2. Methods

2.1. Machine learning

The machine learning discipline contains a lot of powerful methods to solve complex problems. There are several definitions of machine learning. One of the most reasonable is as follows: "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E." [8]. So, if our problem is too complex to be solved directly (with analytical or numeric methods) and we have enough data we can try to build a machine learning model able to extract knowledge from this data. During recent years, we can observe an increasing interest in this field due to several amazing results in different domains; for example, image processing, natural language processing, recommendation systems, games and even art. Such progress in developing machine learning algorithms can be explained by several reasons: the growing available amount of data, growing computational performance and developing effective algorithms.

Common machine learning project workflows consist of five main stages presented in figure 1 [9]. Usually, it is not enough to perform these procedures only one time. Several cycles are needed until the goal is achieved. Although there are automated machine learning (AutoML) techniques which try to automatically cover the complete pipeline, nowadays it does not work well when dealing with complex problems, so we need to release all the stages manually.

19th International Conference "Aviation and Cosmonautics" (AviaSpace-2020) IOP Publishing
Journal of Physics: Conference Series **1925** (2021) 012033 doi:10.1088/1742-6596/1925/1/012033

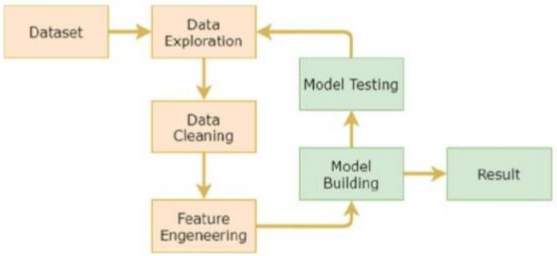


Figure 1. Machine learning project workflow.

2.2. Data set description, feature engineering, validation and metric

At the stage of data exploration available data were examined. There were defined types, distributions, percentages of missing values and information was received from domain knowledge experts to understand which data is possibly useful for the problem solution. After that a data cleaning procedure was performed – outliers were removed, some rows with missing values were dropped. After all these procedures, the data set consisted of more than 4.5 billion samples of sales which were made from 2012 to 2020. There were data for about 89 000 items. An example of one item data is presented in figure 2. All data were divided by 24 categories like "Sport", "Zoo", "Furniture", "Books" etc.

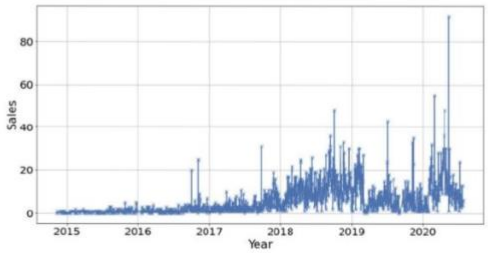


Figure 2. One item history.

The next important procedures in the machine learning pipeline are feature processing and generation. All the data presented in this paper were taken from Ozon company database which stores the history of sales, pre-processed by means of Pandas library, and visualized using Matplotlib library.There are two commonly used approaches of feature generation – use already trained models for feature extraction or manually create features based on the domain knowledge. Figure 3 presents sales distributions by month and figure 4 presents sales by date Obviously, time-dependent features contain useful information for our problem. Other important features are price, promotion and category. Although we could not use data about the history of each item's sales, we used historical data grouped by categories. Examples of formulated features include such features as: "Average brand sales within a

The problem of predicting demand for a new product based on its characteristics and description is

critical for various industrial enterprises, wholesale and retail trade and, especially, for modern highly competitive sector of air transportation, since solving this problem will optimize production, management and logistics in order to maximize profits and minimize costs. Classic demand forecasting methods assume the availability of sales data for a certain historical period, which is obviously not the case when concerning a new product. Most research papers are limited either to a specific category of goods or use sophisticated marketing methods. This paper proposes the use of machine learning methods. We used data about new product demand from the Ozon online store. The input data of the algorithm are characteristics such as the price, name, category and text description of the product. To solve the regression problem, various implementations of the gradient boosting algorithm were used, such as XGBoost, Light GBM, Cat Boost. The forecast accuracy is now about 4.00. The proposed system can be used both independently and as part of another more complex system. Common demand forecasting problems

1. *Forecasting demand too low*

3. Results

The resulting learning curves are presented in figure 6. There are two important results displayed in learning curves. Firstly, despite the complex data, the error decreases during the learning process, which indicates that the algorithm is being trained correctly. Secondly, no increase is observed in the error on the validation set, which means that the effect of overfitting was successfully avoided, and the model will work correctly on new data.

The resulting feature impotencies are presented in figure 7. Feature impotencies show how much, on average, the prediction changes if the feature value changes. The most important features are date, price, and discount, but days of the week and aggregated features also contribute to final accuracy. The values of feature impotencies play an important role in the selection of features and have been used for this purpose.

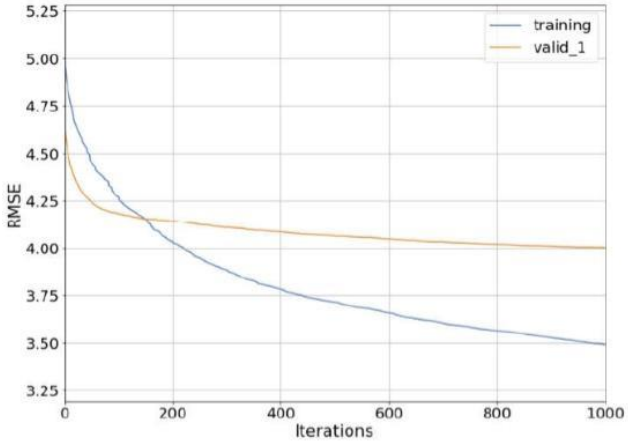


Figure 6. Learning curves.

The best accuracy achieved at testing data set is RMSE = 4.00129. There are no published articles with results on the same data set, so it is not possible to compare results directly. This result was achieved at hyper-parameters presented in table 3. This parameter can be used by other researchers to solve the same or similar problem. In order to clarify how this result was obtained optimization table fragment is shown in table 4.

Table 3. Resulting hyper-parameters of Gradient tree boosting algorithm.

Hyper-parameter	Value
Learning rate	0.051
Number of leaves	64
Minimum data in leaf	12
Number of iterations	243

2. *Forecasting demand too high*
3. *Supply chain dependencies causing last minute changes*
4. *Forecasting demand too early*
5. *Relying too rigidly on forecast demand*

Solutions to demand forecasting problems

1. *Forecast demand frequently and for short periods*
2. *Understand which steps of production you can stagger for flexibility*
3. *Have a plan for responding to expected variations*
4. *Have a plan for handling extreme variations and edge cases*

Product Demand Prediction (Case Study)

A product company plans to offer discounts on its product during the upcoming holiday season. The company wants to find the price at which its product can be a better deal compared to its competitors. For this task, the company provided a dataset of past changes in sales based on price changes. You need to train a model that can predict the demand for the product in the market with different price segments.

The **dataset** that we have for this task contains data about:

5. the product id;
6. store id;
7. total price at which product was sold;
8. base price at which product was sold;
9. Units sold (quantity demanded);

I hope you now understand what kind of problem statements you will get for the product demand prediction task. In the section below, I will walk you through predicting product demand with machine learning using Python.

Product Demand Prediction using Python

Let's start by importing the necessary Python libraries and the dataset we need for the task of product demand prediction:

```
import pandas as pd

import numpy as np

import plotly.express as px

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor


data = pd.read_csv("https://raw.githubusercontent.com/amankharwal/Website-data/master/demand.csv")
```

data.head()

ID	Store ID	Total Price	Base Price	Units Sold
0	1	8091	99.0375	111.8625	20
1	2	8091	99.0375	99.0375	28
2	3	8091	133.9500	133.9500	19
3	4	8091	133.9500	133.9500	44
4	5	8091	141.0750	141.0750	52

Now let's have a look at whether this dataset contains any null values or not:

1

```
data.isnull().sum()
```

```
ID            0
Store ID      0
Total Price   1
Base Price    0
Units Sold    0
dtype: int64
```

So the dataset has only one missing value in the Total Price column, I will remove that entire row for now:

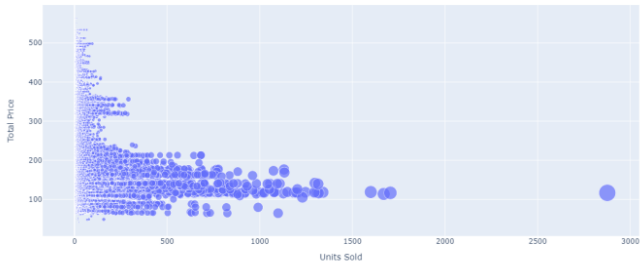
```
data = data.dropna()
```

Let us now analyze the relationship between the price and the demand for the product. Here I will use a **scatter plot** to see how the demand for the product varies with the price

Let us now analyze the relationship between the price and the demand for the product. Here I will use a **scatter plot** to see how the demand for the product varies with the price change:

```
fig = px.scatter(data, x="Units Sold", y="Total Price",
                 size='Units Sold')

fig.show()
```



Product Demand Prediction Model

Now let's move to the task of training a machine learning model to predict the demand for the product at different prices. I will choose the Total Price and the Base Price column as the features to train the model, and the Units Sold column as labels for the model:

```
x = data[["Total Price", "Base Price"]]

y = data["Units Sold"]
```

Now let's split the data into training and test sets and use the decision tree regression algorithm to train our model:

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y,
                                                test_size=0.2,
                                                random_state=42)
from sklearn.tree import DecisionTreeRegressor
model = DecisionTreeRegressor()
model.fit(xtrain, ytrain)
```

Now let's input the features (Total Price, Base Price) into the model and predict how much quantity can be demanded based on those values:

```
#features = [["Total Price", "Base Price"]]

features = np.array([[133.00, 140.00]])
```

```
model.predict(features)
```

```
array([27.])
```