WIRTSCHAFTS UNIVERSITÄT WIEN VIENNA UNIVERSITY OF ECONOMICS AND BUSINESS

Seminar Paper

An Introduction to WEKA: The All-in-One Machine Learning Software in Java

Jakov Pavic

Subject: 4135 - Seminar aus BIS

Supervisor: Univ. Prof. Mag. Dr. Rony G. Flatscher

Date of Submission: 17. June 2023

Vienna University of Economics and Business,
Welthandelsplatz 1, 1020 Vienna, Austria

Abstract

WEKA is a powerful machine learning software offering an all-in-one solution. Starting from data preparation over building machine learning models to visualizing data: WEKA offers it all. The software is written in Java and can be either accessed via a Graphical User Interface (GUI) or the Application Programming Interface (API). Various expanding packages can be installed through a package manager, enlarging the possibilities even more. The goal of this paper is to give a theoretical overview to the topic of machine learning and demonstrate it with simple examples in WEKA. After reading the paper the reader should be able to compute first small machine learning projects.

Contents

Codes

List of Figures

1 Introduction

Intelligence is being explained as "the ability to learn, understand, and make judgments or have opinions that are based on reason" ("intelligence", n.d.). When speaking about intelligence most people think about human intelligence and intelligence by other living things, but since the second half of the 20th century another form of intelligence is evolving and rising quickly in power and capability: Artificial intelligence (AI), intelligence generated by machines. Alan Turing, a British mathematician and computer scientist stated in his paper in 1950 that machines could be capable of making decisions similarly to humans with the help of information and reason. He failed only because computers were very limited in power and availability at the time, but with the enhancements of computers in the following years there are no longer such limitations. Today, Alan Turing is also being referred as the father of artificial intelligence (Rockwell, 2017).

A big field of artificial intelligence deals with machine learning where experiences, in form of already existing data, and different algorithms are being used to gather new information and compute predictions (Mohri et al., 2018). There are many software tools and frameworks in various programming languages available for computing different machine learning tasks. The programming language most associated with machine learning is Python in combination with frameworks like Tensorflow, Sci-Kit learn or Keras, but this paper will show that it is and should not be only limited to one programming language by introducing WEKA.

When thinking about WEKA, a lot of Birdwatcher and New Zealand fans will probably think about the famous New Zealand bird, but in this case we are talking about the software which is offering an all-in-one implementation for machine learning in Java and WEKA stands for Waikato Environment for Knowledge Analysis. But still, there is one thing both the bird and the software have in common, which is their country of origin. WEKA, from now on only talking about the software, was developed by the University of Waikato in New Zealand in 1993 ("Weka (software)", 2023).

2 Overview: Machine Learning

As already stated above, in machine learning computers use data and algorithms to create models fitting to this data in order to gain new information and even be able to predict future outcomes. There are three main tasks in machine learning: description, prediction and causal inference. When talking about description tasks, the goal is to find patterns in the data and "describe" the data. The focus is on the past ("Difference Between Descriptive and Predictive Data Mining - Javatpoint", n.d.). In contrast to that, predictive tasks aim to combine the input to the output, such that predictions for the output can be made by having the input. In this case the focus lies on the future (Hernán et al., 2019). The third task, the causal inference aims to answer the question: What would be the outcome, if we change the input (Gonfalonieri, 2020). Furthermore, machine learning algorithms can be divided into 3 different categories: supervised- , unsupervised- and reinforcement machine learning (Jiang et al., 2020).

2.1 Supervised Machine Learning

Supervised machine learning algorithms are mainly used for prediction tasks. Based on labeled data the machine is being trained to predict outcomes. Labeled data has the characteristics that there is already an existing output for the input in the data set. For example, a data set containing pictures of dogs and cats would already tell which picture shows a dog and which a cat. Supervised machine learning algorithms can be divided into classification and regression depending on the output variable observed. If the output is categorical, such as male/female or yes/no, then classification algorithms are being used. On the other hand, if the output variable is a continuous quantity like prices are, then regression algorithms are being used ("Types of Machine Learning - Javatpoint", n.d.).

2.2 Unsupervised Machine Learning

Unsupervised machine learning algorithms use unlabeled data for description tasks. Unlabeled data does, in contrast to labeled data, not have the outcome. The data set of the cat and dog images would in this case not say whether the picture shows a dog or a cat. The three main use cases for unsupervised machine learning algorithms are: clustering, association rules and dimensionality reduction ("What is Unsupervised Learning? - IBM", n.d.).

2.2.1 Clustering

In Clustering, the goal is to group single data points into clusters with certain similar attributes. There are multiple methods to cluster: hierarchical-, partitioning-, overlapping- and probabilistic methods ("What is Unsupervised Learning? - IBM", n.d.).

A very popular method is the hierarchical one which uses a sequential approach where at the start every data point is a cluster on its own. Step by step, two clusters are being merged into one until only one cluster containing all the entries is left. Depending on the task, either the whole hierarchy can be used or one certain level which leads to a certain amount of clusters (Milligan and Cooper, 1987).

Partitioning methods are also referred as non-hierarchical clustering since these methods only create a specific amount of distinct clusters. There are different procedures for this method differing in starting point, in rules how entries are being assigned to clusters, in the way outliers are being treated and in the amount of clusters, which can either be predefined or decided by the algorithm itself (Milligan and Cooper, 1987).

What makes overlapping methods so special compared to the previous ones is the fact that data points can be assigned to multiple clusters allowing for, as the name already reveals, overlapping clusters. Those methods are significantly less used and also less algorithms support overlapping clusters (Milligan and Cooper, 1987).

Probabilistic methods allow for overlapping clusters just as overlapping methods do. The distinct feature is that in probabilistic clustering entries are being grouped determined by the probability that they are a member of a specific cluster ("What is Unsupervised Learning? - IBM", n.d.).

2.2.2 Association Rules

Association rules are looking for relations between different variables. It checks if one variable is dependent from another. For example, it can be used to analyse the buying behavior of supermarket customers. There may be an association that most people who will buy a bread, will buy milk as well. Therefore, supermarkets could put those items in the opposite corners so people who want to buy those items have to go throughout the whole supermarket increasing the chances of buying more items. In the simplest form, association rules are if-then statements looking for single cardinalities. If a costumer buys bread, then he will buy

milk too ("Association Rule Learning - Javatpoint", n.d.).

2.2.3 Dimensionality Reduction

As a general rule it can be said that the more relevant data there is available, the better the results are going to be since the machine has more data to learn from, but this does not apply always. More variables make the model more complex and require more processing power. Another difficulty when having to much variables is visualizing the data. It can quickly make a plot unreadable and therefore useless. This is why dimensionality reduction aims to reduce those dimensions without much damaging the integrity of the data ("What is Unsupervised Learning? - IBM", n.d.).

2.3 Reinforcement Machine Learning

Reinforcement machine learning is a technique following the slogan: learning by doing. An agent learns by performing actions and checking the results from those actions which can either be positive or negative. The main goal of the agent is to receive the most positive feedback. Since the agent learns by feedback, no labeled data is required similar to unsupervised machine learning techniques. This type of machine learning is especially developed for problems with sequential decision making and a long-term goal like robotics is. An example would be a chess AI where the agent tries different chess moves in order to find the one where the feedback is the best ("Reinforcement Learning Tutorial - Javatpoint", n.d.)

2.4 Semi-Supervised Learning

Semi-supervised learning is something in between supervised and unsupervised learning. It is made for data sets where only a small amount of the data is labeled and the rest not. This has the big advantage that not all the data has to be labeled, something that often has to be done manually which is very time-consuming and therefore costly. On the other hand, having completely unlabeled data restricts the possibilities by a lot ("Introduction to Semi-Supervised Learning - Javatpoint", n.d.). Semi-supervised learning aims to overcome those two problems by repeating following steps, also known as self-training: First, the small amount of labeled data is being used to train an initial model with the help of supervised machine learning. This model is then being used for the so called pseudo-labeling, where based on the model a prediction is being made for the labels of the unlabeled data. Since the

predictive power is limited, the labels are called pseudo. In the next step, the predictions with the most confident results are being added to the labeled part of the data set. This new bigger labeled data set is now again being used to create a new model, which will again be used to pseudo label unlabeled data. This process is repeated multiple times getting more and more labeled data ("Semi-Supervised Learning, Explained with Examples", 2022).

It has to be said that all those learning algorithms used in machine learning are going hand in hand rather than competing with each other. Often, unsupervised machine learning algorithms are being used to preprocess the data by detecting outliers and reducing the dimensions. The prepared data is then the input for supervised machine learning algorithms to compute prediction.

3 Overview: WEKA

Having all this information about machine learning, the software WEKA can now be introduced. WEKA is an all in one software for machine learning. It supports the whole process starting from preparing the data over building different machine learning models to visualizing the data making it easy understandable for everyone. But not only does it support the whole process, WEKA stands out because of the huge variety of algorithms available which are either implemented right away or can be extended via packages. There are two different ways of using WEKA: Either the graphical user interface, which can especially be useful for programming affine people or the application programming interface with the possibility of implementing machine learning models in programming code, can be used.

For the whole paper the WEKA version 3.8.6 is being used. In WEKA if the second number of the version is even, the release is stable, while an odd number indicates a development release. A convention which is also used for Linux kernels ("Development - Weka Wiki", n.d.). Another interesting fact that deserves appreciation is that WEKA runs under the GNU General Public License, which makes it a free software. This allows the user to "run, study, share and modify the software" ("GNU General Public License", 2023).

3.1 Graphical User Interface (GUI)

Probably the easiest way to use WEKA is via the GUI which makes machine learning executable without writing a single line of code. This paper puts the focus on the API and

all the examples below will be accomplished via the API, but in this section a short overview of the GUI will be made. When starting WEKA, the GUI Chooser opens offering five different applications: Explorer, Experimenter, KnowledgeFlow, Workbench and SimpleCLI.

The Explorer is very likely the simplest interface. It offers all the functionalities via menu selection and entering information into forms. To make it even simpler, the explorer helps by already showing options for the forms and explaining shortly what the variable is all about when hovering over it. The only thing users have to do is to understand and correctly interpret the results (Witten et al., 2016).

The Knowledge Flow option is designed for stream processing which is designed for large data sets. The problem with the Explorer is that he automatically loads in the whole data set when working with it, which for small- and medium-sized data set works perfectly fine, but since we are living in a world where more and more data is being collected leading to enormous data sets which are summarized under the term big data, the memory capacity of machines often is not big enough to load all the data into it. Therefore, a data stream is created making it possible to process very large amount of data (Witten et al., 2016).

The third option, the Experimenter, helps to find the best methods and parameters when it comes to supervised machine learning. There is a huge amount of available algorithms with each having parameters which can either improve or decrease the results. Even though often rules of thumb exists, like the 80/20 rule which states that 80% of the data should be used for training and 20% for testing, those rules are not always available or correct since they heavily depend on the data set and the use case. The task of finding the best parameters and methods can of course also be completed manually using the Explorer, but the Experimenter automates this step saving a lot of time (Witten et al., 2016).

The Workbench is a merged version of the first three interfaces including any installed plugins. Last but not least, the Simple CLI standing for Command-Line Interface is a simple panel allowing to insert commands (Witten et al., 2016).

3.2 Application Programming Interface (API)

After doing a short overview of the WEKA GUI the focus will now switch to the other possible method of using WEKA, which is via the API. An API is allowing different applications to communicate with each other. In this case, it will allow the usage of all WEKA functionalities

via programming code. As WEKA is written in Java, this is the main programming language for the API, but other programming languages, like Python, are available too.

To be able to use the API two components have to be installed: First, the WEKA software for the appropriate operating system and second, a supported Java version. An information of the current supported versions can be found on the WEKA website. Additionally, an integrated development environment (IDE) for Java is in fact not required but recommended, since this will make the coding part simpler. After finishing the installation of those components the only step left is adding the weka.jar file as a classpath in Java (gupta_shrinath, 2021). Having added the file as a classpath, all classes can be imported into the environment and the coding part can start.

3.3 Packages

Additionally to the huge amount of algorithms and tools implemented in WEKA there exist over 200 packages which enlarge the possibilities with WEKA even more. Those packages range from adding more algorithms to adding wrapper to make WEKA compatible with other programming languages. The easiest way to add packages is via the package manager which can be found in the Tools section of the GUI Chooser. It shows a list of all official WEKA packages with a description and gives even the opportunity to install unofficial packages. To be able to use the package manager properly, WEKA 3.8.1 or newer is required (Witten, n.d.). Showing all packages would go far beyond the scope of this paper.

3.4 .ARFF

The most popular data formats when using machine learning are probably CSV, JSON and XML. Even though WEKA does support those data formats it is recommended to use the ARFF format which is the short form for Attribute-Relation File Format. An ARFF is a text file encoded in ASCII and is separated into a header section and an information section ("Attribute-Relation File Format (ARFF)", 2008).

The header section contains the attribute information, also known as the columns of a data set. To build a header first a relation declaration is required. The name for the relation is a string and has to be quoted if spaces are included. Now attributes can be added, but keep in mind that the order matters. The first attribute added will be the first column and so on. Additionally, it is important that each attribute-name is unique and starts with an alphabetic

character. Again, the name has to be quoted if spaces are used ("Attribute-Relation File Format (ARFF)", 2008).

```
@RELATION <name>


@ATTRIBUTE <attribute-name> <datatype>
@ATTRIBUTE <attribute-name> <datatype>
```

For declaring the datatype four different options are available: numeric, nominal-specification, string and date. A numeric attribute is either an integer or a real, a nominal-specification allows for providing a list of possible options for this attribute, a string contains textual values and the date is used to denote dates with the possibility to define a specif date format ("Attribute-Relation File Format (ARFF)", 2008).

The second part, the information section, contains all the data. Each line denotes one entry in the data sets and gets separated by a carriage return. Within one entry values are separated by commas, similar to a CSV. Missing values are indicated by a question mark and strings and nominal-specification data types are case sensitive which can quickly lead to errors ("Attribute-Relation File Format (ARFF)", 2008).

```
@DATA
value 1.1, value 1.2, ..., value 1.n
value 2.1, value 2.2, ..., value 2.n
.
.
.
value n.1, value n.2, ..., value n.n
```

The following picture below shows an ARFF file containing information about housing prices in European cities and will also be used for the Data Preparation chapter. The interesting thing about the data is that it was created by artificial intelligence itself in form of Chat-GPT.

```
@relation housing

@attribute city {London,Paris,Berlin,Madrid,Rome,Amsterdam,Vienna,Athens,Stockholm,Dublin}
@attribute country {'United Kingdom',France,Germany,Spain,Italy,Netherlands,Austria,Greece,Sweden,Ireland}
@attribute type {Apartment,House}
@attribute sqr_metre numeric
@attribute price numeric

@data
London,'United Kingdom',Apartment,?,200000
Paris,France,House,120,350000
Berlin,Germany,Apartment,60,150000
Madrid,Spain,Apartment,90,180000
Rome,Italy,House,150,400000
Amsterdam,Netherlands,Apartment,70,250000
Vienna,Austria,House,110,300000
Athens,Greece,Apartment,75,160000
Stockholm,Sweden,Apartment,85,220000
Dublin,Ireland,House,130,380000
```

Figure 1: ARFF File

4 Data Preparation

Enough of the theory. It is now time to focus on concrete examples and show how to do machine learning with the help of WEKA, but before algorithms can be applied and predictions be made, the data set has to be prepared which is the main focus of this chapter.

4.1 Importing Data

The first step is always importing the data into the environment. As already stated, WEKA supports multiple data formats, but it is recommended to use either ARFF or CSV files. JSON and XML are supported too, but they have to be in an ARFF structure meaning that most of the downloadable data sets in JSON and XML are not compatible and first need to be converted. To import JSON which already are in a ARFF structure the JSONLoader class can be used, while XML in ARFF structure is called XRFF and therefor the class to import those files is called XRFFLoader. The following example shows how CSV and ARFF files can be imported:

```
1 DataSource src = new DataSource("./housing.csv");
2
3 Instances housing = src.getDataSet();
```

Code 1: Importing Data

The first step in importing a data set is to create a DataSource instance pointing to a file containing data. It is important to use the right file path, when pointing to a file. In this case, the file is in the same directory as the working environment. Once the DataSource is

initialized, it is possible to read in all the data with the help of the getDataSet method. In WEKA, a single data entry is an instance and therefore a data set which normally has lots of data entries is of the type Instances. With a simple print statement all Instances can be printed out.

4.2 Combining Data

When combining multiple data sets they can either be merged or appended. Merging means to combine multiple data sets with the same entries but different attributes. In a machine learning project it is normal to use multiple data sources since there are not always all information available in a single data set. To complete this task in WEKA the mergeInstances method is being used with both data sets as input. It is important that both Instances have the same number of rows and that all attributes have unique names. The following example expands the housing data set with additional information from a second data set:

```
1 Instances housing_new = Instances.mergeInstances(housing,
    housing_newcol);
```

Code 2: Merging Data

On the other hand, appending data adds additional data entries with the same attributes. This is done with the add method which adds one instance to the other instances with the same attributes in the same order. In this case, two data sets with each having multiple instances are being appended. Hence, a for loop is being used to add each instance one by one to the data set. With the numInstances method it is possible to get the number of Instances in a data set which is being used to denote how often the for loop has to be executed. It is important to start with Index zero, otherwise the first instance will be left out since WEKA indices start with zero.

```
1 for(int i = 0; i < housing_newrow.numInstances(); i++) {
2     housing_new.add(housing_new.instance(i));
3 }
```

Code 3: Appending Data

4.3 Understanding Data

Before one is able to work with data it is always import to understand the data first. Which variables are included in the data set? What is the mean/mode of each variable? What is the maximum/minimum value for each variable? All these information can help in getting an understanding of the data, making all further tasks easier.

The first step in gaining some insights into the data set is to look at it. Since data sets are often very large it can be very confusing to print out the whole data set. In a lot of cases it is already enough to print out the first few instances to get a broad overview.

```
for (int i = 0; i < 5; i++) {
    System.out.println(housing_new.get(i));
}
```

Code 4: Get First 5 instances

Probably the most practical method summarizing a whole data set with only one line of code is the toSummaryString method. It is included in the Instances class and the output gives following information about each variable: name, data types, the amount of missing values, the amount of unique values and the amount of different values. Very useful information which can help in further preparation steps.

```
System.out.println(housing_new.toSummaryString());
```

Code 5: Summarizing a Data Set

Next, it can be helpful to check the mean/mode for each variable. The mean is calculated by summing up all values and dividing by the amount of them, while the mode outputs the value which occurs most frequently. If the variable is numeric, the mean will be calculated, while the mode is being calculated for nominal variables.

```
for (int i = 0; i < housing_new.numAttributes(); i++) {
    System.out.println(housing_new.meanOrMode(i));
}
```

Code 6: Get Mean/Mode

Last but not least, it might be helpful to calculate the maximum and minimum value for each numerical variable. This information does not only give insights about the range of the

variable but could also be the first indicator for an outlier if the minimum or maximum value differs a lot from the mean. The if-statement in the code below makes sure that only numerical variables are being considered by checking their type. Afterwards, the kthSmallestValue method is being used first to output the smallest value and then to output the biggest value by outputting the last smallest value which in fact is the biggest value.

```
for (int i = 0; i < housing_new.numAttributes(); i++) {
    if(housing_new.attribute(i).type() == 0){

        System.out.println("The smallest Value is: " +
   housing_new.kthSmallestValue(i, 1));

        System.out.println("The biggest Value is: " +
   housing_new.kthSmallestValue(i, housing_new.numInstances()))
   ;
     }
}
```

Code 7: Get Maximum/Minimum Value

4.4 Selecting Data

All the data needed is now combined into a single data set but often not all data is relevant for the task. If the task is to predict housing prices, the attribute denoting if the house has a balcony or not might not have an observable impact to the price if all houses observed have a balcony, whilst the size of the houses definitely will have an impact on the price and therefore need to be selected. The following example deletes the attribute showing if the house has a balcony or not:

```
housing_new.deleteAttributeAt(1);
```

Code 8: Deleting Attributes

The deleteAttributeAt method deletes the attribute based on the Index starting with 0. In this case, the second attribute has being deleted, leaving only attributes needed, but not only attributes need to be selected. Instances may to be filtered too. Let's say only houses

for 170,000€ should be observed. The following example shows how to filter instances introducing the concept of Filters:

```
RemoveWithValues filter_expensive = new RemoveWithValues();

String[] options = new String[4];
options[0] = "-C";
options[1] = "5";
options[2] = "-S";
options[3] = "170000";
filter_expensive.setOptions(options);

filter_expensive.setInputFormat(housing_new);

Instances housing_expensive = Filter.useFilter(housing_new,
   filter_expensive);
```

Code 9: Filter Instances

Filter take Instances as input, compute some kind of transformation and output new Instances. There are various filters for all kind of tasks. For filtering values the RemoveWithValue filter and a string array with the options for that filter are being used. "-C" is for selecting the attribute via an index. In this case, the index starts with 1 so the fifth attribute is being selected. "-S" selects all values that are smaller than that value. Combining both options, it selects all instances where the fifth column is less than 170,000. Those options are now given to the filter with the setOptions method. Next, it is important to set the input Format with the instances as an input and then the Filter can be used, which will remove all instances where the value of column 5 is below 170,000.

4.5 Cleaning Data

After an understanding of the data is created and the relevant data is selected the next step is to clean the data which is probably the most time consuming task, but having good quality data is crucial in machine learning to get good results. The main task of data cleaning is to fix errors arising in the data set. Those errors can either be duplicates, structural errors, outliers or missing values (“Data Cleaning in Data Mining - Javatpoint”, n.d.).

Duplicate entries mostly arise when collecting data. Either the same entry is being inserted multiple times or data sets containing mutual entries are being combined. But not every duplicate entry has to be removed. This highly depends on the data set since there could simply be two different entries with the same values. In the best case, the data set has a key variable, like an ID, making every entry unique. The following example removes all duplicate entries with the help of the RemoveDuplicates class which is again applied via filters.

```
RemoveDuplicates filter_duplicate = new RemoveDuplicates();

filter_duplicate.setInputFormat(housing_expensive);

Instances housing_nodup = Filter.useFilter(housing_expensive,
   filter_duplicate);
```

Code 10: Removing Duplicates

Another error that can lead to problems if not removed in the preparation stage are structural errors. This goes from bad naming conventions to bad capitalization. Both, attribute naming conventions and value naming convention should be consistent ("Data Cleaning in Data Mining - Javatpoint", n.d.). For example, no capital letters and no blank spaces in the data set. "New York" and "new york" are two different values. Therefore, defining clear rules from the beginning will avoid a lot of errors later. The following example renames an attribute which had a different capitalization compared to the other attributes:

```
housing_nodup.renameAttribute(5,"rooms");
```

Code 11: Rename Attribute

Sometimes the problem with data is not about the data available but rather the data not available. The sensor collecting the data did not work or the clerk filling in the data missed a field: All this mistakes lead to missing values. There are different notations for missing values ranging from an empty field over "None" to 0. WEKA uses the question mark to denote missing values. There are a lot of different ways of handling missing data. One way is simply deleting the whole entry which is easy to compute but information may get lost since the whole entry is being deleted and not only the missing value. Other methods fill the missing value either with the mean/mode from the column or the value from the most similar data entry. The following example deletes all rows containing missing values:

```
housing_nodup.removeIf(Instance::hasMissingValue);
```

Code 12: Removing Missing Values

Last but not least, Outlier, which are unusual data points varying significantly to all other data points and therefor could bias the model, need to be handled, but since outlier detection is more complex, an algorithm or different data visualizations are used for detecting them ("What are outliers in the data?", n.d.).

4.6 Creating New Data

Data can create new data. Sometimes a specific attribute needed is not available but the data for calculating it is. For example, the housing data set contains the prices for the properties, but since all properties have a different size it is hard to compare them. Therefore, a new column needs to be created containing the prices per square metre making the prices for all cities comparable. This can be archived by expressions which again use the Filters class for applying. The process is quite simple. First, set the expression, in this case a division and second, set the name of the new attribute. The only thing left is applying the filter.

```
AddExpression addExpressionFilter = new AddExpression();

addExpressionFilter.setExpression("a5 / a4");
addExpressionFilter.setName("price_per_sqrm");

addExpressionFilter.setInputFormat(housing_nodup);

Instances housing_prepared = Filter.useFilter(housing_nodup,
   addExpressionFilter);
```

Code 13: Creating New Data

4.7 Train/Test Split

When using supervised machine learning algorithms, the data first needs to be split into training and testing data. The training data is being used for training the model and the testing data afterwards for testing it. For this step, it is important to randomize the data first since often the data is sorted by a column which could bias the results. For example, if

the training data only contains cats and the testing data only contains dogs the results will be bad. After the data is randomized, it can be split into two data sets. There is no specific ratio that has to be used for splitting, but as a rule of thumb the ratio 80/20 is often being used making 80% training data and 20% testing data like in the example below, where the finished data set is being split:

```
1  int seed = 42;
2  double trainPercentage = 80.0;
3
4  housing_prepared.randomize(new Random(seed));
5
6  int trainSize = (int) Math.round(housing_prepared.numInstances
       () * trainPercentage / 100.0);
7
8  int testSize = housing_prepared.numInstances() - trainSize;
9
10 Instances trainData = new Instances(housing_prepared, 0,
       trainSize);
11 Instances testData = new Instances(housing_prepared, trainSize,
        testSize);
```

Code 14: Train/Test Split

First a seed is being set. With this seed the randomize method will always output the same random order. This is important so the results do not change if the code is being run multiple times. Having shuffled the data, the sizes for both training and testing data are being calculated using simple mathematics. Now the data can be split from entry 0 to entry trainsize for the training data and from entry trainsize to the last entry of the data set for the testing data.

5 Machine Learning

The output of the previous chapter, a clean data set, can now be used for the main chapter in this paper, machine learning. The different types of machine learning have already been explained in the second chapter making it now possible to straight dive into different algo-

rithms. The following algorithms were chosen based on experiences and an online research and should cover the most known and important algorithms for machine learning but keep in mind that WEKA offers a much bigger variety of algorithms which blow the scope of this paper.

5.1 Supervised Learning

First, the focus is being put on supervised machine leaning algorithms which work with labeled data. The following seven algorithms will be shown in this chapter: Linear Regression, Logistic regression, Decision Tree, Random Forest, Support Vector Machine, Naive Bayes Algorithm and K-Nearest Neighbor.

5.1.1 Linear Regression

Linear Regression is a very easy understandable machine learning algorithm and therefore the perfect start in this topic. It is being used for predicting a numeric variable, called the dependent variable, with the help of one or multiple other variables, also known as independent variables. To do so, the algorithm is looking, as the name would already suspect, for linear relations between the dependent and independent variables. A graph is a good way to make such relations visible ("Linear Regression in Machine learning - Javatpoint", n.d.).

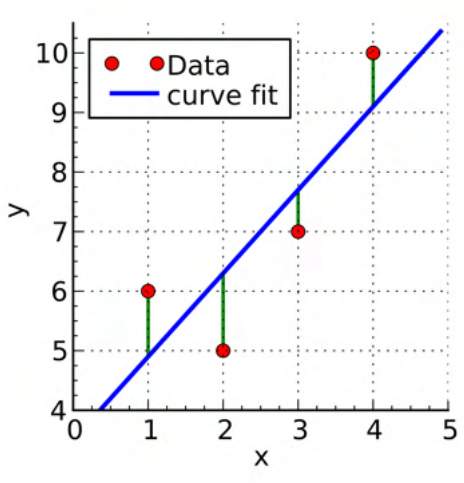

Figure 2: Linear Regression
Source: (Krishnavedala, 2011) Licensed under CC BY-SA 3.0

The regression line denoting the relationship between the variables can either go up or down. An up-going relation line indicates a positive relationship. If the independent variable rises the dependent variable rises too. This is for example the relationship between a house size

and its price. The bigger the house the more expensive it will be if all other variables stay the same. On the other hand, a down-going relationship is if the independent variable decreases, but the dependent variable increases. There is a negative relation. The more damages a house has the less the price is going to be. If the regression line is horizontal there simply is no relation between those variables.

Linear regression can be further divided into simple- and multiple linear regression. A linear regression is called simple if only one independent variable is used to predict one dependent variable. If multiple independent variables are used, it is called multiple linear regression.

In the figure above every dot represents one data entry with the values for the independent and dependent variable. The best case would be to get a regression line that goes through all those points and therefore describing the entries the best which in most cases is impossible. There always will be a certain error between the line and the data points. The goal is to minimize this error by finding the best fitting line using cost functions. In linear regression the cost function is called Mean Squared Error (MSE) which calculates the average squared errors between the prediction and the actual value ("Linear Regression in Machine learning - Javatpoint", n.d.).

In WEKA, building a linear regression model is very easy. The example below builds a linear regression model where a prediction on house prices is being made based on the size of the house and the number of bed- and bathrooms. This data is again generated by ChatGPT and has already been processed and split into training and testing data set.

```
1 LinearRegression lr = new LinearRegression();
2
3 trainData.setClassIndex(trainData.numAttributes()-1);
4
5 lr.buildClassifier(trainData);
```

Code 15: Linear Regression

The procedure in building a supervised machine learning model is for most algorithms quite similar. Only the options that can be provided via a string array, the same way as in chapter 4.4 Selecting Data, differ. The first step is to create a new instance of the model class, like in this case a LinearRegression instance. The next step is to set the class index which is the

dependent attribute which is being predicted. Often this attribute is set as the last attribute so the index can be identified by calculating the number of attributes and removing 1, but other places are possible too. Once the class index is being set, the classifier can be built and the results printed out. The code above outputs following:

```
Price = 1138.9569 * SquareMeters + 57701.7943 * Bathrooms + (-1570.6529)
```

The result is an equation showing the price based on the square metres and the amount of bathrooms. The amount of bedrooms are not in the equation since, by default, there is an attribute selection method used deciding if the attribute is relevant for the regression. The equation can be interpreted as following: for each square metre the price for the house rises by 1138.9569 and for each bathroom the price rises for 57701.7943. The intercept is -1570.6529 which is the value if square metre and bathrooms are both 0, which obviously do not make any sense, but the intercept should never be interpreted on its own but only in combination with all variables.

But how good is the model? To be able to answer this question, the model has to be evaluated, which in WEKA works for most supervised machine learning models identical as seen below:

```
1 Evaluation eval = new Evaluation(trainData);
2
3 testData.setClassIndex(testData.numAttributes()-1);
4
5 eval.evaluateModel(lr, testData);
6
7 System.out.println(eval.toSummaryString());
```

Code 16: Linear Regression Evaluation

To evaluate a model, an Evaluation instance has to be initialized having the training data as input. In the next step, the class index has to be set for the testing data the same way as above for the training data. Now, the evaluation can be done by calling the evaluateModel method giving the linear regression and the testing data as input. There are many different evaluation parameters available. For regression, the toSummaryString gives a good overview as seen in the output.

```
Correlation coefficient                  0.9765
Mean absolute error                  17253.4261
Root mean squared error              20315.711
Relative absolute error                 21.1987 %
Root relative squared error             22.9693 %
Total Number of Instances                9
```

A good variable for evaluating linear regression is the root mean squared error (RMSE). The mean square error (MSE) calculates the mean of the residuals, which are the squared differences from predicted to the real values. RMSE takes the root from this mean to scale it down again to the variable. The smaller this value the better the model (Wheeler, 2021). A value of 20315.711 can be simplified explained as the models predictions are in average about 20,000 wrong, which is below 10% of the mean of this data set making it a good value especially because the data set is very small.

5.1.2 Logistic Regression

Logistical Regression works similar to linear regression, but instead of predicting numeric variables it is used to predict categorical variables. Instead of predicting the price of a house, logistic regression can for example predict if the data entry belongs to a house or a flat. And another difference compared to the linear regression is that the result is not exact but rather a probabilistic value between 0 and 1. In combination with a predefined threshold a prediction to one category can be made. Other than the name would suggest, logistic regression is not being used to solve regression problems but classification problems. Again the goal is to fit a line to the data points but in this case the line is S shaped, lies between 0 and 1 and is described by the Sigmoid function ("Logistic Regression in Machine Learning - Javatpoint", n.d.).

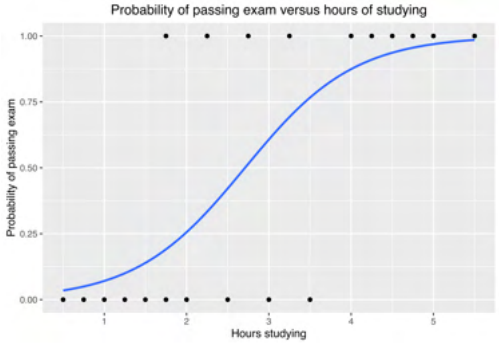

Figure 3: Logistic Regression
Source: (Canley, 2022) Licensed under CC BY-SA 4.0

There are three different types of logistic regression based on the variable that is trying to be predicted. If the variable only has two categories, binomial logistic regression is being used (e.g. Yes/No). If there are more than two categories which are not in an order, multinomial logistic regression is being used (e.g. Dog, Cat, Mouse). If there are more than two categories with an order, ordinal logistic regression is being used (e.g. Low, Medium, High). To evaluate the results, a confusion matrix is being build where correct and wrong predictions are being compared ("Logistic Regression in Machine Learning - Javatpoint", n.d.).

Since all the upcoming algorithms including the logistic regression are mainly used for classification tasks, the same data set with the same task is being computed for the remaining algorithms to be able to compare the results and choose the best algorithm for the task. The data set is called weather.nominal.arff and is included when installing WEKA. It shows if someone played (yes or no) based on the weather conditions: outlook, temperature, humidity and windy. Computing a logistic regression works the same as for the linear regression. The only difference is the different class that has to be initialized.

```
1 Logistic logr = new Logistic();
2
3 trainData.setClassIndex(trainData.numAttributes()-1);
4
5 logr.buildClassifier(trainData);
```

Code 17: Logistic Regression

To be able to compare all the models, an evaluation metrics is needed. To evaluate classification models, the confusion matrix is being used, which can be derived as following:

```
1 Evaluation eval = new Evaluation(trainData);
2
3 testData.setClassIndex(testData.numAttributes()-1);
4 eval.evaluateModel(logr, testData);
5
6 System.out.println(eval.toMatrixString());
```

Code 18: Logistic Regression Evaluation

Again, the evaluation works the same as for the linear regression, but in this case we call the toMatrixString method to create a confusion matrix. The result can be seen below. It

is a table where the rows denote the real values and the columns the predicted ones. In this case the model predicted four times yes(a), but only two predictions were correct, while the other two were wrong. On the other hand it predicted two times no(b) from which one prediction is correct and one wrong. Overall, the model was three times correct and three times wrong.

```
a b   <-- classified as
2 1 | a = yes
2 1 | b = no
```

5.1.3 Decision Tree

The decision tree algorithm uses tree-structures like the family tree for solving both regression and classification problems, but other than representing the family, the internal nodes represent the variables (e.g. Housetype), the outgoing lines represent the possibilities (e.g. House and Flat) and the leaf nodes represent the outcome following those possibilities. The result is a tree containing all possible combinations of variables. The tree is being build based on the Classification and Regression Tree algorithm (CART). The advantage of decision trees is that the logic behind is easy understandable ("Decision Tree Algorithm in Machine Learning - Javatpoint", n.d.).

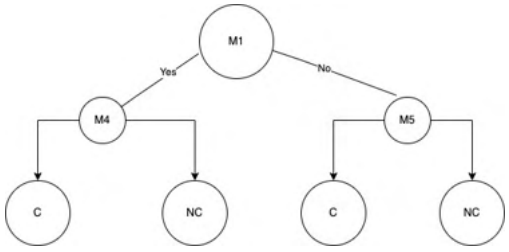

Figure 4: Decision Tree
Source: (Gmcgee15, 2021) Licensed under CC BY-SA 4.0

The way the algorithm comes to a prediction can easily be described with 5 steps. Step 1 is to begin with the root node, which consists of the whole data set. In step 2 Attribute Selection Measure (ASM) is being used, which uses different techniques to detect the best attribute. In step 3 the data set is being divided based on the values for the best attribute. Step 4 is to generate a node for this attribute. The last step is to recursively repeat this until nodes can not be further classified. The last nodes created are called leaf nodes ("Decision Tree Algorithm in Machine Learning - Javatpoint", n.d.).

Those trees can become very large fitting perfectly to the data which is not optimal since

it might lead to an overfitting model, which does well for the data we are training the model on, but very poorly on new data. Therefore, the tree needs to be pruned to the right size to get rid of unnecessary nodes. On the other hand, if too much of the tree is being pruned, too much information might get lost. The two main techniques for pruning are Cost Complexity Pruning and Reduce Error Pruning ("Decision Tree Algorithm in Machine Learning - Javatpoint", n.d.).

To build a decision tree the J48 class is being used. The rest follows the same pattern as the algorithms before.

```
1 J48 dt = new J48();
2
3 trainData.setClassIndex(trainData.numAttributes()-1);
4
5 dt.buildClassifier(trainData);
```

Code 19: Decision Tree

Same applies for the confusion matrix, therefore only the matrix will be shown and not the code behind it. Again we got three correct and three wrong predictions, but the distribution is a bit different.

```
a b   <-- classified as
3 0 | a = yes
3 0 | b = no
```

5.1.4 Random Forest

Similar to a forest made up of trees the random forest algorithm uses multiple decision trees for solving both regression and classification tasks. Each tree contains a subset of the data and gives a result on its own. The result, the majority of the trees agree on, is the final output. The more trees are being used, the higher the accuracy. The algorithms advantage is its predictive power and the efficiency especially with large data sets ("Machine Learning Random Forest Algorithm - Javatpoint", n.d.).

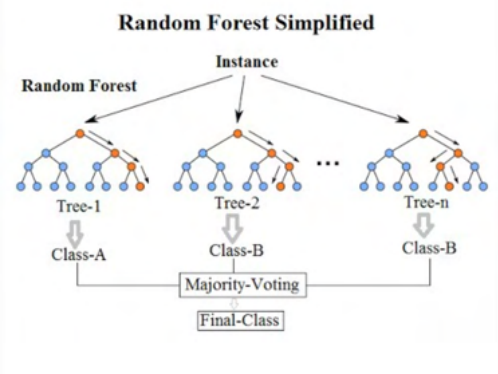

Figure 5: Random Forest
Source: (Jagannath, 2017) Licensed under CC BY-SA 4.0

For building the model the RandomForest class is being used. The rest stays the same.

```
RandomForest rf = new RandomForest();

trainData.setClassIndex(trainData.numAttributes()-1);

rf.buildClassifier(trainData);
```

Code 20: Random Forest

The random forest gives the same results as the logistic regression.

```
a b   <-- classified as
2 1 | a = yes
2 1 | b = no
```

5.1.5 Support Vector Machine (SVM)

Another very popular algorithms used both for regression and classification problems is the support vector machine algorithm. In contrast to trying to fit a line to the data points, the SVM algorithm tries to generate a line that splits the data sets into n-categories. The best line for this task is being called hyperplane which has the maximum margin, the maximum distance between the data points. The hyperplane is being created with the help of extreme points called support vectors. Those points are closest to the hyperplane and therefore have the biggest impact on it. There are two types of SVM: Linear SVM which is used when the data can be separated into two classes by a single straight line and non-linear SVM if it is not possible to do so ("Support Vector Machine (SVM) Algorithm - Javatpoint", n.d.).

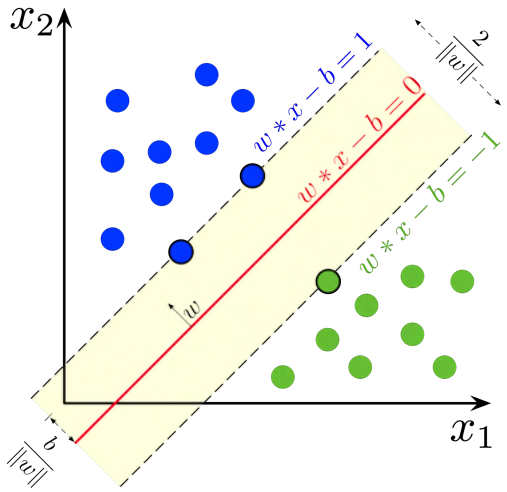

Figure 6: Support Vector Machine
Source: (Larhmam, 2018) Licensed under CC BY-SA 4.0

The class for support vector machines is called SMO. The rest stays the same.

```
1 SMO svm = new SMO();
2
3 trainData.setClassIndex(trainData.numAttributes()-1);
4
5 svm.buildClassifier(trainData);
```

Code 21: Support Vector Machine

Support Vector Machines output the best result so far with guessing four instances right and only two wrong.

```
a b   <-- classified as
3 0 | a = yes
2 1 | b = no
```

5.1.6 Naive Bayes Algorithm

The Naive Bayes Classifier is used for classification and regression problems and follows, same as the logistic regression, a probabilistic approach. The main use case is for classifying text. The algorithm got its name because of its naive thinking that there are no dependencies between the features and its dependency on the Bayes Theorem ("Naive Bayes Classifier in Machine Learning - Javatpoint", n.d.).

```
1 NaiveBayes nb = new NaiveBayes();
2
3 trainData.setClassIndex(trainData.numAttributes()-1);
4
5 nb.buildClassifier(trainData);
```

Code 22: Naive Bayes

The Naive Bayes algorithm predicts the same as the logistic regression and the random forest.

```
a b   <-- classified as
2 1 | a = yes
2 1 | b = no
```

5.1.7 K-Nearest Neighbor (KNN)

To round up the supervised machine learning algorithms, a very simple yet powerful algorithm is being shown. KNN is mainly used for classification problems but can, in principle, be used for regression problems as well. The concept is quite simple: Assign new data to the category with the most similar characteristics to other data in this category. This is why it is called a lazy learner algorithm: It does not use the training data immediately but rather waits for new data to look for similarities in the available data entries ("K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint", n.d.).

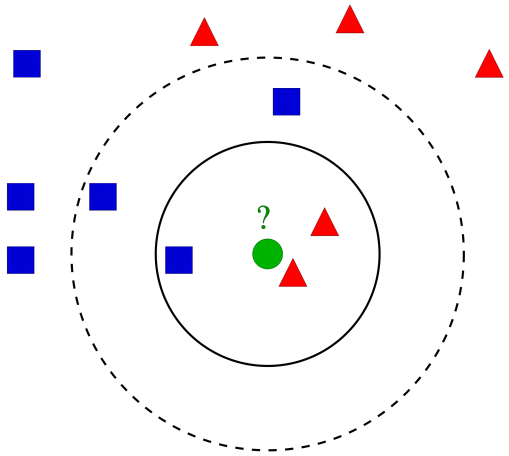

Figure 7: K-Nearest Neighbor
Source: (AnAj, 2007) Licensed under CC BY-SA 3.0

The process behind the algorithm is as easy as the description. First, a number for K, which is the amount of neighbors, is chosen. Next, K nearest neighbors are being selected with the

help of the euclidean distance. The last step is to check the categories from the K-neighbors. The category which the majority of the neighbors belong to is being assigned to the new data entry. The best way to find out a good value for K is trying it out, but often 5 delivers good results ("K-Nearest Neighbor(KNN) Algorithm for Machine Learning - Javatpoint", n.d.).

```
1 IBk knn = new IBk();
2
3 trainData.setClassIndex(trainData.numAttributes()-1);
4
5 knn.buildClassifier(trainData);
```

Code 23: Linear Regression Evaluation

KNN outputs the same as logistic regression, random forest and naive bayes. Therefore, the best model for this task is the Support Vector Machine with the most correct classifications.

```
a b   <-- classified as
2 1 | a = yes
2 1 | b = no
```

5.2 Unsupervised Learning

Data is not always labeled and labeling data can quickly become very time consuming and therefore expensive. That is why unsupervised machine learning algorithms were invented. They do not need any labeling for finding patterns in the data. Often, they are used as predecessor for supervised machine learning algorithms, detecting outlier, finding clusters and reducing the dimension in the data set. Following two algorithms are being shown in this chapter: Principal Component Analysis, K-Means Clustering.

5.2.1 Principal Component Analysis

Data sets are getting larger and larger. Not only row-vise but also column-vise collecting more kinds of data. Columns in a data set are also known as the dimensionality of a data set. A high dimensionalty leads to modeling and visualazing tasks get more and more complex. Therefore dimensionality reduction is being used to reduce the number of features

("Introduction to Dimensionality Reduction Technique - Javatpoint", n.d.).

Probably the most known algorithm for computing such tasks is the principal component analysis. It converts a large number of correlated variables into a set of linearly uncorrelated variables, which are known as Principal Components and are orthogonal, meaning that there is no correlation between them. Also those components decrease in importance making the first principal component also the most important one ("Principal Component Analysis - Javatpoint", n.d.).

Whilst it is possible to do PCA on both numeric and categorical variables, it is mainly recommended for numerical variables. To compute a PCA in WEKA, the filters class is being used. The housing data set which was used in the linear regression chapter has only numerical attributes and is therefore used to compute a PCA even though it only contains three variables. The output of a PCA is a new data set containing only two variables so one dimension less.

```
1 PrincipalComponents pca = new PrincipalComponents();
2
3 pca.setInputFormat(housing);
4
5 Instances transformedData = Filter.useFilter(housing, pca);
```

Code 24: Principal Components Analysis

5.2.2 K-Means Clustering

Another task of unsupervised machine learning algorithms is to find clusters, which are a set of data points with similar characteristics. K-Means Clustering is the most known algorithm for solving this problem. In contrast to KNN, where K denotes the amount of neighbors that are being observed, the K in K-Means denotes the amount of clusters. Every data point is put into exactly one cluster. Each cluster has a randomly chosen centroid where the data points are added to minimizing the sum of distances between data points and centroids. This process is being repeated with improved centroids until the clusters with minimum distance has been found. Deciding the right value for K can be done with the Elbow method, which uses WCSS, the Within Cluster Sum of Squares ("K-Means Clustering Algorithm - Javatpoint", n.d.).

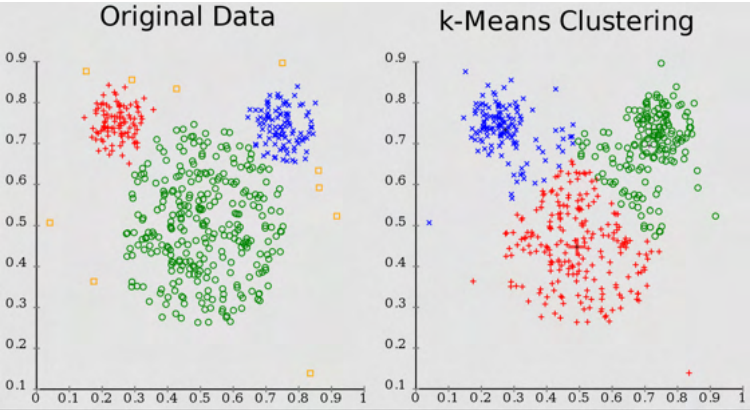

Figure 8: K-Means Clustering
Source: (Chire, 2010) Licensed under Public Domain

The class that can compute K-Means clustering is called SimpleKMeans and can be easily implemented with the builClusterer method.

```
1 SimpleKMeans skm = new SimpleKMeans();
2 skm.buildClusterer(weather);
```

Code 25: K-Means Clustering

6 Data Visualization

When doing machine learning, preparing the data and choosing the right algorithms is very important, but being able to describe the results in a way that people without any machine learning knowledge can understand your findings too, earns at least the same importance. Since a lot of text and numbers can quickly confuse people, visualizations are the way to go when trying to summarize data and make it easy understandable. Also, visualizations can help getting an understanding of the data before starting to prepare it. There are a lot of different ways to present data, which makes the tools making it available very powerful.

Even though this paper focuses on the API, it is worth mentioning that when it comes to visualization the GUI has a huge advantage and makes it easier to compute different kinds of graphs. But still, the API offers possibilities too as shown in the example below:

```
PlotData2D plotData = new PlotData2D(housing);

VisualizePanel panel = new VisualizePanel();
panel.addPlot(plotData);

JFrame frame = new JFrame("Linear Regression Visualization");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(1600, 1200);
frame.getContentPane().add(panel);
frame.setVisible(true);
```

Code 26: Data Visualization

In order to create a graph, first a PlotData2D instance has to be created with the data that is going to be visualized as the input. Next, a panel is being created where the PlotData2D instance is being added. To be able to visualize it, a GUI has to be created, which in Java can be done via a JFRAME. To be able to close the window the setDefaultCloseOperation is being set as well as the size. In the end, only the panel has to be added to the JFRAME and it has to be set as visible. The Output is not just a single graph, but a tool for creating graphs by choosing different variables. This is perfect for getting a first understanding of how the correlation between two variables is or if there are any outliers.

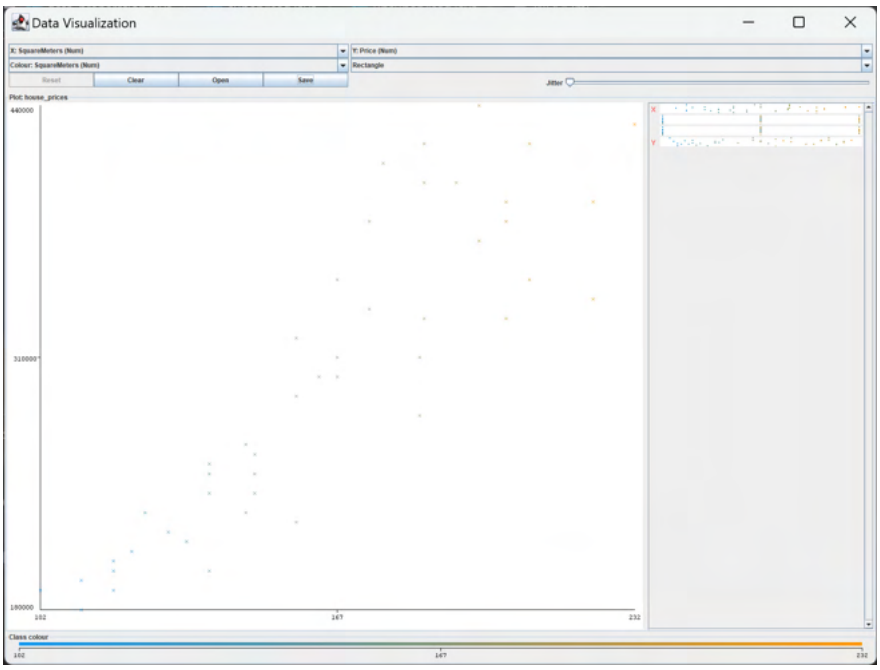

Figure 9: Data Visualization

7 Summary

WEKA is a very powerful software. Data preparation, building machine learning models and visualizing data is normally a task for three different tools and WEKA combines them all. Even more, it offers multiple programming languages, a GUI making it possible to do all this tasks without writing a single line of code and a package manager which is adding even more functions to the software. Especially the machine learning algorithms are very powerful. The possibility to build a model in a few code lines, enhanced with additional tools, like a missing value detection and a variables selection, makes it really easy to compute machine learning projects. It is worth mentioning, that it supports deep learning too, which was out of scope in this paper, but very interesting for further investigations. All the knowledge gained in this paper should be sufficient for computing first small machine learning projects with the help of WEKA.

References

AnAj, W. A. A. (2007, May 28). *K-nearest neighbors algorithm.* https://upload.wikimedia.org/wikipedia/commons/e/e7/KnnClassification.svg

Association rule learning - javatpoint. (n.d.). Retrieved April 14, 2023, from https://www.javatpoint.com/association-rule-learning

Attribute-relation file format (arff). (2008, November 1). Retrieved April 29, 2023, from https://www.cs.waikato.ac.nz/~ml/weka/arff.html

Canley, W. (2022, March 27). *Logistic regression.* https://upload.wikimedia.org/wikipedia/commons/c/cb/Exam_pass_logistic_curve.svg

Chire, W. (2010, October 12). *K-means clustering.* https://en.wikipedia.org/wiki/K-means_clustering#/media/File:ClusterAnalysis_Mouse.svg

Data cleaning in data mining - javatpoint. (n.d.). Retrieved May 3, 2023, from https://www.javatpoint.com/data-cleaning-in-data-mining

Decision tree algorithm in machine learning - javatpoint. (n.d.). Retrieved May 10, 2023, from https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

Development - weka wiki. (n.d.). Retrieved April 27, 2023, from https://waikato.github.io/weka-wiki/development/

Difference between descriptive and predictive data mining - javatpoint. (n.d.). Retrieved April 11, 2023, from https://www.javatpoint.com/descriptive-vs-predictive-data-mining

Gmcgee15, W. (2021, December 10). *Decision tree.* https://upload.wikimedia.org/wikipedia/commons/4/43/Phi_Function_Tree.jpg

Gnu general public license. (2023, June 3). Retrieved April 27, 2023, from https://en.wikipedia.org/wiki/GNU_General_Public_License

Gonfalonieri, A. (2020). Introduction to causality in machine learning - towards data science. Retrieved April 11, 2023, from https://towardsdatascience.com/introduction-to-causality-in-machine-learning-4cee9467f06f

gupta_shrinath. (2021). How to use weka java api. *GeeksforGeeks.* https://www.geeksforgeeks.org/how-to-use-weka-java-api/

Hernán, M., Hsu, J., & Healy, B. (2019). A second chance to get causal inference right: A classification of data science tasks. *Chance*, *32*(1), 42–49. https://doi.org/10.1080/09332480.2019.1579578

Intelligence. (n.d.). Retrieved April 6, 2023, from https://dictionary.cambridge.org/dictionary/english/intelligence

Introduction to dimensionality reduction technique - javatpoint. (n.d.). Retrieved May 29, 2023, from https://www.javatpoint.com/dimensionality-reduction-technique

Introduction to semi-supervised learning - javatpoint. (n.d.). Retrieved April 15, 2023, from https://www.javatpoint.com/semi-supervised-learning

Jagannath, W. V. (2017, March 24). *Random forest*. https://upload.wikimedia.org/wikipedia/commons/7/76/Random_forest_diagram_complete.png

Jiang, T., Gradus, J., & Rosellini, A. (2020). Supervised machine learning: A brief primer. *Behavior Therapy*, *51*(5), 675–687. https://doi.org/10.1016/j.beth.2020.05.002

K-means clustering algorithm - javatpoint. (n.d.). Retrieved June 4, 2023, from https://www.javatpoint.com/k-means-clustering-algorithm-in-machine-learning

K-nearest neighbor(knn) algorithm for machine learning - javatpoint. (n.d.). Retrieved May 24, 2023, from https://www.javatpoint.com/k-nearest-neighbor-algorithm-for-machine-learning

Krishnavedala, W. (2011, June 9). *Linear regression*. https://upload.wikimedia.org/wikipedia/commons/b/b0/Linear_least_squares_example2.svg

Larhmam, W. (2018, October 19). *Support vector machine*. https://upload.wikimedia.org/wikipedia/commons/7/72/SVM_margin.png

Linear regression in machine learning - javatpoint. (n.d.). Retrieved May 7, 2023, from https://www.javatpoint.com/linear-regression-in-machine-learning

Logistic regression in machine learning - javatpoint. (n.d.). Retrieved May 8, 2023, from https://www.javatpoint.com/logistic-regression-in-machine-learning

Machine learning random forest algorithm - javatpoint. (n.d.). Retrieved May 11, 2023, from https://www.javatpoint.com/machine-learning-random-forest-algorithm

Milligan, G., & Cooper, M. (1987). Methodology review: Clustering methods. *Applied Psychological Measurement*, *11*(4), 329–354. https://doi.org/10.1177/014662168701100401

Mohri, M., Rostamizadeh, A., & Talwalkar, A. (2018, December 25). *Foundations of machine learning, second edition*. MIT Press. https://books.google.at/books?hl=de&lr=&id=dWB9DwAAQBAJ&oi=fnd&pg=PR5&dq=machine+learning&ots=AyvQWXu5r-&sig=8v9cwvnW_iw-0chzeFZBmVyKISE#v=onepage&q=machine%20learning&f=false

Naive bayes classifier in machine learning - javatpoint. (n.d.). Retrieved May 20, 2023, from https://www.javatpoint.com/machine-learning-naive-bayes-classifier

Principal component analysis - javatpoint. (n.d.). Retrieved May 31, 2023, from https://www.javatpoint.com/principal-component-analysis

Reinforcement learning tutorial - javatpoint. (n.d.). Retrieved April 15, 2023, from https://www.javatpoint.com/reinforcement-learning

Rockwell, A. (2017, August 28). *The history of artificial intelligence.* Retrieved April 7, 2023, from https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/

Semi-supervised learning, explained with examples. (2022, March 18). Retrieved April 15, 2023, from https://www.altexsoft.com/blog/semi-supervised-learning/

Support vector machine (svm) algorithm - javatpoint. (n.d.). Retrieved May 13, 2023, from https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

Types of machine learning - javatpoint. (n.d.). Retrieved April 12, 2023, from https://www.javatpoint.com/types-of-machine-learning

Weka (software). (2023, May 12). Retrieved April 9, 2023, from https://en.wikipedia.org/wiki/Weka_(software)

What are outliers in the data? (n.d.). Retrieved May 3, 2023, from https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm

What is unsupervised learning? - ibm. (n.d.). Retrieved April 13, 2023, from https://www.ibm.com/topics/unsupervised-learning

Wheeler, W. (2021). Evaluating linear regression models using rmse and r^2. Retrieved May 7, 2023, from https://medium.com/wwblog/evaluating-regression-models-using-rmse-and-r%C2%B2-42f77400efee

Witten, I. (n.d.). Extending weka with "packages". *FutureLearn.* https://www.futurelearn.com/info/courses/data-mining-with-weka/0/steps/25379

Witten, I., Frank, E., & Hall, M. (2016, January 1). *The weka workbench. Online appendix for "data mining: Practical machine learning tools and techniques" morgan kaufmann, fourth edition, 2016.* The University of Waikato. https://www.cs.waikato.ac.nz/ml/weka/Witten_et_al_2016_appendix.pdf