DATA MINING AND PREPARATION

MSc INFORMATION SYSTEMS AND SERVICES

SPECIALIZATION: BIG DATA AND ANALYTICS

«DATA MINING FROM COMMERCE RELATED DATA»

PANAGIOTAKOPOULOS GEORGIOS (ME2030)

<u>SUPERVISOR</u>

PHILIPPAKIS MICHAEL

Abstract

This work consists of four datasets in one main part. In the experimental part, we undertake to apply knowledge mining techniques to four data sets that are directly or indirectly related to the commerce sector. In Chapter 1, we look at a data set that contains data on Facebook commercial pages. Next, in Chapter 2, we study a data set that contains data from electronic transactions. Then, in Chapter 3, we look at a set of data that has to do with users browsing the internet and making purchases. In Chapter 4, we study a set of data concerning the annual costs of wholesale customers for specific product categories. Both the Weka knowledge mining tool and code developed in the Python programming language were used to process the datasets. For each data set we list the steps and techniques we applied in our experiment, and we present our conclusions.

Table of contents

Figure Index

Tables

Introduction

Nowadays, data are everywhere and generated faster than ever. By analyzing data from various sources, using machine learning and knowledge mining techniques, we are able to draw useful conclusions about the nature of the data, the strategies to follow and the decisions to be made. In this way, through the analysis of data, the execution of machine learning algorithms and the interpretation of results, and the extraction of knowledge from data, we can significantly reduce the cost of our activity, provide better products and services and be more competitive.

A necessary condition in order to ensure better results in the extraction of knowledge from the data, is to apply the appropriate pre-processing techniques, in order to transform the data into a format more compatible and comprehensible than the respective machine learning model, to locate and manage with appropriate incomplete or incorrect values, but also to reduce the volume of data, thus helping the algorithms to provide better performance in the results.

In this project, we undertake to study four data sets.

Having the specific data from Dataset 1 and Dataset 2 related to Facebook live sales in Thailand and Online Retail respectively, we try to look for the best ranking method. Specifically we try to compare different methods through the Weka tool and the Python programming language in order to find the best method by changing most of the parameters such as k in K-Nearest Neighbor or Folds in cross fold-validation or train . Weka is a machine learning toolbox that supports many machine learning activities and can be easily expanded or adapted with new classifiers, clusters, feature selection methods, data visualizations, and more. Therefore, in this way we managed to find the best method that will be able to better classify other data by categorizing them by type of sale, in terms of Dataset 1 and by categorizing them by country, in terms of Dataset 2. At the same time many Papers use other or the same (In other parameters) methods. So we try to compare these methods to find the results of the best classification for the specific data.

Datasets 3 and 4 are related to online shopping and wholesale. Specifically, we study a data set (dataset 3) that relates to internet users' browsing data and their correlation with online shopping, and a data set (dataset 4) that relates to wholesale customer costs in various product categories. After processing the data set properly, we execute machine learning algorithms in order to extract knowledge from them..

Specifically, for dataset 3 we use the WEKA knowledge mining tool to pre-process and then run the Naïve Bayes, J48, Random Forest, and Logistic Regression sorting algorithms and compare the results.

For dataset 4, we use the Python programming language to do a proper pre-processing which includes dimensionality reduction with the PCA method and then run the K-means clustering algorithm.

Part 2

1. Dataset 1: Facebook live sellers

The first dataset to be edited is entitled "Facebook live sellers in Thailand" and contains data from commercial Facebook pages of Thai fashion and cosmetics companies. The data includes, among others, the type of publications (video, photos, statuses, and links). The dataset also includes metrics such as comments, shares, and reactions. The data set includes 7,051 data lines.

1.1. Preprocessing

Pre-processing of data consists of removing from the data set four columns, which did not contain data (Column1,..., Column4) (Picture: Chart). WEKA software completes this step by selecting these four columns and clicking the Remove button.

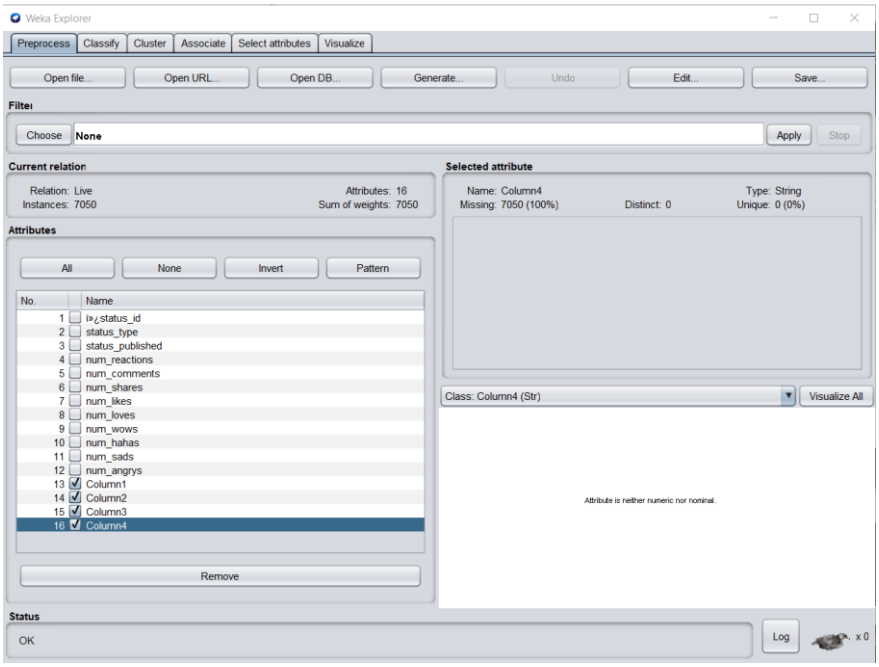

Figure 1. Pre-processing data set with WEKA.

Remove empty columns.

1.2. Classification

Since the size of the dataset is relatively small, the following classification algorithms were chosen to be applied (Article1):

1. KNN
2. Decision table
3. Simple logistic
4. Logistic
5. SVM

1.2.1. Classification with K-Nearest Neighbor algorithm

The Nearest Neighbor (NN) technique is a simple approach to the problem of classification, so a new item is classified using the majority of the categories of k examples that are closest to what is given. To categorize an X element, we identify the k nearest neighbors. For example for κ = 3 we mean 3 closest neighbors while we can give weight to the nearest neighbor. To implement the algorithm, we need to know in advance the value of k and the distance metric. So we enter different number k, different folds, different percentage of training data. Finally, we save the results of the different methods with k, in order to compare them. In this particular Dataset, we applied k = 1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 20 different neighbors to compare the results with each other and find the best result.

The algorithm was executed with two possible configurations supported by the WEKA toolbox. The first setting allows the definition of a different number of folds, while the second allows the application of different percentages of training and test dataset (Percentage split), as shown in Figure 2.

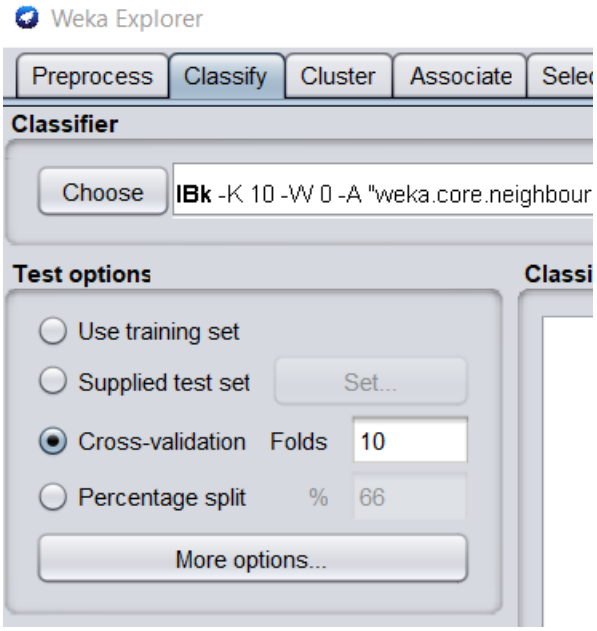

Figure 2. Selection of KNN algorithm execution parameters in WEKA

KNN execution scenarios with folds application are described in the Table 1.

Table 1. Scenarios of KNN algorithm execution with folds implementation

	Cross fold validation 2	**Cross fold validation 4**	**Cross fold validation 6**	**Cross fold validation 8**	**Cross fold validation 10**
k	**Correctness**	**Correctness**	**Correctness**	**Correctness**	**Correctness**
1	75,7021	76,4113	75,9858	76,4255	76,3830
2	75,9291	76,5106	76,8794	76,5532	76,8511
3	77,1348	77,4610	77,4752	77,7021	78,0142
4	77,9858	78,3404	78,4823	78,6383	78,7518
5	78,4965	78,7943	78,9787	79,0213	78,9787
7	78,6950	79,2199	79,0638	79,2340	79,1206
8	78,8652	79,0922	79,2624	79,2624	79,2057
10	78,9787	79,2057	79,4610	79,2908	79,4894
12	78,9929	79,4752	79,3901	79,5461	79,6454
14	78,9220	79,3050	79,2340	79,2766	79,6312
16	78,8085	79,1631	79,2908	79,3333	79,4468
20	78,5390	78,9220	79,2057	79,1773	79,1631

The following diagrams show the performance of the KNN algorithm for different folds and different values of k in each fold.

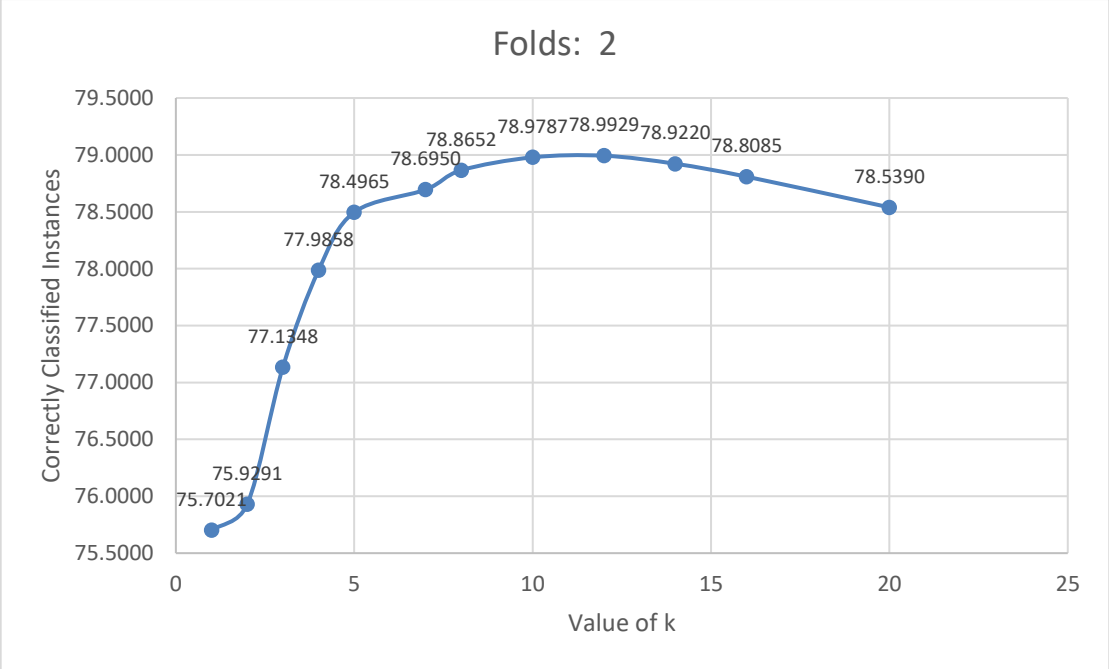

*Figure 3. **KNN** algorithm results for 2 folds and values k = 1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 20*

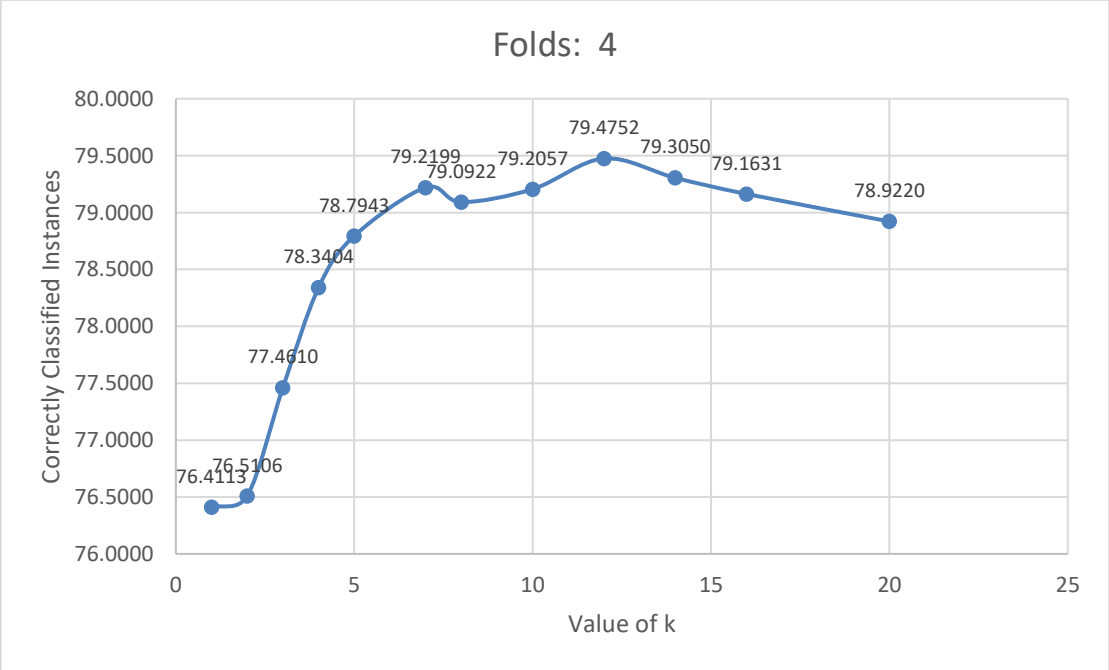

*Figure 4. **KNN** algorithm results for 4 folds and values k = 1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 20*

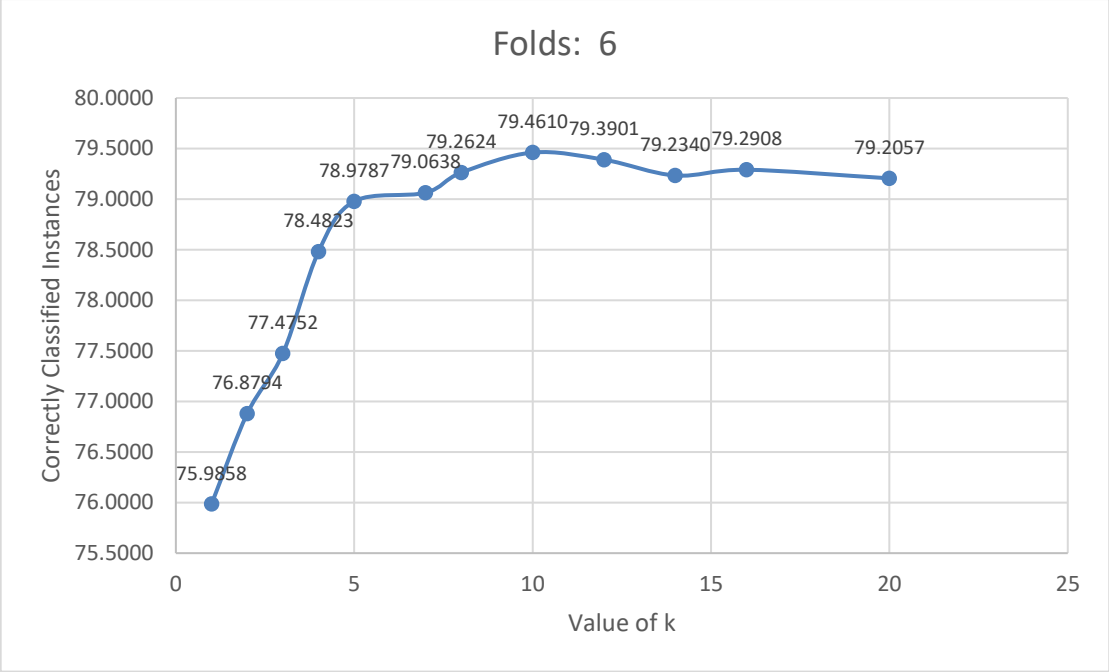

*Figure 5. **KNN** algorithm results for 6 folds and values k = 1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 20*

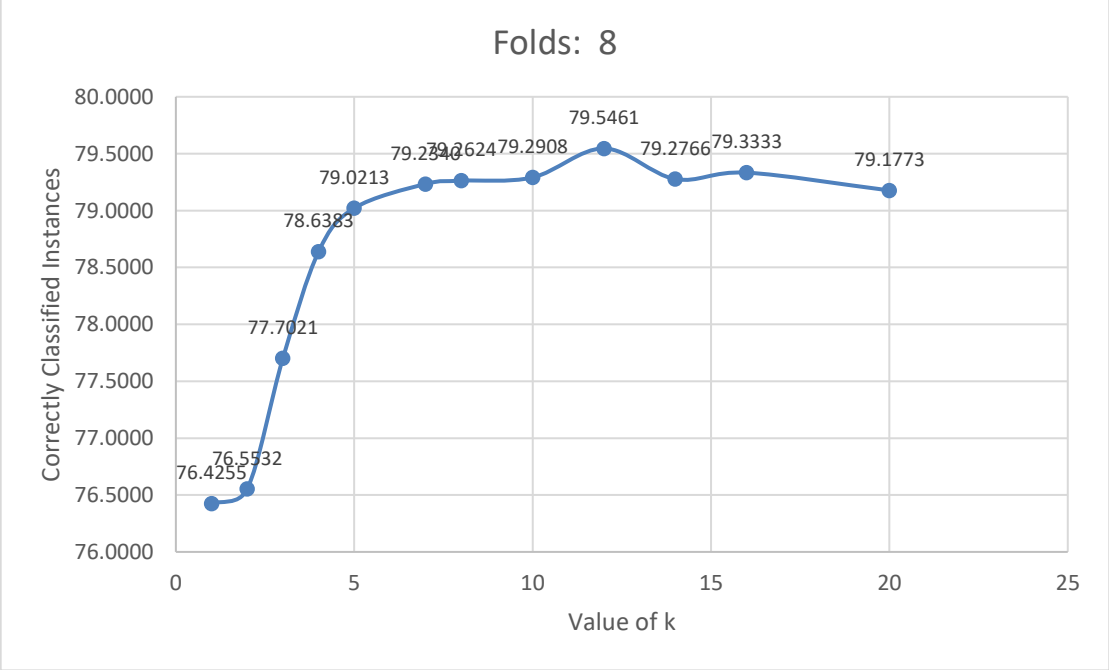

*Figure 6. **KNN** algorithm results for 8 folds and values k = 1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 20*

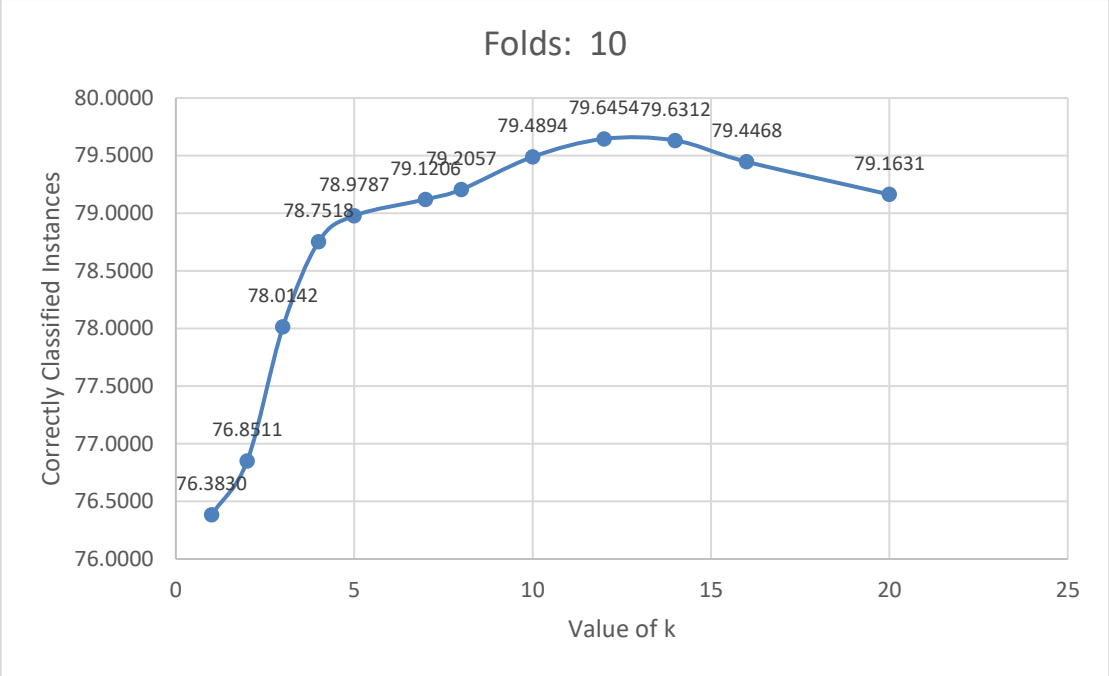

*Figure 7. **KNN** algorithm results for 10 folds and values k = 1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 20*

KNN execution scenarios with percentage split application in training and test dataset are described in Table 2.

Table 2. KNN algorithm execution scenarios with percentage split implementation

	Percentage Split 50%	**Percentage Split 70%**	**Percentage Split 80%**
k	**Correctness**	**Correctness**	**Correctness**
1	74,8369	75,6974	76,0993
2	75,5745	76,0284	77,5177
3	77,6454	77,9669	78,7943
4	78,5816	79,1489	79,9291
5	79,0638	78,5816	79,7163
7	79,3759	79,2435	80,3546
8	79,3475	79,8109	80,5674
10	79,5745	79,7163	81,0638
12	79,3191	79,5272	80,8511
14	79,4610	79,4326	80,7801
16	79,2908	79,2908	80,7801
20	79,0922	79,4326	81,0638

The following diagrams show the performance of the KNN algorithm for different percentages of training and test datasets and values of k in each execution.

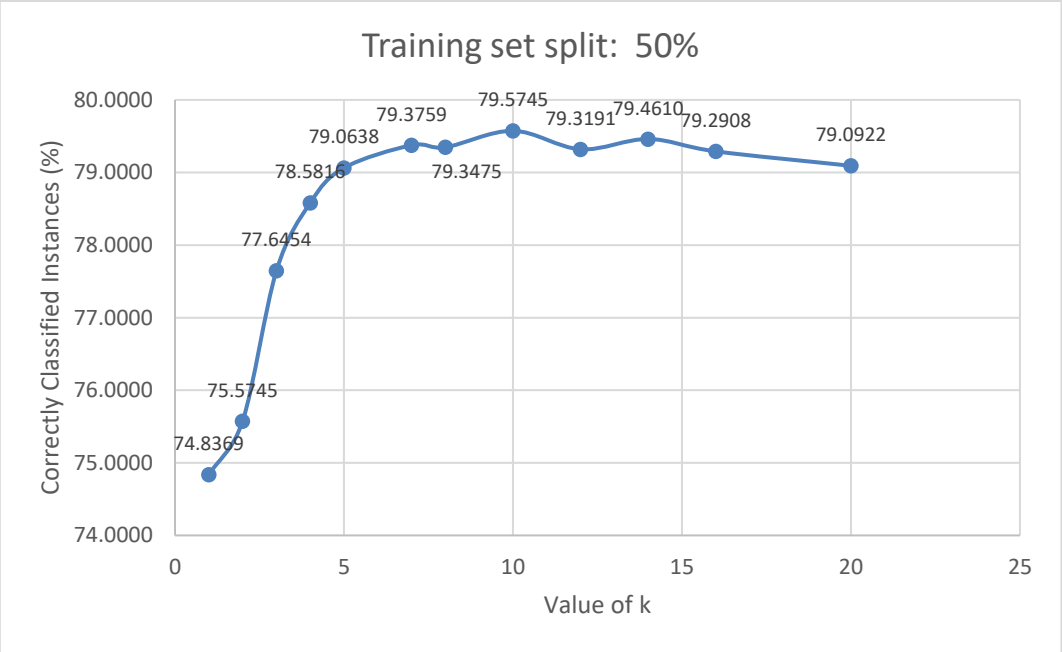

*Figure 8. Results of **KNN** algorithm execution with training set percentage=50% and values k=1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 20*

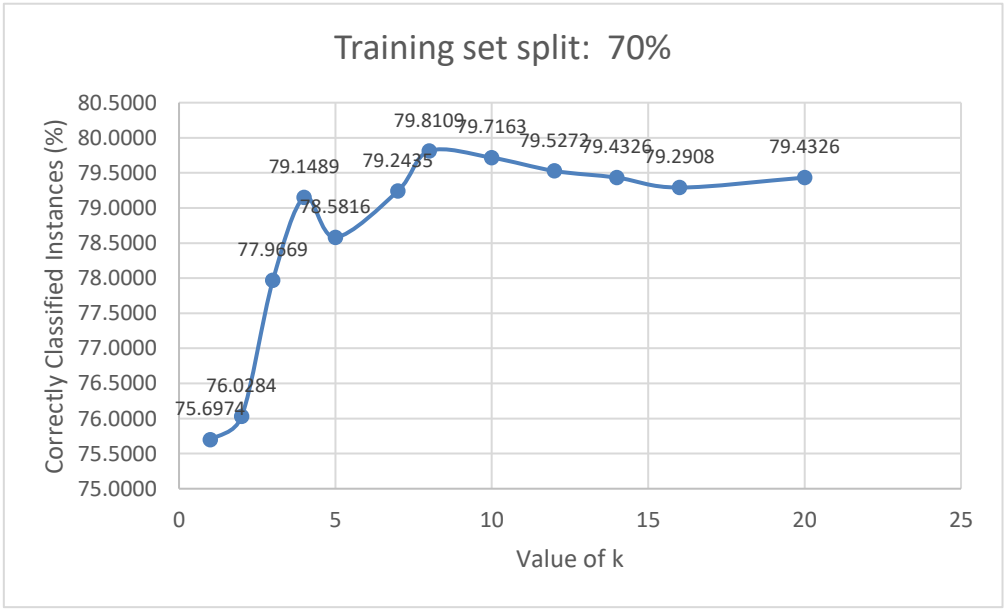

*Figure 9. Results of **KNN** algorithm execution with training set percentage=70% and values k=1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 20*

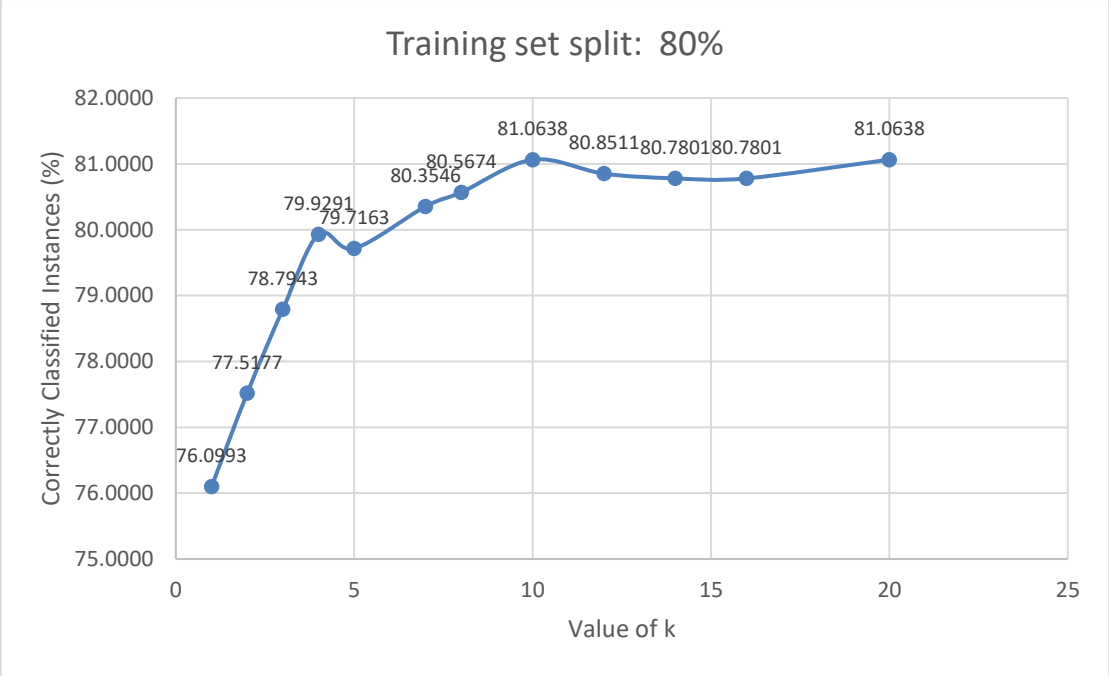

*Figure 10. Results of **KNN** algorithm execution with training set percentage=80% and values k=1, 2, 3, 4, 5, 7, 8, 10, 12, 14, 16, 20*

A summary graph of the correctness results was created for each execution of the KNN algorithm, and the results are shown in Figure 11 below.

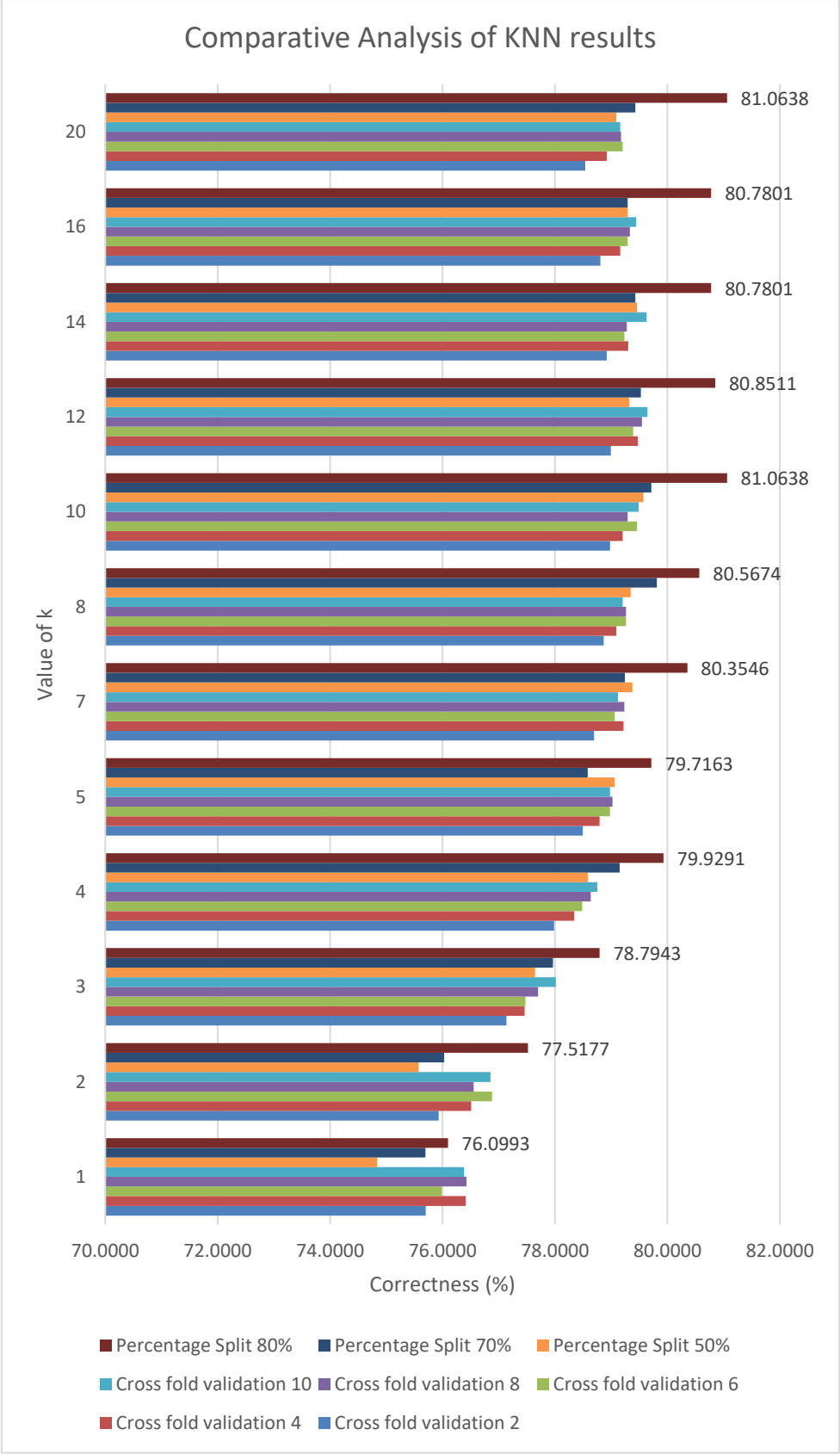

*Figure 11. Aggregate performance graph of the **KNN** algorithm for different execution parameters*

Conclusion

The KNN algorithm offers the best result in terms of correctness ratio in the execution with Percentage split 80% and k = 20 but also in the execution with Percentage split 80% and k = 10. In both cases, the value of correctness was recorded at 81.0638%.

1.2.2. Classification with Decision table Algorithm

The decision table is a compact means of documenting the different decisions or actions to be taken in different sets of conditions. The purpose of a decision table is to structure logic by creating rules derived from the data entered into the table itself. A decision table lists causes (business rule situation) and effects (business action, expected results) which are presented using a table where each column represents a unique combination. If there are rules within a business that can be expressed through the use of standards and data, then a decision board is a technique that can be used to achieve this. Each decision table series collects and stores its data separately and then combines the data with a specific or custom template to create a rule. Decision tables should not be used if these rules do not follow a set of standards (Techopedia, 2021).

For example, what kind of market would consumers prefer through Facebook depending on the different types of sales, such as selling live video, selling through a link, or through photo viewing.

The algorithm was executed with two possible configurations supported by the WEKA tool. The first setting allows the definition of a different number of folds, while the second allows the application of different percentages of training and test dataset (Percentage split) (Figure 12).

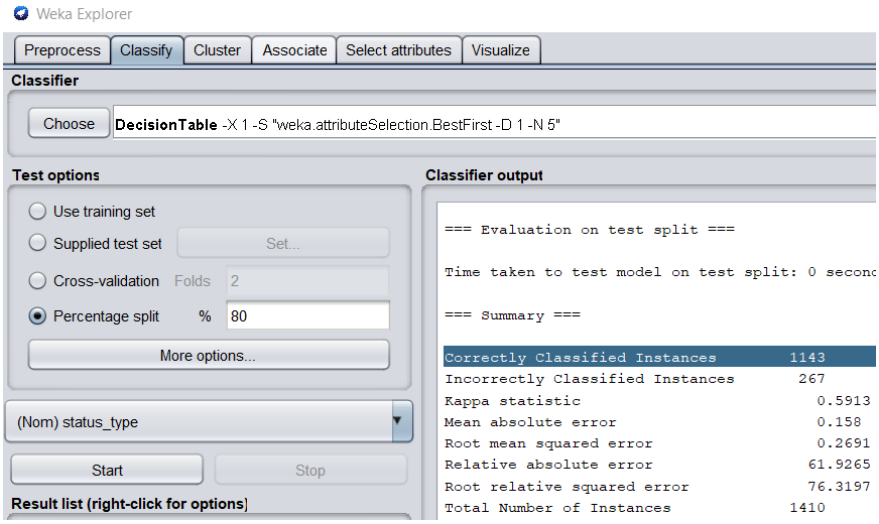

*Figure 12. Selection of execution parameters of the **Decision table** algorithm in WEKA*

The execution scenarios of the Decision table with folds application are described in Table 3.

Table 3. Execution scenarios of Decision table algorithm with application of folds

Folds	**Correctness (%)**
10	79,4752
8	79,3333
6	79,1631
4	78,7518
2	79,4468

The following chart shows the performance of the Decision table algorithm for different folds (Picture 13).

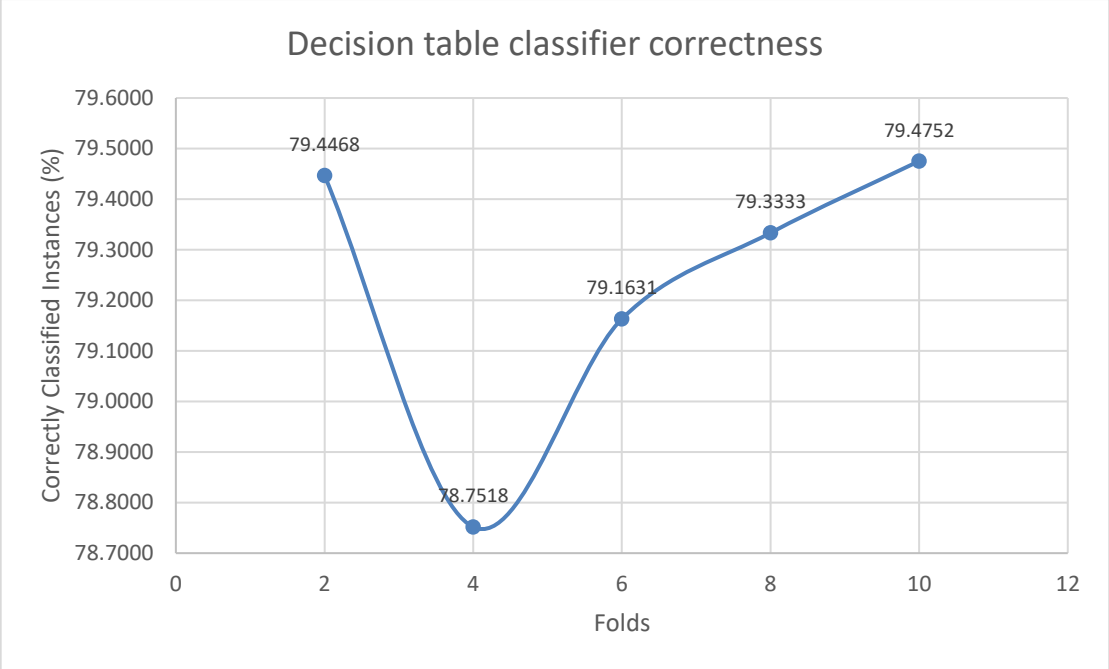

*Figure 13. Results of **Decision table** algorithm for 2, 4, 6, 8 and 10 folds*

The execution scenarios of the Decision table with application of percentage split in training and test dataset, are described in the Table 4.

Table 4. Scenarios of execution of Decision algorithm with percentage split application

Percentage split	**Correctness (%)**
50	78,6099
70	79,9054
80	81,0638
90	77,8723

The following chart shows the performance of the Decision table algorithm for different sets of datasets in train and test sets (Figure 13).

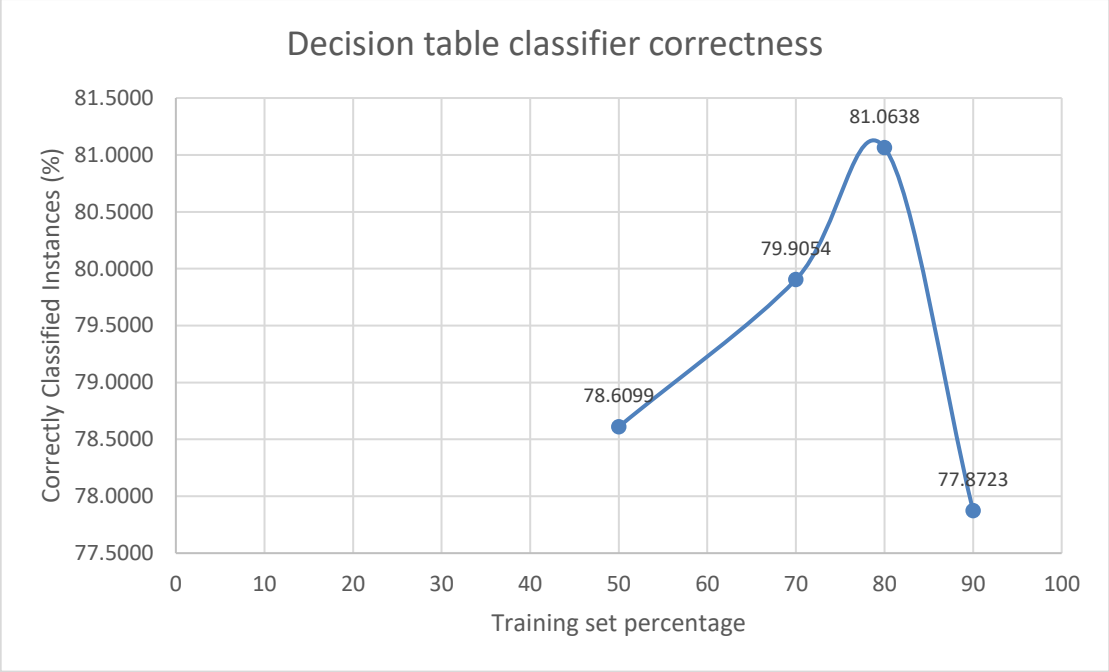

*Figure 14. Results of **Decision table** algorithm for percentage split 50%, 70%, 80% and 90%*

The aggregate chart of the correctness results for each execution of the Decision table algorithm and the results are shown in Figure 11 below.

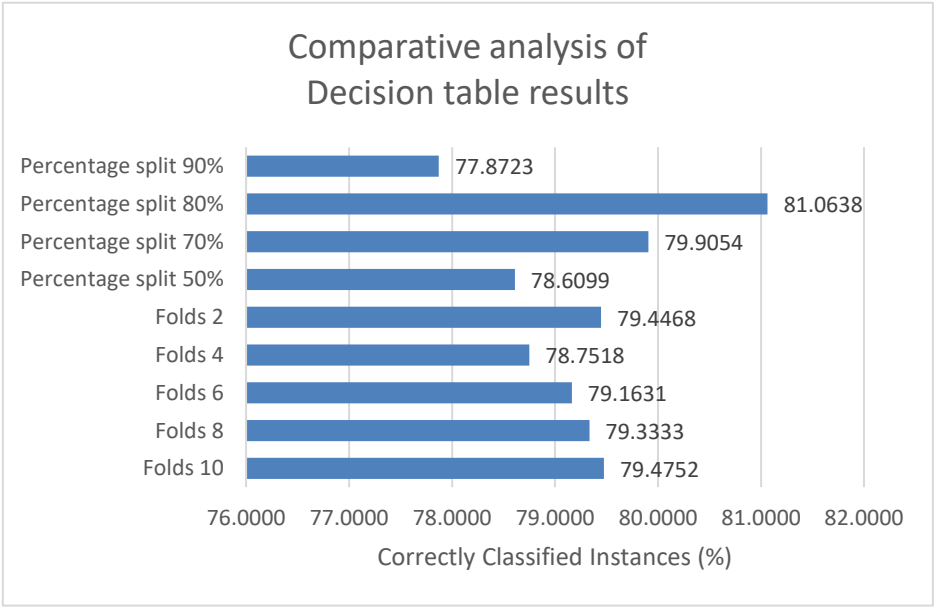

*Figure 15. Aggregate performance results of the **Decision table** algorithm for different execution parameters*

Conclusion

The Decision table algorithm offers the best result in terms of the correctness ratio in execution with percentage split 80%. The value of correctness recorded is 81.0638%.

1.2.3. Classification with Simple logistic algorithm

The Simple Logistic method is a one-parameter logistic regression. Logistic regression comes from the fact that linear regression can also be used to perform a classification problem, but logistic regression is not linear (because it involves exponential transformation and proportion transformation, eg Logit Transformation) Simply by converting the categorical target with constant prices.

The idea of logistic regression is to make linear regression generate probabilities. It is always better to predict class odds than to predict classes (DataCadamia, 2021).

The regression logic estimates the order probabilities directly using the Logit transform.

The algorithm was executed with two possible configurations supported by the WEKA tool. The first setting allows the definition of a different number of folds, while the second allows the implementation of different percentages of training and test dataset (Percentage split), as shown in Figure 16.

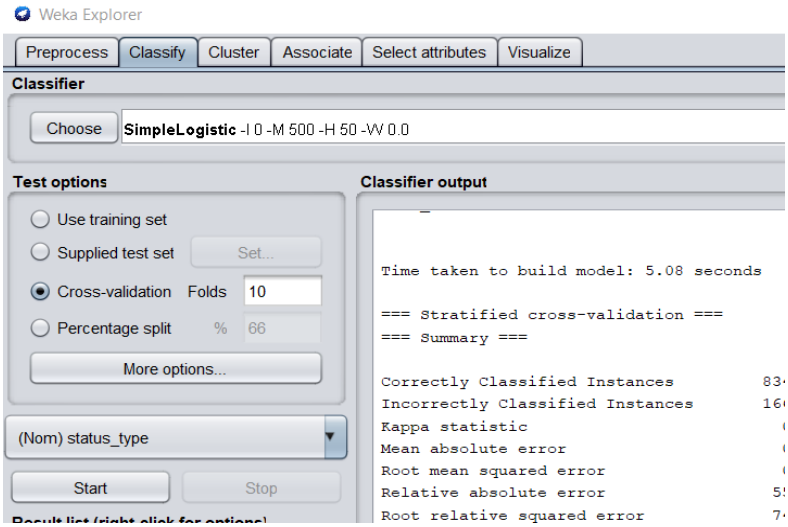

*Figure 16. Parameters selection and execution of **Simple Logistic** Algorithm on Weka*

Simple logistic execution scenarios with folds implementation are described in the Table 5.

Table 5. Simple logistic algorithm execution scenarios with folds implementation

Folds	Correctness (%)
2	82,3000
4	83,3000
6	83,0000
8	83,1000
10	83,4000

The following graph shows the performance of the Simple logistic algorithm for different folds.

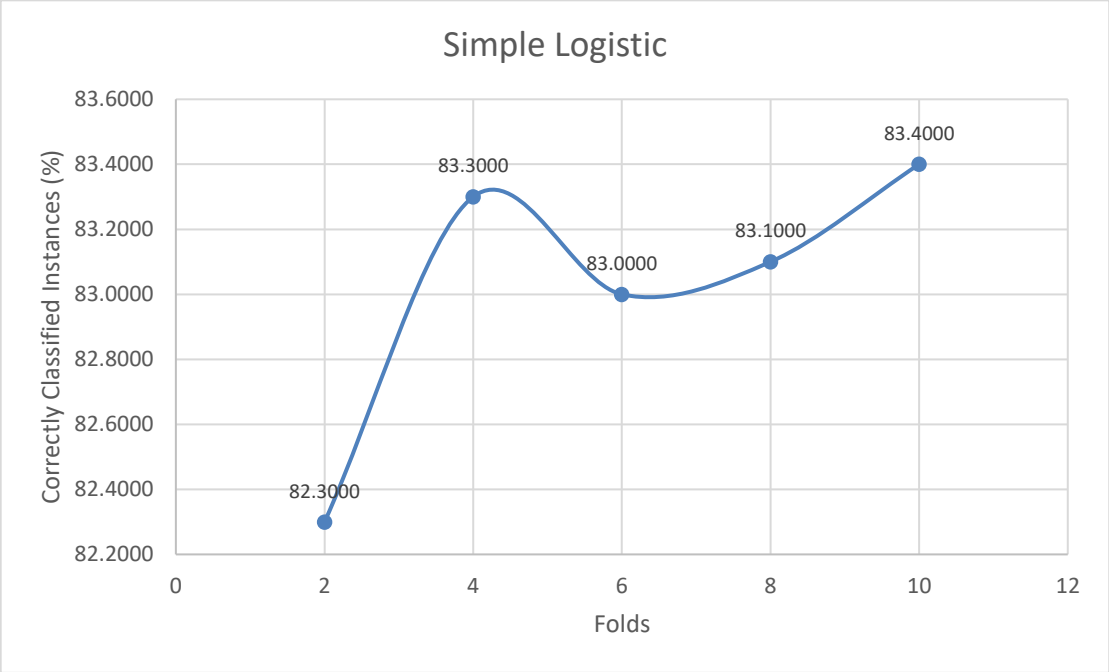

*Figure 17. **Simple logistic** algorithm results for 2, 4, 5, 6, 8 and 10 folds*

The execution scenarios of Simple logistic with percentage split application in training and test dataset, are described in the Table 6.

Table 6. KNN algorithm execution scenarios with percentage split implementation

Percentage split	Correctness (%)
50%	83,2000
70%	83,3333
80%	84,0000
90%	81,0000

The following chart below, shows the performance of the **Simple logistic** algorithm for different percentages of training and test datasets..

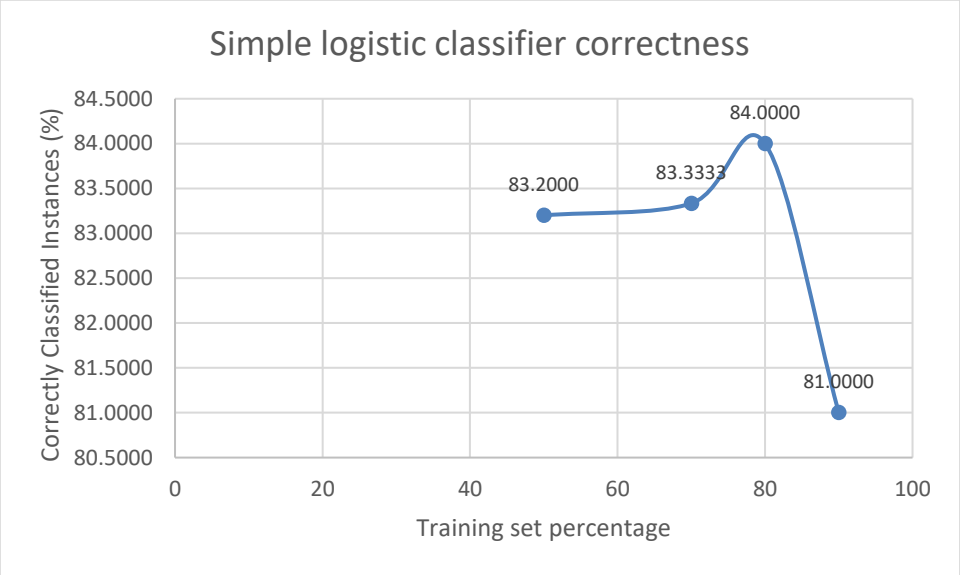

Figure 18. Results of Simple logistic algorithm execution with training set percentage 50%, 70%, 80% and 90%

An aggregate graph of the correctness results for each execution of the Simple logistic algorithm is generated and the results are displayed in Figure 19 below.

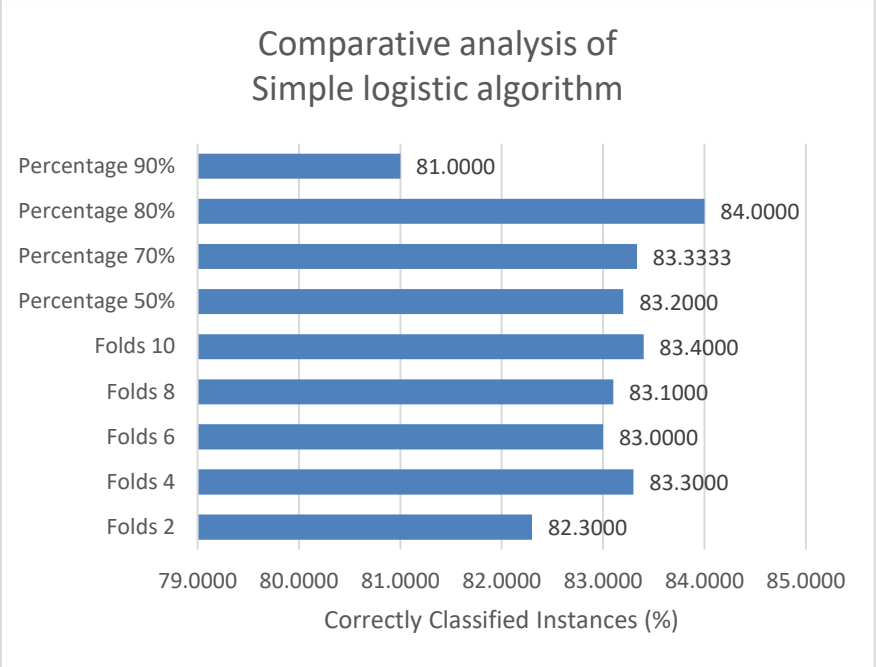

*Figure 19. Aggregate performance results of the **Simple logistic** table algorithm for different execution parameters*

Conclusion

The Simple logistic algorithm offers the best result in terms of correctness ratio in execution with percentage split 80%. The value of correctness recorded is 84.0000%.

1.2.4. Classification with Logistic algorithm

Logistic regression investigates the non-linear effect of a dependent categorical variable on the action of many independent variables. It is characterized, depending on the nature of the categories of the dependent variable, by three categories of models, the binomial regression (with only two categories), the tactical (the categories are ordered in ascending order) and the nominal (qualitative categories). In the examined regression model, the techniques of the optimal selection of the status_type variables and the diagnostic criteria of validity and reliability of the model are applied. An example is the study of the possibility of promoting e-commerce sales, through the facebook website, using the specific regression model. Logistic regression is essentially a model for classifying the values of a Y variable response based on probability theory. In this model where the variable Y is usually binary (takes two values) the goal is to predict this outcome from a number of predictor variables that can be nominal, regular or quantitative (Πετρίδης, 2015).

The algorithm was executed with two possible configurations supported by the WEKA tool. The first setting allows the definition of a different number of folds, while the second allows the application of different percentages of training and test dataset (Percentage split), as shown in Figure 20.

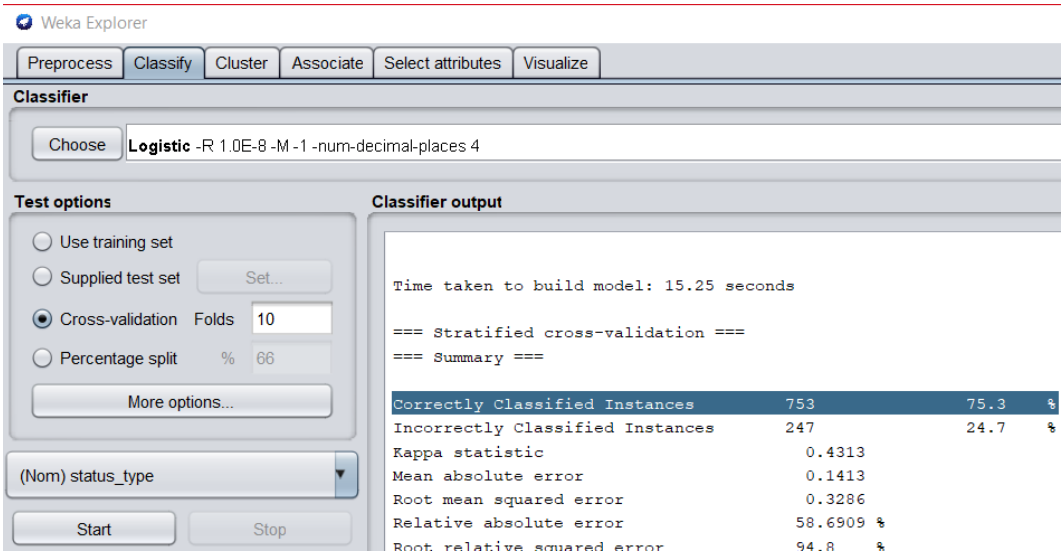

*Figure 20. Selection of execution parameters of the **Logistic** algorithm in WEKA*

Τα σενάρια εκτέλεσης του Logistic με εφαρμογή folds, περιγράφονται στον Πίνακα 7.

Table 7. Logistic algorithm execution scenarios with folds application

Folds	**Correctness (%)**
2	81,3000
4	81,8000
6	78,3000
8	81,3000
10	75,3000

The following line chart shows the performance of the Logistic algorithm for different folds.

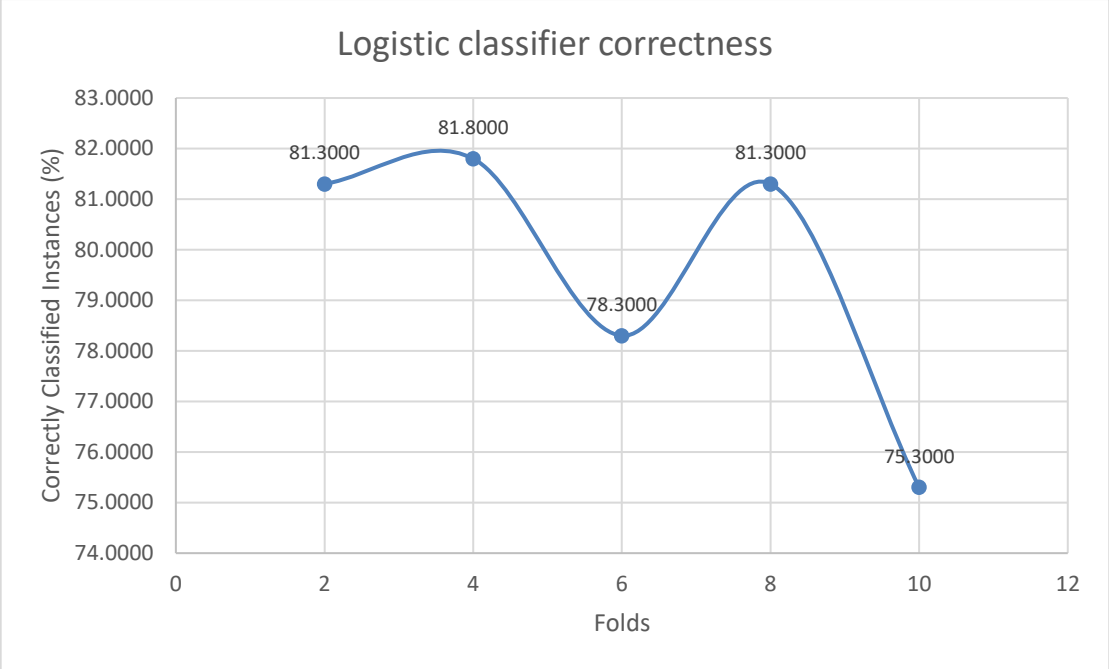

Figure 21. Simple logistic algorithm results for 2, 4, 6, 8 and 10 folds

The execution scenarios of Logistic with percentage split application in training and test dataset, are described in the Table 8.

Table 8. Logistics algorithm execution scenarios with percentage split application

Percentage split	**Correctness (%)**
50	79,6000
70	80,3333
80	81,5000
90	79,0000

The line chart below shows the performance of the Logistic algorithm for different training percentages and test datasets.

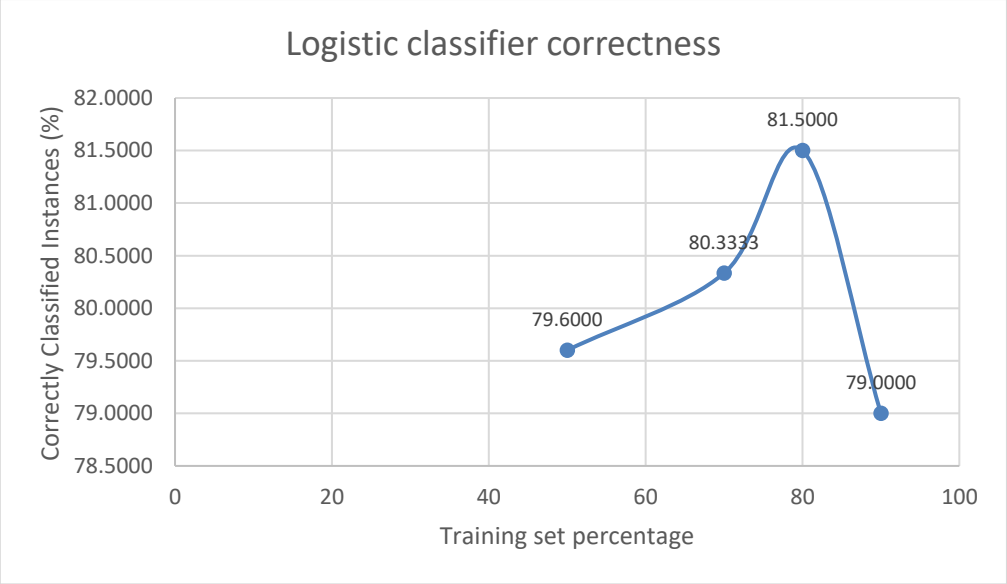

Figure 22. Logistic algorithm execution results with training set percentage of 50%, 70%, 80% and 90%

A summary graph of the correctness results was generated for each execution of the Logistic algorithm and the results are shown in Figure 23 below.

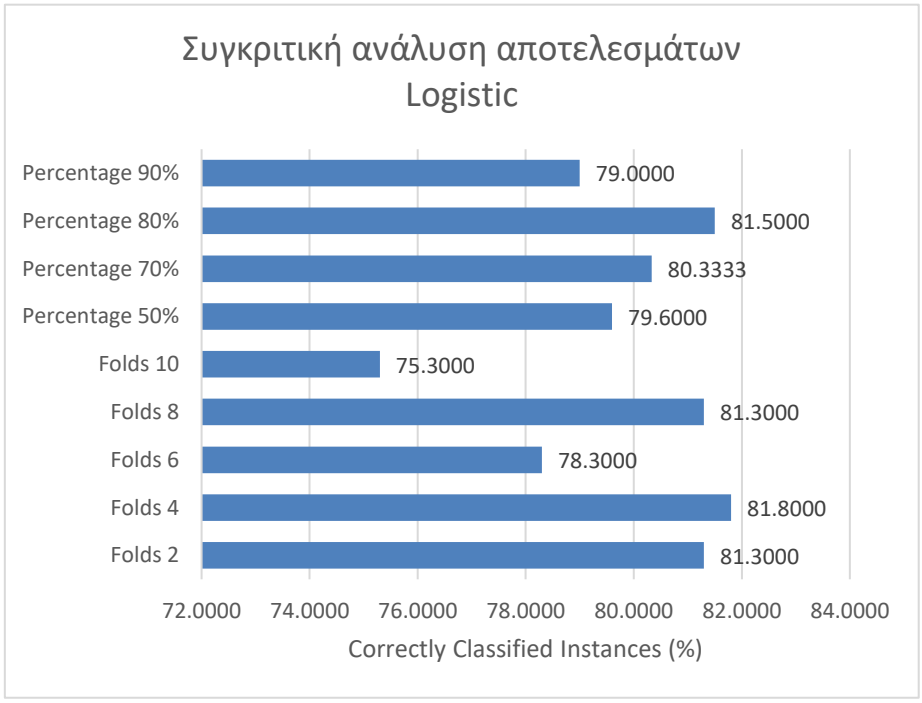

Figure 23. Aggregate performance results of the Logistic algorithm for different execution parameters

Conclusion

The Logistic algorithm offers the best result in terms of the correctness ratio in the execution with 4 folds. The highest value of correctness recorded is 81.8000%.

1.2.5. Classification with SVM algorithm

Support Vector Machines (SVMs) are a set of methods used for sorting and regression. They belong to a family of generalized graphic classifiers. The Support Vector Machines (SVM) is a prediction, sorting and regression tool that uses Mechanical Learning theory to maximize predictive accuracy while avoiding overly fitting data. SVM was originally popular with the research community and is now an active part of machine learning research worldwide. SVM became widely known when pixel usage was introduced, because it gives an accurate comparison to complex neural networks and processed data in a handwriting recognition task. The SVM technique is widely used and has a performance (ie, the error rate for test sets) either similar or significantly better than that of competing methods. (Παπαδάκη, 2012).

The algorithm was executed for different values of parameter ***c*** and percentage split. The selection of their price is done in the corresponding window of the WEKA application (Figure 24).

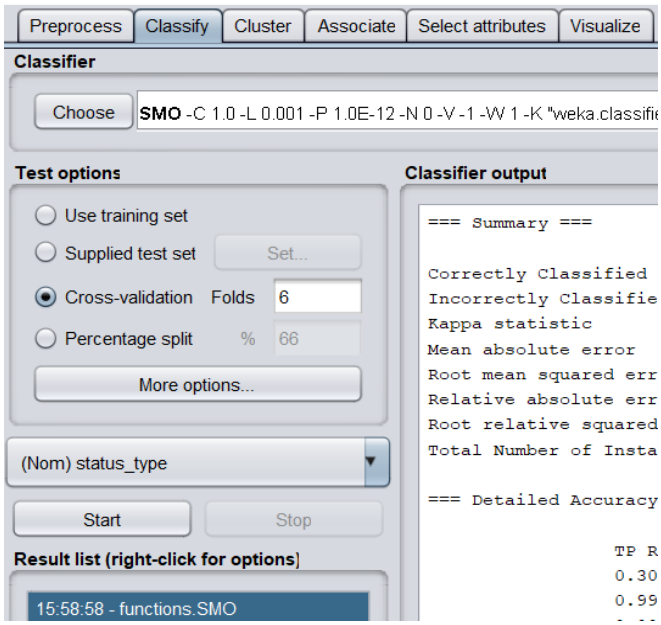

*Figure 24. Configure parameter **c** to execute the SVM algorithm in WEKA*

SVM execution scenarios for different ***c*** and value of folds = 10, are described in

The Table 9.

Table 9. SVM algorithm execution scenarios for different values of parameter c

c	Correctness (%)
0,3	71,9000
0,5	73,5000
0,7	74,4000
1,0	74,3000
1,2	74,3000
1,5	74,3000
1,8	74,3000

The diagram below illustrates the performance of the SVM algorithm for different **c** and 10 folds.

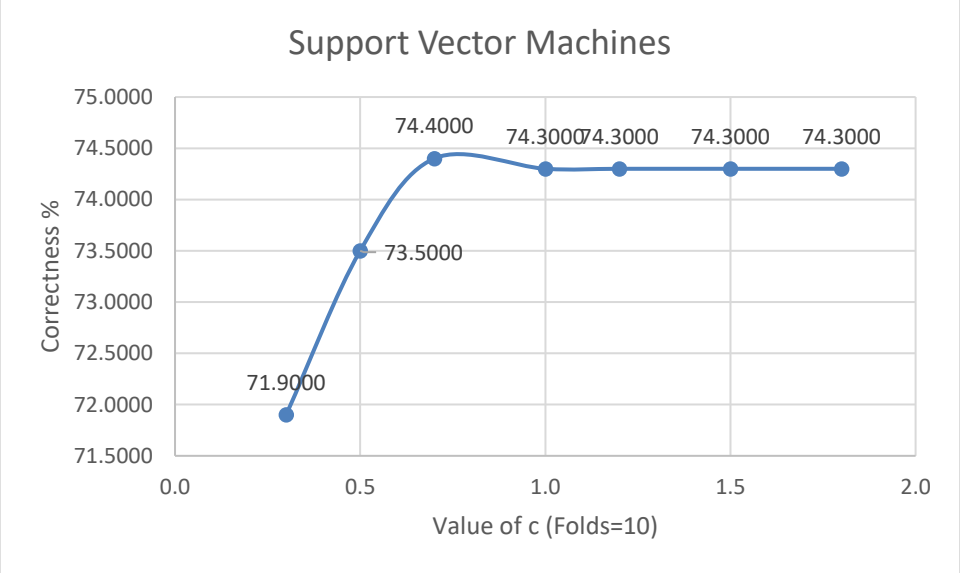

*Figure 25. SVM algorithm results for **c** = 0.3, 0.5, 0.7, 1.0, 1.2, 1.5, 1.8 and 10 folds*

The execution scenarios of SVM with percentage split application in training and test dataset, are described in the Table 10.

Table 10. SVM algorithm execution scenarios with percentage split application

Percentage split	Correctness (%)
50	74,8000
70	74,0000
80	73,5000
90	71,0000

The line graph below shows the performance of the SVM algorithm for different percentages of training and test datasets.

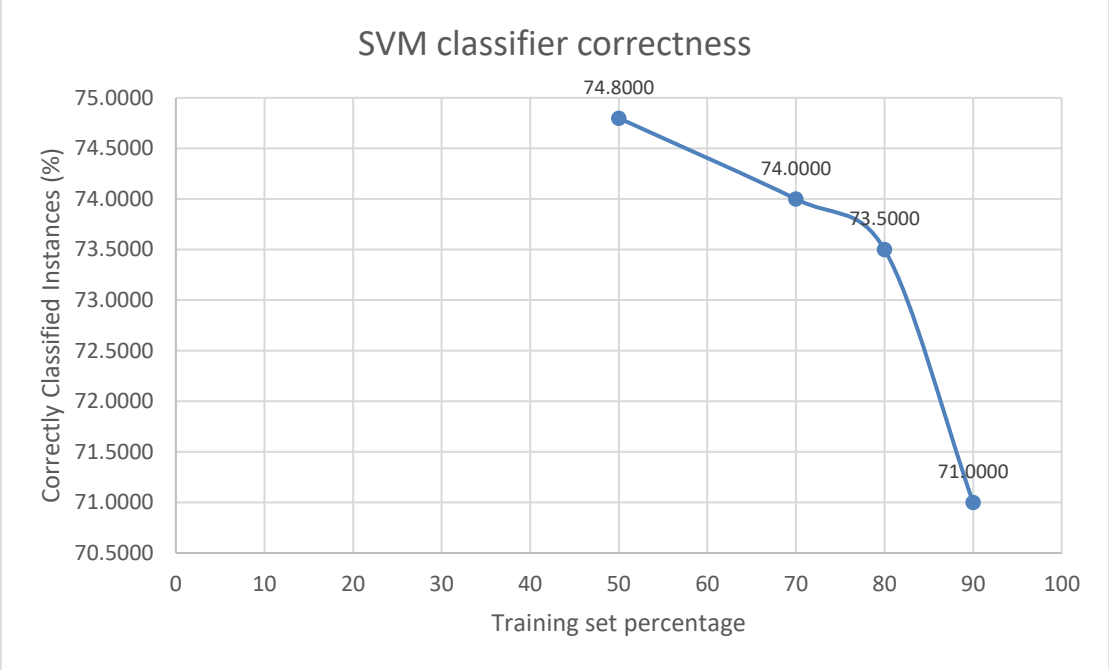

Figure 26. SVM algorithm execution results with training set percentage of 50%, 70%, 80% and 90%

An aggregate graph of the correctness results for each execution of the SVM algorithm is generated and the results are shown in Figure 27 below.

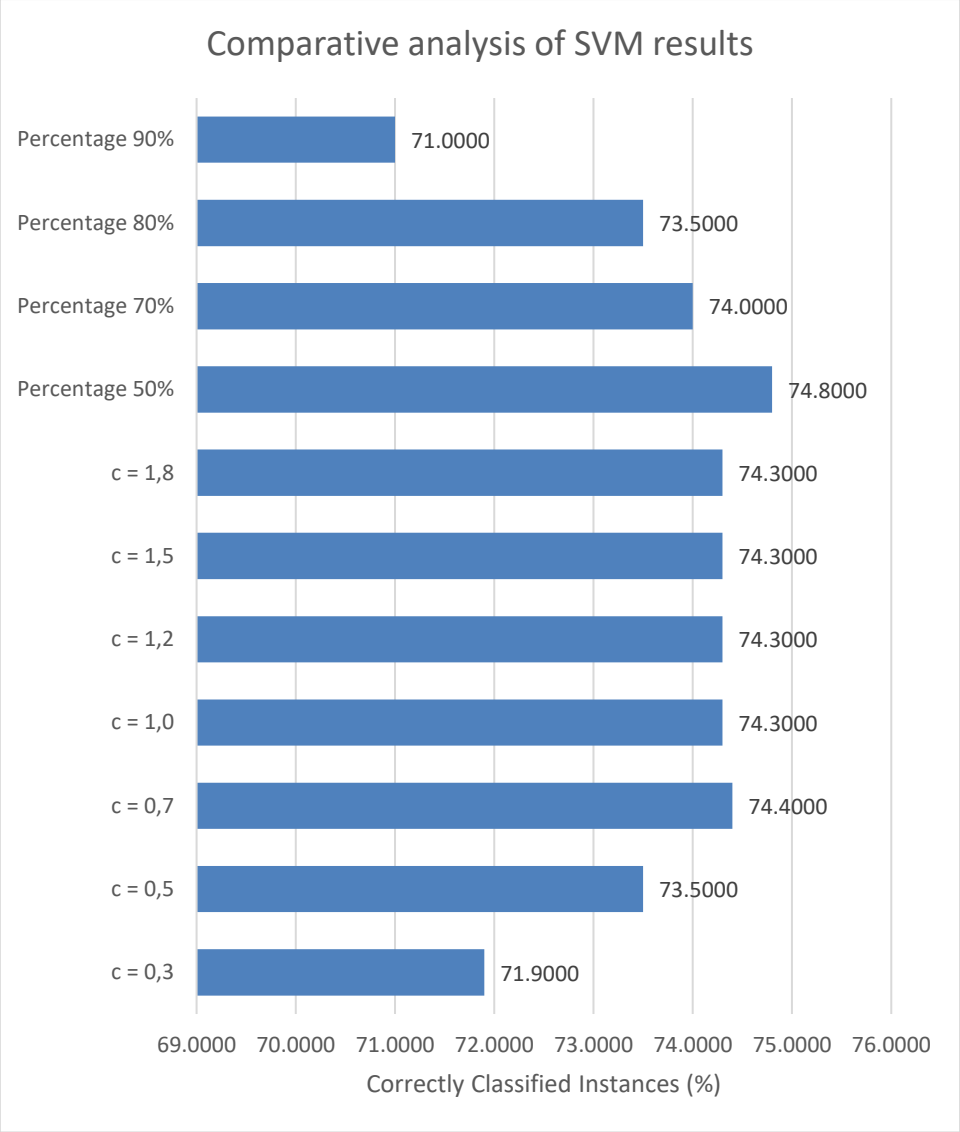

Figure 27. Aggregate performance results of the SVM algorithm for different execution parameters

Conclusion

The SVM algorithm offers the best result in terms of correctness ratio in execution with percentage split 50%. The value of correctness recorded is 74.8%.

1.3. Results / Conclusion

The best result is obtained by using the Simple logistic algorithm, with a training rate of 80%, which gives an estimate of 84.0000%.

With this data, using this particular classifier, we can have predictions with an 84% chance of success for consumer behavior in live stream sales through social networks. The interested party can provide indicative:

1. How many visitors will react depending on the time of publication of the entry.
2. How many visitors will react if the post contains a video or image or link.
3. Combinations of the above conditions.

By studying the ROC curves for the different values of the "status_type" property, we try to identify for which values the model gives the best values for predicting True Positive cases. A good ROC curve should be shifted as far as possible to the vertical axis of true positives. This ensures that the highest percentage of predictions is correctly recorded as true positive and a low percentage as false positive. Accordingly, we study the ROC curves for the four values of the "status_type" property.

The “photo” value produces a relatively good ROC curve, similar to that of the “video” value. Both curves move above the AUC curve, approaching the vertical axis, which means that the FP entries remain low. From the level of TP 0.6 the ROC curve acquires a slope, which means that the TP entries decrease and the FPs increase accordingly.

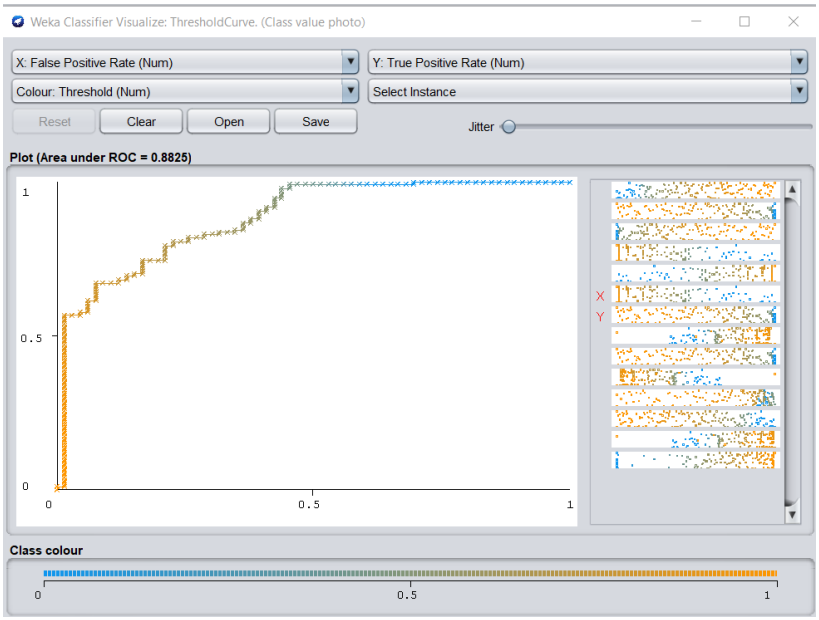

Figure 28. ROC curve for the Simple logistic classifier with 80% percentage split for the “photo” value in the status_type property

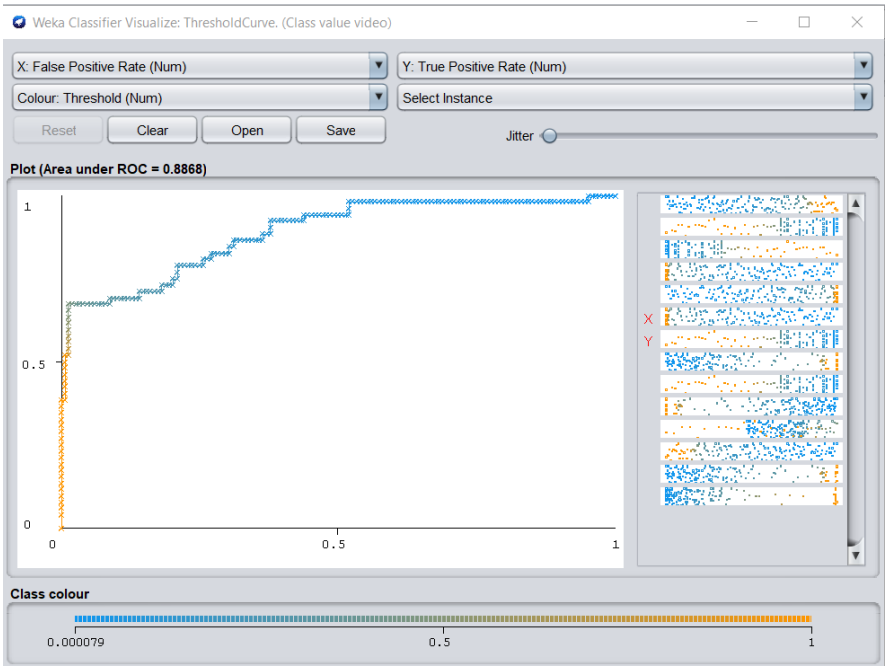

Figure 29. ROC curve for the Simple logistic classifier with percentage split 80% for the value "video" in the status_type property

The "status" value produces a ROC curve, whose true positives are lower than that of photo / video. The curve moves above the AUC curve, but does not approach the vertical axis, which means that TP entries have a large margin of error, while FPs remain high compared to video / photo values.

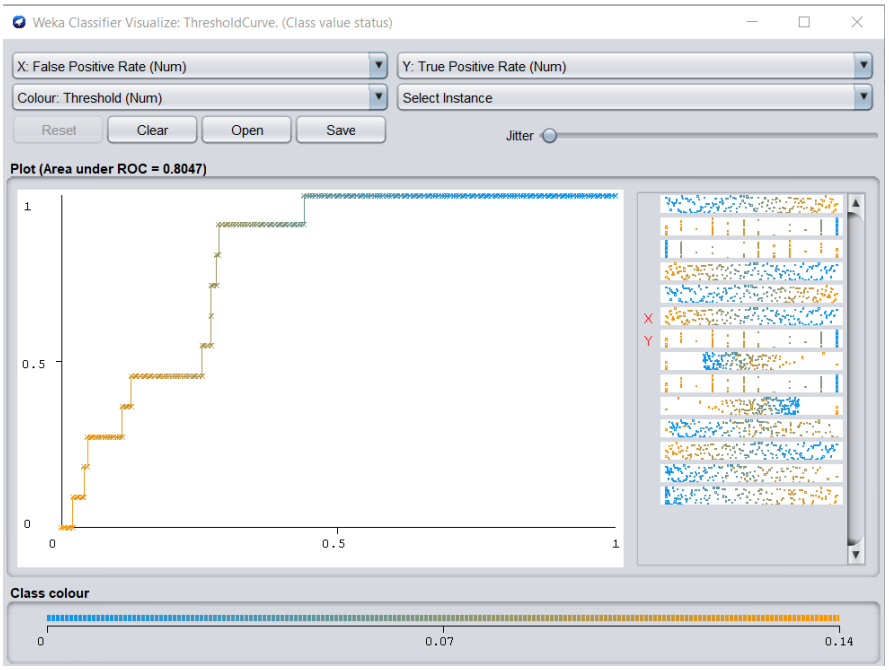

Figure 30. ROC curve for the Simple logistic classifier with 80% percentage split for the value "status" in the status_type property

The "link" value produces a relatively good ROC curve, similar to that of the "video" value. The curve moves above the AUC curve, approaching the vertical axis (to a lesser extent than the ROC curve of the photo / video, which means that the FP entries remain low. that TP entries are reduced slightly, keeping the FP value low.

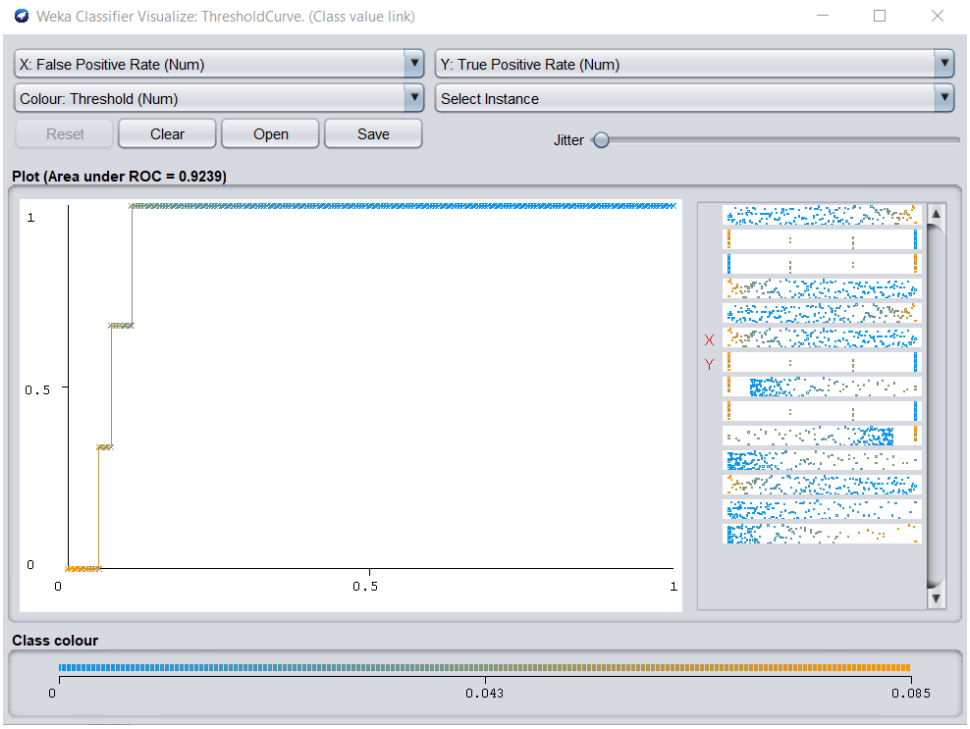

Figure 31. ROC curve for Simple logistic classifier with percentage split 80% for the value "link" in the status_type property

2. Dataset 2: Online retail

The Python language was used to edit the "Online retail" dataset. The reason is that WEKA classifiers in WEKA could not process the nominal properties of the dataset. For editing, the data file was uploaded to a Pandas DataFrame.

2.1. Preprocessing

In the Figure 32 the first 5 datasets of the dataset are displayed so that we can form an opinion about them.

```
In [11]: runfile('C:/Users/Giorgos/PycharmProjects/pythonProject/Online Retail.py', wdir='C:/Users/Giorgos/PycharmProjects/
pythonProject')
  InvoiceNo StockCode                          Description  Quantity  \
0    536365    85123A   WHITE HANGING HEART T-LIGHT HOLDER         6
1    536365     71053                  WHITE METAL LANTERN         6
2    536365    84406B       CREAM CUPID HEARTS COAT HANGER         8
3    536365    84029G  KNITTED UNION FLAG HOT WATER BOTTLE         6
4    536365    84029E       RED WOOLLY HOTTIE WHITE HEART.         6

      InvoiceDate  UnitPrice  CustomerID         Country
0  12/1/2010 8:26       2.55     17850.0  United Kingdom
1  12/1/2010 8:26       3.39     17850.0  United Kingdom
2  12/1/2010 8:26       2.75     17850.0  United Kingdom
3  12/1/2010 8:26       3.39     17850.0  United Kingdom
4  12/1/2010 8:26       3.39     17850.0  United Kingdom
```

Figure 32. Print .head() for the dataset "Online retail"

Figure 33 presents the data structure, data types and the number of non-null values of each column.

```
==========================DATA INFO============================
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  object
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 33.1+ MB
None
```

Figure 33. Result of printing the .info() method in dataframe

Observing the results in Figure 33, we conclude that the only columns that contain null values are the:

1. Description (1454 null values)
2. CustomerID (135.080 null values)

From those values, we decided to delete from the dataset those in which the "CustomerID" column has a null value.

Figure 34 shows the result of applying the .describe() method to the dataframe. We obtain results for the number of entries, the average value of the numeric columns, the standard deviation, minimum and maximum values of the numeric columns as well as the frequencies of display of the values in the areas 25%, 50% and 75%.

```
=========================DATA DESCRIPTION==========================
            Quantity      UnitPrice     CustomerID
count  541909.000000  541909.000000  406829.000000
mean        9.552250       4.611114   15287.690570
std       218.081158      96.759853    1713.600303
min    -80995.000000  -11062.060000   12346.000000
25%         1.000000       1.250000   13953.000000
50%         3.000000       2.080000   15152.000000
75%        10.000000       4.130000   16791.000000
max     80995.000000   38970.000000   18287.000000
```

Figure 34. Results of applying the .describe() method to the dataset.

In order to determine the number of empty (null) values in each column, we applied the .isnull().sum() method to the dataframe. We deleted the rows from the data set that have a null value in the column. The results are shown in the Figure 35.

```
=========================MISSING VALUES==========================
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

Figure 35. Results of the execution of the .isnull().sum() method in the dataframe

In order for the data to be semantically correct, it was decided to exclude entries from the dataset that have zero or negative values in the "Quantity" and "UnitPrice" columns.

We also note that there are 1454 entries that do not have a description in the product while 135,080 entries do not have a value in the "CustomerID" column.

The .describe() method was then applied to derive the basic statistical values for the data set, after deleting the rows with zero or negative values in the Quantity and UnitPrice columns (Figure 36).

```
=======================DATA_NEW DESCRIPTION=========================
            Quantity      UnitPrice     CustomerID
count  530104.000000  530104.000000  397884.000000
mean       10.542037       3.907625   15294.423453
std       155.524124      35.915681    1713.141560
min         1.000000       0.001000   12346.000000
25%         1.000000       1.250000   13969.000000
50%         3.000000       2.080000   15159.000000
75%        10.000000       4.130000   16795.000000
max     80995.000000   13541.330000   18287.000000
```

Figure 36. Result of the .describe() method in the dataframe after processing the data and deleting columns

A new column named "TotalPrice" has been created, as a calculated "Quantity" * "UnitPrice" field (Figure 37).

Two columns titled "Year" and "Month" were also added, where the values were derived from the "InvoiceDate" column, in order to make the calculation of the charts easier.

```
Data columns (total 11 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    397884 non-null  object
 1   StockCode    397884 non-null  object
 2   Description  397884 non-null  object
 3   Quantity     397884 non-null  int64
 4   InvoiceDate  397884 non-null  datetime64[ns]
 5   UnitPrice    397884 non-null  float64
 6   CustomerID   397884 non-null  float64
 7   Country      397884 non-null  object
 8   TotalPrice   397884 non-null  float64
 9   Year         397884 non-null  int64
 10  Month        397884 non-null  int64
dtypes: datetime64[ns](1), float64(3), int64(3), object(4)
memory usage: 36.4+ MB
```

Figure 37. Result of applying the .info() method to the dataframe after adding the TotalPrice column

Figure 38, shows the result of applying the .head() method in order to obtain an image of the data format of the dataset.

	InvoiceNo	StockCode	Description	Quantity	\
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	
1	536365	71053	WHITE METAL LANTERN	6	
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	

	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice	\
0	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30	
1	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	
2	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00	
3	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	
4	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	

	Year	Month
0	2010	12
1	2010	12
2	2010	12
3	2010	12
4	2010	12

Figure 38. Result of applying the .head() method to the processed dataframe

The following diagrams show the statistical chart of the data.

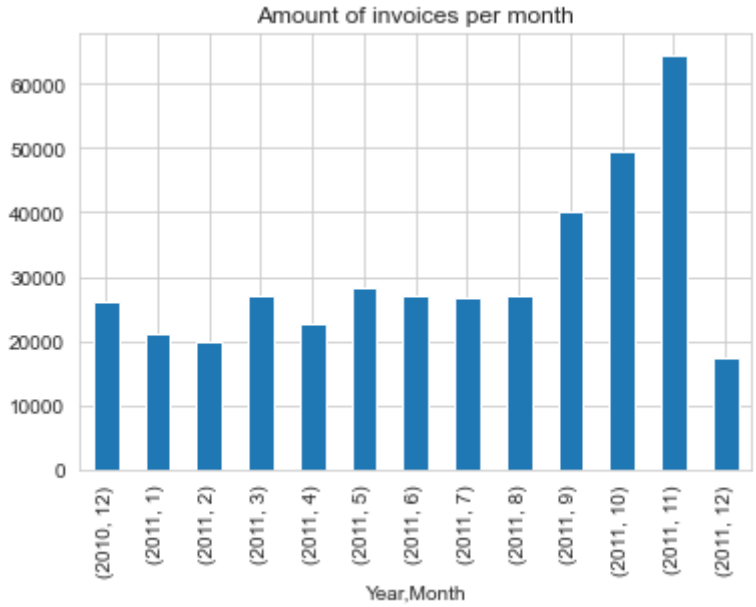

Chart 1. Number of invoices per month

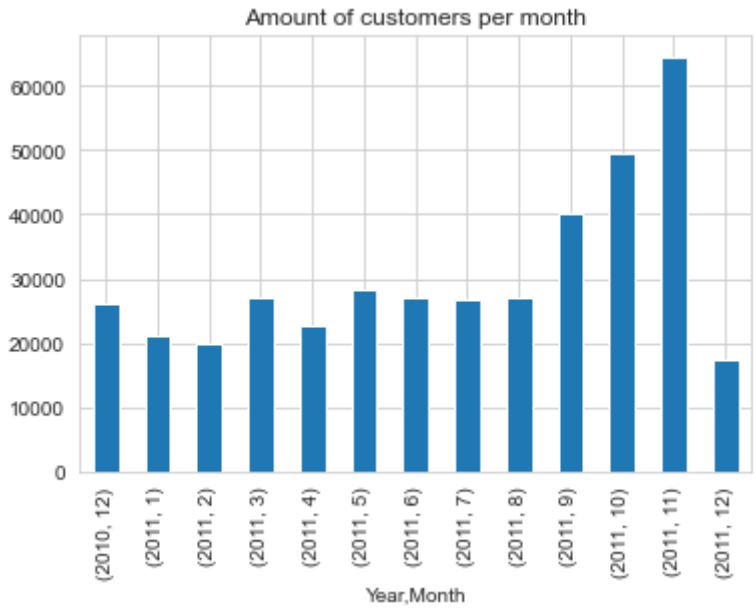

Chart 2. Number of customers who made a transaction per month

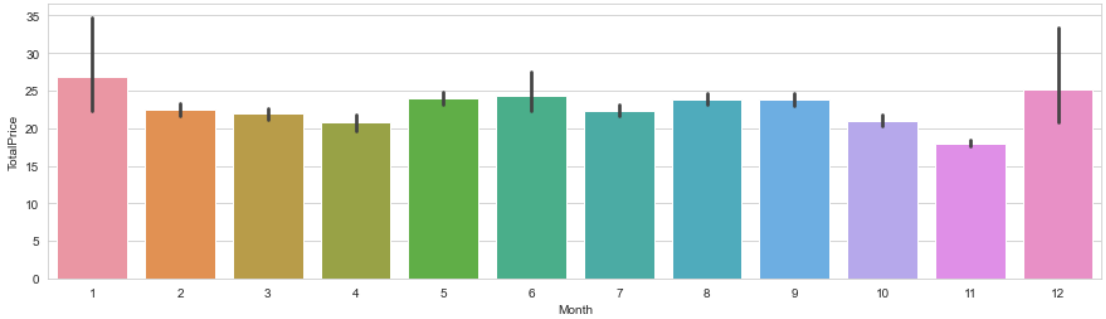

Chart 3. Total amount of sales per month

Figure 39 shows the columns from the original dataset, to which the classifiers will be applied in the next processing step.

2.2. Implementation of Classifiers

	Quantity	UnitPrice	TotalPrice	Year	Month	Country
0	6	2.55	15.30	2010	12	United Kingdom
1	6	3.39	20.34	2010	12	United Kingdom
2	8	2.75	22.00	2010	12	United Kingdom
3	6	3.39	20.34	2010	12	United Kingdom
4	6	3.39	20.34	2010	12	United Kingdom

Figure 39. Result of applying the .info() method to the subset of the original dataframe, which will be processed by the classifiers

2.2.1. Decision tree

The results of the Decision tree classifier were obtained by applying 2 to 10 folds. The results are displayed in the Figure 40

```
Decision Tree
2 cross-fold validation,Result: 0.88944
3 cross-fold validation,Result: 0.89045
4 cross-fold validation,Result: 0.89058
5 cross-fold validation,Result: 0.89068
6 cross-fold validation,Result: 0.89081
7 cross-fold validation,Result: 0.89089
8 cross-fold validation,Result: 0.89068
9 cross-fold validation,Result: 0.8908
10 cross-fold validation,Result: 0.89082
```

Figure 40. Decision tree classifier results with different number of folds

Figure 41 shows the results of the LDA, KNN, CART and Naïve Bayes classifiers. In the case of the KNN classifier, the execution parameters were 10 folds.

```
LDA: 0.889948 (0.001464)
KNN: 0.867954 (0.004955)
CART: 0.888244 (0.001546)
NB: 0.041761 (0.002241)
```

Figure 41. Execution results of LDA, KNN, CART and Naïve Bayes classifiers.

Especially for the KNN classifier was performed additionally with percentage split 67% training set (Figure 42.)

```
K-Nearest Neighbor
Percentage split 50%
Accuracy: 87.98041599999999 % stdev: 0.2041359056923518
K-Nearest Neighbor
Percentage split 67%
Accuracy: 88.081827 % stdev: 0.2277964187587836
K-Nearest Neighbor
Percentage split 70%
Accuracy: 87.88851100000001 % stdev: 0.2560936368598203
K-Nearest Neighbor
Percentage split 80%
Accuracy: 88.042022 % stdev: 0.2933438976145654
K-Nearest Neighbor
Percentage split 90%
Accuracy: 87.925809 % stdev: 0.30289663470773937
```

Figure 42. Result of execution of the KNN classifier with percentage split 50%, 67%, 70%, 80% and 90% training set

Chart 4 shows the comparative correctness results of the classifiers applied to the data set. With the exception of the NB classifier, the other classifiers behaved similarly and the correctness results were very close to each other.

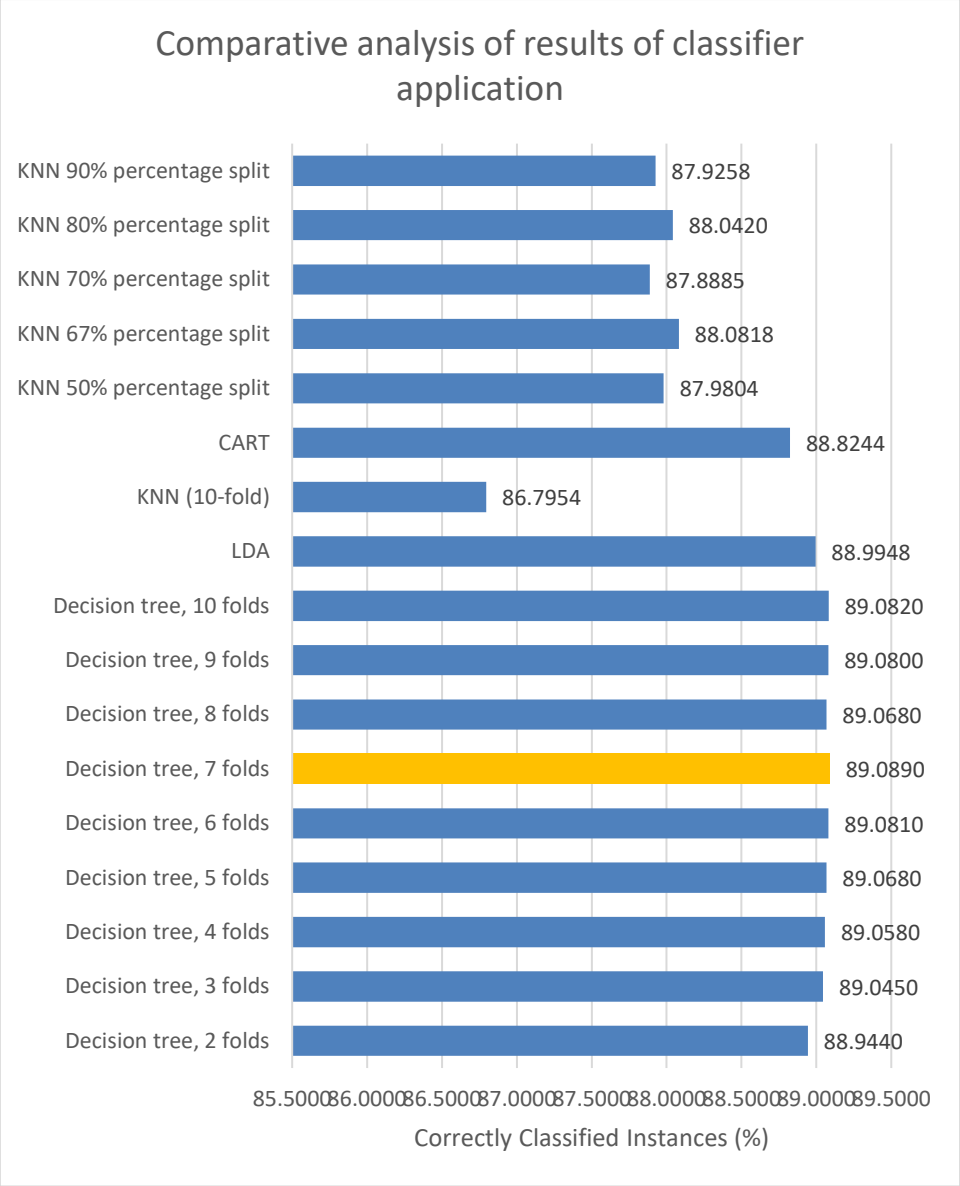

Chart 4. Comparative analysis of classifier application results

3. Dataset 3: Online Shoppers' Purchasing Intention

3.1. Dataset Description

The first set of data we will study is called "Online Shopper's Purchasing Intentions" and comes from the UCI Machine Learning Repository[1] (Sakar, et al., 2019). Contains data related to internet browsing and online shopping. The data was retrieved over a period of one year and each record concerns a session of a different user. The data set contains a total of 12,330 records and 18 variables. The 10 variables are numeric and the 8 are nominal. In the next table, we describe the data set variables:

Table 11. Online Shoppers Purchasing Intention data set variables

A/A	Name	Type	Range
1	Administrative	Numeric	0 - 27
2	Administrative_Duration	Numeric	0 - 3398.75
3	Informational	Numeric	0 - 24
4	Informational_Duration	Numeric	0 - 2549.375
5	ProductRelated	Numeric	0 - 705
6	ProductRelated_Duration	Numeric	0 - 63973.52
7	BounceRates	Numeric	0 - 0.2
8	ExitRates	Numeric	0 – 0.2
9	PageValues	Numeric	0 – 361.76
10	SpecialDay	Numeric	0 - 1
11	Month	Nominal	Feb - Dec
12	OperatingSystems	Nominal	1 – 8
12	Browser	Nominal	1 – 13
14	Region	Nominal	1 – 9

[1]Source:
https://archive.ics.uci.edu/ml/datasets/Online+Shoppers+Purchasing+Intention+Dataset

15	TrafficType	Nominal	1 – 20
16	VisitorType	Nominal	Returning_Visitor, New_Visitor, Other
17	Weekend	Nominal	TRUE, FALSE
18	Revenue	Nominal	TRUE, FALSE

The following is a description of each variable:

Administrative: the number of web pages that the user visited in the specific session and were related to the management of his account.

Administrative_Duration: the total time the user spent on the Administrative category websites.

Informational: the number of websites that the user visited in the specific session and were related to information i.e. an online store.

Informational_Duration: the total time the user spent on the Informational category websites.

ProductRelated: the number of websites that the user visited in the specific session and were related to products.

ProductRelated_Duration: the total time the user spent on the ProductRelated category websites.

BounceRates: the percentage of visitors who visit a site and leave immediately after not navigating further on that site.

ExitRates: the percentage of visitors who leave a site while they are on a particular web page.

PageValues: the average "value" of the websites the user visited before making an online purchase.

SpecialDay: a numeric value that represents how far a particular, important day or holiday (for example, Christmas) was from the specific time of recording.

Month: the month in which the specific registration was recorded.

OperatingSystems: the type of Operating System of the user's device.

Browser: the type of web browser of the user's device.

Region: the geographical area of the recording of the specific registration.

TrafficType: the way (type of movement) in which the user performed the specific session.

VisitorType: the type of user, i.e. whether it is a new visitor or not.

Weekend: whether the time of the session coincides with a weekend or not.

Revenue: concerns whether the user made an online purchase or not.

The variable "Revenue" is the dependent variable of the data set and will have the role of "class" in the categorization models that we will create. That is, we will predict whether the user belongs to the True or False class, ie whether he will eventually make an online purchase or not, taking into account the other (independent) variables.

3.2. Visualization of the dataset

Then we will explore the data set through various graphs, in order to better understand it. Charts were created using the Python programming language.

In the next Figure, we show the number of users per class, ie those who did not make an online purchase and those who made one. We notice that there are many more users who did not make a purchase, so there is an imbalance in the distribution of class prices in our datase.

Figure 43. Illustration of the "Revenue" variable

The next figure shows the number of observations per class, along with the "Administrative_Duration" column.

Figure 44. Display of "Revenue" and "Administrative_Duration" variables

In the next Figure, we display the number of observations per class, along with the "Informational" column.

Figure 45. Illustration of "Revenue" and "Informational" variables

In the next image, we display the number of comments per class, along with the "ProductRelated_Duration" column.

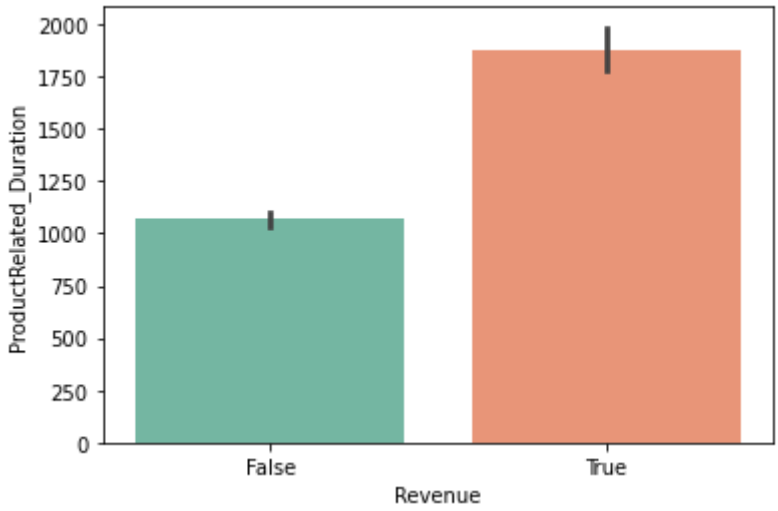

Figure 46. Display of "Revenue" and "ProductRelated_Duration" variables

We observe that users who eventually make online purchases spend more time viewing web pages related to management functions, and visit more pages with general information and pages related to products.

In the next figure, we show the number of online sessions and the time that took place while the color represents the class, i.e. whether a purchase was made or not.

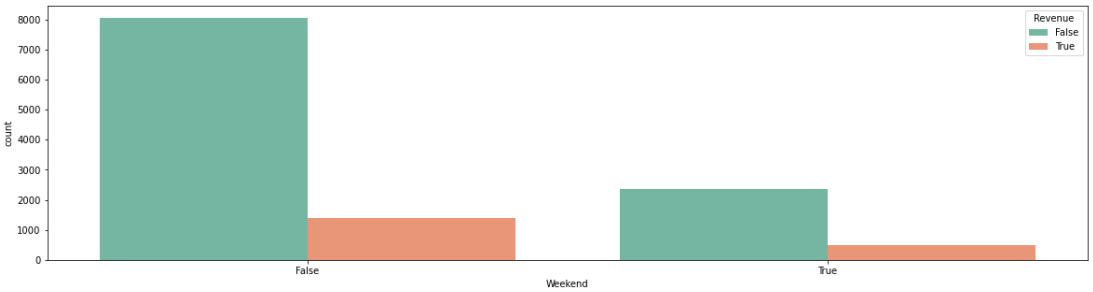

Figure 47. Illustration of "Weekend" and "Revenue" variables

We notice that more visits are recorded but also shopping on weekdays and not on weekends.

In the next image, we display the number of users per TrafficType and in a different color for each class.

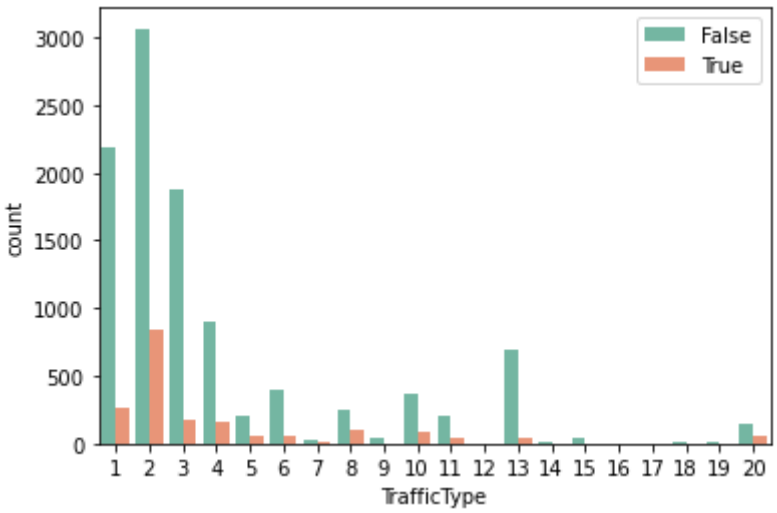

Figure 48. Display of "TrafficType" and "Revenue" variables

We observe that the largest number of visits that either end up in the market or not, is made by users with TrafficType = 2, followed by users with TraffcType types 1 and 3. Assuming that the type TrafficType 1 corresponds to users who directly visit an online store (type "Direct Traffic"), while types 2 and 3 correspond to those who visit the online store through a link ("Referral Traffic") or through a search engine ("Organic Traffic") (Longden, 2020), then we can say that users who visit an online store in an "indirect" way are the ones who are ultimately more likely to make a purchase.

In the next Figure, we show the number of users using each type of browser, and the class.

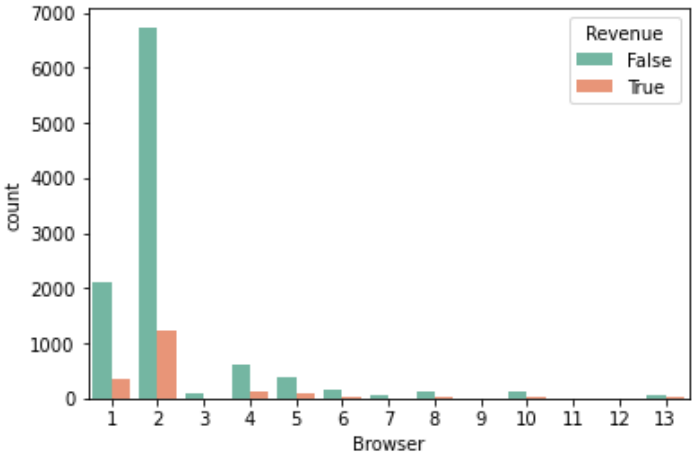

Figure 49. Illustration of "Browser" and "Revenue" variables

We notice that the type 2 browser is the most popular in general.

In the next image, we represent the number of users per month, and the class.

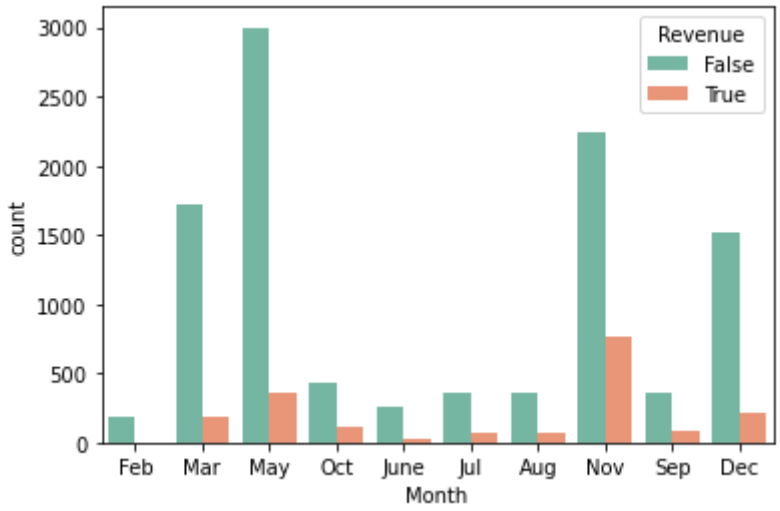

Figure 50. Illustration of "Month" and "Revenue" variables

We observe that most of the visits that end in a market, take place in the month of November, while those that do not end in a market take place in the month of May.

In the next figure we show the number of users per Operating System type that they use in their device.

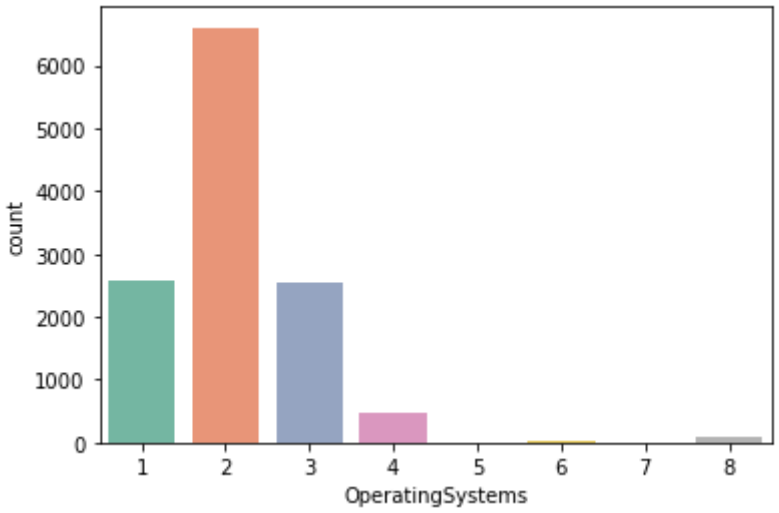

Figure 51. Illustration of variable "OperatingSystems"

We observe that the most popular Operating System is the one that has type 2.

Finally, in the next image we show the number of users per visitor category, and the class.

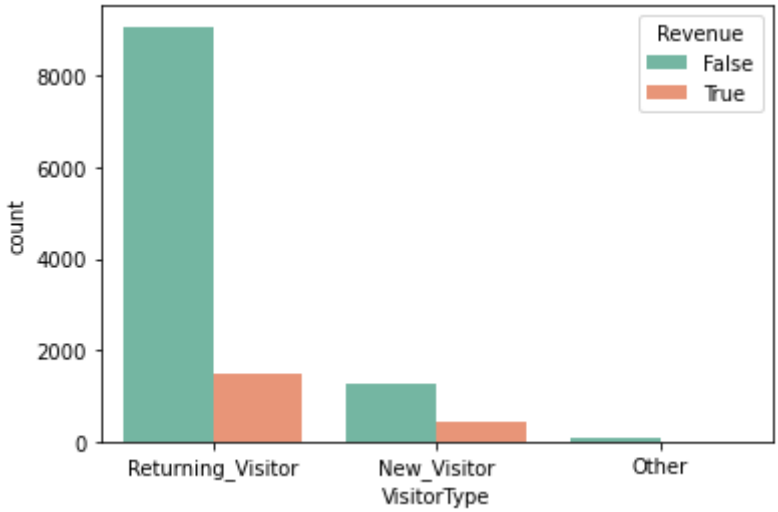

Figure 52 Display of the "VisitorType" and "Revenue" variables

We notice that most visits and purchases are made by those who have visited this online store in the past. They are more likely to make a purchase.

3.3. Pre-processing the dataset

In this chapter, we will pre-process our data set so that we can classify it. The WEKA software tool was used for the pre-processing (but also for the classification).

First, we load our data set into the WEKA software.

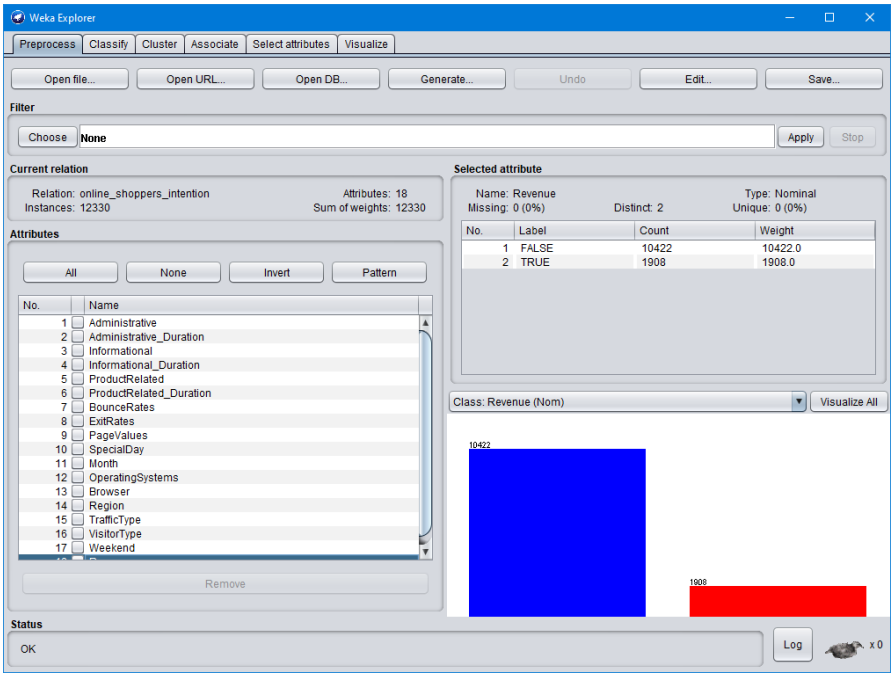

Figure 53. Loading data set into WEKA

Through the display of the class, we see the number of users who did not make a purchase (10422) compared to those who made a purchase (1908), in a total of 12330 users.

By checking the variables one by one, we observe that the data set does not contain empty or incomplete values. It also does not contain duplicate entries.

3.3.1. Convert variables to nominal ones

A subset of variables are essentially nominal, but because they have distinct numeric values, WEKA considers them to be numeric. Therefore, we will convert them to nominal ones. These variables are "OperatingSystems", "Browser", "Region", and "TrafficType". In the image below, we can see the "OperatingSystems" variable, which has 8 distinct values but is recognized as numeric:

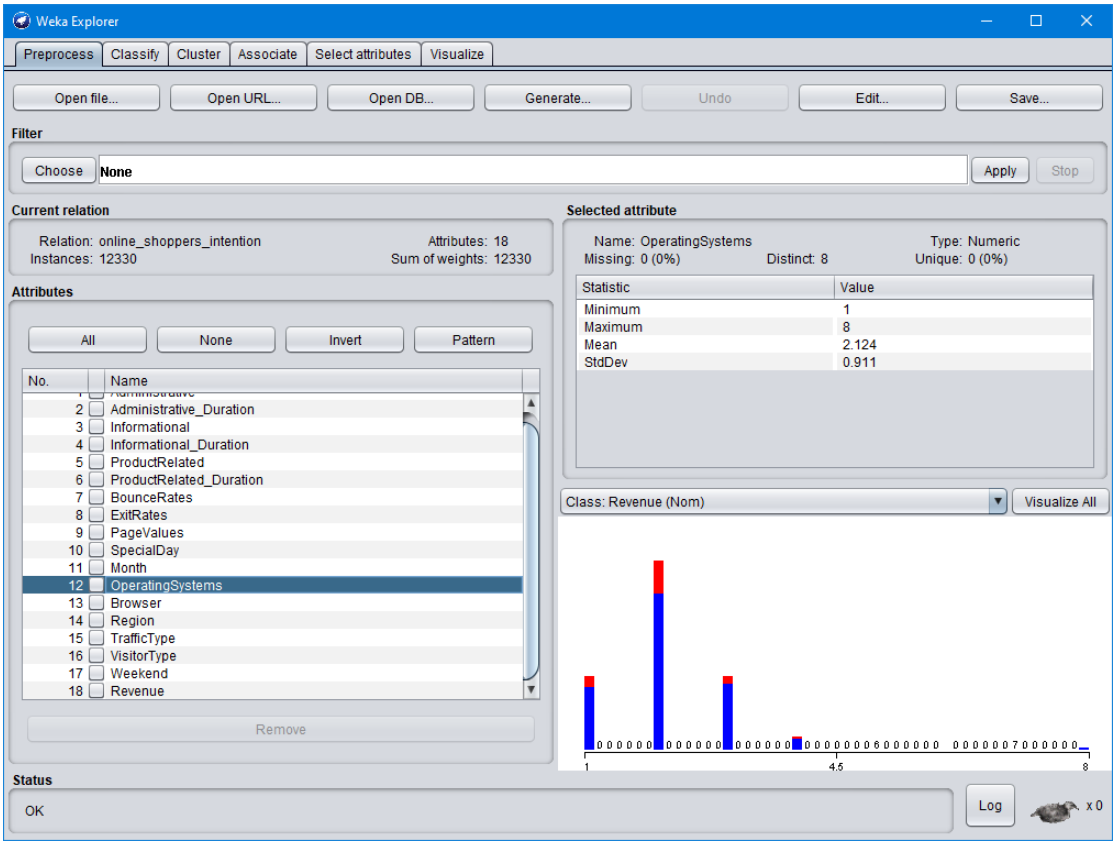

Figure 54. The "OperatingSystems" variable before converting to nominal

Applying the **unsupervised.attribute.NumericToNominal** filter to these four variables, we change their type to nominal. In the following figure, we can see the OperatingSystems variable after the application of the filter.

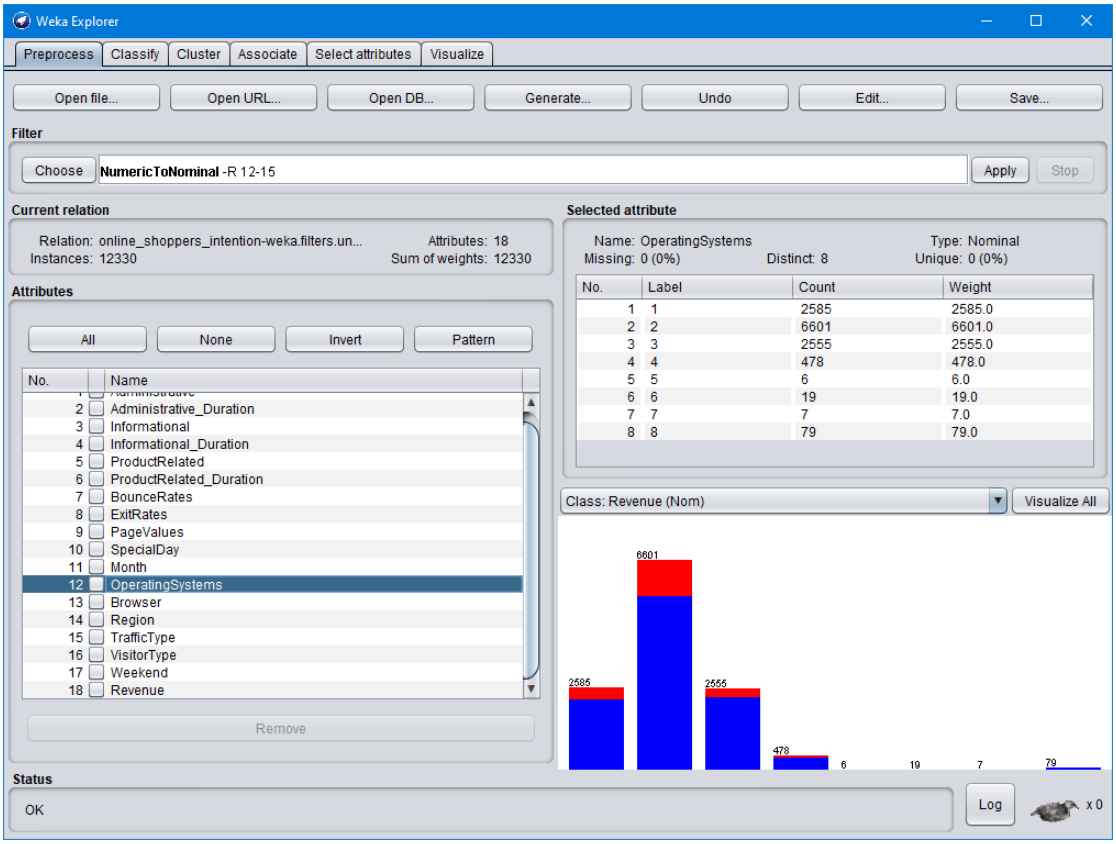

Figure 55. The "OperatingSystems" variable after conversion to nominal

3.3.2. Binary coding of nominal variables

The values of the nominal variables of our data set do not have any ranking. However, because in most cases they are expressed as numbers, the classification algorithm is likely to give different weight to specific values of the variable, which may adversely affect the performance of the algorithm. To avoid this, we will apply binary coding to the named variables ("One-hot-encoding" technique). After applying this technique, as many new variables (columns) will be created as the values of each nominal variable and each of them will have a value of 0 or 1, depending on whether the variable for the specific record has the value of this column or not. For example, for the variable "OperatingSystems" that has 8 distinct values, after applying the technique, 8 new variables with value 0 or 1 will be created. If, for example, in a specific record, the variable has a value equal to 2, then 8 new binary values will be created, the column for the value 2 of the variable will have a value of 1 and the rest will have a value of 0.

To apply this technique, select the **unsupervised.attribute.NominalToBinary** filter and apply it to all named variables in our dataset (except the Revenue variable). Even those that have alphanumeric values, such as the Month variable, binary coding is likely to provide better performance, especially in algorithms that can not work with nominal

variables. The next image shows the data set after applying the filter. We notice that now there are a total of 75 variables, from 18 that existed initially.

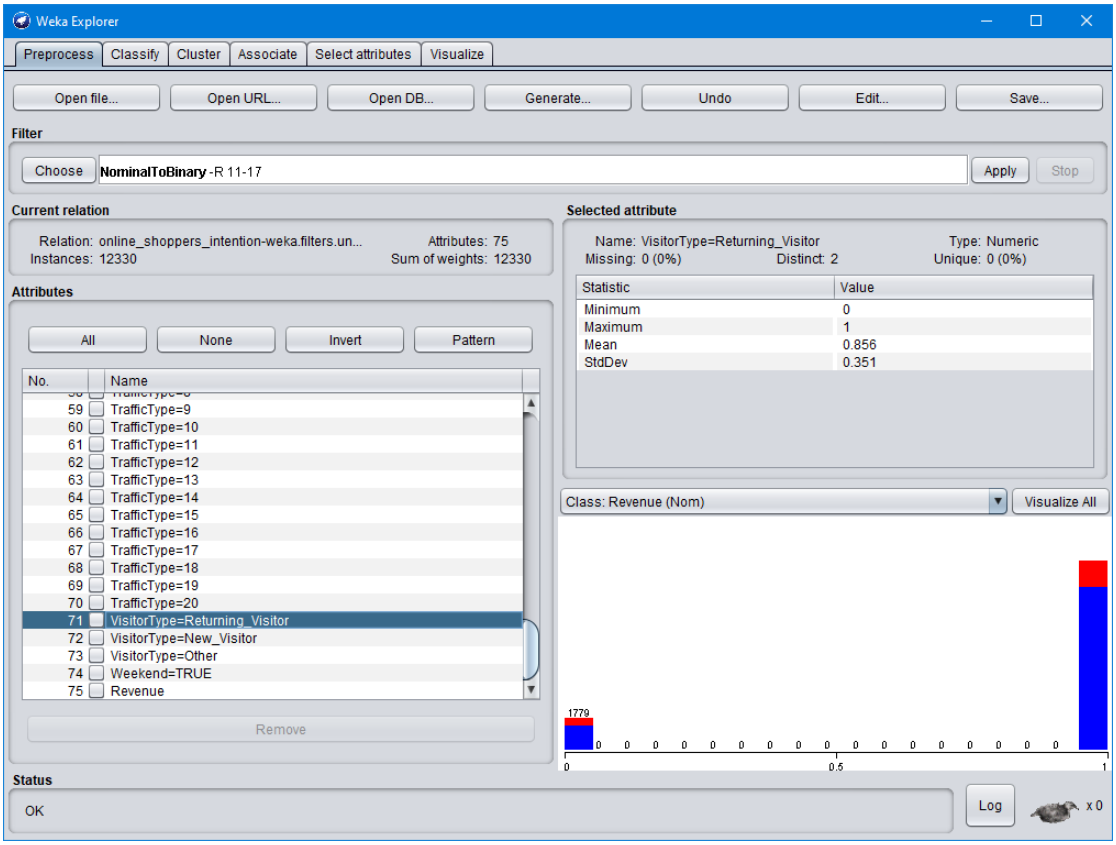

Figure 56. The data set after applying the "NominalToBinary" filter

3.3.3. Outlier detections and deletion

The next important step in pre-processing our data set is to locate and delete the outliers in the numerical variables. To perform this action, we apply the **unsupervised.attribute.InterquartileRange** filter to the numeric values. After applying the filter, two new variables are created regarding whether the specific record contains a value that is considered outlier or generally very wide. We are interested in deleting those entries that contain outliers. In the next image, we see the data set after applying the filter.

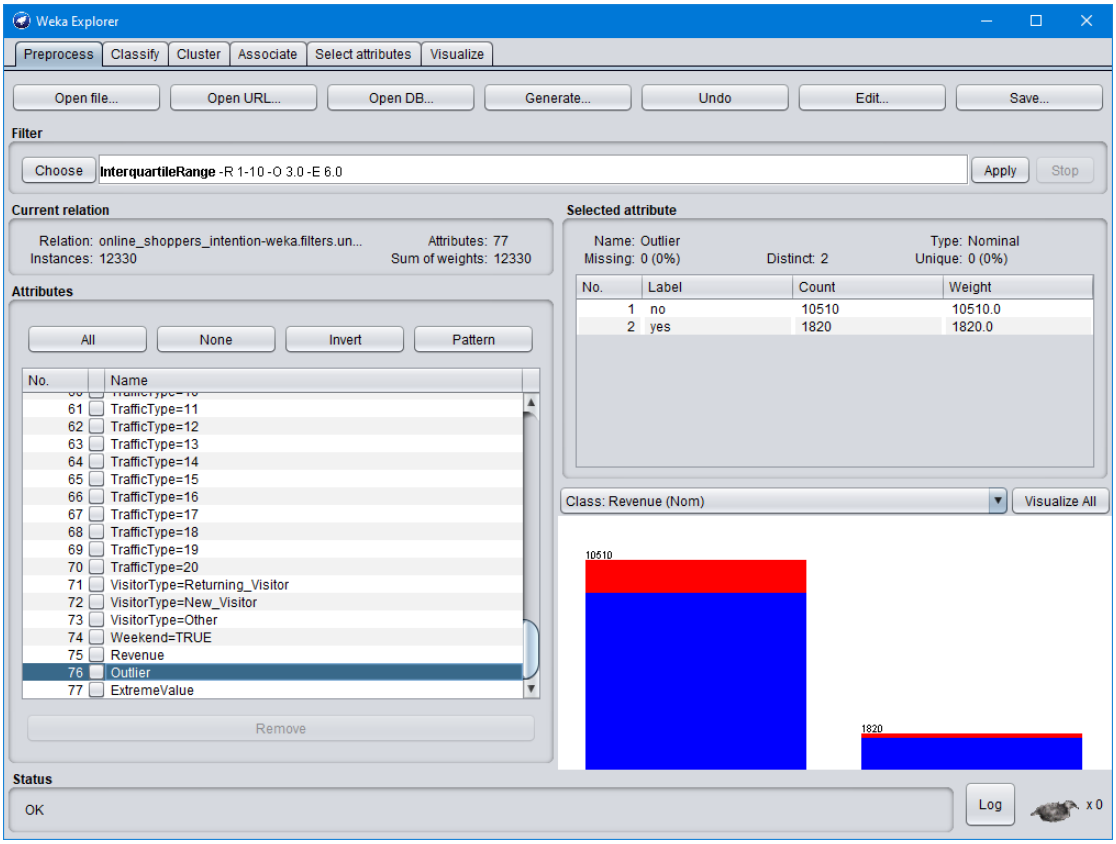

Figure 57. The data set after the application of the "InterquartileRange" filter

We observe that 1820 records have been identified as outliers

Next, we remove these records from the data set by applying the "unsupervised.instance.RemoveWithValues" filter to the newly created "Outlier" variable. Now our data set contains 10510 records. Finally, select and delete the additional variables "Outlier" and "ExtremeValue". The following figure shows the data set after performing the above actions.

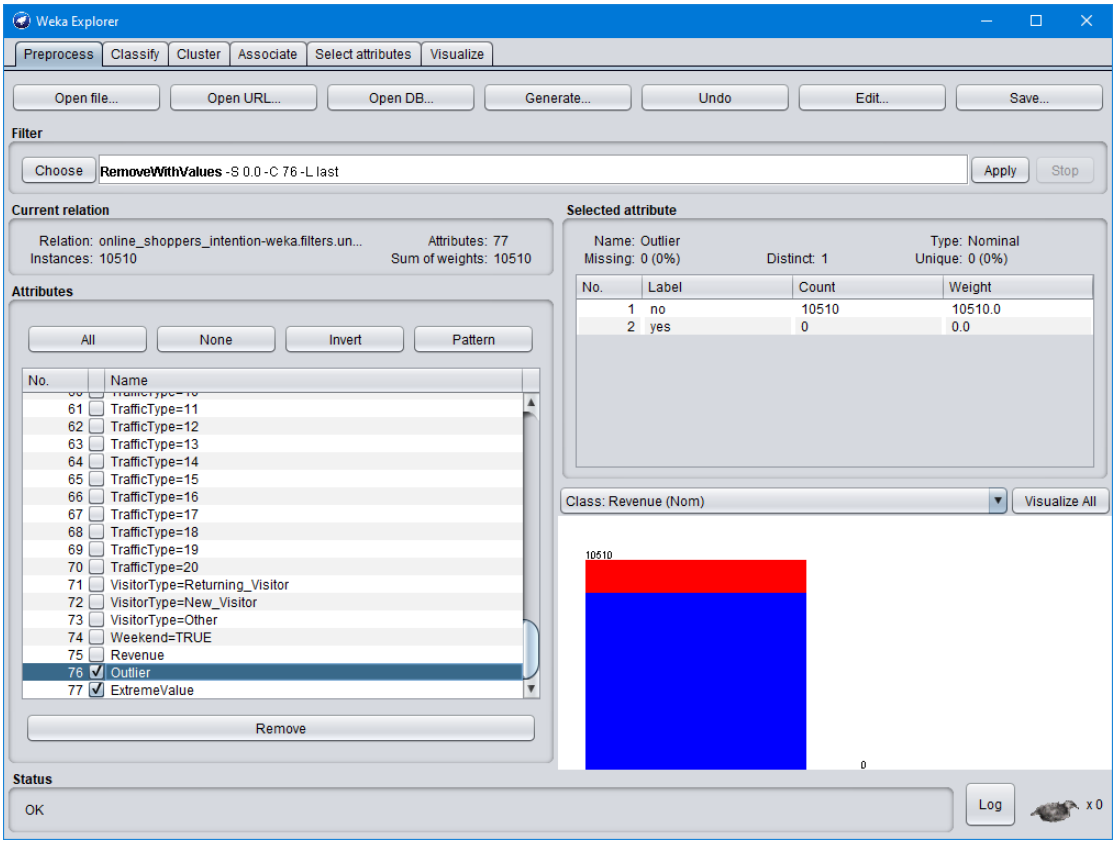

Figure 58. The data set after deleting the outliers

3.3.4. Normalization of numerical variables

In the next pre-processing step, we will normalize the numerical variables in the interval [0-1]. To do this, apply the **unsupervised.attribute.Normalize** filter. In the next image, we see the PageValues variable, which now has values in the range [0-1] from [0-361.76] that it originally had.

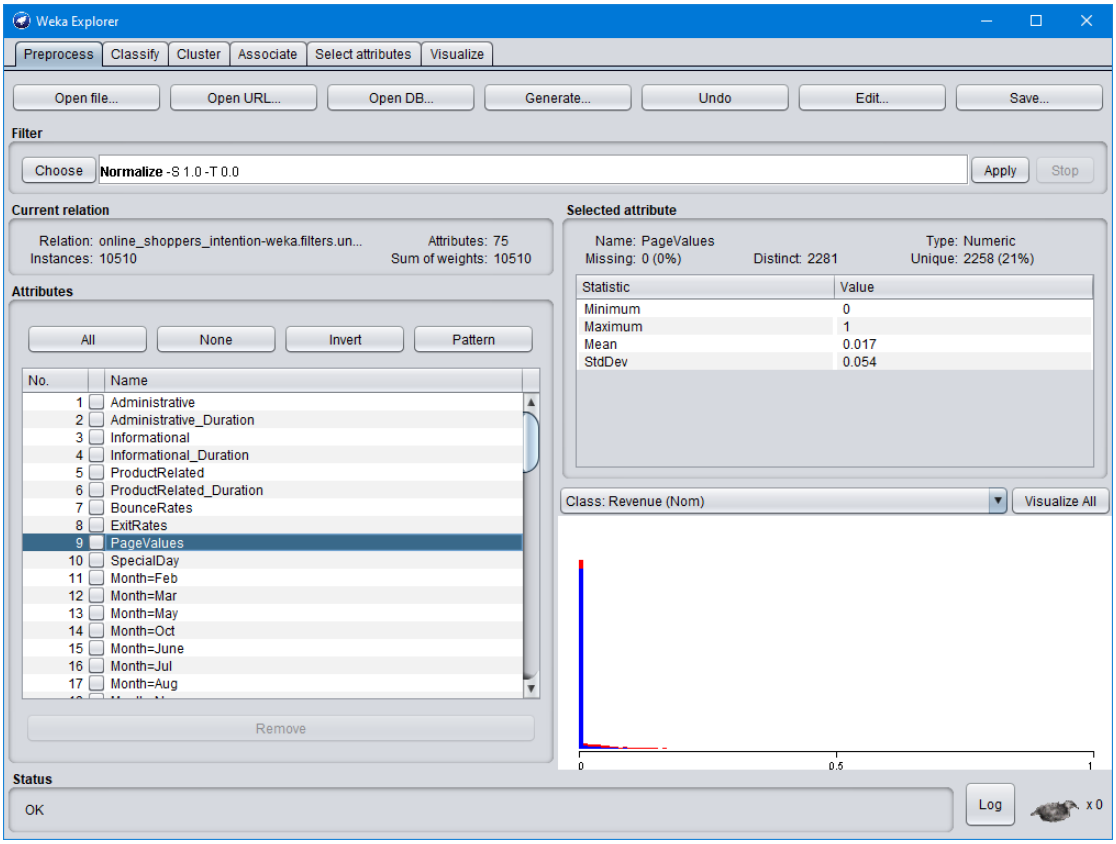

Figure 59. The data set after the Normalize filter is applied

Now, all the variables in our data set have a value of [0-1] and at this point we have successfully completed the pre-processing.

3.4. Execution of classification algorithms

At this point, we are ready to run classification (categorization) algorithms on our processed data set. We will perform the following algorithms:

- KNN
- Naïve Bayes
- J48
- Random Forest
- Logistic Regression

In all cases, we have defined the variable Revenue as a class and we will predict its value: TRUE if the user finally makes an online purchase while browsing, and FALSE if the user does not make an end purchase.

The stratified 10-fold cross validation technique was used to train and test the algorithms. In this technique, the data set is divided into 10 equal-sized sections, of which 9 are used for classifier training and one for testing. This process is performed

10 times (or generally as many times as we have defined through the parameter that defines the folds) and at the end the individual predictions are combined to produce the final prediction. This ensures that all data set records will be used for training, but also for testing, while each observation will only be used once for testing. The stratified method ensures that in each selection of the data set subset, the percentage representation of each class will be maintained. In our case, the TRUE class, which is a minority, will continue to be a minority in the selection of each subset. This is very important to ensure proper training and testing of each algorithm.

To evaluate the performance of the algorithms, we examined the values of the Correctly Identified Instances (CCI), Precision, Recall, and Area Under the Curve (AUC). The CCI value refers to the percentage of items that are correctly classified regardless of class. For the rest of the measures, we focus on the positive class. The Precision value refers to the percentage of correctly sorted items relative to all items sorted by the algorithm. The Recall value refers to the percentage of items that belong to the class and are sorted correctly. Finally, the value of the AUC (Area Under the ROC Curve) is a measure that expresses the ability of the classifier to classify correctly, by examining different thresholds for a particular class. The diagram of the ROC curve on the X-axis contains values that refer to the percentage of False Positives (FP rate), ie those items that were incorrectly classified in this class, and on the Y-axis the values that refer to the percentage of True Positives (TP rate) , that is, those items that were sorted correctly. When the curve is shaped so that it is closer to the Y axis, ie we generally have lower values at the FP rate than at the TP rate, then the area of the area below the curve increases. Higher AUC values indicate higher classification performance.

3.5. Execution of algorithms using all

KNN Algorithm

The K Nearest Neighbors algorithm sorts each new element into a class taking into account the values of the nearest neighbors K based on distance.

We run the KNN algorithm with different values in parameter K (1-10). The results are shown in the following diagram. We display the percentage of correctly sorted records for each run (Correctly Classified Instances - CCI), as displayed by the WEKA tool after each run of the algorithm.

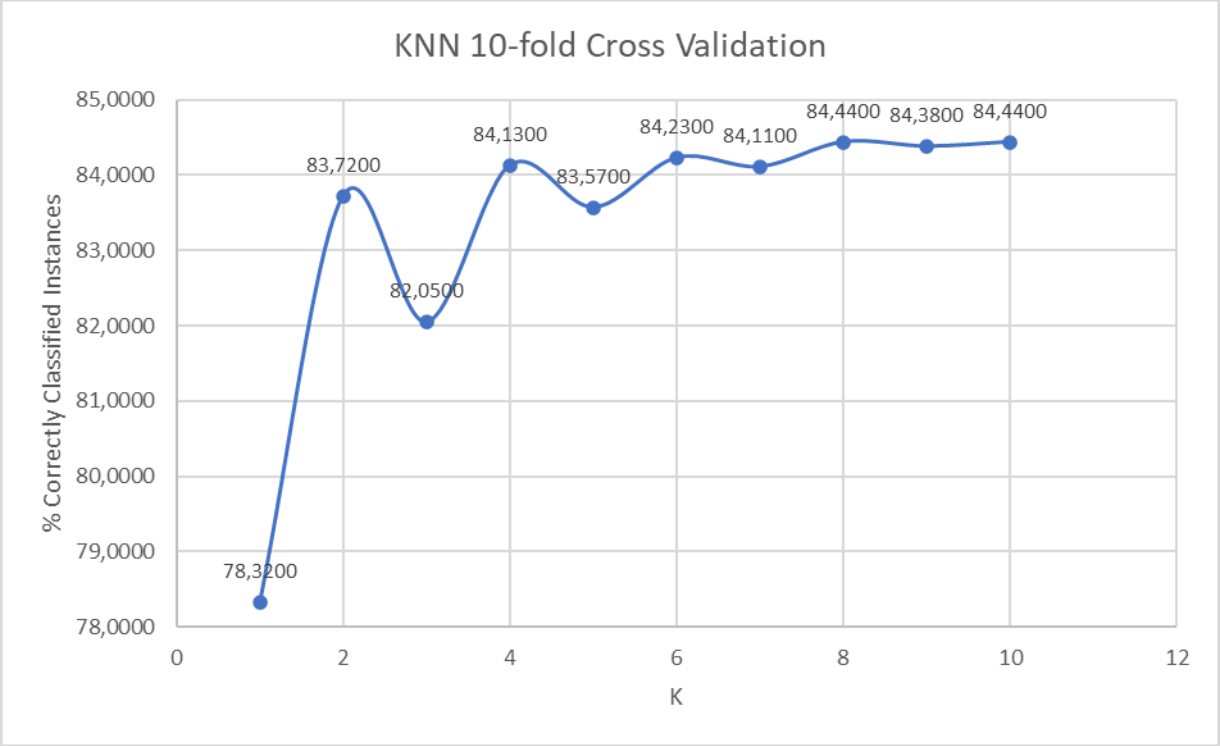

*Figure 60. Execution of the KNN algorithm for different values of the parameter **K***

We observe that for value K = 8, the algorithm shows the best performance, while for K = 9 the efficiency decreases slightly and for K = 10, the efficiency is equal to that for the value of K = 8. We will consider that for these tests, the algorithm had the best performance for K = 8, equal to 84.4%.

For K = 8, and for the positive class, we have Precision values: 58.7% and Recall: 0.59%.

Naive Bayes Algorithm

The Naïve Bayes algorithm classifies data into classes based on the probability distributions of the data. Execution of the Naïve Bayes algorithm resulted in a percentage of correctly registered entries (CCI) equal to 82.9%. Also, for the positive class it resulted in Precision values: 46.9% and Recall: 59.4%. We notice that the comments (users) belonging to the positive class, which are also the ones that interest us the most, are recognized with much greater success.

J48 Algorithm

The J48 algorithm creates a decision tree for sorting. After executing the algorithm, we resulted in a percentage of correctly classified entries (CCI) equal to 88.7%. Also, for the positive class it resulted in Precision values: 67.5% and Recall: 55.6%. We continue

to see an improvement in identifying more registrations that belong to the positive class.

Random Forest Algorithm

The Random Forest algorithm is based on the creation of multiple decision trees. The result of the classification of each tree is taken into account as a "vote" for the specific class and finally the class with the most votes is selected.

After running the algorithm, we resulted in a 90.1% percentage of correctly registered records (CCIs). Also, for the positive class it resulted in Precision values: 78.3% and Recall: 52.5%. We notice that the recall for the positive class has decreased, but the precision has increased. Therefore the classifier made fewer classification errors for the positive category.

Logistic Regression

The Accounting Regression method tries to create a linear model with appropriate coefficients, so as to separate the two classes. Classifies attributes into one of two classes by calculating the ratio of the complementary probabilities of whether an item belongs to a particular class or not. To run the algorithm, we selected the SimpleLogistic model offered by the Weka tool. After executing the algorithm, we resulted in a percentage of correctly classified records (CCI) equal to 88.4%. Also, for the positive class it resulted in Precision values: 75.9% and Recall: 40%.

The following table shows the results of the execution of each algorithm. The values Precision, Recall and AUC refer to the positive class.

Table 12. Results of execution of classification algorithms

ALGORITHM	CCI	PRECISION	RECALL	AUC
KNN (K=8)	84.4%	58.7%	0.59%	0.663
NAÏVE BAYES	82.9%	46.9%	**59.4%**	0.82
J48	88.7%	67.5%	55.6%	0.776
RANDOM FOREST	**90.1%**	**78.3%**	52.5%	**0.922**
LOGISTIC REGRESSION	88.4%	75.9%	40.0%	0.892

Conclusion: We observe that the Random Forest algorithm had the highest accuracy (CCI), the highest precision and the highest AUC for the positive class. Therefore, it made the most correct classifications and the fewest errors in classifying the positive

class observations. But the biggest recall for the positive class was the Naïve Bayes algorithm. He therefore correctly classified most of the observations belonging to the positive class.

In the next figure, we illustrate the ROC curve for running the Random Forest algorithm for the positive class.

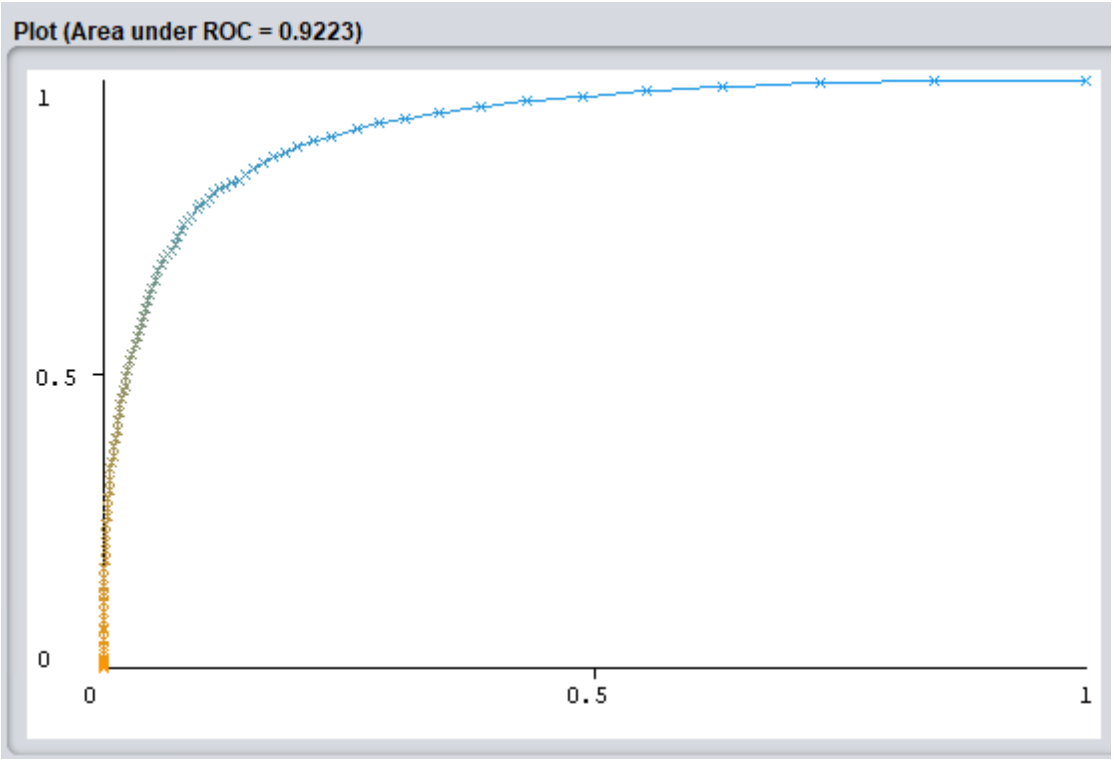

Figure 61. The ROC curve for the Random Forest algorithm

3.5.1. Execution of algorithms by selecting the best variables

At this point, we will try to improve the performance of the classification algorithms by selecting the best variables in our data set. To do this with the Weka tool, go to the "Select attributes" tab and select **CfsSubsetEval** as the evaluation method, retaining the suggested search method, "GreedyStepwise" with the suggested parameters. After performing the method, the columns are evaluated as the best:

- BounceRates
- PageValues
- Month=Nov
- Browser=3
- TrafficType=3

The following figure shows the result of selecting the best variables:

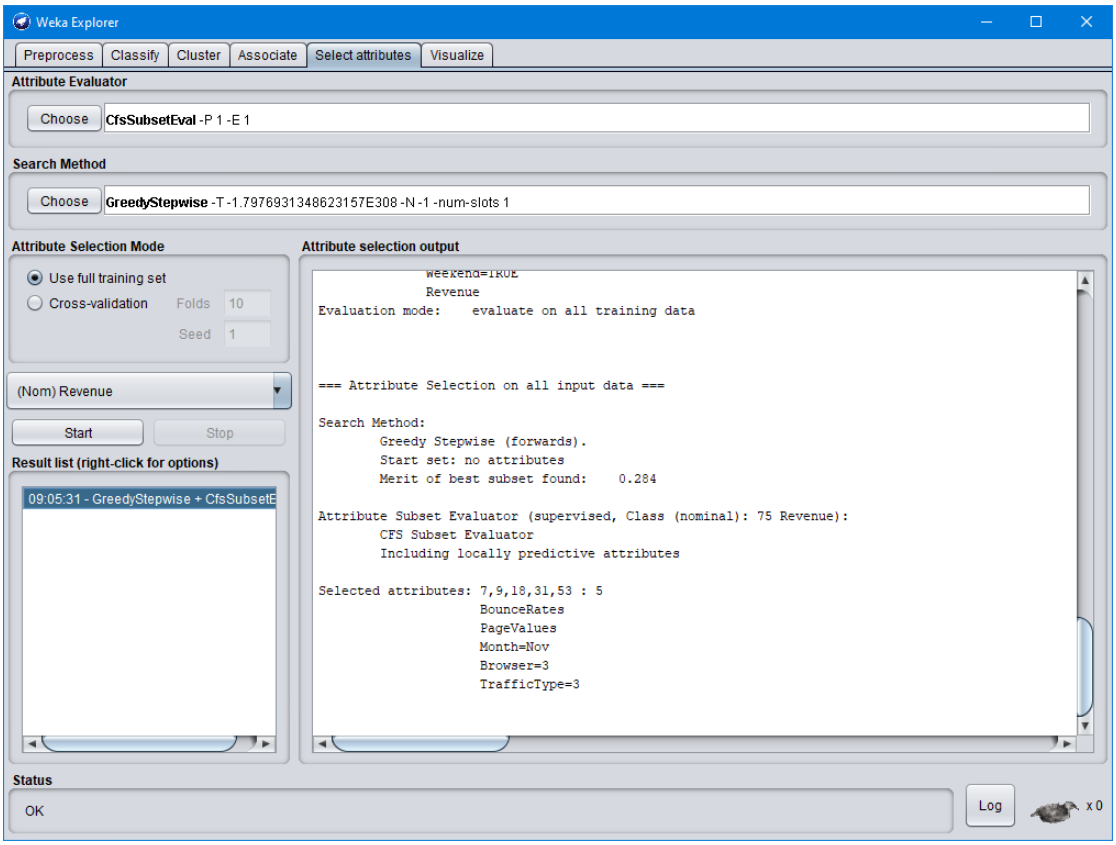

Figure 62. Selection of the best variables

Right click on the results and choose to re-save the data set file with these 5 variables. We then load it to run the same sorting algorithms again.

And in this case, we execute the KNN algorithm by selecting values of the parameter K in the interval 1-10. In the next image we show the percentage of correctly sorted records for each run of the algorithm (Correctly Classified Instances - CCI), as displayed by the WEKA tool after each run.

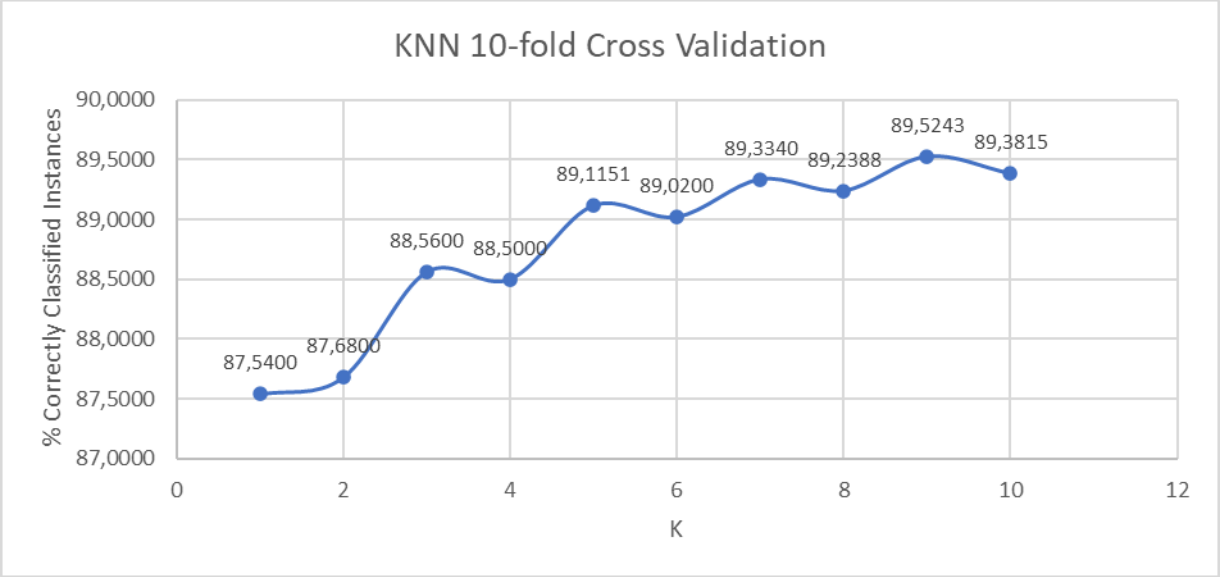

Figure 63. Execution of the KNN algorithm after selection of variables for different values of parameter ***K***

We observe that for value K = 9, the algorithm shows the best performance (89.5%).

The following table shows the results of the execution of each algorithm after selecting the best features. The values Precision, Recall and AUC refer to the positive class.

Table 13. Results of execution of classification algorithms after selection of the best variables

ALGORITHM	CCI	PRECISION	RECALL	AUC
KNN (K=9)	89.5%	70.4%	**58.3%**	0.882
NAÏVE BAYES	88.9%	75.4%	44.5%	0.854
J48	**89.6%**	71.9%	56.7%	0.855
RANDOM FOREST	87.8%	64%	53.1%	0.873
LOGISTIC REGRESSION	88.7%	**78.3%**	40.4%	**0.886**

Conclusion: We observe that after selecting the best variables, we have performance improvements on all models except the Random Forest model. The KNN algorithm in particular benefited greatly from the reduction of variables and had a vertical increase in the value of Recall. So now it successfully sorts a lot more than the positive class items. The highest accuracy (CCI) for the positive class was the algorithm J48 (decision

tree). The highest precision and the highest AUC for the positive class was the Logistic Regression model. Finally, the KNN algorithm had the largest recall for the same class.

In the next Figure, we show the ROC curve for the Logistic Regression model run.

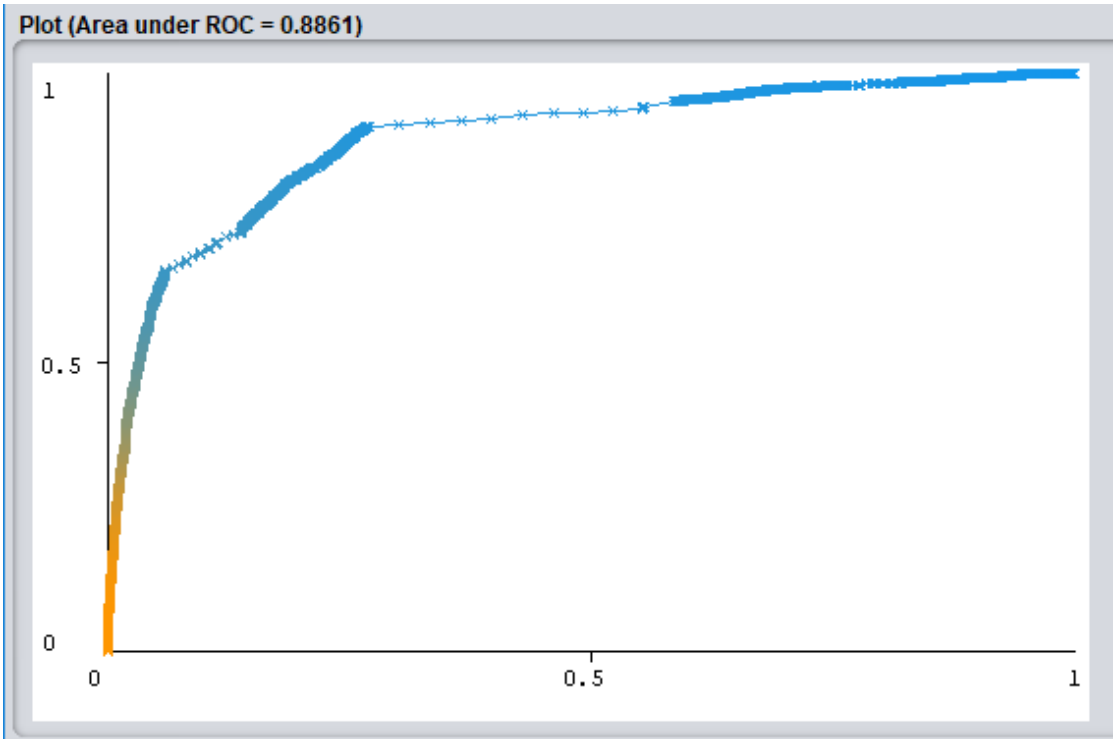

Figure 64. The ROC curve for the ***Logistic Regression*** *model*

3.6. Finding Association Rules

In an effort to better understand our data set and draw conclusions, we will run the Apriori algorithm to derive correlation rules.

First, we reload our original data set with the 18 variables. To run the Apriori algorithm we need to subtract the numeric variables from our data set. We select the numeric variables and subtract them, while converting again to nominal, those variables that have been identified as numeric (OperatingSystems, Browser, Region and TrafficType). We also subtract the Revenue variable. The following figure shows the result of this pre-processing:

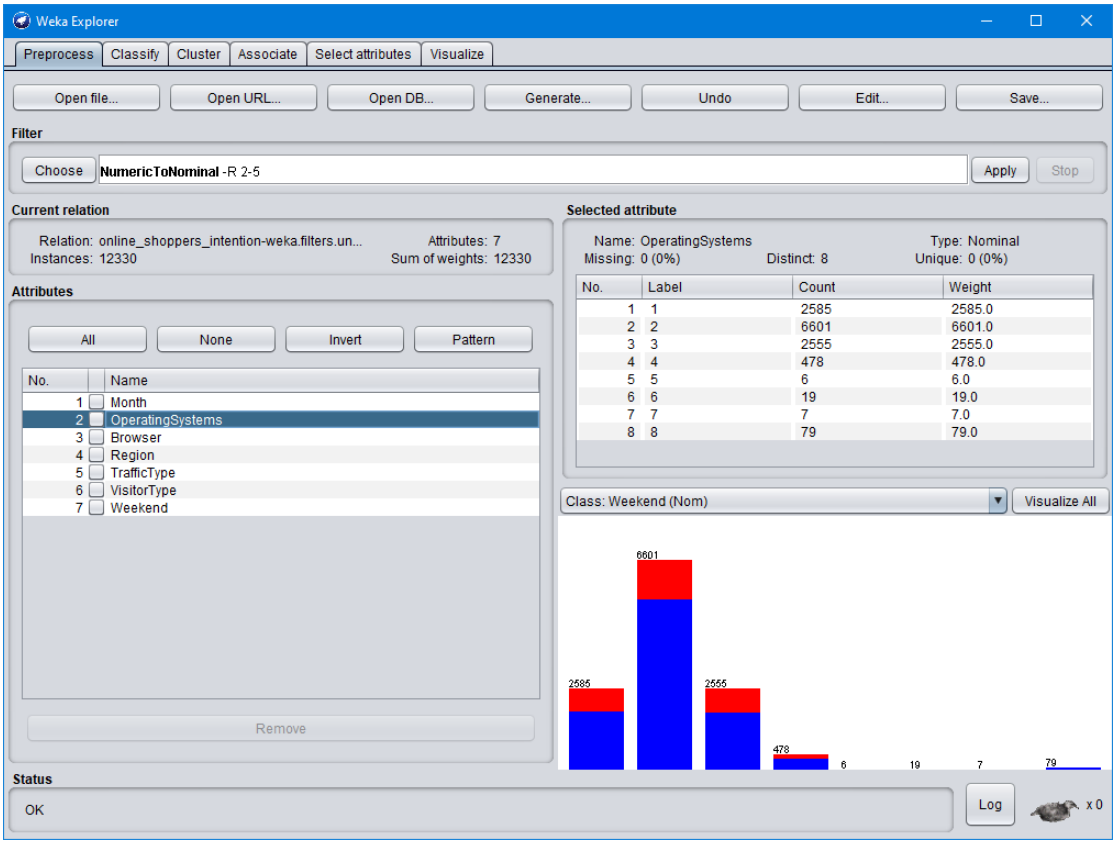

Figure 65. Pre-process the data set to execute the Apriori algorithm

Go to the "Associate" tab and run the Apriori algorithm with the default parameters. The following figure shows the results of the algorithm execution.

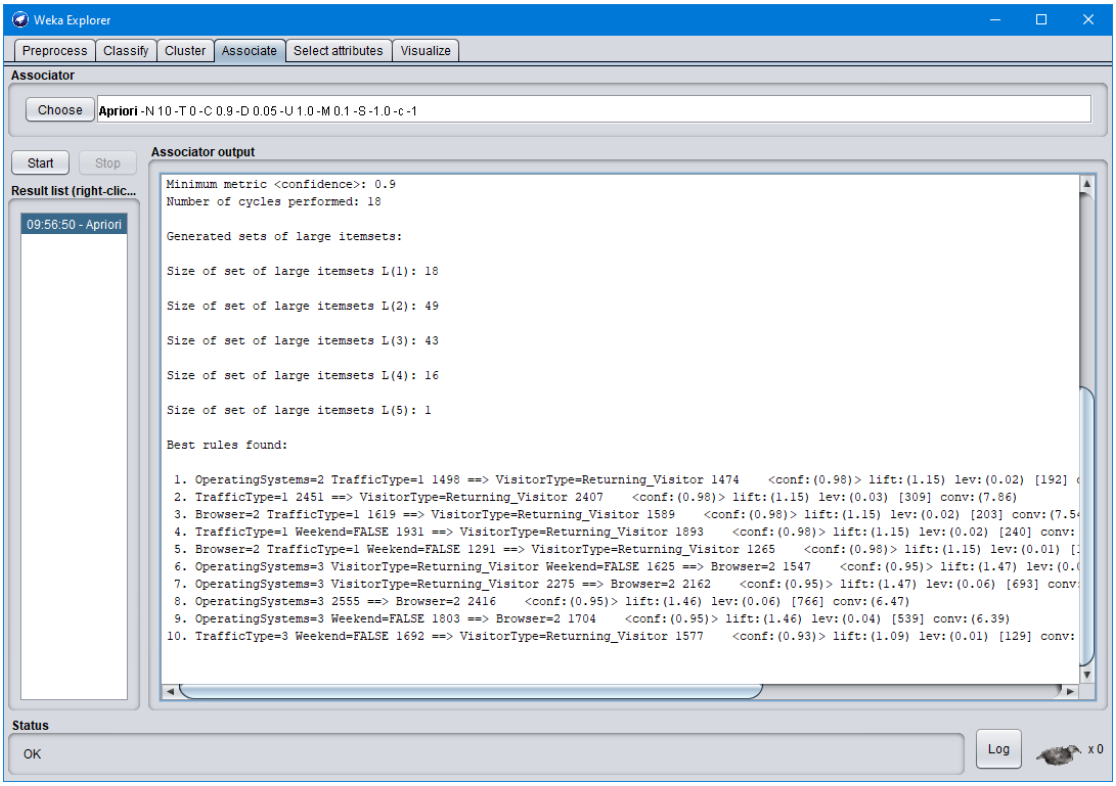

Figure 66. Results of the Apriori algorithm

The WEKA tool highlights the top ten correlation rules exported.

The three best rules are listed in the following table.

Table 14. The most popular association rules that have been exported

Rule	Case Frecuency	Support count	Confidence
OperatingSystems=2, TrafficType=1 ➔VisitorType=Returning_Visitor	1498	1474	98%
TrafficType=1, ➔VisitorType=Returning_Visitor	2451	2407	98%
Browser=2, TrafficType=1 ➔VisitorType=Returning_Visitor	1619	1589	98%

Conclusion: Given the specific rules, we could draw useful conclusions about the behavior of the internet users represented by the data set. Based on the first rule, we conclude that users with devices running the most popular type 2 operating system,

and who visit the online store in a certain way (albeit "directly"), then, tend to be visitors who have visited the specific store or stores in the past. n combination with the 2nd and 3rd rule which concern the same type of Traffic, we can "outline" the profile of the "most frequent visitor", who tends to use a specific operating system, and a web browser and visit the online store with a specific way. To visitors with this profile, for example, we can show targeted ads for an upcoming product with the expectation that as "old" visitors, they will trust us more easily and may buy the product.

4. Dataset 4: Wholesale Customers

4.1. Data set description

The second set of data we will study is called "Wholesale customers" and comes from UCI Machine Learning Repository[2] (Abreu, 2011). Contains data related to the amount spent annually in various product categories by wholesale suppliers of a supplier in Portugal. Each registration concerns a wholesale customer. The data set contains a total of 440 records and 8 variables. The 2 variables are nominal and the 6 are continuous. In the following table, we describe the description of the data set variables:

Table 15. Wholesale customers data set variables

A/A	Name	Type	Range
1	Channel	Nominal	1 - 2
2	Region	Nominal	1 - 3
3	Fresh	Continuous	3 - 112151
4	Milk	Continuous	55 - 73498
5	Grocery	Continuous	3 - 92780
6	Frozen	Continuous	25 - 60869
7	Detergents_Paper	Continuous	3 - 40827
8	Delicatessen	Continuous	3 - 47943

The following is a description of each variable:

Channel: Refers to the sector in which the wholesale customer is active (Hotel/Restaurant/Cafe or Retail).

Region: Refers to the geographical area in which the customer's registered office is located.

Fresh: Refers to the annual amount (expressed in monetary terms) that the customer spends on fresh produce..

[2] Source: https://archive.ics.uci.edu/ml/datasets/wholesale+customers

Milk: Refers to the annual amount (expressed in monetary terms) that the customer spends on dairy products..

Grocery: Refers to the annual amount (expressed in monetary terms) which the customer spends on grocery products.

Frozen: It refers to the annual amount (expressed in monetary terms) that the customer spends on frozen products.

Detergents_Paper: Refers to the annual amount (expressed in monetary terms) that the customer spends on detergents and stationery.

Delicatessen: Refers to the annual amount (expressed in monetary terms) that the customer spends on delicatessen products.

In the data set we will perform clustering. In other words, we will try to identify customer groups with common characteristics.

4.2. Data set visualization

In the next figure, we display the values of each variable. We have excluded the nominal variables.

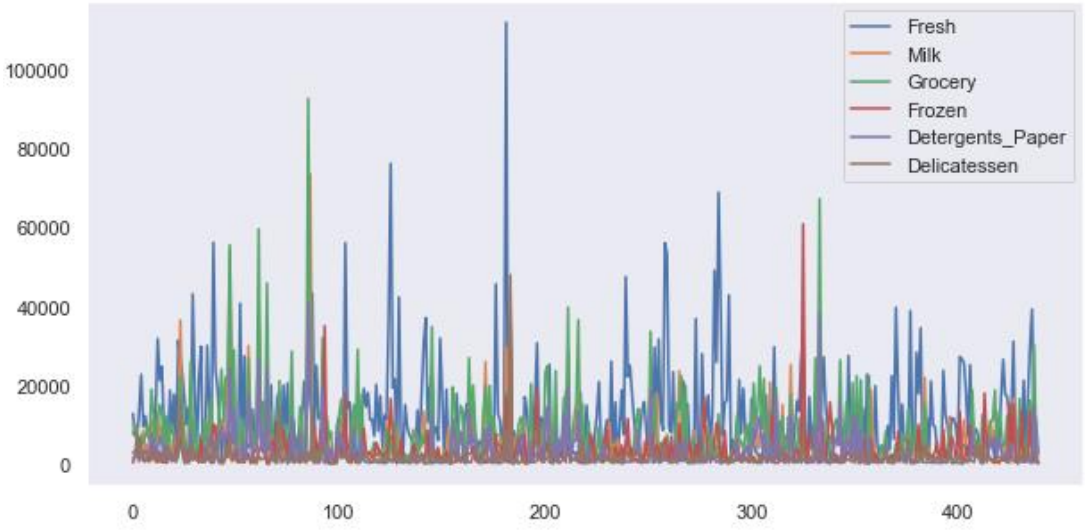

Figure 67. The annual amounts spent by wholesale customers

We observe that for the category Fresh (fresh products), the highest amount recorded has been spent, while in general the variable records large amounts. It is followed by the category of grocery products, which also records large amounts. After all, the demand for this type of product is always higher, so this trend is to be expected. On the other hand, for delicatessen type products, low amounts are spent, therefore, these products have a low demand.

4.3. Data set Pre-processing

In this chapter, we will deal with the pre-processing of our data set so that we can do clustering. The processing and clustering of the data set was performed using the Python programming language.

First, we load the data set. The next image shows the first five rows.

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

Figure 68. The first five rows of the data set

Checking the values of each variable, we find that there are no empty or missing values. Also, the data set does not contain duplications.

4.3.1. Subtraction of nominal variables

To perform the clustering, we choose to subtract the two nominal variables of the data set, namely the Channel and Region variables. We will focus on numerical values. The following figure shows the first five records of the data set after subtracting the nominal variables. Now our data set has 6 variables.

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	12669	9656	7561	214	2674	1338
1	7057	9810	9568	1762	3293	1776
2	6353	8808	7684	2405	3516	7844
3	13265	1196	4221	6404	507	1788
4	22615	5410	7198	3915	1777	5185

Figure 69. The data set after the subtraction of the nominal variables

4.3.2. Dimensionality Reduction with the Principal Components Analysis method

Prior to clustering, we will apply dimensionality reduction the size of the data set so that we can keep most of the data variation, but by using smaller dimensions. This will make the clustering problem easier for the algorithm. For this purpose, we will apply the method of Principal Component Analysis (PCA). Before reducing the dimensions, we must normalize our prices. This step is necessary because the projection in the new dimensions is based on the variation of the data. Therefore, if we have large differences in the range of values of the data set variables, then some variables may prevail over others and this will negatively affect the accuracy of the PCA method. During normalization, subtract the mean value of the corresponding variable from each observation and divide by the standard deviation of the variable.

Στην επόμενη εικόνα, διακρίνουμε το σύνολο δεδομένων μας μετά την εφαρμογή της τυποποίησης. Έχουμε πλέον «κεντράρει» τα δεδομένα μας.

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen
0	0.052933	0.523568	-0.041115	-0.589367	-0.043569	-0.066339
1	-0.391302	0.544458	0.170318	-0.270136	0.086407	0.089151
2	-0.447029	0.408538	-0.028157	-0.137536	0.133232	2.243293
3	0.100111	-0.624020	-0.392977	0.687144	-0.498588	0.093411
4	0.840239	-0.052396	-0.079356	0.173859	-0.231918	1.299347

Figure 70. The dataset after the implementation of Standardization

At this point, we are able to apply the PCA method. In order to find the ideal number of dimensions in which the method will display our data, we first apply the PCA method to all dimensions and create the cumulative variance graph, which shows us the variation of our data by number of dimensions. In the next figure, this line-chart is shown.

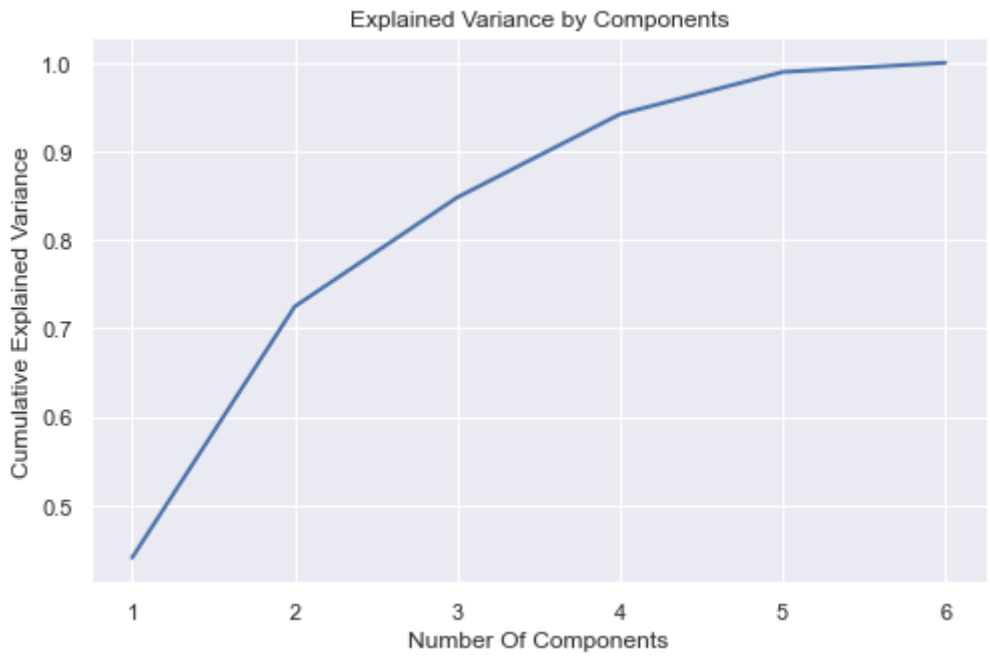

Figure 71. Cumulative variation diagram by number of dimensions

On the Y Axis we can see the variation of the data, while on the X axis the number of dimensions that we will include each time. Ideally, we want to keep the initial data fluctuation at a rate of over 90% (approximately 92% according to the diagram). The more variation we decide to keep, the more information we will keep from the original data, but of course we will have more dimensions.

In our case, we choose to maintain 4 dimensions, a number which maintains the initial data variation of more than 90%

Thus, we apply the PCA method for four dimensions. The result is the display of the original data in 4 dimensions, instead of the original 6. The first records of the result of the application of the method, are shown in the following image.

	PC1	PC2	PC3	PC4
0	0.193291	-0.305100	-0.140878	-0.486432
1	0.434420	-0.328413	0.319007	-0.178830
2	0.811143	0.815096	1.523416	-1.254082
3	-0.778648	0.652754	0.163012	0.380060
4	0.166287	1.271434	0.066279	-0.826227

Figure 72. Result of PCA method application in 4 dimensions

We will apply the K-means algorithm for the clustering of our data set, giving as input the new data set expressed in the 4 new dimensions, instead of the original ones. Random centers of gravity are initially defined for each cluster. At each step of the iterative process, each element is classified in the cluster of the center of gravity from which it is the shortest distance. When the clusters are created, the centers of gravity are recalculated based on the average of the cluster elements and the elements are re-assigned to the clusters. The algorithm terminates when the clusters remain constant or when they have changed very little.

4.4. Execution of the K-means algorithm

At this point, we will run the K-means algorithm to detect clusters in our data based on the number of features that resulted from the application of the PCA method.

First, we need to determine the number of clusters we want the algorithm to create.

To find the ideal number of clusters, we execute the algorithm many times by defining a different number of clusters and after each execution, we calculate the value of the Within Cluster Sum of Squares - WCSS. With these results, we create a diagram where we project on the X axis the number of classes in each run of the algorithm and on the Y axis the WCSS value we calculated. This method is known as the "elbow method" (Kodinariya & Makwana, 2013). The line-chart is shown in the following figure.

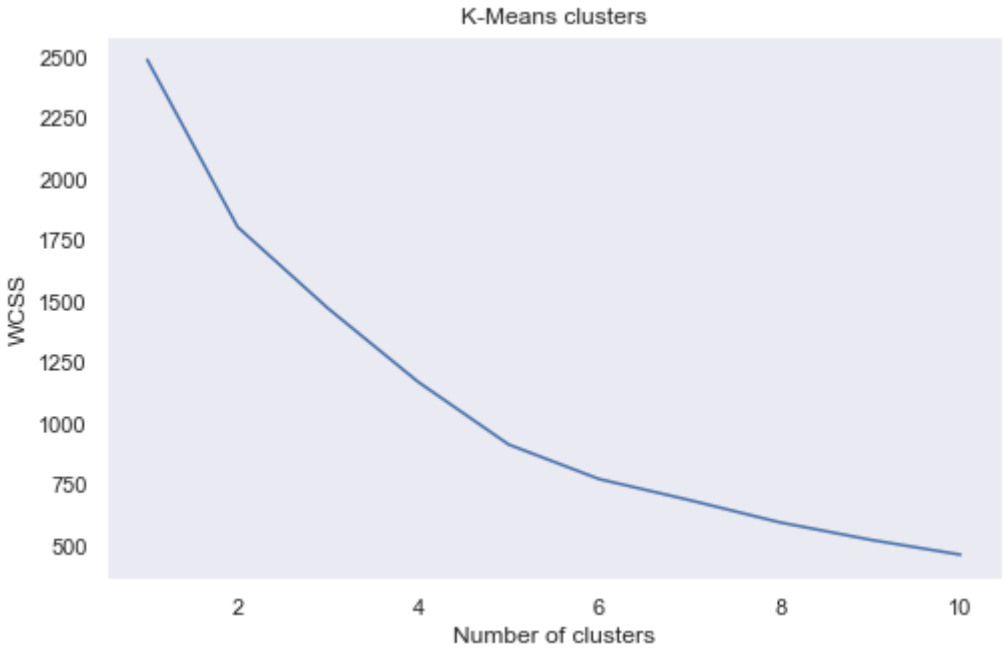

Figure 73. Number of classes in each run of the algorithm

According to the elbow method, the ideal number of classes is determined by the point at which the curve starts to become smoother from steep and thus an "elbow" is formed. In our case, we observe that this happens for a number of clusters equal to 2. Therefore, we will run the K-means algorithm with a number of clusters of 2.

After running the algorithm, we have now mapped each observation in our dataset to a cluster. The following figure shows the data set as it appears after the execution of the algorithm. In each record we have the values of the original attributes, the values of the principal components derived from the PCA, and the cluster in which the record is placed.

	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicatessen	PC1	PC2	PC3	PC4	K-means Cluster
0	12669	9656	7561	214	2674	1338	0.193291	-0.305100	-0.140878	-0.486432	Cluster 1
1	7057	9810	9568	1762	3293	1776	0.434420	-0.328413	0.319007	-0.178830	Cluster 1
2	6353	8808	7684	2405	3516	7844	0.811143	0.815096	1.523416	-1.254082	Cluster 1
3	13265	1196	4221	6404	507	1788	-0.778648	0.652754	0.163012	0.380060	Cluster 1
4	22615	5410	7198	3915	1777	5185	0.166287	1.271434	0.066279	-0.826227	Cluster 1
...	...	...	...	...	...	...	...	...	...	...	...
435	29703	12051	16027	13135	182	2204	0.870602	2.220845	-0.605500	1.049263	Cluster 1
436	39228	1431	764	4510	93	2346	-0.902520	1.676916	-1.418980	-0.572274	Cluster 1
437	14531	15488	30243	437	14841	1867	3.465704	-1.039838	-0.713161	0.033408	Cluster 2
438	10290	1981	2232	1038	168	2125	-0.918023	-0.030047	0.258408	-0.524578	Cluster 1
439	2787	1698	2510	65	477	52	-1.105137	-0.861338	0.305154	-0.114377	Cluster 1

440 rows × 11 columns

Figure 74. The data set after the application of the K-means algorithm

Finally, we highlight the two clusters that were created. We select the first two main components (PC1 and PC2), which retain the highest percentage of variability of the original data and display in these clusters that have been created.

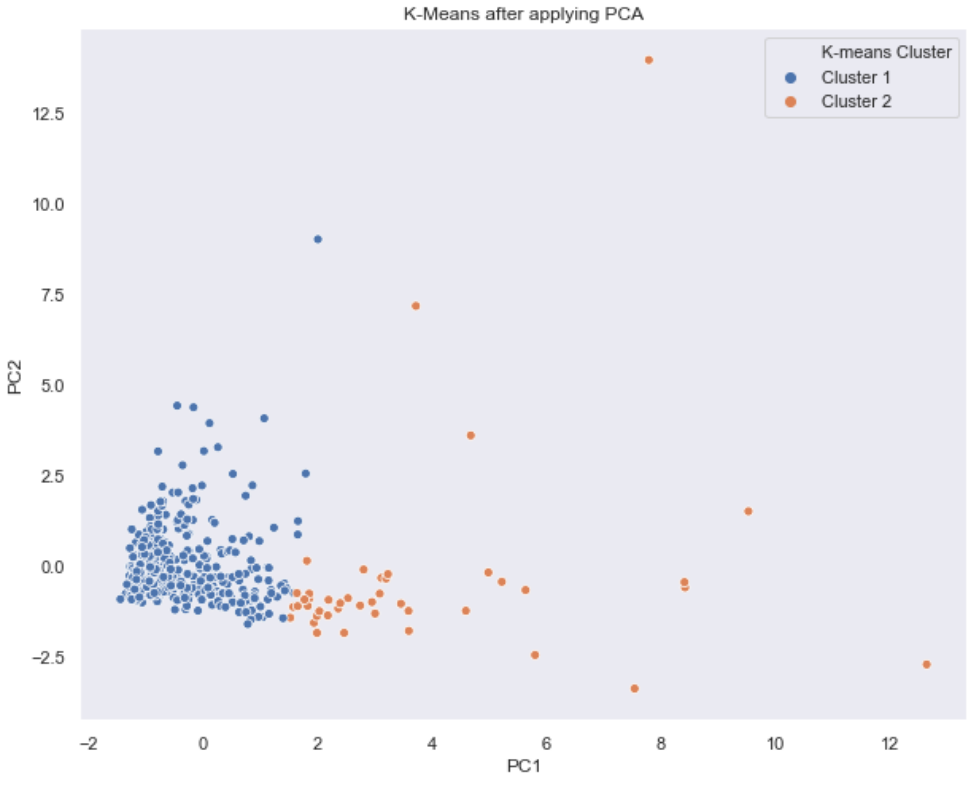

Figure 75. View clusters in the first two main components

Conclusion: We observe that the elements of the first cluster are more and with a smaller distance between them. The points of the second cluster are less and more sparse, ie they are farther apart

Bibliography

Abreu, N. G. C. F. M. d., 2011. *Análise do perfil do cliente Recheio e desenvolvimento de um sistema promocional,* Λισαβόνα: ISCTE-IUL.

Aly A. Farag, S. Y. E., 2008. *A Tutorial on Data Reduction - Linear Discriminant Analysis*. [Ηλεκτρονικό]
Available at: https://www.lsv.uni-saarland.de/fileadmin/teaching/dsp/ss15/DSP2016/matdid437773.pdf

Kodinariya, T. & Makwana, P., 2013. Review on Determining of Cluster in K-means Clustering. *International Journal of Advance Research in Computer Science and Management Studies*, pp. 90-95.

Longden, K., 2020. *What is Website Traffic?*. [Ηλεκτρονικό]
Available at: https://activeinternetmarketing.co.uk/what-are-the-different-types-of-website-traffic/

Nasreen, S., 2014. *A Survey Of Feature Selection And Feature Extraction Techniques In Machine Learning*. s.l., Science and Information (SAI).

Raschka, S., 2015. *Python Machine Learning*. Birmingham: Packt Publishing Ltd.

Sakar, C., Polat, S., Katircioglu, M. & Kastro, Y., 2019. Real-time prediction of online shoppers' purchasing intention using multilayer perceptron and LSTM recurrent neural networks. *Neural Computing and Applications*, p. 6893–6908.

Santosa, F. & Symes, W. W., 1986. Linear inversion of band-limited reflection seismograms. *SIAM Journal on Scientific and Statistical Computing,* p. 1307–1330.

Techopedia, 2021 . *Decision Table (DETAB)*. [Ηλεκτρονικό]
Available at: https://www.techopedia.com/definition/18829/decision-table-detab
[Πρόσβαση 2 March 2021].

Παπαδάκη, Μ., 2012. *Μηχανές διανυσματικής υποστήριξης (SVMs) και εφαρμογές σε πραγματικά σεισμολογικά δεδομένα,* Αθήνα: Εθνικό Μετσόβειο Πολυτεχνείο / Διπλωματική εργασία.

Πετρίδης, Δ., 2015. *Κεφάλαιο 3: Λογιστική Παλινδρόμηση*. [Ηλεκτρονικό]
Available at: https://repository.kallipos.gr/bitstream/11419/2128/1/04_chapter03.pdf
[Πρόσβαση 2 March 2021].