1 Evaluation methods

We are using a wide variety of evaluation methods, because the more feedback on the models we have, the better, but the main two metrics are MEDAE and MAE, with the MSE being on the close third place. Other metrics are interesting to see, but they are not primarily used for regression evaluation. The list of used metrics with short explanation:

1. **MEDAE** - Median absolute error reaches the best value at 0.0. It is calculated by taking the median difference of all absolute differences between the real value and the predicted one. It is great when we are dealing with the outliers. The formula for calculating MEDAE is:

$$MedAE(y, x) = median(|y_1 - x_1|, ..., |y_n - x_n|)$$

   where x_i is the predicted value of ith sample and y_i is the corresponding true value.

2. **MAE** - Mean absolute error is a measure of the error between real and predicted values. In other words, it is the mean of the sum of absolute errors divided by the number of samples. In the formula it looks like this:

$$MAE = (\frac{1}{n}) \sum_{i=1}^{n} |y_i - x_i|$$

   and in this case, again x_i and y_i are real and predicted values, with n being the sample size. Similar to the formerly mentioned MEDAE, the best value possible is 0.0.

3. **MSE** - The formula for mean squared error one is:

$$(\frac{1}{n}) \sum_{i=1}^{n} (y_i - x_i)^2$$

   This formula is similar to MAE, even the letters represent the same values - n is the sample size, x_i represents real and y_i the predicted value of the sample. The mean squared error is the last metric primarily meant for regression

evaluation. The best value is once again 0.0, however, this method might not be very suitable for all scenarios, since its error loss rate is quadratic.

4. **Accuracy** - Accuracy score is the fraction of exact matches between the predicted and true values. The best value for accuracy is 1.0.

5. **Recall** - Recall represents the ability of the model to correctly classify truly positive values. It is done by comparing true positives to the real positives (true positives + false negatives). If given the parameter setting of the weighted average (which we did), it calculates the metrics for each label and finds their average weighted by the support (the number of true instances for each label). The best result for recall, precision, and F1 is 1.0

6. **Precision** - Precision looks similarly to recall, but it rather shows the ability of the model to correctly mark a negative value as negative. Again, with the usage of weighted average, the metric is usable for multi-class targets.

7. **F1** - F1 score can be interpreted as a harmonic mean of precision and recall. It is calculated as the multiplication of recall and precision divided by their sum. In the multi-class score, this is the average of the F1 score of each class with weighting depending on the average parameter.

8. The average of these methods - We are calculating two extra values from these methods - **regress**, which is the mean value of the three main regressor values. The second value is **total** which calculates the mean distance from the best score of each metric. The best value is 0.0 for both of them. Additionally, other approaches to these metrics are used,with MEDAE and MAE being valued the most. As we have seen in the analysis, and as we have experienced, these two metrics are great for evaluating predictions.

4.6.2 Comparison of the models

As we have explored, there are many evaluation metrics we can use and test on our data. It can be really interesting to try and look at the model prediction performance from many different angles and we can play around with the evaluation methods, explore how the models are doing on train and test splits, etc.

Table 2: Comparison of the regressors

Metric	Function	XGB	CAT	ADA
Total	Mean	0.1314	0.1501	0.1593
Total	Median	0.1011	0.1216	0.12
MAE	Mean	0.1497	0.2053	0.2149
MAE	Median	0.1362	0.1862	0.189
MEDAE	Mean	0.0282	0.0827	0.0864
MEDAE	Median	0.0253	0.0758	0.0758
Total	best_result	151	72	61
MAE	best_result	172	58	53
MEDAE	best_result	229	23	31

Of course, all values in the table are coming from the evaluation of predictions over the test part of the clusters. There is an opportunity to compare train results in the code, but the models should be evaluated and compared based on the prediction of the unseen data.

The table 2 contains the column named function. This column explains how we evaluated the model's performance over the clusters. We are also exploring an interesting comparison coming from the nature of our data and its split. The best_split function in the table shows how many clusters the given regressor gets the best number for over the given metric.

Based on the evaluation results, we can state that XGBoost regressor is the best of the ones evaluated, with Cat and Ada being much closer. What is interesting to see is that XGBoost is the best overall, but it does not apply to every single cluster - so we might want to use different regressors based on the cluster we are predicting for.

These models are doing well, compared to the papers described in the analysis part. Some of the models described in that have reached better numbers, but usually not by much. We were expecting those results to not be ideal, as the year 2022 is very hard for e-commerce so far. As we have seen in the EDA, it its first time after a long time that the trend is falling and Muziker is no exception in this. With the unstable situation in Europe and fast inflation rate, people are spending less money than was expected based on the previous years. Other big factor is the lockdown ending after almost two years, which means people are visiting shopping centres

more, therefore shopping via the internet less. Therefore, despite the fact that the models are not 100% perfect, they are better than we expected, as, for now, our machine learning does not take into consideration outside factors.

As the last part of the evaluation, we will show how the prediction looks - this is the cluster (1, 1) predicted by the best performing regressor - XGBoost. We can see that the prediction is almost accurate, but as we mentioned - the model in the test part is expecting the sales to be slightly higher than they in reality are.

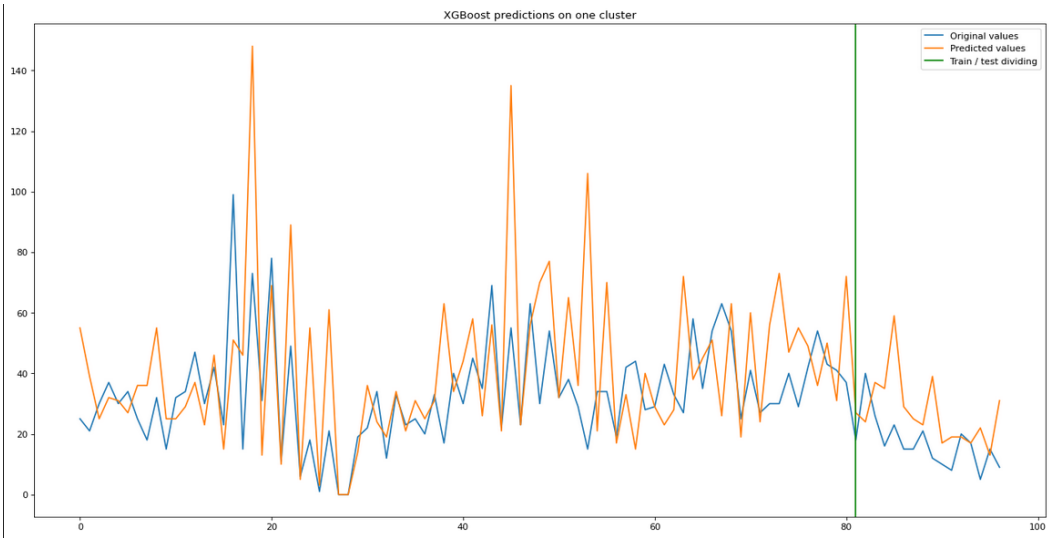

Fig. 9: *XGBoost predictions on cluster*

5 Conclusion

Machine learning is a growing field, which has the big potential for the future. Time series forecasting is still being explored and not fully known, but it is also a very exciting topic, because it offers a great trade-off. E-commerce is also growing and it has great potential of benefiting from the Machine Learning algorithms, if they are implemented correctly.

We read several different articles, especially from recent years and found various approaches to the problem solving of demand forecasting. While exploring this field, we found out that several different techniques were proven to be working well on data forecasting. We have also explored different ways of data preparation, since as we learned, it is a big part of creating machine learning models.

After that, we obtained, explored, analysed, and cleared the data we were given from the e-Shop. We have used different recognised techniques to prepare the data for machine learning - encoding values, dividing the data into smaller parts with the usage of different clustering types and normalising the values in them by choosing the suiting normalisation approaches for the data.

We have decided to try whether our data are suitable for regression models. For that, we had to prepare and fit the data, we had to select the right features and hyperparameters and optimise the models for the best results possible. We also had to explore the evaluation metrics and to implement them in our solution.

With the usage of the Python language and its great machine learning libraries, we were able to prepare everything needed for the task of comparing 3 different regressors - XGBoost, AdaBoost, and CatBoost. We have proven the data of the company to be suitable for machine learning forecasts and found out that XGBoost is, according mainly to MAE and MEDAE evaluation, the best regressor to use for these data. The models are able to comfortably predict 4 months of sales. The other regressors were not too far behind, in some circumstances, they might even be a better choice.

5.1 Future work

As we have mentioned, different machine learning technologies are used in other companies, so the future work might consist of attempting to fit different forecast approaches to the data Muziker has - some options were observed in the analysis of this work, but we believe there are other great solutions to be explored. Another possible route would be to improve this solution by changing the way of data storing, so that the models used in this solution could have easier access to more features from the history of products. However, this is long-term work, because we would need to gather several months worth of information before we could continue.

The last possible route to take from this point is to use the existing models in practice - for example, automate the forecasting of availability in the warehouse.

6 Resumé

6.1 Analýza aktuálnej situácie

Záujem ľudí o nakupovanie na internetových obchodoch rastie každým rokom. V online priestore potom prebieha konkurenčný boj o každého zákazníka. Všetko je dostupné na pár klikov, produkty a obchody sa dajú ľahko porovnávať a zamieňať.

Existuje niekoľko faktorov, ktoré ovplyvňujú rozhodnutia zákazníka. Jedným z hlavných parametrov, ktoré ľudia bez pochýb sledujú je cena. Veľmi jednoducho povedané: čím nižšia cena, tým väčšie predaje sa dajú očakávať. Každá spoločnosť sa snaží maximalizovať svoje zisky a každý zákazník sa snaží nakúpiť najlepšie, ako vie.

Obchod, s ktorým spolupracujeme, v súčasnosti využíva na odhadovanie dopytu a nastavovanie cien historické dáta a manuálne úpravy. Tento proces nie je veľmi efektívny a je časovo i finančne náročný. Avšak, keby vieme, koľko produktov bude predaných za danú cenu, vedeli by sme z toho profitovať napríklad zvýšením cien, alebo objednaním vyššieho počtu kusov tovaru.

Na to, aby sme vedeli predpovedať dopyt po produkte, potrebujeme pochopiť chovanie zákazníkov a ich nákupné zvyky. Tento prístup je používaný na niekoľkých online obchodoch a zdá sa, že je to efektívna cesta ku zlepšeniu predajov [13] [33].

Naša práca sa preto bude zaoberať predpovedaním dopytu po tovare s využitím strojového učenia. Samozrejme, neočakávame 100% úspešnosť, keďže ľudia sa často správajú nepredvídateľne. Avšak, ak sa nám podarí vhodne zozbierať dáta a navrhnúť model, takáto predikcia môže byť cenným pomocníkom.

Strojové učenie je stále rýchlejšie rozvíjajúce sa odvetvie. Na predikcie časových radov sa dajú využiť rôzne spôsoby ako napríklad neurónové siete, modely autoregresného integrovaného kĺzavého priemeru (ARIMA) [11], iné typy regresií a ďalšie, rôznorodé štýly. Napríklad pri neurónových sieťach to môžu byť rekurentné neurónové siete, pre regresory zasa existujú rôzne modely postupného zosilňovania, ARIMA model potom môže mať rôzne konfigurácie.

Všetky typy strojového učenia majú jednu spoločnú vlastnosť a tou je potreba dobrých dát. Požadované dáta získame z internetového obchodu vďaka spolupráci so spoločnosťou SoftPoint.

Dáta, ktoré dostaneme bude pravdepodobne potrebné rozdeliť do skupín. Na to môžeme využiť kategórie, do ktorých produkty prirodzene spadajú, alebo techniky zhlukovania. Tieto techniky sa dajú rozdeliť do dvoch hlavných skupín - oddielové a hierarchické. Oddielové metódy majú preddefinovaný počet finálnych zhlukov, do ktorých dáta rozdeľujú. Hierarchické zhlukovanie samo určuje finálny počet skupín, podľa preddefinovanej mierky podobnosti. Toto zhlukovanie sa delí ďalej - na aglomeratívne a rozdeľujúce.

Okrem toho, že dáta treba rozdeliť do skupín je potrebné, aby boli pripravené na účely strojového učenia. Platí napríklad to, že modely strojového učenia dokážu spracovať iba numerické hodnoty. Jeden z ďalších prístupov ošetrovania týchto hodnôt, predstavuje normalizácia dát - teda ich preškálovanie na iný interval tak, aby sa všetky vlastnosti dát nachádzali v približne rovnakých hodnotách. Inak by sa mohlo stať, že by program vyšším hodnotám automaticky pridelil vyššiu dôležitosť (napríklad mesiacov je iba 12, ale číslo roku je 2022 - pritom obe majú podobnú dôležitosť, dokonca očakávame, že mesiac bude podstatnejší). Taktiež existuje niekoľko štýlov normalizácie. Konkrétne príklady sú MinMax normalizácia na interval <0, 1>, cyklické kódovanie, ktoré je vhodné na cyklické dáta ako sú mesiace, či dni. Využívaná je aj štandardizácia, SoftMax, či klasická normalizácia. Rôzne metódy ale spejú k jednému cieľu - lepšej príprave dát pre strojové učenie.

V posledných rokoch sa tematika predikcie časových radov stáva veľmi populárnou po celom svete. Rôzne obchody využívajú rôzne prístupy a skúmajú, ktorá verzia funguje najlepšie. Oblasť umelej inteligencie sa vyvíja rýchlym tempom a tak za posledné roky vzniklo pomerné množstvo zaujímavých článkov o tematike predikcií predaja cez internetový obchod. Ako príklad si môžeme vziať internetový obchod Rue La La [13], pre ktorý bola modelovaná predikcia predaja tovaru na základe historických dát s použitím regresných modelov. Ako najlepšie sa ukázali algoritmy využívajúce regresné stromy. Podobný prístup bol testovaný aj na trhu nehnuteľností [12], kedy boli využité rôzne prístupy k regresii. Vidíme ale aj príklady využitia neurónových sietí [3] či auto regresných modelov [11]. Takže skutočne, všetky tieto prístupy našli uplatnenie na reálnom trhu a ich popis vykazuje dobré výsledky. Výsledky sú zvyčajne získavané štatistickými metódami - ako napríklad výpočtom strednej absolútnej chyby, strednej percentuálnej absolútnej chyby, alebo aj výpočtami chýb s využitím hodnoty mediánu.

Riešenie problému

Cieľom tejto práce je získať správne dáta z internetového obchodu, spoznať a opísať tieto dáta, pripraviť ich na strojové učenie a následne implementovať modely umelej inteligencie na predikciu dopytu tovaru. Následne je potrebné takéto modely ohodnotiť, aby sme zistili úspešnosť ich predpovedí. Na vypracovanie tejto práce bude využitý jazyk Python, spoločne s jeho knižnicami, s použitím technológií Jupyter Notebookov.

Prvou časťou pochopenia problému bolo vyznať sa v databáze, v ktorej sú uložené dáta online obchodu Muzikeru. Táto databáza využíva databázový jazyk PostgreSQL a na prístup do vnútra je potrebné dostať prihlasovacie údaje od firmy, ktorá ju spravuje. Databáza tohto internetového obchodu je naozaj obrovská: obsahuje niekoľko schém, ktoré spoločne predstavujú vyše 300 tabuliek, pričom počet záznamov v tabuľkách sa pohybuje od jednotkových čísel až po milióny riadkov s desiatkami stĺpcov.

Zorientovať sa v dátach zabralo značné úsilie a čas, ale keď sa nám to podarilo zistili sme, že podstatné informácie pre náš problém sa dajú vytiahnuť z databázy dopytovým spojením 8 tabuliek. Takto získané tabuľky nám poskytli podstatné informácie o objednávkach, košíkoch objednávok, predmetoch v košíku, ktoré sú nemapované na produkty v databáze. Vieme, aký produkt bol v ktorý deň predaný koľko krát, za akú cenu, v akej krajine a aké predmety boli predané spolu s ním. Vieme zistiť, akú má produkt kategóriu, do akého patrí segmentu, akej je značky, vieme i to, či je ešte v predaji, odkedy sa nachádza v databáze a mnoho ďalších, potenciálne užitočných parametrov produktu.

Tieto údaje sú, samozrejme, rozložené medzi osem spomínaných tabuliek, ktoré majú medzi sebou logicky vytvorené spojenia cez cudzie kľúče, ako je štandardom pri databázových usporiadaniach SQL. Konkrétne tabuľky sú: objednávky, meny, košíky, predmety v košíku, tabuľka predmetov, kategórie, segmenty a značky. Dáta, ktoré sme získali týmto štýlom uložíme do .csv súboru. Takéto, čisté a neupravené dáta ešte stále podliehajú licenčnej zmluve spoločnosti, preto nie je možné verejne zdieľať tento dátový set.

Po získaní dát je ďalšou podstatnou časťou pochopiť, čo sa vlastne v dátach nachádza, prípadne si pozrieť nejaké vzťahy medzi dátami a pod. Inými slovami sa

presúvame do prieskumnej analýzy údajov, ktorej cieľom je celkové pochopenie dostupných dát. V rámci našej analýzy príde aj k čiastočnému prečisteniu dát, keďže máme rovno na mysli finálny produkt - modely umelej inteligencie a vďaka tomu vieme, ktoré dáta môžeme začať vyhadzovať.

Najskôr sme zistili rozsah dát - dokopy máme približne 3.6 milióna záznamov o objednávkach, s 58 stĺpcami charakterizujúcimi každú objednávku. 3 z týchto stĺpcov obsahujú dátumy, 28 stĺpcov má číselné hodnoty, v 5 sú hodnoty Pravda / Nepravda a 22 stĺpcov má iný dátový typ.

Ďalej sme z dát vyhodili objednávky, ktoré boli zrušené (asi 100 000 záznamov) a 25000 záznamov, ktoré obsahujú prázdne hodnoty. Potom sa chceme pozrieť na to, aké typy produktov a kategórií sú definované. Produkty mali pôvodne 9 rôznych typov, ale po vyhodení darčekov, darčekových kariet a ich variácií nám ostali iba 3. Pri kategóriách to bolo jednoduchšie, keďže 3/3 kategórii sme mohli ponechať.

Muziker je medzinárodný obchod a preto nás zaujíma aj zdroj objednávok a použitá mena. Zistili sme, že viac ako 30% objednávok pochádza z Českej republiky, druhé je Slovensko a za ním Maďarsko. Najpoužívanejšou menou je Euro, ale ceny pri nákupoch produktov boli vždy v databáze uvedené v pôvodnej hodnote. Preto sme s využitím stĺpca kurzu prekonvertovali všetky cenové čísla na Eurá - vďaka tomu máme možnosť jednoduchých porovnaní.

Potom sme zistili, že sa jedná o predaje približne 150 000 produktov, pričom ceny sa pohybujú od pár centov až po 22 799 eur. Okrem toho sa potvrdila naša domnienka, že najsilnejšie predaje sú v okolí Vianoc. Trend predajov je dlhodobo rastúci, avšak v poslednej dobe nastalo prvé klesanie v histórií, čo by mohlo negatívne ovplyvniť modely, ktoré môžu predpokladať zlepšenia.

Ďalšou časťou riešenia je pred-spracovanie dát. Prvým krokom je zmena nečíselných parametrov na numerické. Ako prvé zmeníme dátumy. Pri dátumoch aplikujeme dva prístupy - dátum dokumentu rozdelíme do viacerých stĺpcov - dni, týždne, roky, mesiace a dni v týždni. Druhý využitý prístup bol pri počte dní, počas ktorých sa produkt nachádza v obchode. Pôvodne sme mali uložený dátum pridania do obchodu, ale bolo jednoduché zmeniť ho na číslo dní, ktoré sa nachádza v obchode. Ďalej sme zakódovali zvyšné nenumerické hodnoty na čísla - pri niektorých sme využili, že ich reprezentuje id z ich tabuľky, na iné sme použili kódovanie.

Po zakódovaní dá sme identifikovali odľahlé hodnoty a vyhodili tieto údaje z databázy. Odľahlé hodnoty sme identifikovali ako tie, ktoré ležia ďalej ako trojnásobok štandardnej odchýlky od priemeru.

Následne dáta roztriedime do 303 zhlukov. Proces zhlukovania prebieha v dvoch iteráciách - v prvej sa celé dáta rozdelia do k (101) zhlukov s využitím zhlukovania metódou najbližších susedov a následne jednotlivé zhluky potom ešte rozdelíme hierarchickým zhlukovaním. Po tejto fáze vygenerujeme .csv súbor *data_after_preprocessing.csv*. Tento súbor obsahuje 2.7 miliónov riadkov v 44 stĺpcoch.

Posledné dva úseky predspracovania sa nachádzajú v poslednej časti kódu. V tejto časti napočítame týždenné predaje jednotlivých produktov. Naše predikcie budú založené na týždennej báze, nakoľko je to výhodnejšie a naše dáta sú príliš riedke na to, aby sa nám oplatilo riešiť denné predikcie. Po rozhovore s ľuďmi s firmy sme zistili, že týždenné predikcie na 4 mesiace dopredu (teda 16 predikovaných hodnôt pre každý produkt) sú plne dostatočné. Napokon dáta normalizujeme s využitím cyklického kódovania na mesiace a týždne, rok normalizujeme a na zvyšné hodnoty použijeme MinMax normalizačnú techniku.

Dostali sme sa do poslednej časti riešenia, kedy prišiel čas vytvárať modely. V našom riešení budeme testovať tri rôzne typy regressorov - konkrétne CatBoost, XGBoost a AdaBoost regressory. Dáta máme pripravené, zostáva nám rozdeliť ich na trénovaciu a testovaciu množinu. Testovacia množina predstavuje predaje z roku 2022 - teda niečo okolo 4 mesiacov, zvyšné hodnoty (roky 2019-2021) budú slúžiť na trénovanie modelov.

Súčasťou trénovania modelov je výber vlastností, ktoré budú reprezentovať dáta. Používame len tie hodnoty, ktoré prešli testom rozptylu - teda sú dostatočne rozdielne na to, aby ovplyvňovali model. Následne modely vytvoríme a vyberieme im hyperparametre - s využitím náhodnej vyhľadávacej mriežky.

Posledná časť je vyhodnotenie modelov. Potom, čo sme každý model natrénovali a vylepšili mu hyperparametre, vyskúšame jeho nasadenie a predikovanie pre každý zhluk z databázy. Zo štatistických testov medzi predikovanými a reálnymi hodnotami následne dostaneme úspešnosť predikcie modelu nad zhlukom.

Dve najpodstatnejšie metriky pre vyhodnotenie predikcií ktoré používame sú:

1. **Stredná absolútna chyba (MAE)** - priemer všetkých absolútnych odchýlok v predikciách.

2. **Mediánová absolútna chyba (MEDAE)** - medián absolútnych odchýlok predikovaných hodnôt od skutočných.

Zhodnotenie

Po úspešnom natrénovaní a ohodnotení modelov sme zistili, že XGBoost dosahuje najlepšie výsledky - koknrétne je jeho priemerné MAE naprieč zhlukmi 0.14, zatiaľčo CatBoost (0.20) a AdaBoost (0.21) dopadli o niečo horšie. Čo je ale hlavné je, že sa nám podarilo úspešne natrénovať modely strojového učenia na dátach obchodu, s využitím 25 rôznych parametrov dát. C.3

Oblasť umelej inteligencie a strojového učenia sa stále rozrastá a mení, ale aktuálne sa zdá, že má pred sebou veľkú budúcnosť. Pre online obchody je veľkou výhodou, ak vedia spracovať dáta, ktoré im o sebe zákazníci poskytnú.

V našej práci sa ukázalo, že Muziker dáta sú vhodné na využitie algoritmov strojového učenia. Najlepšie výsledky z vyskúšaných algoritmov nám priniesol XGBoost regresor, ale do budúcna môže byť veľkým prínosom skúmanie aj iných technológií umelej inteligencie.

References

[1] AGHABOZORGI, S., SHIRKHORSHIDI, A. S., AND WAH, T. Y. Time-series clustering–a decade review. *Information Systems 53* (2015), 16–38.

[2] AL-MUSAYLH, M. S., DEO, R. C., ADAMOWSKI, J. F., AND LI, Y. Short-term electricity demand forecasting with mars, svr and arima models using aggregated demand data in queensland, australia. *Advanced Engineering Informatics 35* (2018), 1–16.

[3] BANDARA, K., SHI, P., BERGMEIR, C., HEWAMALAGE, H., TRAN, Q., AND SEAMAN, B. Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *International conference on neural information processing* (2019), Springer, pp. 462–474.

[4] BEDI, J., AND TOSHNIWAL, D. Deep learning framework to forecast electricity demand. *Applied energy 238* (2019), 1312–1326.

[5] BROCKWELL, P. J., BROCKWELL, P. J., DAVIS, R. A., AND DAVIS, R. A. *Introduction to time series and forecasting*. Springer, 2016.

[6] CAI, X., AND XU, T. Dtwnet: a dynamic timewarping network. *Advances in Neural Information Processing Systems 32 (NeurIPS 2019) 32* (2019).

[7] CHIMMULA, V. K. R., AND ZHANG, L. Time series forecasting of covid-19 transmission in canada using lstm networks. *Chaos, Solitons & Fractals 135* (2020), 109864.

[8] COHEN-ADDAD, V., KANADE, V., MALLMANN-TRENN, F., AND MATHIEU, C. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM) 66*, 4 (2019), 1–42.

[9] DE AMORIM, R. C., AND MIRKIN, B. Minkowski metric, feature weighting and anomalous cluster initializing in k-means clustering. *Pattern Recognition 45*, 3 (2012), 1061–1075.

[10] DEY, R., AND SALEM, F. M. Gate-variants of gated recurrent unit (gru) neural networks. In *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)* (2017), IEEE, pp. 1597–1600.

[11] FATTAH, J., EZZINE, L., AMAN, Z., EL MOUSSAMI, H., AND LACHHAB, A. Forecasting of demand using arima model. *International Journal of Engineering Business Management 10* (2018), 1847979018808673.

[12] FEDOROV, N., AND PETRICHENKO, Y. Gradient boosting–based machine learning methods in real estate market forecasting. In *8th Scientific Conference on Information Technologies for Intelligent Decision Making Support (ITIDS 2020)* (2020), Atlantis Press, pp. 203–208.

[13] FERREIRA, K. J., LEE, B. H. A., AND SIMCHI-LEVI, D. Analytics for an online retailer: Demand forecasting and price optimization. *Manufacturing & Service Operations Management 18*, 1 (2016), 69–88.

[14] HANCOCK, J. T., AND KHOSHGOFTAAR, T. M. Catboost for big data: an interdisciplinary review. *Journal of big data 7*, 1 (2020), 1–45.

[15] JAMIL, R. Hydroelectricity consumption forecast for pakistan using arima modeling and supply-demand analysis for the year 2030. *Renewable Energy 154* (2020), 1–10.

[16] KRISLOCK, N., AND WOLKOWICZ, H. Euclidean distance matrices and applications. In *Handbook on semidefinite, conic and polynomial optimization*. Springer, 2012, pp. 879–914.

[17] MADHULATHA, T. S. An overview on clustering methods. *arXiv preprint arXiv:1205.1117* (2012).

[18] PANDIT, S., GUPTA, S., ET AL. A comparative study on distance measuring approaches for clustering. *International Journal of Research in Computer Science 2*, 1 (2011), 29–31.

[19] PATRO, S., AND SAHU, K. K. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462* (2015).

[20] PERONE, G. An arima model to forecast the spread and the final size of covid-2019 epidemic in italy. *HEDG-Health Econometrics and Data Group Working Paper Series, University of York (2020)* (2020).

[21] PRYZANT, R., CHUNG, Y., AND JURAFSKY, D. Predicting sales from the language of product descriptions. In *eCOM@ SIGIR* (2017).

[22] SAIF, A. S., GARBA, A. G., AWWALU, J., ARSHAD, H., AND ZAKARIA, L. Q. Performance comparison of min-max normalisation on frontal face detection using haar classifiers. *Pertanika J. Sci. Technol. 25* (2017), 163–171.

[23] SAK, H., SENIOR, A. W., AND BEAUFAYS, F. Long short-term memory recurrent neural network architectures for large scale acoustic modeling.

[24] SAXENA, A., PRASAD, M., GUPTA, A., BHARILL, N., PATEL, O. P., TIWARI, A., ER, M. J., DING, W., AND LIN, C.-T. A review of clustering techniques and developments. *Neurocomputing 267* (2017), 664–681.

[25] SCHUBERT, E., SANDER, J., ESTER, M., KRIEGEL, H. P., AND XU, X. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS) 42*, 3 (2017), 1–21.

[26] SHILONG, Z., ET AL. Machine learning model for sales forecasting by using xgboost. In *2021 IEEE International Conference on Consumer Electronics and Computer Engineering (ICCECE)* (2021), IEEE, pp. 480–483.

[27] SIAMI-NAMINI, S., TAVAKOLI, N., AND NAMIN, A. S. A comparison of arima and lstm in forecasting time series. In *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)* (2018), IEEE, pp. 1394–1401.

[28] SINGH, A., YADAV, A., AND RANA, A. K-means with three different distance metrics. *International Journal of Computer Applications 67*, 10 (2013).

[29] SONG, H., QIU, R. T., AND PARK, J. A review of research on tourism demand forecasting: Launching the annals of tourism research curated collection on tourism demand forecasting. *Annals of Tourism Research 75* (2019), 338–362.

[30] TABAK, J. *Geometry: the language of space and form*. Infobase Publishing, 2014.

[31] TEMUR, A. S., AKGÜN, M., AND TEMUR, G. Predicting housing sales in turkey using arima, lstm and hybrid models.

[32] TUKEY, J. W., ET AL. *Exploratory data analysis*, vol. 2. Reading, MA, 1977.

[33] VAKHUTINSKY, A. Retail markdown price optimization and inventory allocation under demand parameter uncertainty. *Available at SSRN 3875689* (2021).

[34] VAN KIEN PHAM, THU HA DO THI, THU HOAI HA LE. A study on the covid-19 awareness affecting the consumer perceived benefits of online shopping in vietnam. *Cogent Business & Management 7*, 1 (2020), 1846882.

[35] WANG, X., SMITH, K., AND HYNDMAN, R. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery 13*, 3 (2006), 335–364.

[36] WANG, Y., AND GUO, Y. Forecasting method of stock market volatility in time series data based on mixed model of arima and xgboost. *China Communications 17*, 3 (2020), 205–221.

[37] ZHANG, G. P. Time series forecasting using a hybrid arima and neural network model. *Neurocomputing 50* (2003), 159–175.

[38] ZHANG, X.-L., LUO, Z.-G., AND LI, M. Merge-weighted dynamic time warping for speech recognition. *Journal of Computer Science and Technology 29*, 6 (2014), 1072–1082.

[39] ZIA, T., AND ZAHID, U. Long short-term memory recurrent neural network architectures for urdu acoustic modeling. *International Journal of Speech Technology 22*, 1 (2019), 21–30.

A Appendix: Work plan in the winter semester

Table 1 shows the plan of work during the first semester of work.

Table 1: Work plan for the first semester

Week of the semester	Planned work	Plan compliance success
1.	Explore the field of demand prediction.	The plan was fulfilled
2.	Read articles provided in thesis's description	The plan was fulfilled
3.	Gather more articles discussing the topic of demand forecast and it's variations.	The plan was fulfilled
4.	Gain experience in the work with neural networks.	The plan was fulfilled
5.	Create and adjust LaTeX document for the purpose of thesis. Write the motivation part of introduction,	The plan was fulfilled
6.	Finish introduction and start with text analysis of clustering techniques.	The plan was fulfilled
7.	Adjust clustering analysis part of document.	The plan was fulfilled
8.	Finish clustering studies and explore distance computing.	The plan was fulfilled
9.	Look at the real data and explore available information	The plan was fulfilled
10.	Start the chapter 3 - write about time series forecasting and LSTMs.	The plan was fulfilled
11.	Explore and write about data normalisation	The plan was fulfilled
12.	Finalise chapter 5 - write about related studies and draft of solution. Modify the document's style and fix errors.	The plan was fulfilled during this week and following extra weeks

B Appendix: Work plan for the summer semester

Table 2 shows the plan of work during the second semester of work.

Table 2: Work plan for the second semester

Week of the semester	Planned work	Plan compliance success
1.	Finish exploring the database and write SQL to get required data.	The plan was fulfilled
2.	Start working on EDA.	Postponed due to illness
3.	Analyse products in EDA and write about it.	The plan was fulfilled
4.	Finish EDA and write description of it.	The plan was fulfilled
5.	Add the first data preprocessing layer.	The plan was fulfilled
6.	Explore clustering and implement it into your code.	The plan was fulfilled
7.	Write parts of the document according to your work from past weeks.	The plan was fulfilled
8.	Try to explore regression models and fit at least part of the data onto them.	The plan was fulfilled
9.	Clear data further, refactor and clean code.	The plan was fulfilled
10.	Finalise the code of the regressors and fit data onto them.	The plan was fulfilled
11.	Test the models, hyperparameters and make evaluations.	The plan was fulfilled
12.	Finish the parts of solution description that are missing.	The plan was fulfilled
13.	Finalise the document and prepare everything for the final submission.	The plan was fulfilled

C Appendix: Technical documentation

The goal of technical documentation is to allow users to run the programme on their own device and to explain some additional parts of the code compared to the parts explained in the implementation.

C.1 Installing necessary parts

Python is the programming language necessary for the code to run. The program was written in Python version 3.8.10, but any newer version is expected to work fine with the code as well. Python can be downloaded for free on the website of this programming language. There is a division based on the operating system, so every user can choose what suits them. On *Linux and MacOS* systems, the installation of pip is recommended. That can be done by the following: `python get-pip.py` command on MacOS and Linux. On Windows the command is: `py get-pip.py`.

Jupyter - Two variations of the Jupyter exist - Jupyter Notebook and Jupyter labs. Do not confuse Jupyter Notebook technology with Jupyter notebook files - .ipynb files. They share the same name, but one of them is the file type used across this project and the other is an application to run such files. The code in this work is runnable on both - Jupyter Notebook as well as Jupyter Lab, but we recommend using the latter, as the programme was developed on Lab and is a newer technology.

On **Linux** systems, the installation of jupyter is, once again, done with one single command `pip install jupyterlab`. We were developing on Linux based systems, but Jupyter applications are also available on MacOS and Windows, although we recommend the usage of Conda on those operating systems. Anaconda can be downloaded from its website for Linux, MacOS, or Windows. After downloading it, the user can simply download packages, including Jupyter Notebook or Jupyter Labs.

The last part of the installation is to install Python libraries used in solution. This can be done very conveniently, thanks to the attached requirements.txt file. Pip uses `pip install -r requirements.txt` console command, when the console is open in the directory with requirements.txt file. Conda's alternative

for this is `conda install –file requirements.txt`, but be aware of the fact that Conda installs a list of packages into a specified Conda environment.

Here is the list of the used libraries, alongside with their versions used by us and links to the websites of their documentation and the short use cases of the solution.

- Catboost - ver. 1.0.5, https://catboost.ai/. Library containing CatBoost regressor.
- Category-encoders - ver. 2.4.0, https://contrib.scikit-learn.org/category_encoders/. Used in the preprocessing to convert categorical data to integers.
- Feature-engine - ver. 1.2.0, https://feature-engine.readthedocs.io/en/1.2.x/. Useful for data normalisation techniques.
- Json5 - ver. 0.9.6, https://json5.org/. Used to store and Python dictionaries as JSON files.
- Matplotlib - ver. 3.5.1 https://matplotlib.org/. Very powerful tool for drawing graphs and plots.
- Numpy - ver. 1.22.3 https://numpy.org/. NumPy is an elementary package for scientific computing in Python.
- Pandas - ver. 1.4.1 https://pandas.pydata.org. Pandas is perfect for data analysis and storing.
- Python-dotenv - ver. 0.19.2 https://pypi.org/project/python-dotenv/. Python dotenv is the right tool for environment variables setting.
- Psycopg2-binary - ver. 2.9.3 https://pypi.org/project/psycopg2/. Psycopg2 is used to connect to the SQL database from Python.
- Scipy - ver. 1.8.0 https://scipy.org/. Scipy is fundamental for computing in Python. It is helpful for clustering.
- Seaborn - ver. 0.11.2 https://seaborn.pydata.org/. Useful to visualise data, amazing in combination with matplotlib.

- Scikit-learn - ver. 1.0.2 https://scikit-learn.org/stable/. Basic machine learning library in Python, built on NumPy, SciPy and matplotlib.
- Statsmodels - ver. 0.13.2 https://www.statsmodels.org/. Ideal for finding trends and seasonality in data.
- Xgboost - ver. 1.6.0 https://xgboost.readthedocs.io. XGBoost regressing.

After downloading all of these, the programming environment is ready to go.

C.2 The data

The data structure and obtaining is described deeper in the document. The code is written in 4 notebooks - with their names, but also numbers 00-04. The notebook 00 generates the data by downloading them from the database and storing them into 'data.csv' file. This file, however, contains data that can not be shared freely. If you need these data, please feel free to contact us.

We are sharing 250 000 rows of 3.6 millions in the file called 'data_snippet.csv'. The reason we did not describe this in the main documentation part is because the snippet only contains a small part of the data and so it is likely that some part of the code will not function properly. If you want to load this file into 01 notebook, you need either change the path to 'data/data_snippet.csv' or you can rename this file to 'data.csv'. The purpose of this file is to give the user at least some opportunity to explore the notebooks 01 and 02 - since the 02 notebook's input is the output of the notebook 01, even though the results will not be as representative as they are in the main document part.

However, for the 03 notebook, we still **strongly recommend** using the original 'data_after_preprocessing.csv' which was generated by us, as the result of the big dataset coming from 00 through 01 and 02.

Essentially, two data files are freely available. One of them is 'data_after_preprocessing.csv', which is the same file we used when writing this work and 'data_snipped.csv', which is not meant to work perfectly, but rather show a part of our work, that would be otherwise hidden.

C.3 Code parts

As mentioned previously, the code is divided into four files. Their exact names are:

- 00.Data_Generation.ipynb
- 01.EDA.ipynb
- 02.Preprocessing.ipynb
- 03.Models.ipynb

We believe that their names are explanatory of their purposes.

When it comes down to the code parts - most of the important ones were described in the main document and the rest contains comments with the usage of docstrings in functions. The code was submitted in .ipynb notebooks, but browser-generated .pdf files of the code were also part of the submission, so that anyone can read any part of the code they would like.

There are, however, two thing about the code we would like to point out. The first is the following function (it is longer in the code, but parts the part relevant for explanation are preserved):

```
def prepare_weekly_csvs(df : pd.DataFrame, normalized »
  » :bool = True):
for kmns in range(101):
    for hier in range(1, 4):
        file_path = DATA_CLUSTERS_PATH+»
          » STANDARD_FILE_NAME+'_kmeans-'+str(kmns)+'»
          » _hier-'+str(hier)+'.csv'
        if normalized:
            file_path = DATA_CLUSTERS_PATH+»
              » NORMALIZED_FILE_NAME+'_kmeans-'+str(»
              » kmns)+'_hier-'+str(hier)+'.csv'

        if not os.path.exists(file_path):
```

```
prepare_cluster_data(df, hier, kmns, »
  » normalized).to_csv(file_path, index=»
  » False)
```

On an average laptop, the function was executed for around 8 hours and requires 6.1 GB free space. Its purpose is to prepare weekly sales of every product, normalise new data (if chosen so), and save the files on your disk.

This is crucial, because creating weekly sales for each product of the cluster takes a long time. The clusters are needed for the model evaluation, thus if they were not prepared in advance, it would take exponential amounts of time to do each time we evaluate.

If someone can not afford to leave the notebook to run the code for 8 hours, they can adjust the range of kmns (currently 101) to smaller number - let say 20. That would only create the first 20 kmeans clusters (so 60 files, as each kmeans cluster is further divided into hierarchical clusters), but the program would shut down afterwards. Then the user can simply change the number from 20 to 40 and run the programme again - the first 20 cluster creations will be skipped (if they were not removed or moved away from their file). This allows us to batch the workflow and to save a lot of time further on.

The last thing we believe should be mentioned is the list of all features used in the final iteration of prediction models in the solution. All 25 feature columns used are:

- year
- item_unit_price_with_vat
- product_purchase_price
- item_total_discount_with_vat
- product_id
- category_id
- brand_id

- segment_id
- reviews_count
- reviews_average_score_price
- reviews_average_score_quality
- reviews_average_score_properties
- reviews_average_score_overall
- reviews_average_score
- product_status
- ancestor_count
- days_in_shop
- sold_1_weeks_ago
- sold_2_weeks_ago
- sold_3_weeks_ago
- sold_4_weeks_ago
- week_sin
- week_cos
- month_sin
- month_cos

D Appendix: Electronic media content

Registration number: **FIIT-5212-103121**

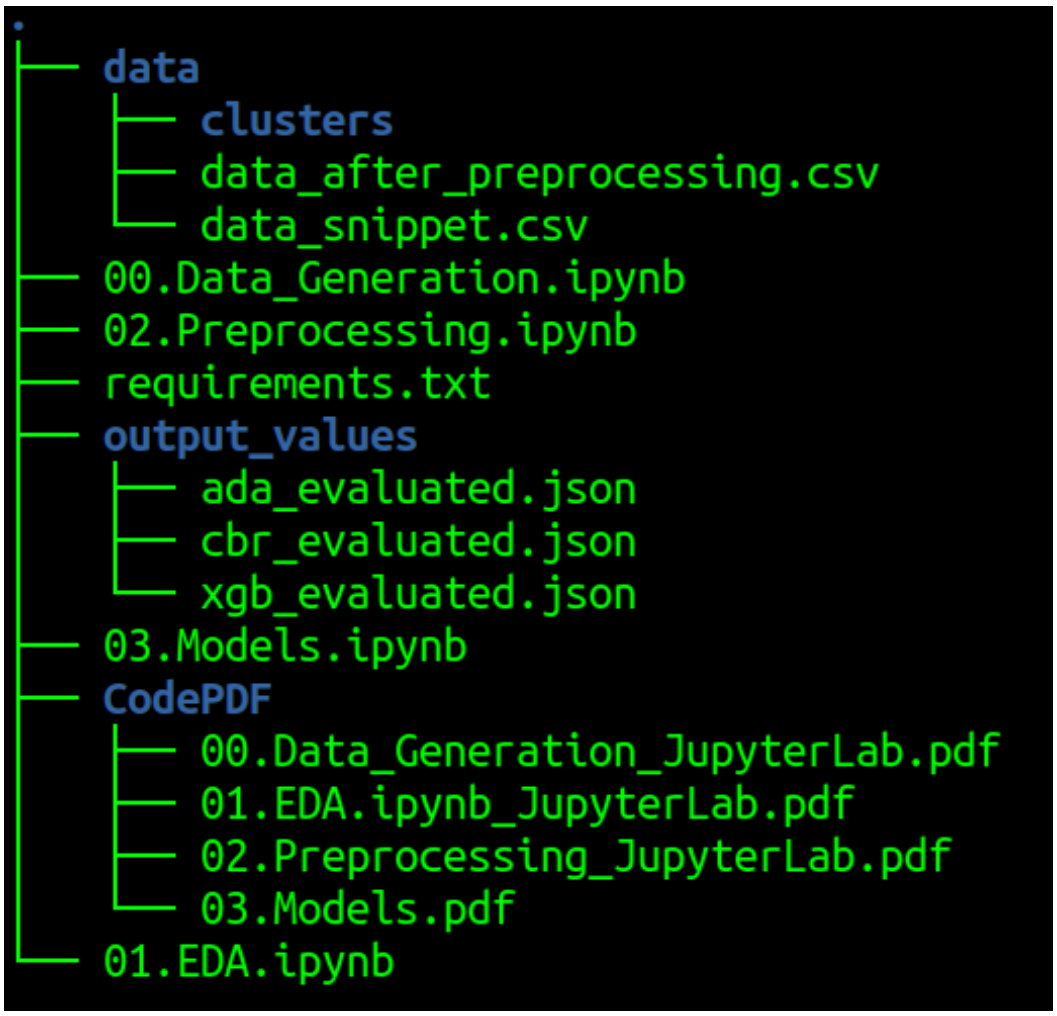

Fig. 10: *Tree of submitted files*