LAB01
Khai thác dữ liệu

21KHMT01

Thành viên nhóm:

21127471 – Nguyễn Hoàng Anh Tuấn

21127687 – Phan Huy Đức Tài

Contents

1. Introduction

The purpose of this lab session is to assist students to better understand data exploration with WEKA and data preprocessing without the use of utility libraries. The report will contain everything required in the assignment document; however the code section will only contain result demonstration.

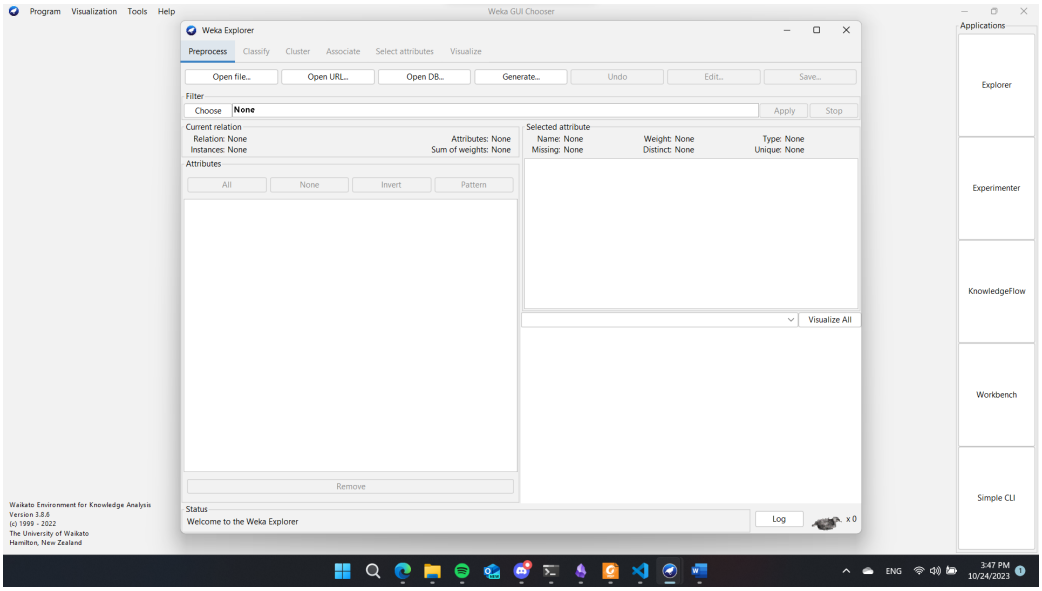
2. Assignment overview

Student ID	Student name	Assigned task	Task contribution
21127471	Nguyễn Hoàng Anh Tuấn	* Read .CSV file and code the required functions. * Run test cases	50 %
21127687	Phan Huy Đức Tài	* Perform data exploration with WEKA (Installing and interacting) * Compile and write report	50 %

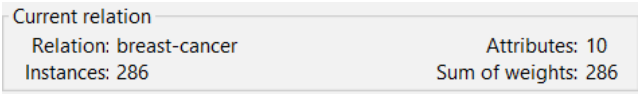
3. Detailed solutions

3.1 Installation

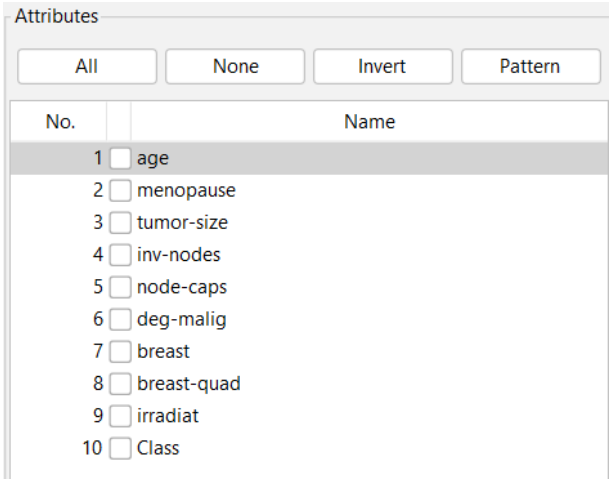
- Here is software opened with the “Explorer” menu on top



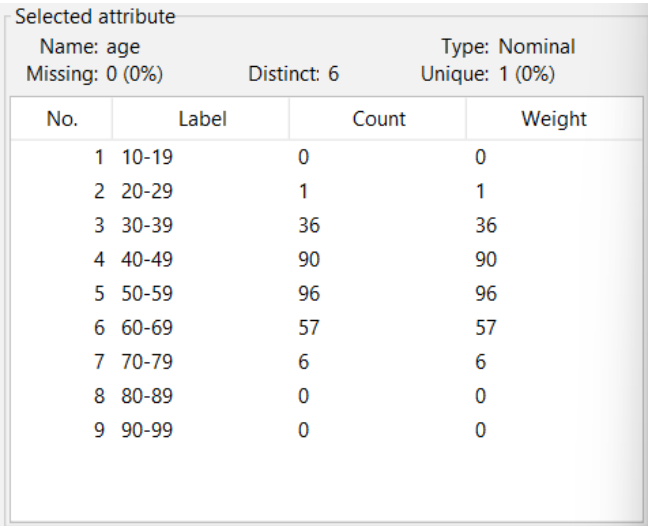
- **Current relation:** shows the current relation (dataset name), number of instances, attributes and the total sum of weights.



- **Attributes**: the list of attributes in the relation



- **Selected Attributes:** this tab shows the number of different values for the selected attributes, alongside its type and general report about the distribution of value and their weights

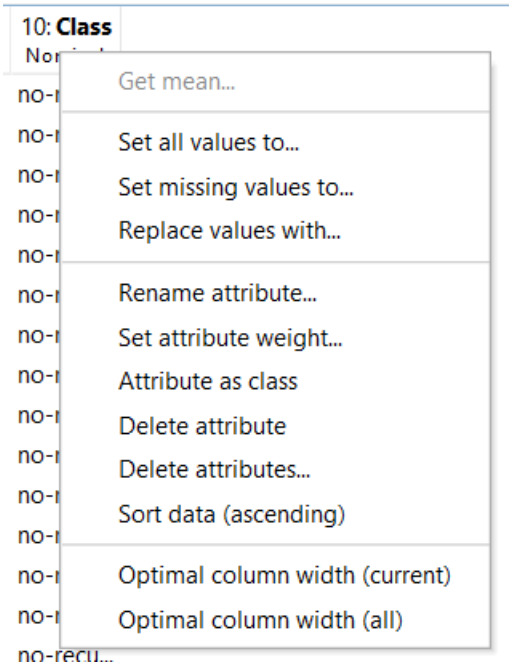


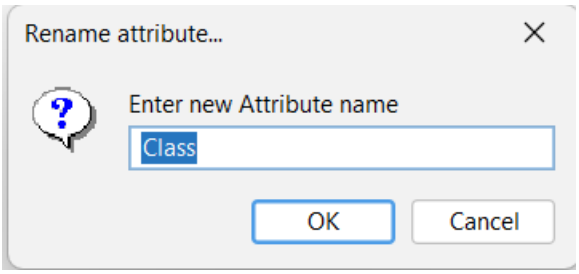
No.	Label	Count	Weight
1	10-19	0	0
2	20-29	1	1
3	30-39	36	36
4	40-49	90	90
5	50-59	96	96
6	60-69	57	57
7	70-79	6	6
8	80-89	0	0
9	90-99	0	0

3.2 Getting acquainted with WEKA

3.2.1 Exploring Breast Cancer data set

- This dataset has **286** instances
- This dataset has **10** attributes
- The **Class** attribute is used for the label, and it can be changed. The user can simply right-click on the **Class** attribute cell and choose "**Rename attribute**". We then get the following 2 screens:



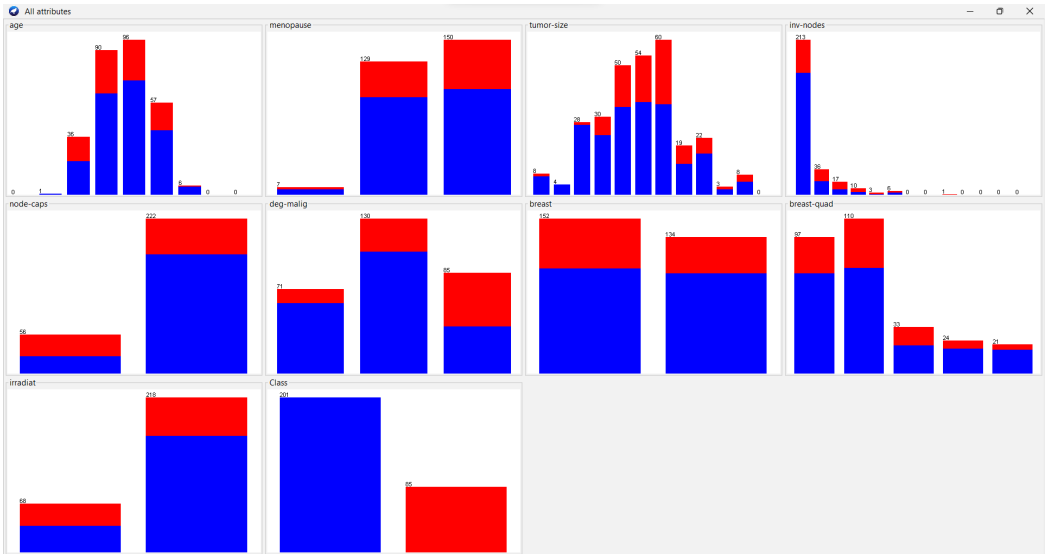


- Here are the meanings of each attribute:
    - **Age**: the patient's age
    - **Menopause**: the patient's stage of menopause (when a woman is longer capable of having a period)
    - **Tumor-size**: the patient's tumor's greatest diameter measurement in mm
    - **Inv-nodes**: the number of axillary lymph nodes containing metastatic breast cancer (range: 0 – 39)
    - **Node-caps:** if the cancer does metastasize to a lymph node, although outside the original site of the tumor it may remain "contained" by the capsule of the lymph node
    - **Deg-malig:** abbreviated version of degree of malignancy, the increasing value range denotes the abnormality of the tumors (range: 1 – 3)
    - **Breast:** the patient's breast that cancer may occur in
    - **Breast-quad:** the quadrant that the cancer takes place

    - **Irradiation**: reveals whether the patient has radiation therapy, which uses high-energy x-rays to destroy cancer cells.
    - **Class**: either **recurrent** or **no-current**, describing the patient's current cancer state
- There are two general ways to solve missing values, and that depends on the data set we're given.
    - **Filling missing values**: This can be done through using the column's average for numerical and continuous values or using the highest occurrence value for object values (string, date, …).
    - **Dropping missing values**: In cases where the missing values are too little to pay any attention to, it is better off to drop the instance containing the missing values. However, this could result in dropping a large number of instances due to having multiple columns containing missing values.

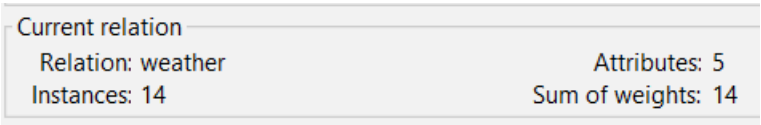
  Aside from the two mentioned above, there are other methods to do this, either by predicting the missing values, using k-nearest neighbors, or putting a fixed set of values depending on the data type.

- There are two attributes that have missing values: **Node-caps** and **Breast-quad**. For **Node-caps**, there are only two possible values **"Yes"** or **"No"**, we can fill either value depending on whether the patient has **recurrent** or **no-recurrent** breast cancer. Or fill them with a default value that appears the most. For **Breast-quad**, there seems to be only one missing value, this seems insignificant, and the value range for this attribute isn't binomial like **Node-caps**, so we're better off dropping the entire row.
- By clicking "**Visualize all**" in the chart section of the explorer, we can view the dataset distribution by ranges of values. Each column in the graph represents a different range of value that the attribute contains. Since all attributes are nominal, each column represented in the graph represents the number of instances of that nominal value. These columns are colored either red or blue, but in most cases they are colored both. This refers to the split in positive (red) and negative (blue) labels that every group belongs to. This visualizes the univariate distribution of all attributes with respect to the "**Class**" attribute.



3.2.2 Exploring Weather dataset.

- From this tab below we can identify that the dataset has 5 attributes(including class) with a total of 14 instances.



And from the table below we see that only two attributes have categorical data type, which are **temperature** and **humidity**. The data type used for the label **play** is categorical.

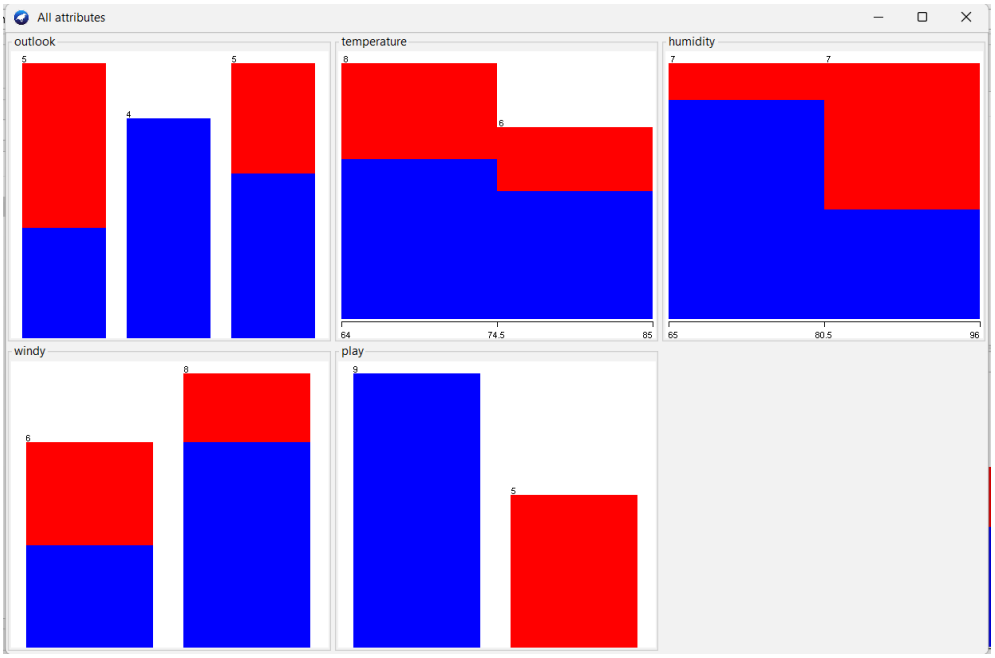
No.	1: outlook Nominal	2: temperature Numeric	3: humidity Numeric	4: windy Nominal	5: **play** Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no

- **Five-number summary:**

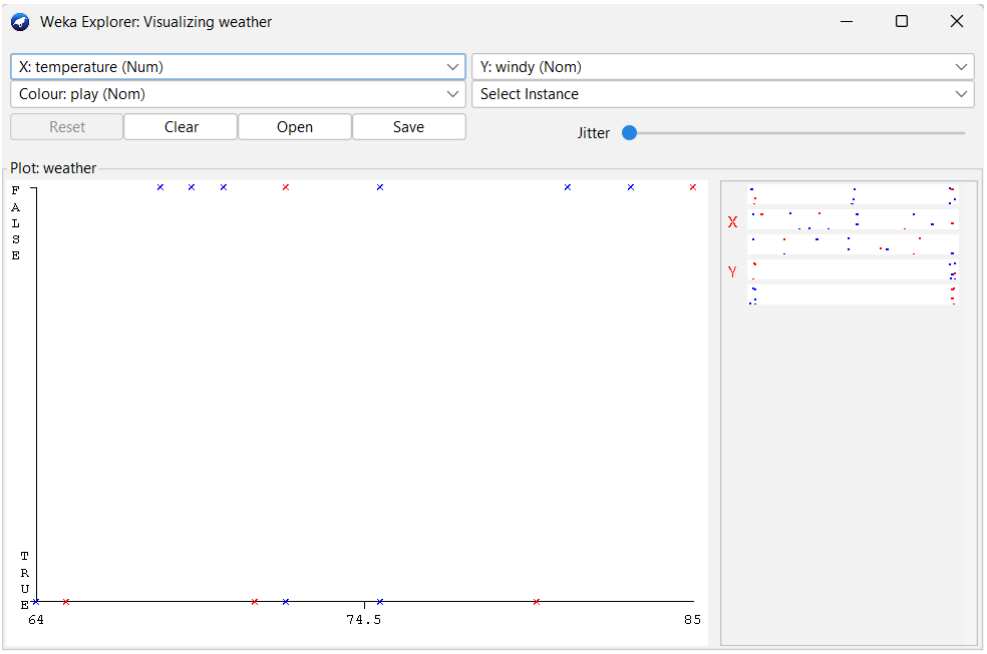
	Temperature	Humidity
Min	64	65
Max	85	96
Median	73.5	82.5
Q1	69.75	70
Q3	81.5	90.25

These calculations are not provided by WEKA by default except for Min and Max, the other statistics are calculated via Excel.

- Here is the all-chart-view of the dataset. And like the above breast cancer dataset, the nominal columns are represented by bar charts. However, the two **temperature** and **humidity** columns are now represented by a histogram since both attributes are numerical and continuous.

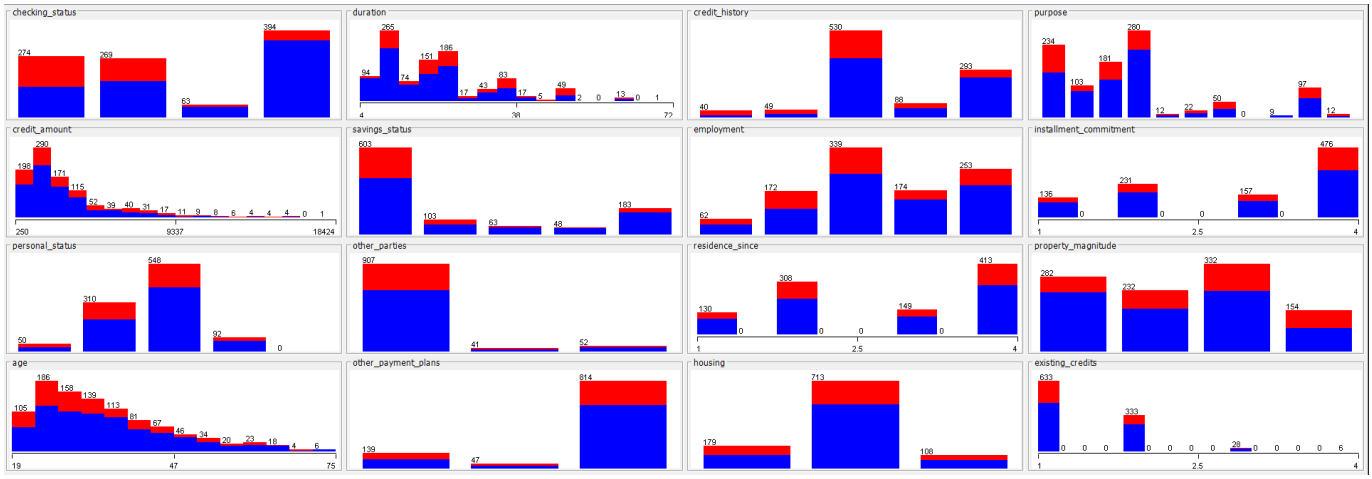


- These charts are called scatter chart, or scatter plot. Due to the dataset not having too many datapoints to properly visualize, the result might be inaccurate. However, we can see a clear separation in scatter points when viewing the graph between the **temperature** or **humidity** and other nominal values. This means that the temperature and/or humidity with other attributes have a close relationship with one another.

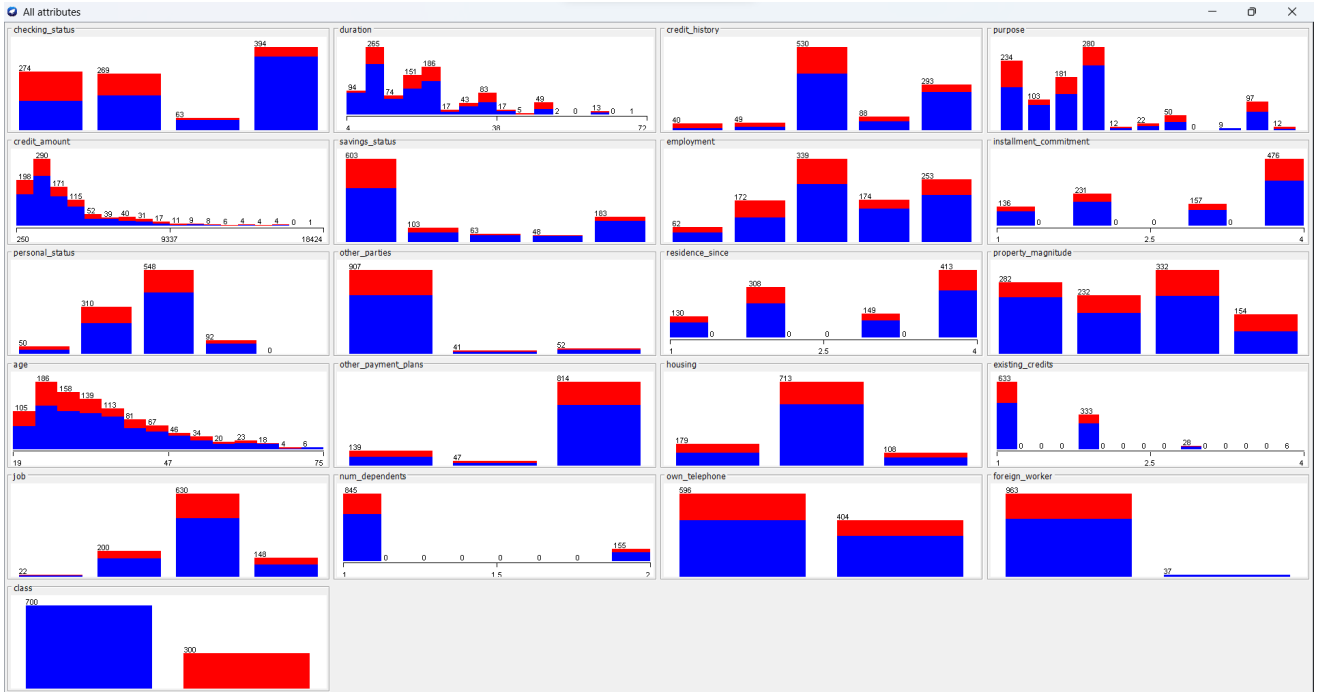


3.2.3 Exploring Credit in Germany dataset

- The comment content of the dataset when viewed with a text editor contains the title of the dataset, its source information and further information about the dataset. From here, we can see the dataset has 1000 instances with 21 classes (including the label). Here are the description of 5 attributes:
  - **Duration (numerical)**: duration in month
  - **Age (numerical)**: age in years
  - **Purpose (nominal):** purpose of credit usage (has 10 total possible values such as car, furniture/equipment, education, …)
  - **Housing (nominal):** type of housing (has 3 possible values: rent, own, for free)
  - **Foreign_worker (nominal):** checking whether the person is a foreign worker (has 2 Y/N values)
- The **class** attribute is used for the label
- The screenshot below shows a few numerical attributes in chart form, and we can see they are right skewed for the most part (duration, age, credit_amount, …).



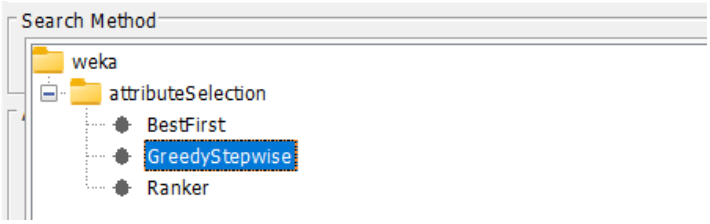
- All the charts from the attributes above can be seen below. These charts now show a clearer mix between numerical and nominal attributes as well as having a better represented graph due to having a lot more datapoints. Most if not all these numerical attributes do not follow normal distribution because they are skewed positively, as mentioned above. Whereas some other nominal attributes have a rather uneven distribution, this could pose a problem in oversampling.



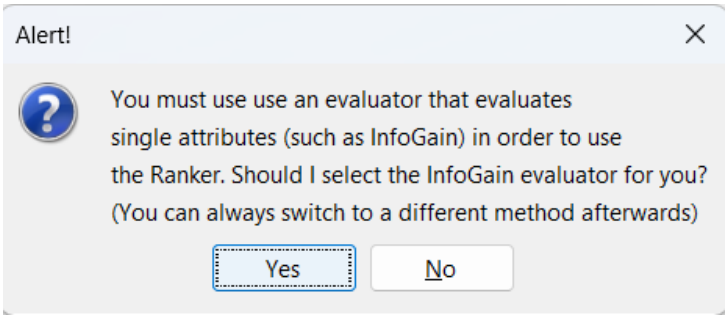
- There are two possible options to choose:
  - **Attribute evaluator:** this is the technique that evaluates the attributes in your dataset in the context of the output variable.
  - **Search method:** this is the technique that navigates through different combinations of attributes in the dataset to "handpick" some features that could suit the model the best

Both are used simultaneously to perform the process "Feature extraction" to reduce the input size of the dataset.

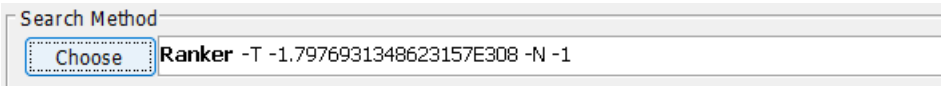
- Step-by-step for feature extraction:
  1. In order to select the top 5 attributes with the highest correlation, we must pick the search method that could "rank" these 5 attributes. This can be expanded to selecting the top n attributes. WEKA's default search methods allowed access to the **Ranker** search method, which evaluates every attribute correlation and ranks them. We start by selecting **Choose** in the **Search method** menu. Afterwards we choose **Ranker** in this menu.



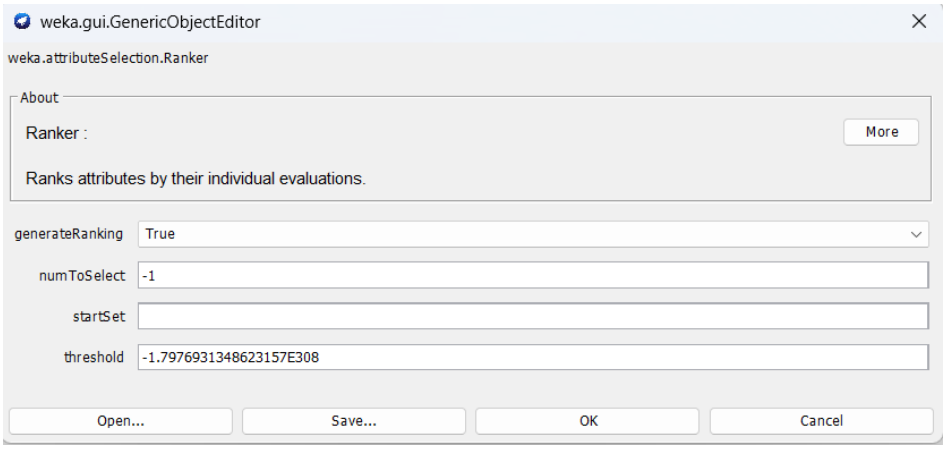
2. The following menu will pop up if your **Attribute Evaluator** isn't set to **InfoGain**. This is because **Ranker** requires the evaluator to be information gain, as it is an efficient way to rank the correlation between attributes. You can choose **Yes** in this menu for the program to automatically do it for you, or you can do it manually later.



3. We then click on the **Search method** tab

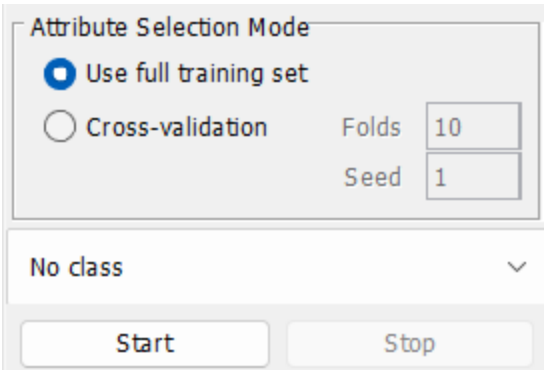


Which opens this menu:



Now we need to modify the **numToSelect** value to **5**, or any **n** value we want. When you're finished, click Ok.

4. Now navigate to this menu and press **Start**, attempting to change the **Attribute Selection Mode** to **K-Fold Cross-validation** will not produce the correct result. So it is advised against doing this.



5. This screenshot can be found in the bottom of the **Attribute selection output** tab, and that is the answer we're looking for. We have come to the conclusion that the 5 best ranked attributes are **checking_status, credit_history, duration, savings_status** and **purpose** respectively.

```
=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 21 class):
        Information Gain Ranking Filter

Ranked attributes:
 0.0947    1 checking_status
 0.0436    3 credit_history
 0.0329    2 duration
 0.0281    6 savings_status
 0.0249    4 purpose

Selected attributes: 1,3,2,6,4 : 5
```

3.3 Data preprocessing with Python

The below demonstration contains a snippet of code that showed the run command and its expected results after running the program in the terminal.

The default format of the run command is as below, for every test case you can run the command without any extra arguments for ease of usage, however some functions require extra arguments due to the nature of the problem.

python function.py data.csv

After running the command without passing any arguments, an output file with format **function_name.csv** would be created on the root directory. Here are the detailed instructions for every function and their corresponding test cases.

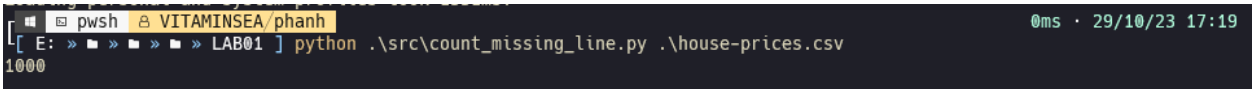
3.3.1 Extract columns with missing values

- Run command(s): **python .\src\list-missing.py .\house-prices.csv --out=[filename].csv**
- Test case: **python .\src\list-missing.py .\house-prices.csv**
- Expected result: a .csv file with the same name including all columns with missing values.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	LotFronta	Alley	MasVnrTy	MasVnrAr	BsmtQual	BsmtCond	BsmtExpo	BsmtFinTy	BsmtFinTy	Fireplace(	GarageTyp	GarageYrB	GarageFin	GarageQu	GarageCo	Fence	MiscFeature	
2	83		Stone	0	Gd	TA	Av	Unf	Unf		Attchd	2007	RFn	TA	TA			
3	70			0							CarPort	1962	Unf	TA	TA			
4	50			0	TA	TA	No	Unf	Unf	Gd	BuiltIn	1929	RFn	TA	TA			
5	52			0	Gd	TA	Mn	Rec	Unf		Detchd	1925	Unf	Fa	TA			
6	None			0	TA	TA	No	ALQ	Rec	Po	Attchd	1960	RFn	TA	TA	GdWo		
7	65			0	TA	TA	No	Unf	Unf		Detchd	1967	Unf	TA	TA			
8	80			0	TA	TA	No	Rec	Unf		Detchd	1963	Unf	TA	TA	MnPrv		
9	32		BrkFace	116	Ex	TA	No	GLQ	Unf		Attchd	1998	Unf	TA	TA			
10	71			0	Ex	TA	No	ALQ	Unf	TA	BuiltIn	2001	Fin	TA	TA			

3.3.2 Count the number of lines with missing data

- Run command(s): **There are no additional arguments to be passed in this function, you can use the basic command.**
- Test case: **python .\src\count_missing_line.py .\house-prices.csv**
- Expected result: the console will display a single number showing the number of row that contains missing data (a row is considered that way only requires at least one empty column)



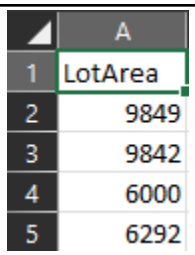
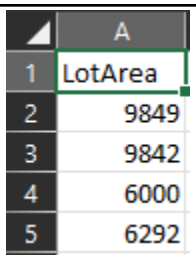
3.3.3 Fill in the missing value using mean, median (for numeric properties) and mode (for categorical properties)

- Run command(s): **python .\src\impute.py .\house-prices.csv --method=[default = median or mean, mode] --columns=[default = all columns or list_of_columns] --out=[filename].csv**
- Test case: run the commands below
  python .\src\impute.py .\house-prices.csv --method=mean --columns=LotArea --out=mean.csv

  python .\src\impute.py .\house-prices.csv --method=median --columns=LotArea --out=median.csv

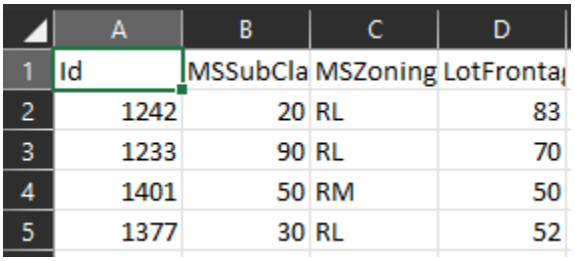
python .\src\impute.py .\house-prices.csv --method=mode --columns=MSZoning --out=mode.csv

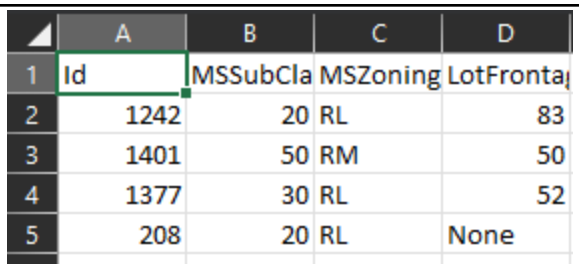
- Expected result: each run should create a file with the name and format like you have specified in the command arguments and the missing values in the original file should be replaced with the corresponding method. The **mean** and **median** method when called on categorical values will not work, refer to using **mode** instead.

		
mean	**median**	**mode**

3.3.4 Deleting rows containing more than a particular number of missing values

- Run command(s): **python .\src\deleteRows.py .\house-prices.csv --threshold=[default = 0.5 or self_defined_threshold]**
- Test case: **python .\src\deleteRows.py .\house-prices.csv --threshold=0.1**
- Expected result: the default value for the threshold argument is 0.5, hence running the command without any arguments passed should remove any rows with the number of missing columns that exceeds 50% of the total columns. Otherwise, the threshold will be set to the user-defined value, and the command will perform similarly.

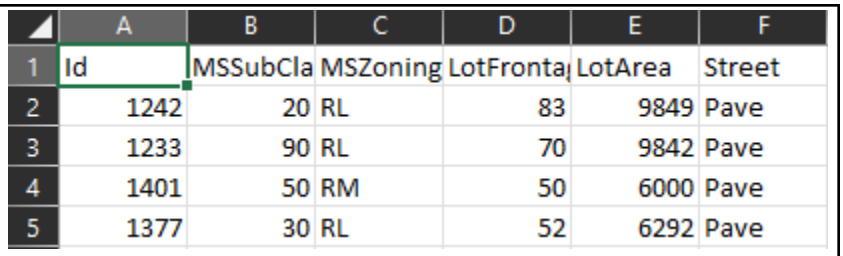
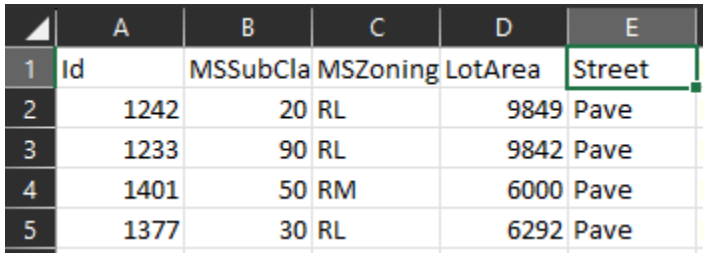
Before deleting	

After deleting	

	A	B	C	D
1	Id	MSSubCla	MSZoning	LotFronta
2	1242	20	RL	83
3	1401	50	RM	50
4	1377	30	RL	52
5	208	20	RL	None

3.3.5 Deleting columns containing more than a particular number of missing values

- Run command(s): **python .\src\deleteCols.py .\house-prices.csv --threshold=[self_defined_threshold]**
- Test case: **python .\src\deleteCols.py .\house-prices.csv --threshold=0.1**
- Expected result: the expected result is also similar to the section above, the only difference is that the command will list all columns with the percentage of missing values that exceeds the threshold.

Before deleting	
After deleting	

	A	B	C	D	E	F
1	Id	MSSubCla	MSZoning	LotFronta	LotArea	Street
2	1242	20	RL	83	9849	Pave
3	1233	90	RL	70	9842	Pave
4	1401	50	RM	50	6000	Pave
5	1377	30	RL	52	6292	Pave

	A	B	C	D	E
1	Id	MSSubCla	MSZoning	LotArea	Street
2	1242	20	RL	9849	Pave
3	1233	90	RL	9842	Pave
4	1401	50	RM	6000	Pave
5	1377	30	RL	6292	Pave

3.3.6 Delete duplicate samples

- Run command(s): **python .\src\deleteDuplicate.py .\house-prices.csv --out=[filename].csv**
- Test case: **python .\src\deleteDuplicate.py .\house-prices.csv**
- Expected result: the command will create a file that stores all rows from the original dataset without their duplicate.

Total rows before deleting	997 1190 60 RL 60 998 192 60 RL 999 990 60 FV 65 1000 982 60 RL 98 1001 862 190 RL 75
Total rows after deleting	713 985 90 RL 75 714 582 20 RL 98 715 668 20 RL 65 716 1190 60 RL 60 717 192 60 RL None

3.3.7 Normalize a numeric attribute using min-max and z-score methods

- Run command(s): **python .\src\scaling.py .\house-prices.csv --method=[default = z-score or min-max] --columns=[default = all columns or list_of_columns] --out=[filename].csv**
- Test case: run the commands below

  python .\src\scaling.py .\house-prices.csv --method=z-score --columns=LotArea --out=z-score.csv

  python .\src\scaling.py .\house-prices.csv --method=min-max --columns=LotArea --out=min-max.csv
- Expected result: each command will correspondingly create a file that contains all columns from the original dataset with their values normalized using either min-max scaling or z-score.

	A	B
1	LotArea	
2	-0.03953	
3	-0.04029	
4	-0.45877	
5	-0.42697	
6	0.24846808127927444	
7	-0.1381	
8	-0.15205	
9	-0.62216	
10	0.21753368991850458	

	A	B
1	LotArea	
2	0.039163953444856105	
3	0.039131	
4	0.021158	
5	0.022524418996295047	
6	0.051532502526103065	
7	0.034930391826653195	
8	0.034331611840874215	
9	0.014141499195389394	
10	0.050203959432655966	

z-score	min-max scaling

3.3.8 Performing addition, subtraction, multiplication, and division between two numerical attributes.

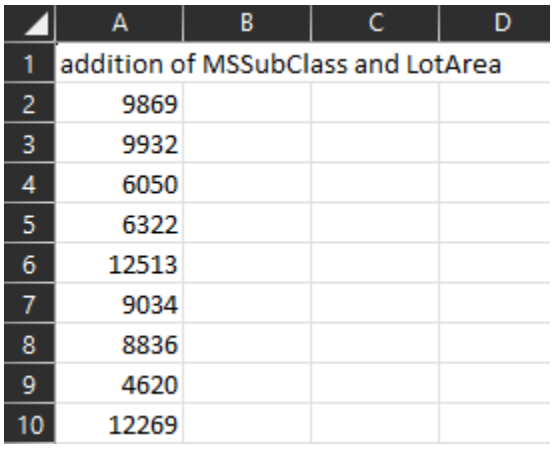
- Run command(s): **python .\src\ahrimeticColumnWise.py .\house-prices.csv [column_A] [column_B] --operation=[default = addition or multiplication, subtraction, division] --out=[filename].csv**
- Test cases: run the commands below

  python .\src\ahrimeticColumnWise.py .\house-prices.csv MSSubClass LotArea --operation=subtraction --out=subtraction.csv

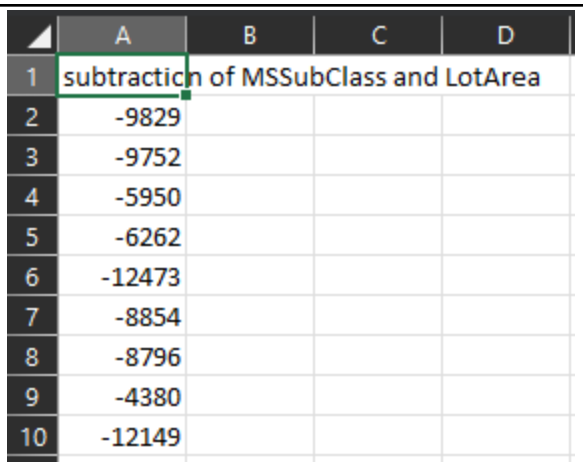
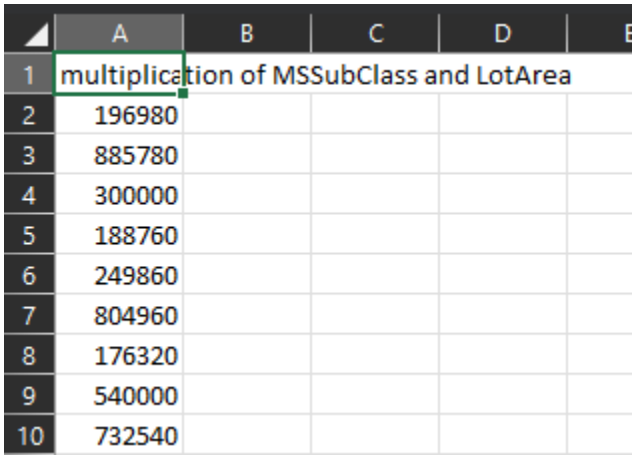
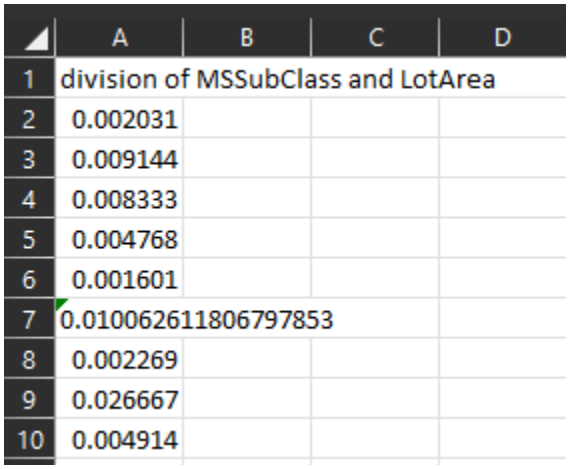
  python .\src\ahrimeticColumnWise.py .\house-prices.csv MSSubClass LotArea --operation=multiplication --out=multiplication.csv

  python .\src\ahrimeticColumnWise.py .\house-prices.csv MSSubClass LotArea --operation=division --out=division.csv

  python .\src\ahrimeticColumnWise.py .\house-prices.csv MSSubClass LotArea --operation=addition --out=addition.csv
- Expected result: This section requires extra mandatory arguments from the user which is the columns name of A and B. The operation will be performed from left to right, whichever variable comes first. Another requirement is that both columns must be numerical. If the operation flag is not called, **A + B** will be the default operation, and the default output will be a file called **columnWise.csv**. To customize output, refer to using the **–out** argument. The output after running the test cases are as follows:

Addition	

	A	B	C	D
1	addition of MSSubClass and LotArea			
2	9869			
3	9932			
4	6050			
5	6322			
6	12513			
7	9034			
8	8836			
9	4620			
10	12269			

Subtraction	
Multiplication	
Division	

4. References

- Weka 3 - Data Mining with Open Source Machine Learning Software in Java

- Lecture slides
- Build Command-Line Interfaces With Python's argparse