Application of open source procedures solving data mining tasks

Master's Thesis

Submitted to the

Department of Computer Science and Languages at

Anhalt University of Applied Sciences

in fulfillment of the requirements for the degree of

Master of Science

Student: V. Boiarshinova

(Matr. Nr.: 4065060)

supervisors: Dr. B. Krause, L.Mylnikov

September 2017

Annotation

The master thesis is devoted to the consideration of various approaches to solving the data mining tasks. The work contains a theoretical description of the various methods and approaches, and a practical part that presages a Python program code, which shows how you can apply these methods to the example of a task from the data mining cap 2012.

The work is relevant nowadays. Since for many enterprises working in the field of sales, at the moment there are problems of processing and drawing knowledge from the data to obtain benefits.

The approaches, analysis of methods and developed software, which are considered in this paper, are also applicable to other different data mining tasks.

Agenda

1 Introduction 4

1.1 What is Data Mining 4

1.2 Comparison of tools for Data Mining 5

1.3 About Anaconda 8

2 How to use data frame Anaconda (Python) to solve a task of Data Mining. 9

2.1 The task for this thesis is 10

2.3 Something about time-series data 10

3 Data Preparation 13

3.1 Formalization of data 13

3.2 Types of errors in data 14

3.3 Presentation and minimum volumes of required data 16

3.4 Hypothesis creation 17

3.5 The application of the theory of data preparation in practice 18

Finding new features 21

4 Modelling 30

4.1 The application of the theory of modelling in practice 31

4.2 Preliminary preparation of data before building a model 32

4.3 Choosing a model for forecasting 33

4.3.1 Random Forest 34

4.3.2 Support vector machine 35

4.3.3 Logistic Regression 37

4.3.4 kNN (k-Nearest Neighbors) 38

5 Evaluation 41

5.1 Application of the ROC-curves method to evaluate selected models 42

5.2 Application of the K-fold cross-validation method to evaluate selected models 46

6 Results.. 49

7 References 51

1 Introduction

Nowadays, almost all sectors have large amounts of data, and very often there is a need to obtain knowledge from this data in order that later they brought benefit. What is clear is that manually processing the data is very costly and unprofitable. So, this is why, in this article I want to discuss the problem of solving the task of Date Mining.

1.1 What is Data Mining

The analysis technique using Data Mining mechanisms is based on various algorithms for extracting regularities from the initial data, the result of which are the models. There are a lot of such algorithms, but in spite of their abundance they can’t guarantee a quality solution. No most sophisticated method in itself will give a good result, because the question of the quality of the initial data becomes crucial[25].

These were the foundations of any Data Mining. There is a wide variety of models, they are all based on the above principles, as an application to this task, I decided to choose one. I decided to start from the CRISP-model, because this methodology is very popular among many organizations and enterprises. This model divides the whole process of analysis and data processing in six steps: business understanding, data understanding, data preparation, modeling, evaluation and deployment. All these stages are very closely connected with each other. After every of stages there is the possibility to return to the previous one, or several steps backwards to debug any originating errors or inaccuracies.

Here I want to describe in general (not only like in CRISP-model) how any process of Data Mining can be divided into several stages [1, 21, 22]:

1. **Understanding and formulating the task of analysis**.

At this first stage, the task is comprehended, the goals to be achieved by the methods of Data Mining are specified, the methods for evaluating the results of the study are determined. Incorrect choice of goals and methods will lead to distortion of the final results.

2. **Preparing data for automatic analysis.**

At this second stage, the data is formatted and, if necessary, edited, in order to apply certain methods of data mining: searching for errors and omissions, converting data to a single format, etc.

3. **Applying Data Mining methods and building models.**

At this third stage, the selected Data Mining methods are applied. There are many options for their application: from the use of one or two algorithms to complex combinations of a variety of methods, which allows you to perform a comprehensive analysis.

4. **Verification of the constructed models (Evaluation step).**

The fourth stage is testing of the received models. The following method is widely used: the entire data array being examined is divided into two unequal parts. Most of them (training) is the source material for building models by Data Mining methods, and the smaller one is a test group, on which the received models are checked. The criterion by which the model is evaluated is the difference in accuracy between the groups.

5. **Deployment of a model.**

At the fifth stage, the models are interpreted by man. This stage is no less important than the previous one, because an incorrect interpretation of the results obtained will negate all the work done previously. The final evaluation of models can be given only after the practical application of the results obtained.

1.2 Comparison of tools for Data Mining

Without powerful analytical tools, extracting useful knowledge hidden in huge amounts of raw, raw data is a difficult, almost impossible task. The best Data Mining software tools provide a variety of machine learning algorithms for modeling (such as Regression, Neural Network, Decision Tree, Bayesian

Networks, Support Vector Method, Random Forest, and so on) and those tools can solve some kind of problems like: Problems of forecasting, classification, clustering, grouping by kinship, link analysis, multivariate analysis etc. [26].

Here is the comparison (table 1) of most popular tools for Data Mining nowadays. All of those tools are free to download [2, 3, 22].

Table 1. The comparison of the most popular tools for Data Mining

	language	**benefits**	**license**	**functions**
Rapid Miner	Java	1.Without writing any code 2.More service than program	AGPL	pre-processing and data visualization, predictive analytics, statistical modeling. Support of training schemes, models and algorithms from WEKA, and scripts for R.
Weka	Java	1.Users can modify the program to fit their needs 2.It is used in various fields	GNU General Public License	Support of several standard tasks of data mining, including data preparation, predictive modeling, visualization, and feature selection.
R-language	Firstly: C and Fortran. Then: R	Scalability and ease of use	GNU General Public License	Using for developing statistical software and data analysis. Support of linear and nonlinear regression, classical statistical methods, time series analysis, and so on.
Orange	C++ and python	Visual programming + Python interface	GNU General Public License	Support of combination of visual programming, Python scripting. Also, there are components for machine learning, bioinformatics and text analysis.
Knime	Java, Eclipse	1. The best tool for data preparation. 2. Open source	GPL General Public License	The graphical user interface that allows design data processing. There are various components for machine learning and data

		software for data analysis and reporting.		mining, providing, together with the concept of building a pipeline of individual modules capabilities for business analytics and financial analysis of data.
IBM-modeler		1. There is the cloud-based version. 2. It can be deployed on the desktop, and integrate into existing systems. 3. Without writing any code	Usually Trial	Improve decisions and results. To help extract value from data, to simplify integration into existing systems
Python	Python	1. It can be embedded in other applications 2. Supports testing and debugging of individual modules and whole programs 3. The presence of multiple packages and modules	PSFL (GPL)	It can be embedded in other applications. Supports testing and debugging of individual modules and whole programs. The presence of multiple packages and modules (which greatly simplify the processing, analysis and forecasting of big amount of data

Since earlier I already had experience working with data mining tasks and using various programs for this, I can on my own say that one of the more popular and easier programs for working with data now is IBM SPSS Modeler. Its main advantage is that you do not need to write any program code, you need only the understanding of what needs to be done.

I want to examine here an alternative and not as popular solutions to the problem. Having considered all the proposed tools, there was the problem of choosing between the R language and Python. Both languages are very similar to each other and, it would seem that there is a problem of choice. But, the comparison was to find such an interesting framework as Anaconda. In the end, I decided to stay on it.

1.3 About Anaconda

This is the free distribution of python which includes many scientific and engineering libraries. The framework Anaconda includes several different tools, with which help it is possible to solve any task about Data Mining.

For this work were used two tools from the Anaconda: Python 3.6 and Anaconda.Fusion

1. Python is a high-level general-purpose programming language that has a higher performance and an easier code structure. Python has a minimal syntax. But, also, the standard Python library includes a large number of different functions [2]. Python supports several types of programming (structural, object-oriented, functional, and so on). The code in Python is organized into functions and classes that can be combined into modules (they in turn can be combined into packages) [3].
2. Anaconda Fusion integrates the Anaconda Platform with Microsoft Excel. This tool provides these core benefits [3, 15]:
    - connect MS Excel to Open Data Science;
    - leverage Data Scientists' notebooks easily through MS Excel
    - access all the power of Python and Big Data, natively embedded inside MS Excel

2 How to use data frame Anaconda (Python) to solve a task of Data Mining.

The solution of the problem data mining is always a process consisting of several stages of working with data. Actions to be performed on the data may be different, and there is big amount of the solutions to the problem, logically. But all the known trips are united by the fact that the main stages of the solution are the same [25].

When solving any task related to the processing of data, firstly, it is always necessary to understand what kind of data is, why we do actions on them and what we want to get as a result [26].

There is always such a stage as the preparation of the initial data, where the various operations are carried out with data that will facilitate the work with the data and will reduce the probability of obtaining a large error in the future.

Also, there is always such a stage as modeling. That is, creating a model for predicting the desired result for the data. The model should have a substantiation both from the mathematical side and from the implementation side. The model should be optimal, i.e. error in forecasting data should be minimal.

And there should always be a stage of evaluation and optimization of simulation results. As mentioned earlier, the error of the forecasting results should tend to zero, but, as everyone understands, this is possible only in an ideal situation. And in real conditions, of course, this will not happen. To do this, we need this stage to evaluate the model so competently and how to adjust something and improve it so that the result is more "truthful".

So, based on these stages, I want to give a theoretical and practical justification for solving the problem of the data mining task, on the example of the DMC 2012.

2.1 The task for this thesis is

The solving of a problem of any task is better to show on a concrete example. I would like to underline the attention for problems with time-series data. To not much to invent, I decided to stop on the task of data mining cup DMC_2012 (the first part). This is a task about forecast sales figures. I chose this task because of some reasons: the first one - there is real time-series data (570 products and changing its prices and the number of sales within 42 days). The second one – the task is not littered with big number of parameters and variables (it makes this example easy to understand). The third one - the data are a part of special interest for research, because there are no real product names, which leads to the next sequence of tasks (tasks not only according to the prediction data, but also tasks related to machine learning).

Also, in the process of working with the task, some additional features will be identified, which will be mentioned later, in the section on the date of preparation.

2.2 How the task looks like:

The prices and associated sales figures are stated for a specific period and for selected products. With the aid of this data a model is to be developed that describes the dependencies between the data. Based on this model, the sales figures of the subsequent period should be forecasted in an application phase [23].

2.3 Something about time-series data

Proceeding from the task, it is easy to understand that we will deal with the problem of forecasting, i.e. to extract knowledge from the available data in order to predict their magnitude in the future. In other words, the basis for forecasting will be the information stored in some database given (in this case in Excel) in the form of time series data.

The time series data is a sequence of observable values of a characteristic ordered at specific times. In the process of determining the structure and regularities of the time series, there is a high probability to detect: noise and emissions, seasonal or cyclic component, trend.

Noises and emissions are quantities that complicate the process of analyzing time series, but they are not basic. The main influence is borne by the trend and the seasonal component.

Trend - a function that is formed under the influence of general tendencies that affect the time series. And the seasonal component is a component of the series, which is repeated periodically. Very often, both trend and seasonality can be present in the time series simultaneously. An important task is to discover them and correctly compare them with the goal [24, 27].

Since this article will focus on predicting time series data, it is worth noting the fact that there are also situations where forecasting time series can be almost impossible because of some special behavior of that data. This happens if there is:

1. **Autocorrelation process.** One of the prerequisites for regression analysis is the independence of the random term in any observation from its values in all other observations [6]. Informally this means that the data of one observation does not affect the data of other observations. If this condition is not met, then it is said that the random term is subject to autocorrelation.
2. If we have in our data something like **the Random-walk process**. A random walk is a mathematical object, known as a stochastic or random process, that describes a path that consists of a succession of random steps

on some mathematical space such as the integers [7]. The probability of changing the value from one to the other is the same for all data.

3. **The autoregressive process** is a time series model in which the values of the time series at a given moment are linearly dependent on the previous values of the same series. Or there is still an **autoregressive distributed lags model**, a model of a time series in which the current values of a series depend both on the past values of this series, and on the current and past values of other time series.

3 Data Preparation

For first stage to solving the task of data mining is to prepare all data set to the modeling step. The data preparation phase aims to obtain the final dataset that will be used in the simulation of original heterogeneous and multi-format data. Data preparation tasks may be performed many times without any preassigned order. They include the selection of tables, records and attributes, as well as conversion and cleaning of data for modeling. Most often, the training data includes the following stages [25]:

- Selection of data
- Cleaning of data
- Combination of data
- Provide data in the desired format

3.1 Formalization of data

Next, you need to get ahead of the way data is represented by selecting one of the four types - number, string, date, logical variable. It is quite simple to define the way of representation, i.e. To formalize some data, for example, the sales volume in rubles is a certain number. But quite often there are situations when it is not clear how to present the factor. Most often, such problems arise with qualitative characteristics. For example, the volume of sales is affected by the quality of the product. Quality is a rather complex concept, but if this indicator is important, then you need to come up with a way to formalize it [27].

It is also necessary to estimate the cost of collecting the data needed for analysis. The fact is that some data is easily accessible, for example, they can be extracted from existing information systems. But there is information that is not easy to gather, for example, information about competitors. Therefore, it is necessary to evaluate the costs of data collection.

The collected data must be converted to a single format, for example, Excel, a text file with delimiters or any DBMS. The data must be unified, i.e. The same information should be described everywhere equally. Usually problems with unification arise when collecting information from heterogeneous sources. In this case, unification is a serious task, but its discussion is beyond the scope of this article.

3.2 Types of errors in data

There is also many different errors in the data (we are not talking now about the difference in the encoding) which we would like to talk about separately. There are a lot of such mistakes. There are also errors that are characteristic only for a subject area or task. But let's look at those that do not depend on the task [27, 28, 29]:

- Conflicting information;
- Omissions in the data;
- Abnormal values1;
- Noise;
- Errors in data entry.

To solve each of these problems there are well-tried methods. Of course, errors can be corrected manually, but with large amounts of data, this becomes quite problematic. Therefore, we consider the options for solving these problems in an automatic mode with minimal human participation.

Conflicting information – there are two main methods for dealing with this type of error:

- If several inconsistent entries are found, delete them. Sometimes this is quite enough.

- Correct the conflicting data. You can calculate the probability of occurrence of each of the inconsistent events and choose the most probable. This is the most competent and correct method of dealing with contradictions.

Omissions in the data – is very serious problem. Most forecasting methods assume that the data is received by a uniform constant flux. In practice, this is extremely rare. To combat this phenomenon, you can use:

- Approximation. If there is no data at any point, we take its neighborhood and calculate by the known formulas the value at this point, adding the corresponding record to the repository.

Abnormal values - quite often there are events that are strongly out of the scene. And it is best to adjust these values. This is because the means of forecasting do not know anything about the nature of the processes. Therefore, any anomaly will be perceived as a perfectly normal value. Because of this, the picture of the future will be greatly distorted. Some random failure or success will be considered a regularity. And the solution for it is:

- Robust estimates. These methods are resistant to strong perturbations. The available data are evaluated for everything that goes beyond the allowed limits, and either the value is deleted, or is replaced by the nearest boundary value.

Noise – doesn’t bring any useful information, but only prevents clearly to see the “picture”. There are several methods to solve this:

- Spectral analysis. With it, we can cut out the high-frequency components of the data. And, changing the width of the spectrum, you can choose what kind of noise we want to remove.

- Autoregressive methods. This common method is actively used in the analysis of time series and reduces to finding a function that describes the process plus noise. The noise after this can be removed and leave the main signal.

Errors in data entry - the number of types of such laws is too large, for example, typos, conscious data corruption, format mismatch, and it was not considered typical errors related to the features of data entry applications. To solve those problems there are well-tried methods. Some things are obvious, for example, before entering data, you can check the formats. Or, for example, you can correct typos based on a different kind of thesauri. But to simplify the task, it will be assumed that in our case we are dealing only with reliable data.

3.3 Presentation and minimum volumes of required data

For the analyzed processes of a different nature, the data must be prepared in a special way.

Since this paper focuses on time series data (the so-called "ordered data"), then the share will be made only on this part. But we need to understand perfectly well that there are other types of data in tasks related to Data Mining: disordered (tasks where the time factor does not matter), transaction data (i.e. several objects or actions grouped into a logically connected unit) and so on.

So, what is the ordered data? Such data are needed to solve forecasting problems, when it is necessary to determine how the process will behave in the future based on the available historical data. Most often, one of the facts is the date or time, although it is not necessary, it can also refer to certain counts, for example, data collected periodically from sensors [28].

For ordered data, one factor corresponds to each column, and events arranged in time are entered each row with a single interval between the lines. It is not allowed to have groupings, totals, etc., that is, a regular table is needed.

If the process is characterized by seasonality/cyclicality, it is necessary to have data for at least one complete season/cycle with the possibility of varying the intervals (weekly, monthly and so on), because cyclicity can be complex, for example, within an annual cycle of quarterly, and within a quarter of a week, then you need to have complete data for at least one longest cycle.

In general, the time for which reliable forecasts can be constructed is limited not only by the amount of data. Based on the assumption that the factors determining the development of the process will have an impact in the future about the same as at the current time. This assumption is not always true. For example, if the situation changes too quickly, new significant factors appear, and so on. This rule does not work. Therefore, depending on the task, the volume requirements can vary greatly. The use of too much data for analysis is just as inexpedient, because in this case, we will build a model from the old history, and therefore factors that may have lost their significance can be considered.

3.4 Hypothesis creation

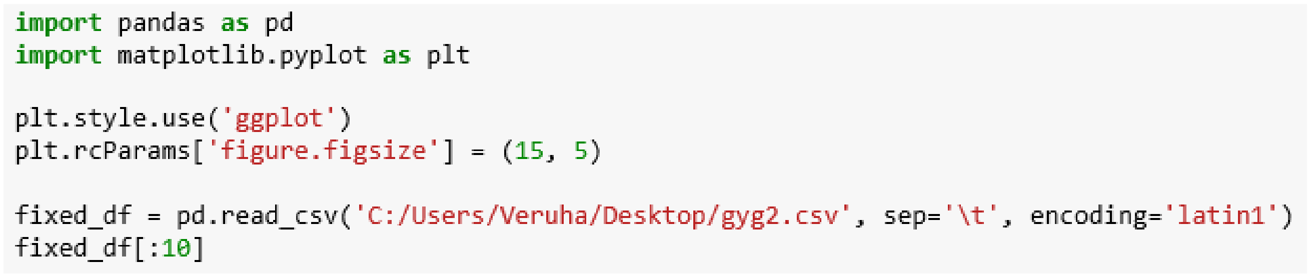
A hypothesis in this case will be assumed to be the assumption of the influence of certain factors on the problem we are investigating. The form of this dependence in this case does not matter, i.e. we can say that sales are affected by the deviation of our price for goods from the market average, but do not indicate how, in fact, this factor affects sales. To solve this problem, Data Mining is used. It is not possible to automate the process of hypothesizing, at least at the current level of technology development. This task should be solved by experts - experts in the field of knowledge. The result of this step should be a list with a description of all factors [29].

Factors can be of the following kind: season, discounts, day of the week, price change among competitors, brand, etc. In the process of selecting the influencing factors, it is necessary to abstract as much as possible from information systems and available data. After preparing the table with a description of the factors, it is necessary to assess expertly the significance of each of the factors. This assessment is not final, it will be the starting point. In the process of analysis, it may well turn out that the factor that the experts considered to be very important is not in fact so, and, on the contrary, the factor, insignificant from their point of view, can have a significant impact.

3.5 The application of the theory of data preparation in practice

Now we apply this stage to our specific task.

First, you need to see what the whole of our data represented (i.e. upload them to Python and see what happens). To implement this all is quite simple, you just need to get on the next command (fig. 1):

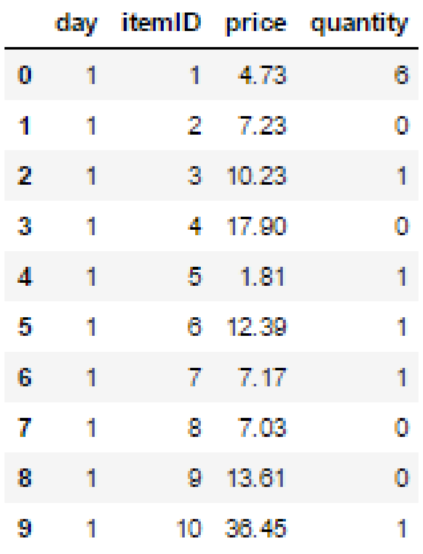


```
import pandas as pd
import matplotlib.pyplot as plt

plt.style.use('ggplot')
plt.rcParams['figure.figsize'] = (15, 5)

fixed_df = pd.read_csv('C:/Users/Veruha/Desktop/gyg2.csv', sep='\t', encoding='latin1')
fixed_df[:10]
```

Figure 1 – How to upload csv-file in Python



	day	itemID	price	quantity
0	1	1	4.73	6
1	1	2	7.23	0
2	1	3	10.23	1
3	1	4	17.90	0
4	1	5	1.81	1
5	1	6	12.39	1
6	1	7	7.17	1
7	1	8	7.03	0
8	1	9	13.61	0
9	1	10	36.45	1

Figure 2 – How the uploading data look like

In order to not clutter up the work and as an example only the first ten records are given. The result you can see on the Figure 2.

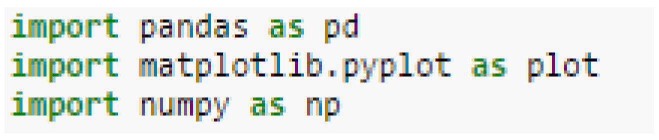
As we see, we have four columns (parameters) for which we can evaluate our data: day, product id, product price on that day and how many pieces of a particular product were sold on a particular day (quantity).

It is necessary to understand that when working with data in Python special packages and libraries are needed. At this stage, three basic ones are used: Numpy, Matplotlib, and Pandas (fig. 3). Why do we need each of them?

Numpy is a Python library that adds support for large multidimensional arrays and matrices, along with a large library of high-level (and very fast) mathematical functions for operations with these arrays.

Matplotib is a two-dimensional graphics library for the python programming language with which you can create high-quality drawings of various formats. Matplotlib is a package module for python.

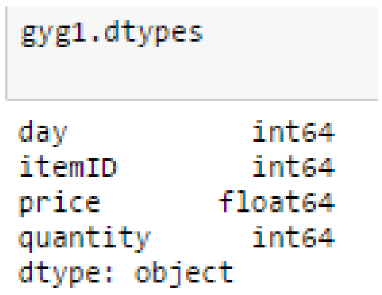
Pandas - this package makes Python a powerful tool for data analysis. The package provides the ability to build summary tables, perform grouping, provides easy access to tabular data, and if you have a package Matplotlib allows you to draw graphics on the resulting data sets.



```
import pandas as pd
import matplotlib.pyplot as plot
import numpy as np
```

Figure 3 – The useful packages for the firsts steps of the work with Python for Data Preparation.

Further it is very interesting to get acquainted with what kind of data are the variables in this table, for which you need to do the following step (fig. 4):

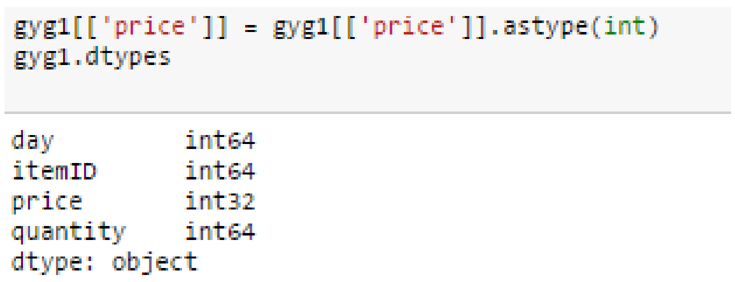


```
gyg1.dtypes
```

```
day            int64
itemID         int64
price        float64
quantity       int64
dtype: object
```

Figure 4 – Definition of data type for every variable of data set.

As we can see, one of the variables (namely price), has a different type of float64. In the beginning it is not essential, but in consequence of the decision of a problem it can lead to difficulties, so it is best to give all variables the same data type. To change the type of the variable is necessary to do (fig. 5):



```
gyg1[['price']] = gyg1[['price']].astype(int)
gyg1.dtypes
```

```
day         int64
itemID      int64
price       int32
quantity    int64
dtype: object
```

Figure 5 – Changing of data type for variable "Price" (from float to integer)

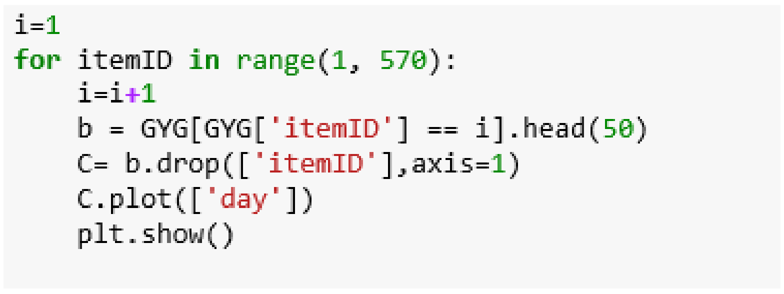
After the data is converted and have a common format, it would be better to see in a graphic form what is happening inside the entire table.

Very often situations arise when the data is not complete, which greatly hinders the solution of the problem, affects the accuracy. All this leads to a procedure like "data recovery." But first you need to check all the data for empty lines or cells.

In this task, we are very lucky, because the data is presented in its entirety, which partially reduces the work on data preparation. But, if it was not so, the ways of solving this problem will be presented below.

Finding new features

Since we are dealing with a time series date, it would be nice to find some dependencies both among days (holidays, weekends, work days, sales), and among the "popularity" of products. For a common vision, write the code that will give us the price chart, the quantity of goods sold per day for one product. In order not to write this code for all products separately, we use the "for" cycle which one you can see on the Figure 6.



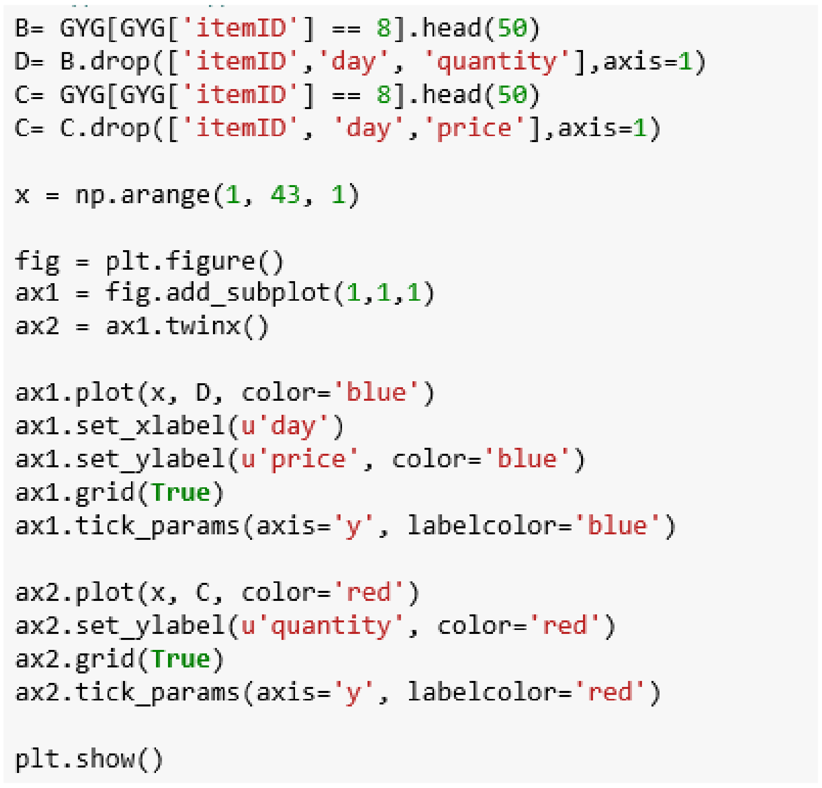
```
i=1
for itemID in range(1, 570):
    i=i+1
    b = GYG[GYG['itemID'] == i].head(50)
    C= b.drop(['itemID'],axis=1)
    C.plot(['day'])
    plt.show()
```

Figure 6 – the program code to get a graph for price and quantity of each product

Such an implementation of the code is only suitable for comparing such parameters, where the values of the values are approximately in the same range. In this case, we want to see the relationship between the price and the quantity of the product sold over time (that is, the spread of the values of the "quantity" and "price" parameters may be large).

To solve the problem of visibility and scale, it was decided to use an additional scale, which will be placed on the right. And then on the left scale it will be possible to determine the values for the price, and on the right scale for the quantity. And in this case, you can clearly see the relationship between them. To do this, you need to change a bit in the code which is given above. I will show these changes using the example of the product 8 (Fig. 7).

As you can see from the code, the main part you need is to divide the data into two samples: in the first one there will be only price changes for a specific product for 42 days, and in the second one - changes in the quantity of sales of this product during the same 42 days. Well, accordingly, it is done to ensure that one scale is responsible only for the parameter "price", and the second for "quantity".



```
B= GYG[GYG['itemID'] == 8].head(50)
D= B.drop(['itemID','day', 'quantity'],axis=1)
C= GYG[GYG['itemID'] == 8].head(50)
C= C.drop(['itemID', 'day','price'],axis=1)

x = np.arange(1, 43, 1)

fig = plt.figure()
ax1 = fig.add_subplot(1,1,1)
ax2 = ax1.twinx()

ax1.plot(x, D, color='blue')
ax1.set_xlabel(u'day')
ax1.set_ylabel(u'price', color='blue')
ax1.grid(True)
ax1.tick_params(axis='y', labelcolor='blue')

ax2.plot(x, C, color='red')
ax2.set_ylabel(u'quantity', color='red')
ax2.grid(True)
ax2.tick_params(axis='y', labelcolor='red')

plt.show()
```

Figure 7 - Adding a second scale for better visualization of data

As it is obvious, on output 570 different schedules have turned out. It is necessary to consider them to understand what is happening there. After studying, it was found that four types of graphs are often repeated. Based on all this, you can try to make assumptions and extract information from the resulting picture.

It is necessary to talk separately about each of the four graphs because each of them deserves a separate consideration and statement of assumptions.

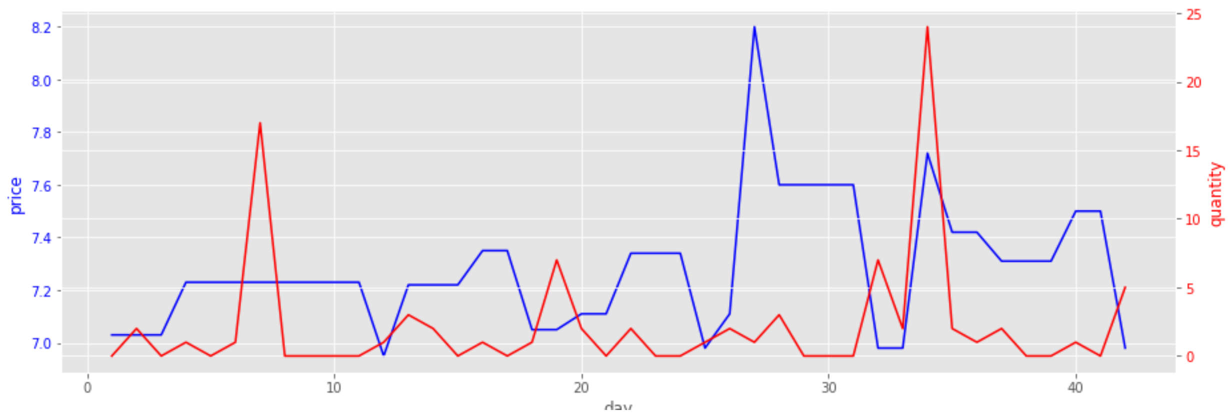


Figure 8 – The first type of graph “weekend”, on example of product 8

The first type of graph, which one you can see on the Figure 8, looks like this. It clearly expresses three "leaps" of sales of a certain product. So far, without making specific conclusions, it can be assumed that these horse racing occurred on some special days of the week (probably the weekend). Because there is no strong dependency between price and quantity (we can’t suppose that there were sales or something like that)

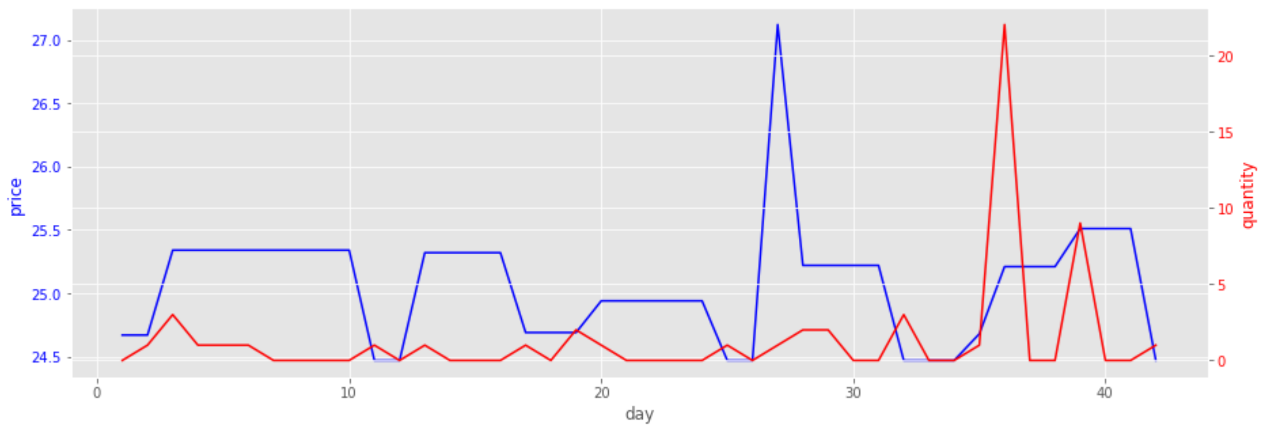


Figure 9 - The second type of graph “before the big celebration”, on example of product 11

The second type of the graph, which one you can see on the Figure 9, represents stable sales throughout most of the period, but at the end two sharp leaps in terms of product purchasing are visible. Also, without confirmation, you can try to assume that this may be “before the big celebration”.

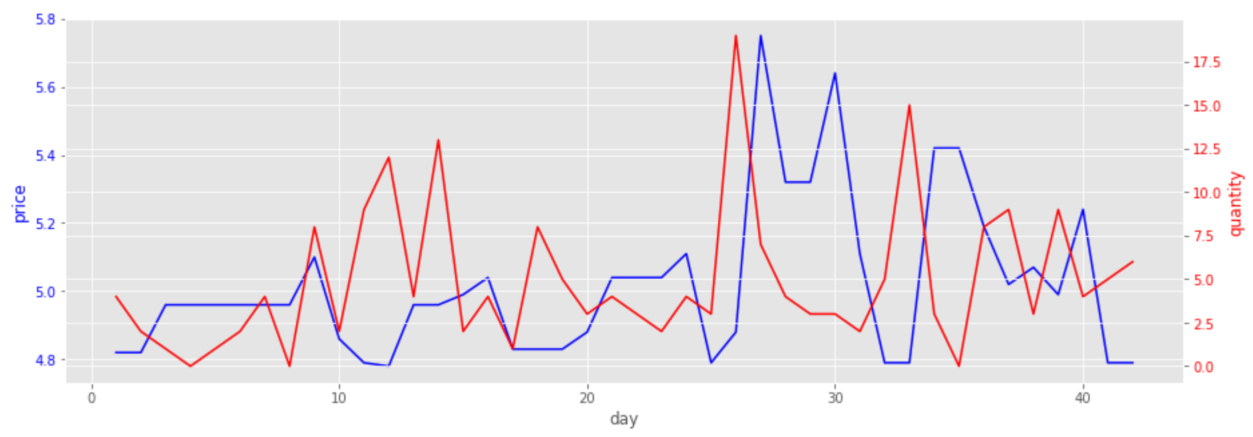


Figure 10 - The third type of graph “prime necessity”, on example of product 100

The third type of graph represents this picture (fig 10). The product has good sales almost every day and the quantity of sales very strong depends on price of this product (The lower the price - the more sales, and vice versa). So, you can conclude that this can be a product of prime necessity.

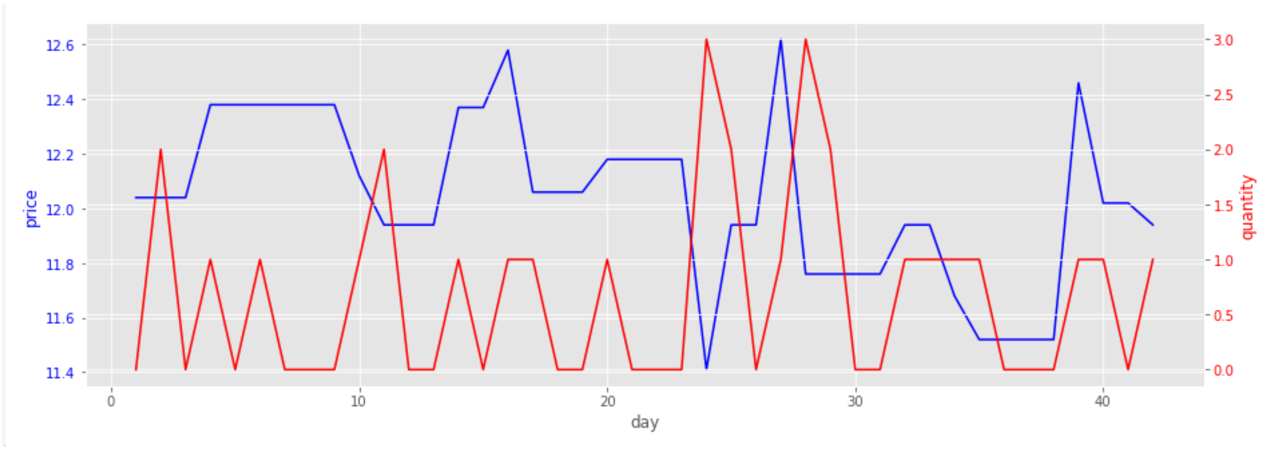


Figure 11 - The second type of graph “unpopular product”, on example of product 12

And the last type of graph, which one you can clearly see on the Figure 11, frequent-type of graphs. It represents very rare sales and in a small amount, regardless of the time. We can assume that this is some unpopular product.

We must now understand how to pull out specific information from the resulting picture. Here you can use a bunch of different methods and consider the dependencies of the parameters among themselves. But first I would like to understand the products themselves.

Also, I would like to mention here that any time series data has its typical behavior. There is always a certain dependence, from which one can make an assumption about the days of the week, or about any time / period of time, and so on. For a quality forecast, it is always very important not to miss the details. At this stage, I did not focus much on this section, it was not my main goal. The task was to show how and in what direction it is necessary to work.

For each online-store there are products that bring the greatest profit and are "basic". There are less profitable, but also relatively often purchased products and there are products that practically all the time lie in their places and have almost no relation to the profit of the store. In this task, I would also like to identify the most significant and most insignificant products. Since, in the future, these data will also help forecasting.

Therefore, a little more work with the four selected types of graphs. Based on their definition (which was given above), I will assign them figures according to the degree of significance. These additional data will be added to the main ones for their next analysis. Thus, I assign such types of products like "weekend" and "prime necessity" to number 2 (since they are significant, always with sales and carry the main profit). I assign type of product "before a big celebration" to number 1, because with small sales throughout the whole month entails a very large sale at the end of the month. And, finally, the last type of product "unpopular product" will be assigned as 0, since the product is not a top-seller and does not bear a big profit from its sales.

To solve this problem of identification of products, it was decided to use the ABC analysis methodology and the tool from the framework Anaconda - Anaconda Fusion

I would like to clarify why it was decided to use the ABC analysis for this task. The main reason is to remove the data, which is least informative for this task. And the next reason is to identify products that are the most valuable and which, conversely, are not the main source of profit.

From the theoretical part is known that [4]: ABC-analysis - analysis of commodity stocks by dividing into three categories:

- A - the most valuable, 20% - assortment; 80% of sales
- B - intermediate, 30% - assortment; 15% - sales
- C - least valuable, 50% - assortment; 5% - sales

In order to not complicate the work, all the necessary preparation of the data was carried out with the use of MS Excel. It should also be added that to the original Excel file, four types of features, which were previously defined, for each product were added (i.e. depending on the type of price/quantity graph for each product).

So, then this preparation for ABC-analysis consisted of the following stages:

1. To begin with, it is necessary to determine the profit that was received every day from each product (all this is done by multiplying the quantity of the sold goods by its price, and so for each product)
2. The basis of ABC-analysis is the principle of sorting, so this will be the next step (sorting products in order of decreasing profits from their sales)
3. Next, you need to determine and calculate the contribution of each element to the common total.

Further, it would be possible to calculate purely mathematically which commodity belongs to which group. But firstly, we are dealing with a large amount of data, and secondly it is not always clear where there is a boundary between groups A and B and between groups B and C. To solve this problem, it was decided to take advantage of the acquired knowledge from the course Machine Learning.

Thus, it was decided to use the method of clustering the prepared data. As methods of clustering, the following were selected: KMeans, agglomerative clustering (Ward) and random sampling. The number of clusters, logically, was assumed to be equal to 3. Because of this clustering, the following results for Ward were obtained on the Figure 12:

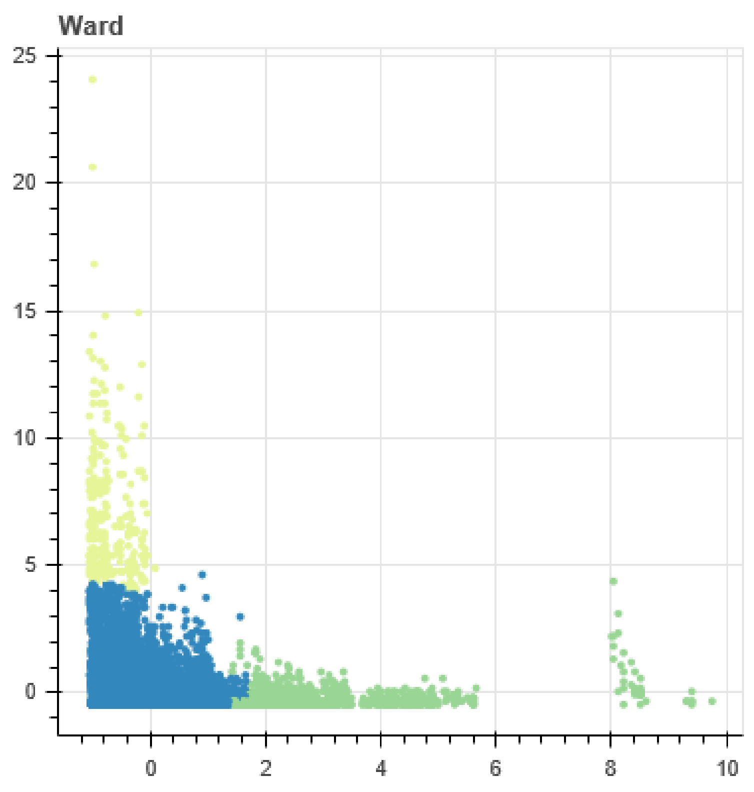


Figure 12 – The results of ABC-analysis for Ward (agglomerative clustering), by AnacondaFusion.

Here there is an important thing that I would like to mention - the parameters of each of the methods that was used for this data analysis. In Anaconda Fusion, the only parameter that you need to configure for each method is the number of clusters. All other internal parameters of methods can not be configured because all this is written in the program code. There is a standard set of parameters for each method that is used. If there is a desire to work with the setting parameter, then for this Anaconda Fusion is not suitable, and it's easier to do it directly in

Jupyter in Python language. I will do the same work in the second and third parts of this article when I will work with the modeling and evaluation of the results.

In the process of working with Random Sampling and with KMeans clustering - the results were identical, so I will give here only results for one of them in a graphical form (fig. 13).

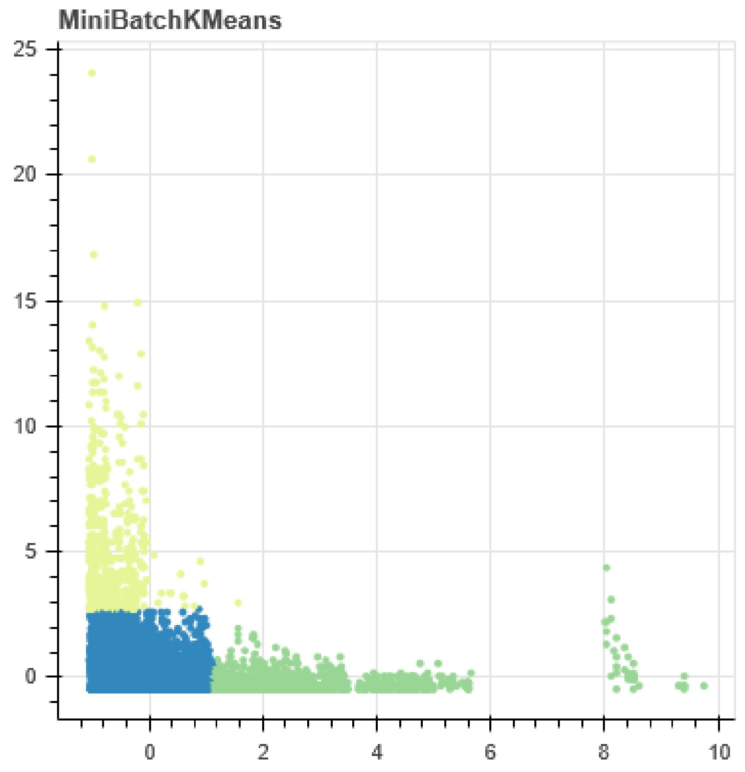


Figure 13 - The results of ABC-analysis for Random Sampling and KMeans (on example KMeans), by AnacondaFusion.

When comparing the resulting graphs between the collection, several facts immediately become apparent, which will help in the future when working with data. First, a small number of points on the right on both graphs can be easily discarded (because they are far away from the main mass of data and can be considered non-informative). And secondly, the fact that KMeans algorithm chooses more strict boundaries between classes, in contrast to agglomerate clustering, catches the eye.

To solve the problem with dropping points, it is enough just to specify the number of clusters equal to 3, and 4 to AnacondaFusion. And as a result of clustering, we get exactly those points on the right, in a separate cluster, marked with a separate digit in the Excel file, which we will simply exclude from our sample in the future, as a kind of noise.

One of the criteria in comparing results and choosing the right method for me was a separate mathematical calculation, which I performed myself with the help of the same Excel. And then I compare my results with the results of clustering in Anaconda Fusion. In fact, even according to the schedule it can be seen that Ward has more blurred boundaries between classes, which is more like the truth. And results of the Ward-clustering are almost completely coincided with my mathematical results.

As a result, as a necessary method for determining the importance of the product, I decided to use Ward (agglomerate clustering), since this method produced a more accurate result, compared to the ABC analysis done manually using Excel.

4 Modelling

There are a lot of mechanisms for constructing models, but each of them has its limitations and solves a certain class of problems, so in practice, it is often possible to succeed by combining the methods of analysis. At the same time, the earlier the attempt is made to test the obtained model in practice – is the better way to see if the model is correct, because there are no other ways to really assess its quality.

In general, the following recommendations can be given, independent of the processing algorithm [21]:

1. Pay great attention to data cleaning. By collecting the data in the right amount, you can’t be sure that they will be of good quality. More often the quality of the data leaves much to be desired, so you need to pre-process them. To do this, there are many methods: removing noise, smoothing, editing anomalies and so on.
2. Combine the methods of analysis. This allows you to look at the problem more broadly. Moreover, the use of different methods to solve the same problem can lead to valuable ideas;
3. Do not pursue absolute precision and start using when the first acceptable results were given. Anyway, it's impossible to get an ideal result. If we got the result, even if it's not perfect, but better than it was before, that is the reason to start using it. First, it allows you to get a quick return. Secondly, only in practice it is possible to really evaluate the result obtained. Thirdly, it is possible and necessary to work in parallel on improving the model considering the results obtained in practice;
4. If it is not possible to obtain acceptable results, you should return to the previous steps of the scheme. Unfortunately, mistakes can be made at any step: the initial hypothesis may be incorrectly formulated, problems may arise with the collection of necessary data, and so on. To this you need to

be ready. If such problems arise, return to the previous paragraphs and consider alternative solutions;

4.1 The application of the theory of modelling in practice

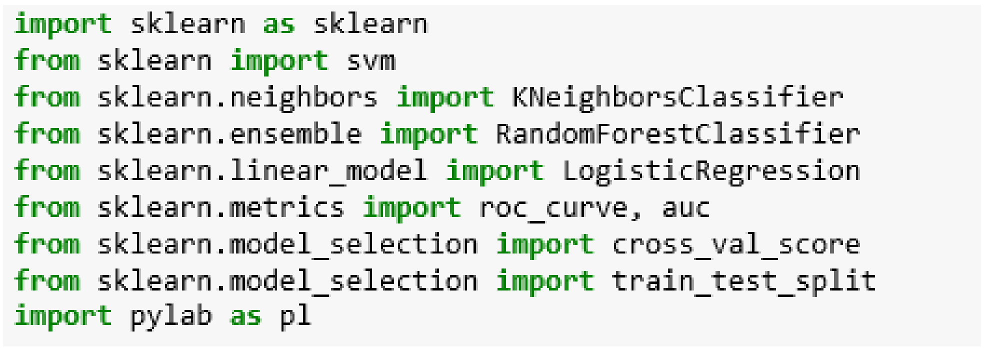
Well, the data is processed and you can start building the model, but first you need to determine how we will check the accuracy of the model. For this test, we will use cross-validation and ROC-curves. We will perform the check on the training sample, after which we apply it to the test sample.

So, let's consider several algorithms of machine learning:

- Support vector machine
- K Nearest Neighborhood Method
- Random Forest
- Logistic regression

For this we will use the next useful package from Python – Scikit-Learn (in program language “sklearn”). *Sklearn* is a Python module integrating a wide range of state-of-the-art machine learning algorithms for medium-scale supervised and unsupervised problems. This package focuses on bringing machine learning to non-specialists using a general-purpose high-level language. [9, 14, 21]

First, we import all the methods we need from this package (fig.14):



```
import sklearn as sklearn
from sklearn import svm
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import roc_curve, auc
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import train_test_split
import pylab as pl
```

Figure 14 – The needed packages from Scikit-Learn for Modeling and Evaluation parts

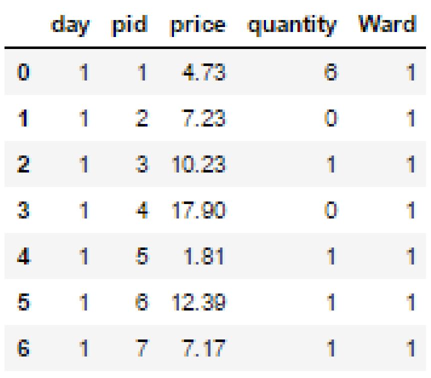
As can be seen from the picture, for modeling (classification) 4 types of models have been chosen: kNN (k-Nearest Neighbors), Random Forest Classifier, Support Vector Machine and Logistic Regression. A little digression to explain: in fact, there are a lot of model variants, and if you have such a resource as time, then you can try all the options and stop at the one with a high probability of the forecast. Therefore, in order not to obstruct the work with all existing methods, it was decided to stop at these four. It is necessary to understand that the principle of construction is the same for all, because all models are already collected in the package “sklearn”. The main difference is the configurable parameters of the model [14]. Therefore, you need to have a full understanding for what cases this model will work, and for which - not. Therefore, a detailed consideration of all functions of the selected models will be given below.

4.2 Preliminary preparation of data before building a model

Since the modeling will be carried out on the basis of the ABC-classification given in the Date Preparation section, it is first necessary to add new information to the source data, which product belongs to which class. This is very important for further stages of forecasting, which would be relevant to the chosen task, because it is always very important to understand which of the products are top-sellers, and which are not, because each seller is always interested in making more profit.

So, in the end, the New dataset consists of five columns:

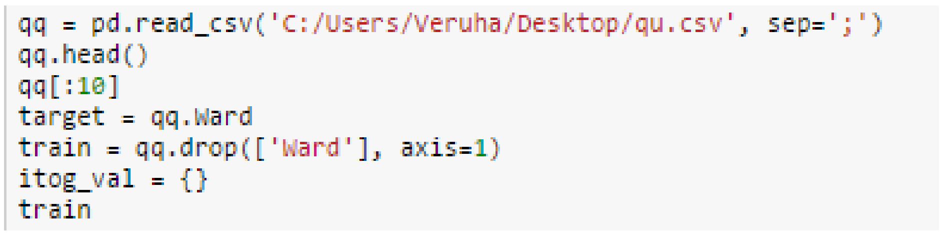
- day;
- product ID;
- price;
- quantity;
- Ward – ABC-significance (there are three variants of groups, for the simplicity of the program code of the designated digits, not letters (actually, the code can also contain letters, just implementation is more difficult) 0 – for the group A, 2 – for the group B and 1 - for the group C).



	day	pid	price	quantity	Ward
0	1	1	4.73	6	1
1	1	2	7.23	0	1
2	1	3	10.23	1	1
3	1	4	17.90	0	1
4	1	5	1.81	1	1
5	1	6	12.39	1	1
6	1	7	7.17	1	1

Figure 15 – How the new dataset looks like

Our training sample will look like this (fig. 15). To build a model, we need to break it down into a sample indicator, which we will investigate (in this example, we will consider ABC-meaning, i.e. to which class the product belongs) and the attributes that determine this indicator.



```
qq = pd.read_csv('C:/Users/Veruha/Desktop/qu.csv', sep=';')
qq.head()
qq[:10]
target = qq.Ward
train = qq.drop(['Ward'], axis=1)
itog_val = {}
train
```

Figure 16 – Splitting an existing sample into an indicator and attributes

As can be seen from the figure 16, two samples (two arrays) were created. The first one is a target, an array of parameters, and the second is a train - array of values. From the array “train”, the column Ward (ABC-significance), which we plan to forecast was deleted.

4.3 Choosing a model for forecasting

This section is completely devoted to the work with the *sklearn* package, so for all the description and all the work with it is mainly based on information taken from the source [14]. The *sklearn* package offers a huge number of models for both classification and clustering. If there is a great desire to get a good forecast result, and there is a time resource, then you can check almost all the models for accuracy. In this case, there is no big quality race, since the main purpose of this work is to show how you can work with Python to predict time series data. In this case, I decided to compare four variants of models for data classification.

4.3.1 Random Forest

Random Forest is a composition of many decisive trees, which reduces the problem of retraining and improves accuracy in comparison with one tree. The forecast is obtained as a result of aggregating the responses of a set of trees. The training of trees takes place independently of each other (on different subsets), which not only solves the problem of constructing identical trees on the same data set, but also makes this algorithm very convenient for application in distributed computing systems.

The construction of the model is already laid out in the module *sklearn*, so the appearance of the program code looks quite simple.

But the main thing is an understanding of how this algorithm works in order to adjust the parameters of the model in such a way that they give the best result.

By default, the "insides" of this model look like [14, 18, 19]:

sklearn.ensemble.RandomForestRegressor *(n_estimators=10, criterion='mse', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_split=1e-07, bootstrap=True, oob_score=False, n_jobs=1, random_state=None, verbose=0, warm_start=False)*

Now you can see individually what and which impact each of the parameters has on the customizable model.

n_estimators – The greater the number of trees provides the higher the quality of training. With an increase in the number of trees, it is even possible to obtain 100% accuracy in the training sample, but on the test sample the quality goes to the asymptote.

criterion - Splitting criterion. In the *sklearn* library for regression, two criteria are implemented: "*mse*" and "*mae*", correspond to the error functions they minimize ("*mse*" for the mean squared error and "*mae*" for the mean absolute error). For classification, the criteria "*gini*" and "*entropy*" are implemented, which correspond to the classical splitting criteria: *Gini* and *entropy*.

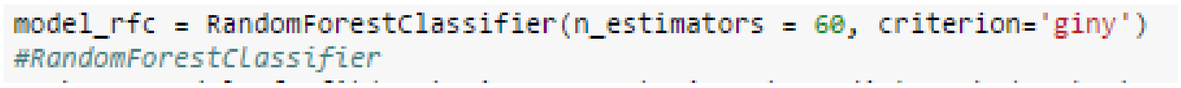
max_features - Number of characteristics for the choice of splitting. As the max_features increases, the time for building a forest increases, and the trees become "more monotonous". By default, it is sqrt (n) in classification tasks and n / 3 in regression problems. This is the most important parameter.

max_depth – Maximum depth of trees. With increasing depth, the quality of learning sharply increases, but, as a rule, it increases as well. It is recommended to use the maximum depth (unless there are too many objects and very deep trees that take a long time to build). When using shallow trees, changing the parameters associated with limiting the number of objects in the sheet and for dividing does not result in a significant effect. Shallow trees are recommended for use in tasks with a lot of noise.

min_samples_split и min_samples_leaf – The minimum number of objects and leaves at which splitting will be performed. It is recommended to leave the values as defaulted. When the parameter is increased, the quality on learning drops, and the Random Forest's build time is reduced.

All other parameters are initially optimally tuned, therefore it is not recommended to change them. The only thing you can do with the parameter n_jobs is to increase the computation speed, because this parameter is responsible for the number of processors on which the calculation is performed.

For the selected task, I settled on setting the following parameters: Number of trees and setting criteria for classification (fig. 17) [14]:



```
model_rfc = RandomForestClassifier(n_estimators = 60, criterion='giny')
#RandomForestClassifier
```

Figure 17 - program code for building a model for Random Forest Classifier

4.3.2 Support vector machine

This method solves the problems of classification and regression by constructing a nonlinear plane that separates solutions.

Due to the peculiarities of the nature of the feature space in which the decision boundaries are built, the support vector machine method has a high degree of flexibility in solving problems of classification of various levels of complexity [16].

By default, the content of the model in the sklearn packet looks like this: [14]

sklearn.svm.SVC *(C = 1.0, kernel = 'rbf', degree = 3, gamma = 'auto', coef0 = 0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=1, decision_function_shape=None, random_state=None)*

Important parameters that need to be adjusted from the point of view in question are *C, kernel*, and *gamma*.

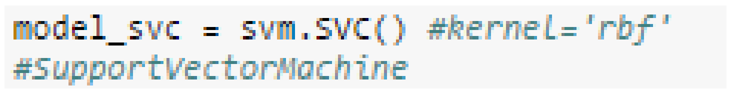
C - the regularization parameter. It also controls the relationship between the smooth boundary and the correct classification of the points in question. The smaller the value of this parameter provides the stronger regularization.

Kernel – that parameter is responsible for the hyperplane, at this point you need to choose which type of kernel will be used from the options:

- linear - on the output we will have a linear division of classes. This separation will be convenient to use, when it is initially known about the clarity of the boundaries between classes.
- rbf (radial basis function) - today is the most popular type of kernel. This is mainly due to its localized and final responses over the entire spectrum of the real axis x.
- poly (for nonlinear)
- sigmoid

gamma — width of *RBF* kernel. If *gamma* = 0, the linear case is considered. It is involved in fitting the model and may be the reason for retraining.

When you configure this type of model in Python, there is one difficulty - the process itself builds and generates the model takes a very large amount of time. And if you are limited in this resource, then this method is better to be omitted, because when choosing the optimal parameters for this type of model, a large amount of time will be spent. As a result, all parameters of this model were left for default (fig. 18):



```
model_svc = svm.SVC() #kernel='rbf'
#SupportVectorMachine
```

Figure 18 - program code for building a model for Support Vector Machine

4.3.3 Logistic Regression

The method of constructing a linear classifier, which makes it possible to evaluate the posteriori probabilities of belonging to classes [20]. The advantage of this algorithm is that at the output for each object we have the probability of belonging to the class.

sklearn.linear_model.LogisticRegression*(penalty='l2', dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='liblinear', max_iter=100, multi_class='ovr', verbose=0, warm_start=False, n_jobs=1)*

Parameters that you should pay special attention to when configuring this type of model: *penalty*, *solverc*, *tol* and *C* [14].

Penalty – Penalization of logistic regression parameters. There are two options for such penalization *L1* and *L2* (default). The difference between these parameters is in the metric, for *L2* we mean the Euclidean metric, and for *L1* the distance of the city quarters

Solver – The algorithm that is used to solve the optimization problem. There are 4 options for choosing the algorithm: *'newton-cg'*, *'lbfgs'*, *'liblinear*', 'sag'. For small amounts of data - *'liblinear'*, *'sag'* - is faster on large data, while the data must be on the same scale and only *'newton-cg'* and *'lbfgs'* are used for multi-class tasks.

Dual – This parameter determines that a direct or dual optimization problem is chosen. Dual wording is supported only with *liblinear*. It's better to use dual = False when *n_samples*> *n_features*

C – The regularization parameter, plays an analogous role, as in the case of Support Vector Machine.

Tol - Tolerance for stopping criteria, changing the value of this parameter has a significant impact on the quality of the model.

When receiving a logistic regression model that will produce high quality models, the following parameters were selected and set (fid. 19):

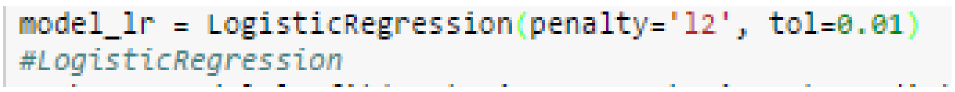

```
model_lr = LogisticRegression(penalty='l2', tol=0.01)
#LogisticRegression
```

Figure 19 - program code for building a model for Logistic Regression

4.3.4 kNN (k-Nearest Neighbors)

The method of k-Nearest Neighbors is a method based on the use of memory and, unlike other statistical methods, does not need preliminary training (i.e. does not fit the models).

The work of the method is based on the intuitive assumption that close-lying objects most likely belong to one category. Thus, forecasts are compiled on the basis of a set of prototype samples that predict new (ie not yet observed) values, using the principle of "victory by majority vote" for classification.

This is the problem of assigning an object to one of the predetermined classes on the basis of its formalized characteristics. Each of the objects in this problem is represented as a vector in an N-dimensional space, each measurement in which is a description of one of the features of the object [14, 17].

sklearn.neighbors.KNeighborsClassifier *(n_neighbors = 5, weights = 'uniform', algorithm = 'auto', leaf_size = 30, p = 2, metric = 'minkowski', metric_params = None, n_jobs = 1)*

The main parameters that you should pay attention to when setting up are:

n_neighbors – Determines the number of nearest neighbors. For *n = 1*, the nearest neighbor algorithm is unstable to noise emissions: it gives erroneous classifications not only on the emission objects themselves, but also on the objects of other classes nearest them. For *n = m*, on the contrary, the algorithm is excessively stable and degenerates into a constant. Thus, the extreme values of *n* are undesirable.

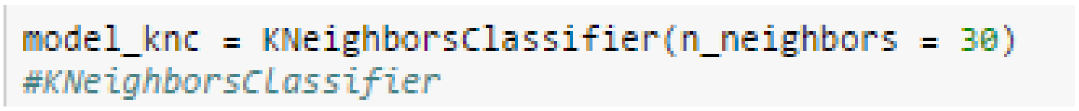
Weights – What weights for neighbors can be chosen. There are three choices: *uniform* - when all weights for neighbors are the same; *Distance* - when the weight of neighbors will depend on their distance, i.e. the closer it sucks to the requested point, the greater its weight (importance) and vice versa. And also, the user can determine the function necessary for its task, which will return weights, taking into account the array of distances.

Algorithm – The choice of the algorithm for calculating nearest neighbors. There are 4 types of algorithms in the *sklearn*: *ball_tree*, *kd_tree*, *brute* and *auto*. Where the first two represent a hierarchical ordering of objects through a system of simple nested figures (balls and rectangles, respectively). Brute - brute-force computation of distances between all pairs of points in the dataset and auto - when the algorithm itself decides which method will give the most desired result.

metric - This is the most difficult of all problems. If objects are described by numerical vectors, the Euclidean metric is often taken. It should be remembered that all the signs must be measured "on the same scale", and, best of all, are correlated. Otherwise, the sign with the largest numeric values will dominate the metric, the remaining attributes, in fact, will not be taken into account.

If there are too many attributes, and the distance is calculated as the sum of deviations by individual characteristics, then the curse of dimension arises. Setting the metric depends on the parameter P. Most often, p = 1 and p = 2 are used. If P = 1, then this will be considered the metric "distance of urban quarters" and if P = 2 (default value), then the Euclidean metric will be used.

When configuring the parameters in this example, the conclusion was made to stop only on setting the number of neighbors (which it was decided to set to 30), and leave all other parameters in default. Therefore, the program code for the K-nearest neighbor method is as follows (fig. 20):



```
model_knc = KNeighborsClassifier(n_neighbors = 30)
#KNeighborsClassifier
```

Figure 20 - program code for building a model for K-nearest neighbor

On the choice and construction of models, the entire solution of the problem is not limited. The most important thing remains ahead - an estimation of accuracy of the constructed models. This is done in order to further select the model, the predictions of which were with the highest accuracy, in order to obtain the highest accuracy for predicting the test data, respectively.

Therefore, after this stage, we need to move on to the next one – Evaluatioin.

5 Evaluation

To evaluate the adequacy of the results obtained, it is necessary to involve experts in the subject area. Interpretation of the model, as well as the promotion of hypotheses, can and should be done by an expert, because it requires a deeper understanding of the process, which goes beyond the data analyzed. In addition, we need to use formal methods for assessing the quality of the model: to test the constructed models on different samples to assess their generalizing abilities, i.e. it should be the ability which will be able to produce acceptable results on data that was not provided to the system when building the model [21].

Some analysis mechanisms can "remember" the data presented to it and show excellent results on them, but at the same time completely lose the ability to generalize and on the test data (from previously unknown data) to give very poor results. With a formal assessment, one can start from the idea that if the model gives acceptable results on the test data, then it has the right to live [22].

When you get acceptable results, you need to start using the resulting models. The beginning of the application is not the completion of the Data Mining project. Work on improving models is always necessary, because After the passage of time, there will necessarily come a time when again the described cycle will have to pass. In addition, after receiving the first satisfactory results, the question usually arises of increasing the accuracy.

It is also necessary to periodically assess the adequacy of the current situation model, because even the most successful model ceases to correspond to reality in due course.

Since there are no unnecessary conversions with data, we will break our data into two subsamples: the training sample and the test sample. This is done to calculate ROC-curves in the future. For cross-validation there is already a built-in function in this case.

Now some theoretical knowledge about ROC-curves and Cross-validation.

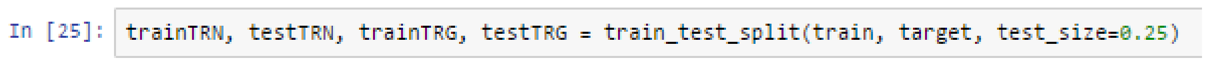
Cross-validation (CV) is the procedure for empirically evaluating the generalizing ability of algorithms that are trained by use cases. A certain set of partitions of the initial sample is fixed into two subsamples: the teaching and the control subsamples. For each partition, the algorithm for the training subsample is tuned, then its mean error is estimated at the objects of the control subsample. The estimation of the sliding control is the mean for all partitions of the error value in the control subsamples [10].

Roc-curves - a graph that allows to evaluate the quality of a binary classification reflects the ratio between the proportion of objects from the total number of characteristic carriers that are correctly classified as bearing a characteristic and the proportion of objects from the total number of objects not carrying a characteristic erroneously classified as bearing a characteristic when the threshold of the decision rule is varied [11].

5.1 Application of the ROC-curves method to evaluate selected models

I decided to concentrate my attention on these two methods of estimating the accuracy of models, because it's very good to have something that you can fight with. I.e. always need to have several different methods of estimating accuracy, just like in the situation with the choice of model. And there can be much more methods for estimating accuracy.

Let's start with the Roc-curves. For this method, you need to add a splitting of the metrics we received (the train array) into two sub-sets: test and training. This is also done by writing program code (Fig. 21):



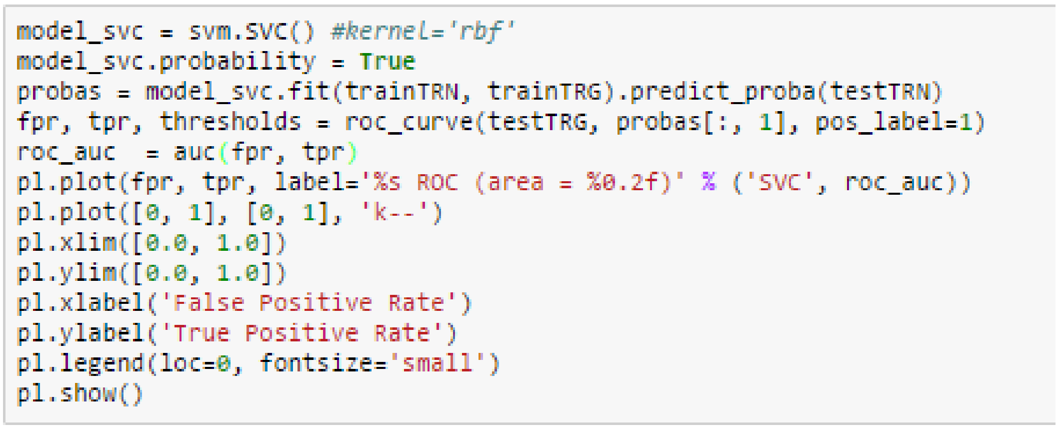
```
In [25]: trainTRN, testTRN, trainTRG, testTRG = train_test_split(train, target, test_size=0.25)
```

Figure 21 – the program code for the creation of arrays for ROC-curves

At the output of the function four arrays of parameters will be received:two of them - a new training array of parameters and a new array of indicators; and the other two are test array of indicators and test array of parameters.
Inside this function, the following parameters are configured: an array of parameters (target), an array of values of indicators (train), and there is a ratio (test-size) - set in which our sample will be divided (in this case we have six weeks, and therefore we want to take the 1/6 ratio, so that one week is for test and the rest weeks are for training).

And further it is already possible to apply this technique of an estimation of accuracy to all four above-stated models.

So, we will start with an example for Support Vector Machine. The program code in Python for ROC-curves is presented on the Figure 22. And how the estimation curve looks like you can see on the Figure 23.



```
model_svc = svm.SVC() #kernel='rbf'
model_svc.probability = True
probas = model_svc.fit(trainTRN, trainTRG).predict_proba(testTRN)
fpr, tpr, thresholds = roc_curve(testTRG, probas[:, 1], pos_label=1)
roc_auc  = auc(fpr, tpr)
pl.plot(fpr, tpr, label='%s ROC (area = %0.2f)' % ('SVC', roc_auc))
pl.plot([0, 1], [0, 1], 'k--')
pl.xlim([0.0, 1.0])
pl.ylim([0.0, 1.0])
pl.xlabel('False Positive Rate')
pl.ylabel('True Positive Rate')
pl.legend(loc=0, fontsize='small')
pl.show()
```

Figure 22 – The program code for ROC-curves in Python on example of Support Vector Machine

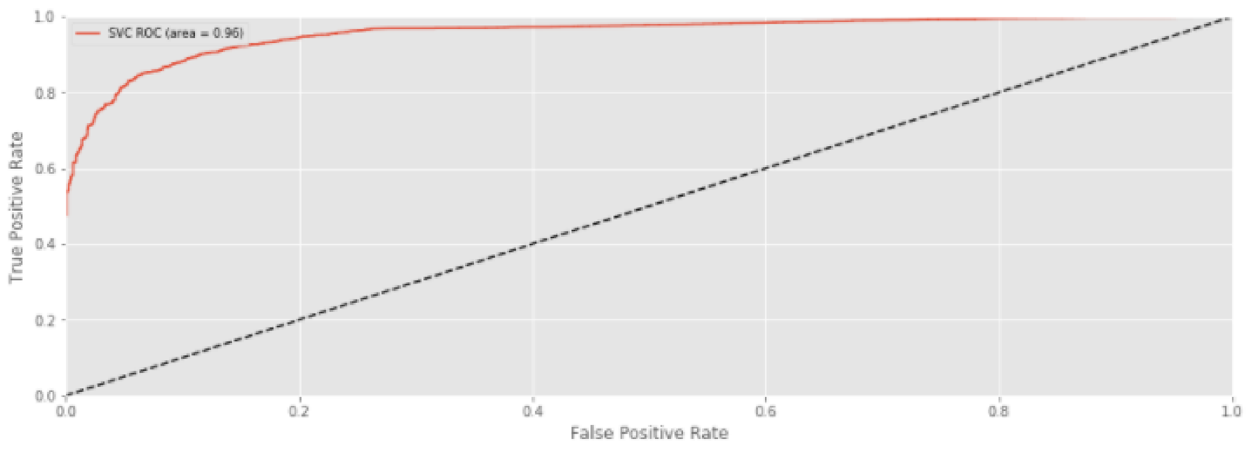


Figure 23 – The result of ROC-curve estimation for Support Vector Machine model

Since the evaluation of all models via ROK-curves occurs in a similar way, the program code for each of them looks the same except for the first two lines of code (but, actually, those lines were shown in the Modelling part of this article). Those, to show here the code for the rest of the models does not make sense. Therefore, the graphs of the results obtained for Random Forest (fig.24), Logistic Regression (Fig. 25) and K-Nearest Neighbor (Fig. 26) are shown below.

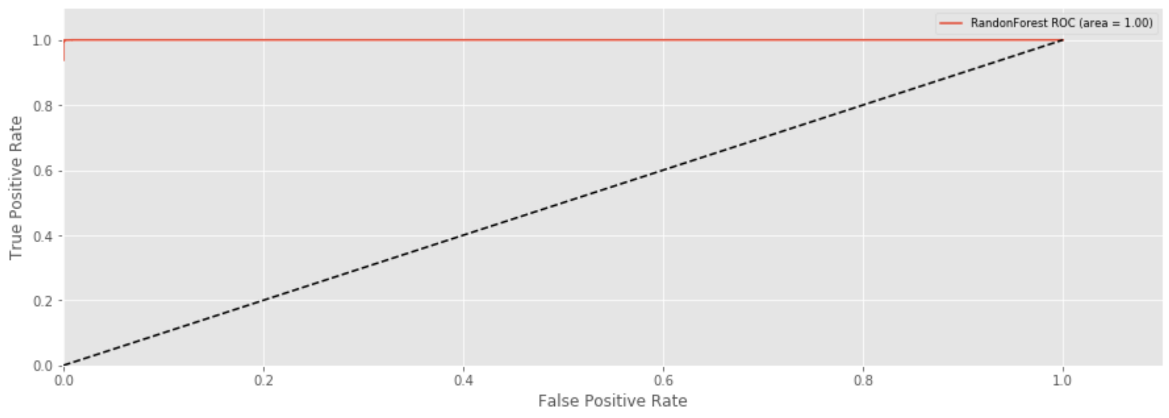


Figure 24 – The result of ROC-curve estimation for Random Forest model

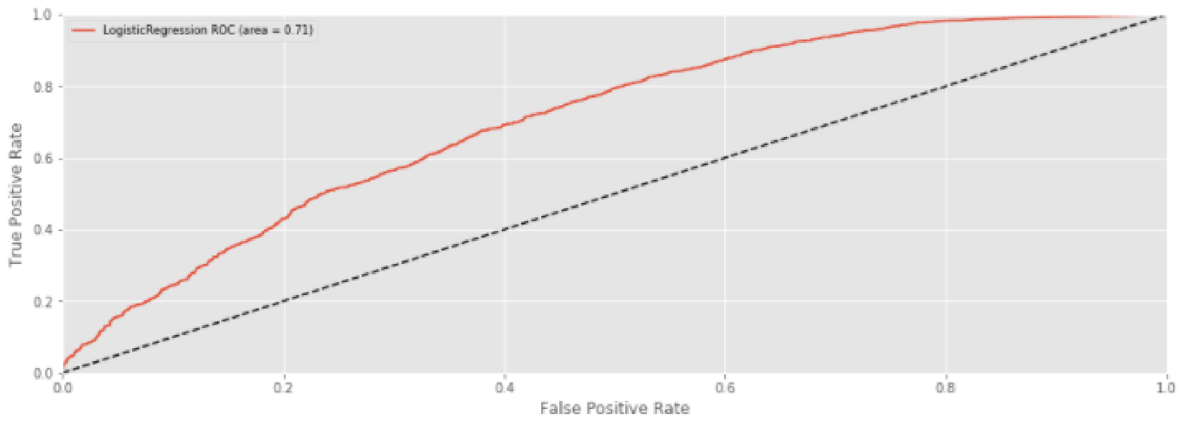


Figure 25 – The result of ROC-curve estimation for Logistic Regression model

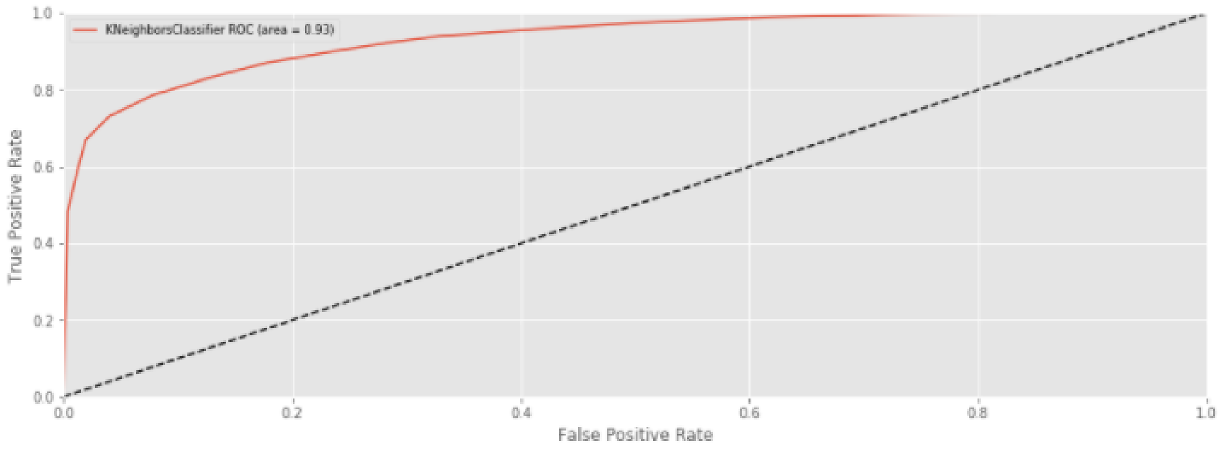


Figure 26 – The result of ROC-curve estimation for K-Nearest Neighbor (KNN) model

Based on the results of the ROC-curves, one can see with what accuracy each of the methods makes predictions. The following probabilities were obtained:

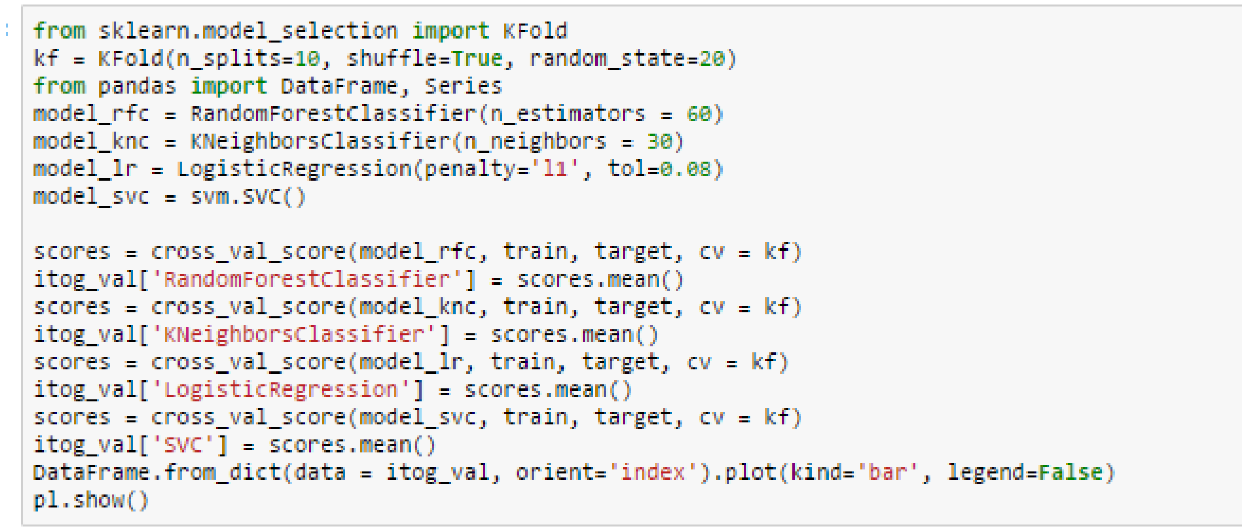
1. Support Vector Machine – 96%
2. Random Forest Classifier - 100%
3. Logistical Regression – 73%
4. K-Nearest Neighbor (KNN) – 93%

Obviously, the Random Forest method showed itself best (prediction with a probability of 100%). And in general, each of the methods produced a very high probability of prediction. Why this happened so - will be explained below.

5.2 Application of the K-fold cross-validation method to evaluate selected models

In this case, the original data set is divided into *K* blocks of the same size. Of the *K* blocks, one is left to test the model, and the remaining *K-1* blocks are used as a training set. The process is repeated *K* times, and each of the blocks is used once as a test set. *K* results are obtained, one for each block, they are averaged or combined in some other way, and give one estimate [12]. The advantage of this method before the random subsampling is that all observations are used for both training and testing of the model, and each observation is used for testing exactly once. Cross-validation is often used on 10 blocks, but there are no definite recommendations for choosing the number of blocks [13].

To present the results of cross-validation was decided in the form of a bar chart for all models at once, for better visibility (fig. 28). The code also for all models is shown below on the Figure 27.



```
from sklearn.model_selection import KFold
kf = KFold(n_splits=10, shuffle=True, random_state=20)
from pandas import DataFrame, Series
model_rfc = RandomForestClassifier(n_estimators = 60)
model_knc = KNeighborsClassifier(n_neighbors = 30)
model_lr = LogisticRegression(penalty='l1', tol=0.08)
model_svc = svm.SVC()

scores = cross_val_score(model_rfc, train, target, cv = kf)
itog_val['RandomForestClassifier'] = scores.mean()
scores = cross_val_score(model_knc, train, target, cv = kf)
itog_val['KNeighborsClassifier'] = scores.mean()
scores = cross_val_score(model_lr, train, target, cv = kf)
itog_val['LogisticRegression'] = scores.mean()
scores = cross_val_score(model_svc, train, target, cv = kf)
itog_val['SVC'] = scores.mean()
DataFrame.from_dict(data = itog_val, orient='index').plot(kind='bar', legend=False)
pl.show()
```

Figure 27 - The program code for K-fold cross-validation in Python

As a result of this code execution, the following graphic result was obtained for cross validation:

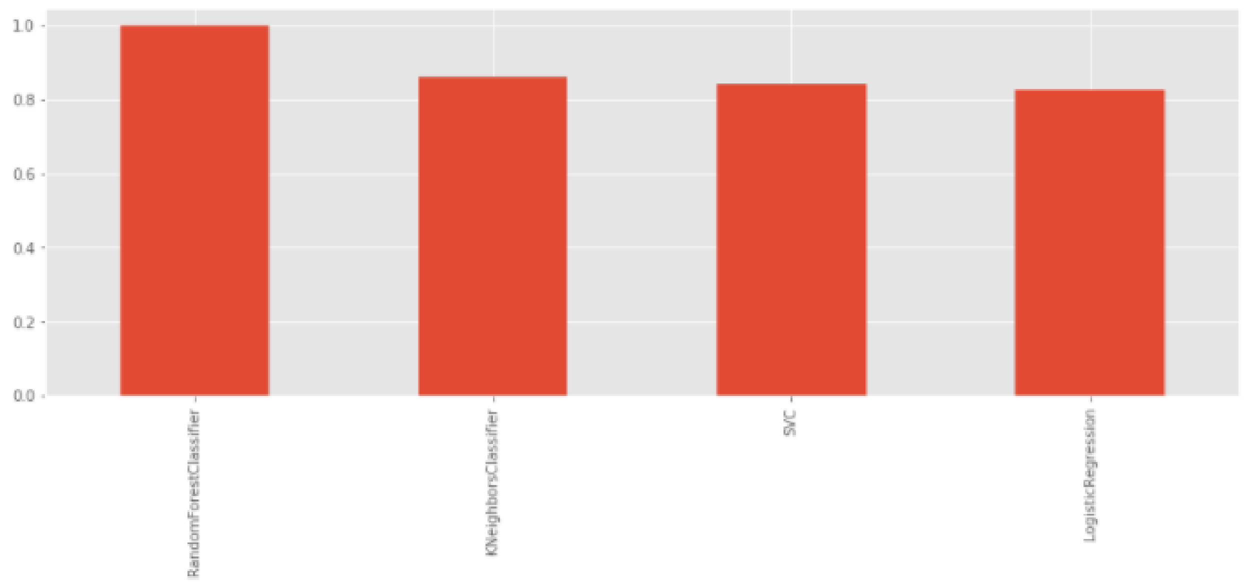


Figure 28 - The result of K-fold cross-validation estimation for all explored models

Based on the results of the K-fold cross-validation, one can see with what accuracy each of the methods makes predictions. The following probabilities were obtained:

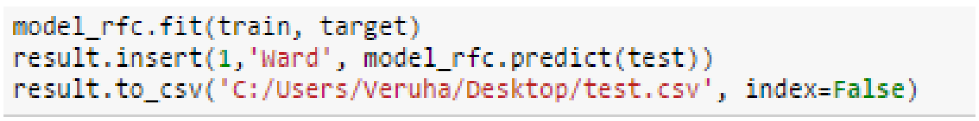
1. Support Vector Machine – 84%
2. Random Forest Classifier - 100%
3. Logistical Regression – 82%
4. K-Nearest Neighbor (KNN) – 88%

And again, as can be seen from the graph, again with a 100% probability according to the prediction, the Random Forest Classifier wins.

We need to make a reference to why the methods give such a high probability. Obviously, in a real situation, this can’t be. The main and main reason is that all actions for solving this problem were carried out on the same set of data: in the case of Data Preparation, division into classes A, B and C from the set of data of the trains was made. And then on the same set, there was a learning of the models and further evaluation of their accuracy. In reality, these actions are unacceptable (but to solve this problem, it was not possible to use other data).

Also need to make a small reference, why it was the Random Forest Classifier that won both times with the highest prediction accuracy. This classifier works very well if all its parameters (mentioned above) are optimally tuned. Well, also in the problem under consideration, division was performed in only three classes (this is a fairly small number of classes), thanks to which the high accuracy was obtained.

Based on a comparison of the two methods for estimating models, it can be concluded that it is best to make a prediction that the data of the test sample belong to three classes (A, B or C), preferably using the Random Forest Classifier. The result of course will also be written in csv format and the code for this action will look like this (Figure 29):



```
model_rfc.fit(train, target)
result.insert(1,'Ward', model_rfc.predict(test))
result.to_csv('C:/Users/Veruha/Desktop/test.csv', index=False)
```

Figure 29 - The program code for prediction and recording the results of classification in csv-file for test-data in Python

So, after getting this result of classification for our test data, you can continue work with the discovering new knowledge in the data, because the main goal for the task 2012 is to predict the quantity for each product. We just predicted here the class of significance to which the product would belong. This was done in order to rely on this prediction, to facilitate the work with another prediction (to increase its accuracy).

6 Results

In this article, an example of the solution of the Data Mining problem was shown and analyzed using the dataframe Anaconda.

As a basis, the task was chosen from DMS2012, in the solution of which all stages of the Data Mining were disassembled. It should also be noted that this task was chosen because it involves dealing with time series. Therefore, the description and main features of time series are given in the article.

In the stage of the Data Preparation, additional features were identified for the existing data set. And an ABC-analysis was conducted, which resulted in the division of the general data set into three classes of significance: A - the most significant product, B - a product of medium importance and C - a product of lower significance.

At the modeling stage, 4 variants of models were chosen as an example for comparison (although in practice there are a lot more of them): Support Vector Machine, Random Forest, K-Nearest Neighbors and Logical Regression. The parameters of each of the selected methods were considered and the codes were given - how the construction of the model looks in the Python language.

At the stage of the evaluation of the results, two methods were chosen to estimate the models obtained (although there may also be much more). These methods are: ROC-curves and cross-validation. Based on the result of both methods, with an accuracy of 100 percent, the Random Forest method has won. This result does not have a place to be in reality, and in this situation it can be explained by the fact that the classification both times was made on the same data (ABC-analysis in the Date Preparation part and the evaluation of models in the Evolution part).

Ideally, you can still work and work with this task. You can return to the stage of data preparation and try to identify new features, as well as at the modeling stage you can work with other types of models, with the adjustment of their parameters. The same thing can continue to explore in the evaluation step. It is important to note that the result of this work is not final, because there was only a forecast for the product belonging to one of the three classes (another reason why Random Forest gave a high result). And that work needs to be done to solve the problem can still continue and continue, but this was not the main task of this article. The task was to show how you can work with time series using Python.

The whole programming code is given on the CD. The data which were used for the task is given on the official site of Data Mining Cup.

7 References

[1] C.P.I.J. van Hinsbergen1, A. Hegyi1 J.W.C. van Lint1 H.J. van Zuylen. "Bayesian neural networks for the prediction of stochastic travel times in urban networks". Research Article for IET Intelligent Transport Systems. 2011. P. 1-10.

[2] Python. URL: <https://ru.wikipedia.org/wiki/Python>, 30.05.2017

[3] Anaconda documentation. URL: < https://docs.continuum.io/anaconda/ > 30.05.2017

[4] Joffrey Collignon, Joannes Vermorel. ABC-analysis (inventory). February 2012. URL: < https://www.lokad.com/abc-analysis-(inventory)-definition>, 02.06.2017

[5] Дмитрий Езепов. ABC-анализ в Excel. URL: <http://statanaliz.info/metody/gruppirovka/30-abc-analiz-v-excel>, 14.07.2016

[6] Random walk. URL: <https://en.wikipedia.org/wiki/Random_walk>

[7] Проблема автокорреляции.URL:< http://studopedia.ru/4_78215_problema-avtokorrelyatsii.html>, 31.05.2017

[8] Луньков АД., Харламов А.В. Интеллектуальный анализ данных. 2009 р. 17-34.

[9] Scikit-learn: Python модуль для машинного обучения. URL: < http://mathlingvo.ru/2015/02/scikit-learn-python/>, 07.06.2017

[10] Cross-validation (statistics). URL: <https://en.wikipedia.org/wiki/Cross-validation_(statistics)>, 15.07.2017

[11] Receiver operating characteristic. URL: <https://en.wikipedia.org/wiki/Receiver_operating_characteristic>, 29.06.2017

[12] Cross-validation and bootstrap. SLDM III © Hastie & Tibshirani - February 25, 2009

[13] Cross-validation: evaluating estimator performance. Scikit learn. URL: <http://scikit-learn.org/stable/modules/cross_validation.html>, 29.06.2017

[14] SCIKIT LEARN. 2017. URL: < http://scikit-learn.org/stable/index.html >, 15.07.2017

[15] AnacondaFusion documentation. URL: <https://docs.continuum.io/anaconda/fusion/>, 05.05.2017

[16] Jason Brownlee. Support Vector Machines for Machine Learning. 2016. URL: < http://machinelearningmastery.com/support-vector-machines-for-machine-learning/ >, 25.06.2017

[17] Tavish Srivastava. Introduction to k-nearest neighbors: Simplified., 2014. URL: < https://www.analyticsvidhya.com/blog/2014/10/introduction-k-neighbours-algorithm-clustering/ >, 25.06.2017

[18] Leo Breiman and Adele Cutler. Random Forests. 2013. URL: < https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm >, 25.06.2017

[19] Kalinin Alexandr. Random Forest: прогулки по зимнему лесу. 2017. URL: < https://habrahabr.ru/post/320726/ >, 13.07.2017

[20] Logistic regression. MADCALC, 2017. URL: < https://www.medcalc.org/manual/logistic_regression.php >, 03.07.2017

[21] Yuri Borodin, Процесс Data Mining. Построение и использование модели. 2016. URL: < http://www.intuit.ru/studies/courses/6/6/lecture/196 >, 25.06.2017

[22] Chubukova Irina. Data Mining. 2015. URL: < http://portal.tpu.ru:7777/SHARED/a/AAPONOMAREV/metod/Tab/lections-data_mining.pdf >, 27.06.2017

[23] The DMC task 2012. URL: < http://www.data-mining-cup.de/rueckblick/rueckblick/article/dmc-2012.html >, 06.05.2017

[24] Time series data. Wiki, 2017. URL: < https://en.wikipedia.org/wiki/Time_series >, 06.06.2017

[25] Ian H. Witten, Elbe Frank. Data Mining: Practical Machine Learning Tools and Techniques. - «Morgan Kaufmann» 2005. P. 10-56.

[26] Н. Паклин, В. Орешков. Бизнес-аналитика: от данных к знаниям. - «Питер» 2009. P. 46-59

[27] James D. Hamilton. Time Series Analysis. - Princeton university press, 1994. P.820.

[28] Walter Enders. Applied economic time series. - «Wiley», 2nd-publ., 2004. P. 480

[29] Ruey S. Tsay. Analysis of financial time series. - Wiley, 1st-publ., 2005. P. 640