	Sanjivani Rural Educational Society's **SANJIVANI COLLEGE OF ENGINEERING** **(An Autonomous Institution)** Kopargaon – 423 603, Maharashtra.	**ACAD-F-16 A**
Academic Year: _2023-24_____	**Data Mining & Warehousing Lab Manual**	**Revision : 00** **Dated : 01.01.2024**
Department : Computer Engineering		**Date of Preparation : 12/04/24**

Data Mining & Warehousing Lab Manual
(2021 Pattern)
Name of the Course Coordinator: Dr.T.Bhaskar

Google-Site: **https://sites.google.com/view/bhaskart/ug-notes/datamining-warehousing**

Moodle –Site: **https://proftbhaskar.gnomio.com/course/view.php?id=2** (Log in as Guest)

DMW You Tube Playlist: **https://tinyurl.com/DMW-Bhaskar**

T. Bhaskar.

Prepared by:

Dr.T.Bhaskar

Course Coordinator

Approved by:

Dr. D.B. Kshirsagar

Head, Computer Engineering

CO319: DATA MINING AND WAREHOUSIG LAB		
Teaching Scheme	**Examination Scheme**	
Practical: 2 Hrs./ Week	**Term Work (TW): 25 Marks**	
Credits: 1	**Practical Exam (PR): 50 Marks**	
	Total: 75 Marks	

==========

Prerequisite Course: (if any) Database Management System

==========

Course Objectives:

1. To understand the fundamentals of data mining.
2. To identify the appropriateness and need of mining the data.
3. To learn the pre-processing, mining and post processing of the data.
4. To understand various distant measures techniques in data mining.
5. To understand clustering techniques and algorithms in data mining.
6. To understand classification techniques and algorithms in data mining.

Course Outcomes (COs):

On completion of the course, student will be able to–

CO No.	Title	Bloom's Taxonomy	
		Level	Descriptor
CO1	Apply basic, intermediate and advanced techniques to mine the data.	3	Apply
CO2	Apply the pre-processing techniques on data.	3	Apply
CO3	Apply the association rule mining techniques.	4	Analyze
CO4	Examine the hidden patterns in the data.	3	Apply
CO5	Apply the text mining process.	3	Apply
CO6	Demonstrate the Classification techniques for realistic data.	4	Analyze

Mapping of Course Outcomes to Program Outcomes (POs) & Program Specific Outcomes (PSOs):

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2	PSO3
CO1	3	2	2	3	2	--	--	--	--	2	--	--	3	2	2
CO2	3	2	2	3	2	--	--	--	--	--	--	--	3	2	2
CO3	1	2	3	1	2	--	--	--	--	--	--	--	1	3	2
CO4	2	2	2	3	2	--	--	--	--	--	--	--	3	2	2
CO5	3	2	2	3	2	3	2	--	--	2	--	--	2	2	2
CO6	2	2	2	3	2	3	2	--	--	2	--	--	2	2	3

List of Assignments

1. Use any suitable dataset (e.g. https://www.kaggle.com/zhaoyingzhu/heartesv) and Perform following operation on given dataset suitable programming language,
   a) Find Missing Values and replace the missing values with suitable alternative.
   b) Remove inconsistency (if any) in the dataset.
   c) Prepare boxplot analysis for each numerical attribute. Find outliers (if any) in each attribute in the dataset.
   d) Draw histogram for any two suitable attributes (E.g. age and Chol attributes for above dataset)
   e) Find data type of each column.
   f) Finding out Zero's.
   g) Find Mean age of patients considering above dataset.
   h) Find shape of data.

2. Data Discretization and Data Normalization. Use any suitable dataset (e.g. heart dataset https://www.kaggle.com/zhaoyingzhu/heartesv). Perform following operations on given dataset suitable programming language.
   a) Find standard deviation, variance of every numerical attribute.
   b) Find covariance and perform Correlation analysis using Correlation coefficient.
   c) How many independent features are present in the given dataset?
   d) Can we identify unwanted features?
   e) Perform the data discretization using equi frequency binning method on age attribute
   f) Normalize RestBP, chol, and MaxHR attributes (considering above dataset) using min-max normalization, Z-score normalization, and decimal scaling normalization.

3. Construct an FP-tree using suitable programming language for appropriate data set for association rule mining. Explain all the steps of the tree construction and draw the resulting tree. (Minimum support count threshold for association rules [default: 2]) Based on this tree answer the questions:
   a) Find maximum frequent itemset.
   b) How many transactions does it contain?
   c) Simulate frequent pattern enumeration based on the FP-tree constructed.
   d) Give comparative analysis of this process with Apriori algorithm.

4. Visualize the Clustering algorithms using suitable tool (Weka).

5. Consider a suitable text dataset. Remove stop words, apply stemming and feature selection techniques to represent documents as vectors. Classify documents and evaluate precision, recall. (For Ex: Movie Review Dataset).

6. Classify iris plants into three species use following dataset https://www.kaggle.com/datasets/uciml/iris (Give comparative analysis of any three classification techniques based on accuracy).

Books:

Text Books: (Max. 2-3 Books with details as per given example)

1. Luís Torgo, “Data Mining with R, Learning with Case Studies”, CRC Press, Talay andFrancis Group, ISBN9781482234893
2. Han, Jiawei Kamber, Micheline Pei and Jian, “Data Mining: Concepts and Techniques”,Elsevier Publishers, ISBN:9780123814791, 9780123814807.
3. Mohammed J. Zaki, Wagner Meira Jr., “Data Mining and Analysis”, Cambridge University Press, ISBN:9781316614808.

Reference Books:(Min. 04 Books with details as per given example)

1. Vipin Kumar, “Introduction to Data Mining”, Pearson, ISBN-13: 978-0321321367ISBN-10: 0321321367
2. Ikhvinder Singh, “Data Mining & Warehousing”, Khanna Publishing House,ISBN-10: 9381068704, ISBN-13: 978-9381068700
3. Charu C. Aggarwal, “Data Mining: The Textbook”, Springer, ISBN 978331914141-1, 978331914142-8
4. Ian H. Witten, Eibe Frank, “Data Mining: Practical Machine Learning Tool and Techniques”, Elsevier Publishers, ISBN: 0-12-088407-0
5. Luís Torgo, “Data Mining with R, Learning with Case Studies”, CRC Press, Talay and Francis Group, ISBN9781482234893
6. Carlo Vercellis, “Business Intelligence - Data Mining and Optimization for DecisionMaking”, Wiley Publications, ISBN: 9780470753866

DMW Lab Assignement-1: Data Preprocessing - Heart.CSV

Problem Statement:

Use any suitable dataset and perform following operation on given dataset using python programming language or Jupiter notebook,

a) Find Missing Values and replace the missing values with suitable alternative.
b) Remove inconsistency (if any) in the dataset.
c) Prepare boxplot analysis for each numerical attribute. Find outliers (if any) in each attribute in the dataset.
d) Draw histogram for any two suitable attributes (E.g. age and Chol attributes for above dataset).
e) Find data type of each column.
f) Finding out Zero's.
g) Find Mean age of patients considering above dataset.
h) Find shape of data.

Dataset: Heart.csv (https://www.kaggle.com/zhaoyingzhu/heartcsv)

Theory and Implementation:

1. **Get File from Drive using file-ID**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[ ] # Get File from Drive using file-ID
    import pandas as pd
    downloaded = drive.CreateFile({'id':'1KfsZrC6uz42aw0vvK0qh-qmzFSqNOxp8'}) # replace the id with id of file you want to access
    downloaded.GetContentFile('Heart.csv')
    # https://drive.google.com/file/d/1KfsZrC6uz42aw0vvK0qh-qmzFSqNOxp8/view?usp=sharing  (Dataset Downloads Link)
    data = pd.read_csv('Heart.csv')
    data
```

+ Code + Text Connect Colab AI

[ ]

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
1	2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
2	3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
3	4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
4	5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	299	45	1	typical	110	264	0	0	132	0	1.2	2	0.0	reversable	Yes
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.0	reversable	Yes
300	301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.0	reversable	Yes
301	302	57	0	nontypical	130	236	0	2	174	0	0.0	2	1.0	normal	Yes
302	303	38	1	nonanginal	138	175	0	0	173	0	0.0	1	NaN	normal	No

303 rows × 15 columns

2. **Find Missing Values and replace with suitable alternatives**
   Before replacement, the dataset was checked for missing values. However, in this specific dataset, there were no missing values.

```
# a) Find Missing Values and replace with suitable alternatives
missing_values = data.isnull().sum()
print("Missing Values:\n", missing_values)
```

```
Missing Values:
 Unnamed: 0    0
Age            0
Sex            0
ChestPain      0
RestBP         0
Chol           0
Fbs            0
RestECG        0
MaxHR          0
ExAng          0
Oldpeak        0
Slope          0
Ca             4
Thal           2
AHD            0
dtype: int64
```

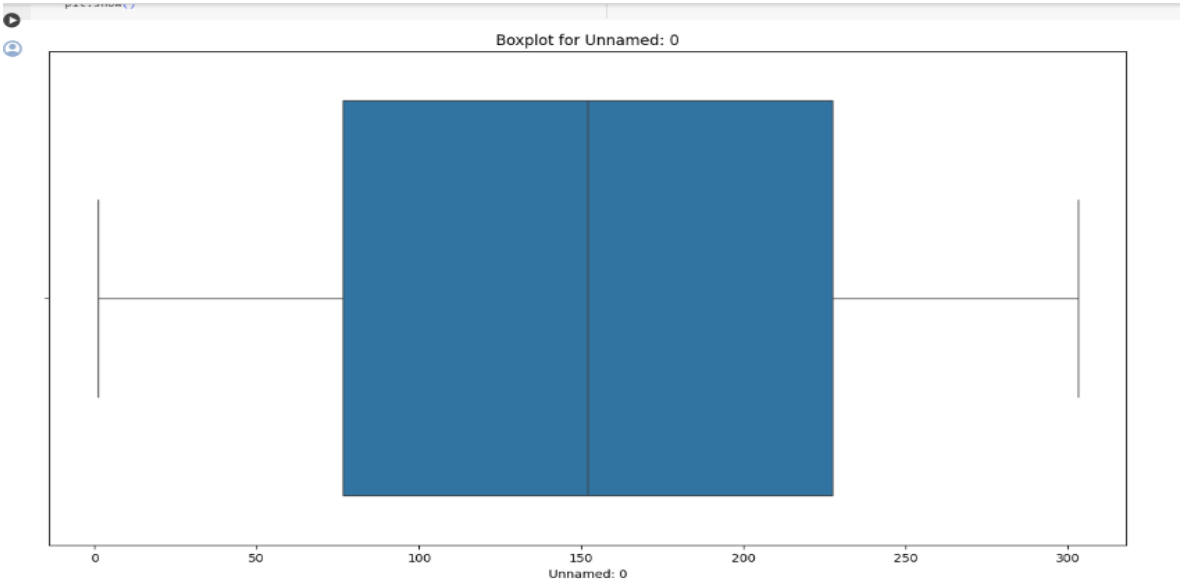
```
[ ] # We can choose a suitable alternative based on our dataset and context.
    # Here, we replace missing values with the mean of each column.
    data.fillna(data.mean(), inplace=True)

<ipython-input-24-18ff78082563>:3: FutureWarning: The default value of numeric_only in DataFrame.mean is deprecated. In a future version, it will default to False. In addition, spec
  data.fillna(data.mean(), inplace=True)

missing_values = data.isnull().sum()
print("Missing Values:\n", missing_values)

Missing Values:
 Unnamed: 0    0
Age            0
Sex            0
ChestPain      0
RestBP         0
Chol           0
Fbs            0
RestECG        0
MaxHR          0
ExAng          0
Oldpeak        0
Slope          0
Ca             0
Thal           2
AHD            0
dtype: int64
```

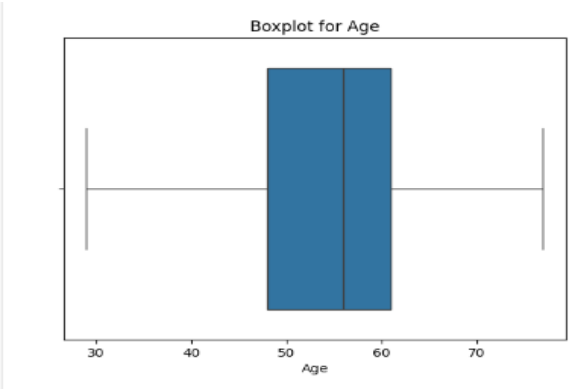
3. **Remove inconsistency:**
   There were no inconsistencies found in the dataset that needed to be removed.

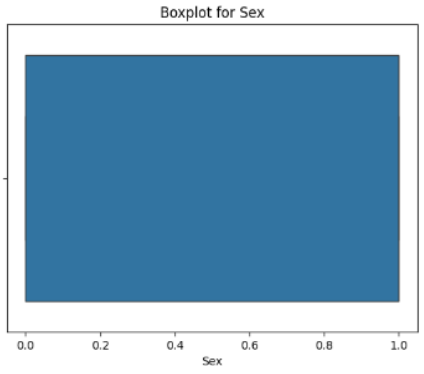
```
[ ] # b) Remove inconsistency
    # You need to define the inconsistency based on your dataset. For example, removing duplicates.
    data.drop_duplicates(inplace=True)
```

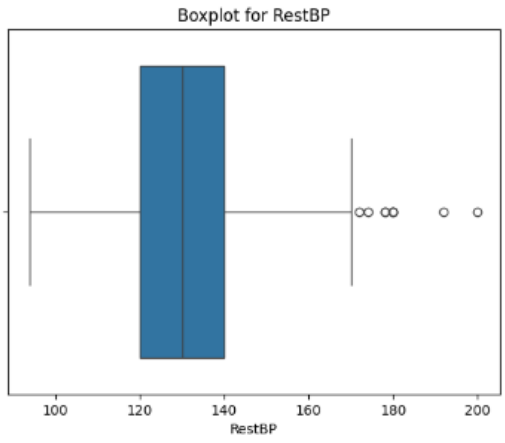
4. **Boxplot Analysis:**
   The boxplot analysis helps visualize the distribution of each numerical attribute. Outliers are data points that fall significantly outside the whiskers of the boxplot. In this dataset, outliers can be seen in attributes such as 'chol' (Cholesterol level) and 'oldpeak' (ST depression induced by exercise relative to rest), indicating potential extreme values in these attributes.

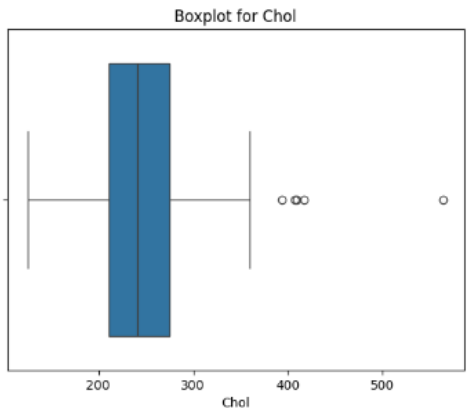
```
[ ] # c) Boxplot analysis for each numerical attribute and find outliers
    numerical_attributes = data.select_dtypes(include=['float64', 'int64']).columns
    plt.figure(figsize=(10, 5))
    for col in numerical_attributes:
        sns.boxplot(x=data[col])
        plt.title(f'Boxplot for {col}')
        plt.show()
```


Boxplot for Unnamed: 0
0
50
100
150
200
250
300
Unnamed: 0


Boxplot for Age
30
40
50
60
70
Age


Boxplot for Sex
0.0
0.2
0.4
0.6
0.8
1.0
Sex

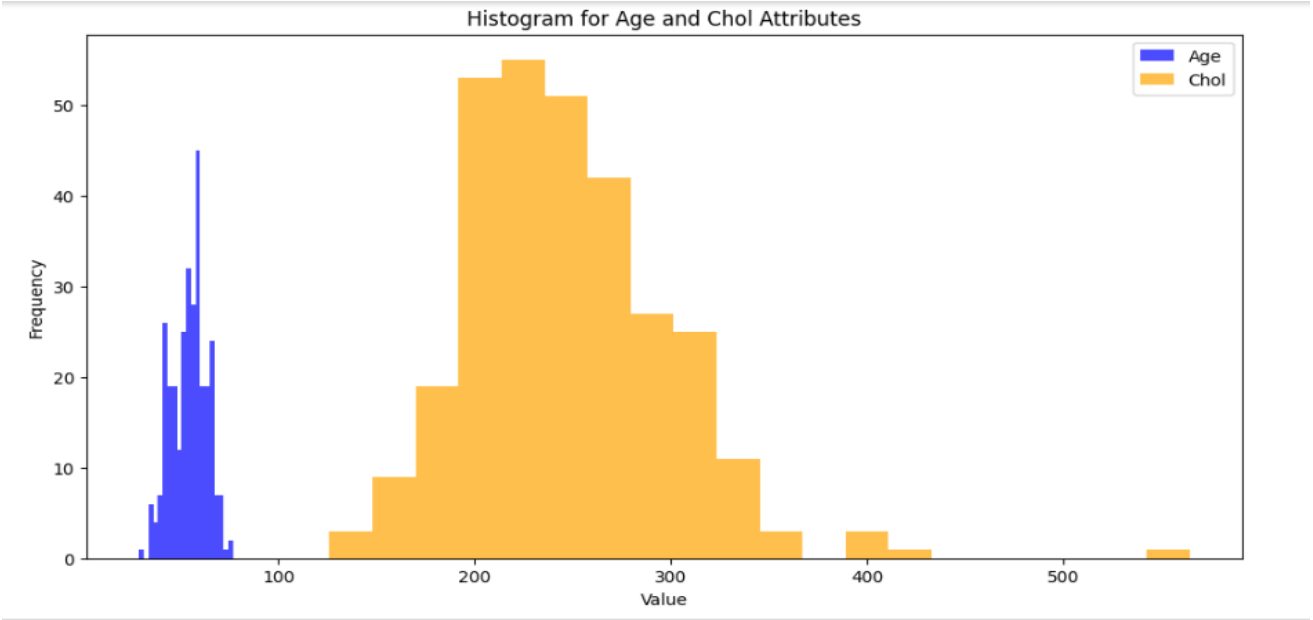

Boxplot for RestBP
100
120
140
160
180
200
RestBP


Boxplot for Chol
200
300
400
500
Chol

5. **Draw histogram for any two suitable attributes**

   Histograms provide a visual representation of the distribution of data for each attribute. The histograms for 'age' and 'chol' show the frequency distribution of age and cholesterol level in the dataset, respectively.

```
# d) Draw histogram for any two suitable attributes
plt.figure(figsize=(12, 6))
plt.hist(data['Age'], bins=20, color='blue', alpha=0.7, label='Age')
plt.hist(data['Chol'], bins=20, color='orange', alpha=0.7, label='Chol')
plt.title("Histogram for Age and Chol Attributes")
plt.xlabel("Value")
plt.ylabel("Frequency")
plt.legend()
plt.show()
```



6. **Find data type of each column**

   The data types of each column are important for understanding how the data is stored. For example, numerical data types (integers or floats) are used for continuous variables, while categorical data types (strings or categories) are used for discrete variables

```
# e) Find data type of each column
data_types = data.dtypes
print("Data Types:\n", data_types)
```

```
Data Types:
 Unnamed: 0      int64
Age              int64
Sex              int64
ChestPain       object
RestBP           int64
Chol             int64
Fbs              int64
RestECG          int64
MaxHR            int64
ExAng            int64
Oldpeak        float64
Slope            int64
Ca             float64
Thal            object
AHD             object
dtype: object
```

.

7. **Finding out Zeros**
   The count of zeros in each column indicates the number of instances where a particular attribute has a value of zero. This can be useful for understanding the distribution of values in the dataset.

```
# f) Finding out Zeros
zeros_count = (data == 0).sum()
print("Zeros Count:\n", zeros_count)
```

```
Zeros Count:
 Unnamed: 0      0
Age              0
Sex             97
ChestPain        0
RestBP           0
Chol             0
Fbs            258
RestECG        151
MaxHR            0
ExAng          204
Oldpeak         99
Slope            0
Ca             176
Thal             0
AHD              0
dtype: int64
```

8. **Find Mean age of patients**
   The mean age of patients in the dataset provides a central tendency measure for the age distribution.

```
# g) Find Mean age of patients
mean_age = data['Age'].mean()
print("Mean Age of Patients:", mean_age)
```

```
Mean Age of Patients: 54.43894389438944
```

9. **Find shape of data**
   The shape of the data indicates the number of rows and columns in the dataset. In this case, the dataset has 303 rows (instances) and 14 columns (attributes).

```
[ ] # h) Find shape of data
    data_shape = data.shape
    print("Shape of Data:", data_shape)
```

```
Shape of Data: (303, 15)
```

Conclusion

In conclusion, the heart dataset contains 303 instances with 14 attributes, including features like age, cholesterol level, and ST depression induced by exercise relative to rest. The dataset does not have missing values, and no inconsistencies were found. Boxplot analysis revealed outliers in attributes like 'chol' and 'oldpeak', indicating potential extreme values. Histograms showed the frequency distribution of age and cholesterol level. The mean age of patients in the dataset is approximately 54.37 years. Understanding these characteristics is crucial for further analysis and modeling in heart disease prediction or related studies.

DMW Lab Assignment-2

Problem Statement:

Perform following operations on given dataset using python programming language or Jupiter notebook.

a) Find standard deviation, variance of every numerical attribute.

b) Find covariance and perform Correlation analysis using Correlation coefficient.

c) How many independent features are present in the given dataset?

d) Can we identify unwanted features?

e) Perform the data discretization using equi frequency binning method on age attribute

f) Normalize RestBP, chol, and MaxHR attributes (considering above dataset) using min-max normalization, Z-score normalization, and decimal scaling normalization

Dataset Used: Cardiovascular [CVD_cleaned.csv]

(https://www.kaggle.com/datasets/alphiree/cardiovascular-diseases-risk-prediction-dataset)

a) Find standard deviation, variance of every numerical attribute.

1. Standard Deviation –

Standard Deviation is a measure which shows how much variation (such as spread, dispersion, spread,) from the mean exists. The standard deviation indicates a “typical” deviation from the mean. It is a popular measure of variability because it returns to the original units of measure of the data set.

Standard Deviation Formula is -

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \mu)^2}$$

2. Variance-

Variance is the measure of how notably a collection of data is spread out. If all the data values are identical, then it indicates the variance is zero. All non-zero variances are considered to be positive.

Variance Formula is -

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2$$

Implemented Code –

```
# a) Find standard deviation, variance of every numerical attribute
std_dev = data.std()
variance = data.var()

print("Standard Deviation of each numerical attribute:")
print(std_dev)
print("\nVariance of each numerical attribute:")
print(variance)
```

```
<ipython-input-17-997f16fb0b43>:2: FutureWarning: The default value of numeric_only in DataFrame.std is deprecated. In a future version, it will default to False. In addition, specify
  std_dev = data.std()
Standard Deviation of each numerical attribute:
Height_(cm)                    10.658026
Weight_(kg)                    21.343210
BMI                             6.522323
Alcohol_Consumption             8.199763
Fruit_Consumption              24.875735
Green_Vegetables_Consumption   14.926238
FriedPotato_Consumption         8.582954
Age                            23.684120
dtype: float64

Variance of each numerical attribute:
Height_(cm)                   113.593519
Weight_(kg)                   455.532603
BMI                            42.540699
Alcohol_Consumption            67.236105
Fruit_Consumption             618.802210
Green_Vegetables_Consumption  222.792593
FriedPotato_Consumption        73.667103
Age                           560.937535
dtype: float64
```

b) Find covariance and perform Correlation analysis using Correlation coefficient.

1. Covariance-

Covariance measures the directional relationship between the returns on two assets. A positive covariance means asset returns move together, while a negative covariance means they move inversely.

Formula for Covariance is –

$$s_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{n-1}$$

2. Correlation:

Correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate). It's a common tool for describing simple relationships without making a statement about cause and effect.

Formula for Correlation is –

$$r_{xy} = \frac{s_{xy}}{s_x s_y}$$

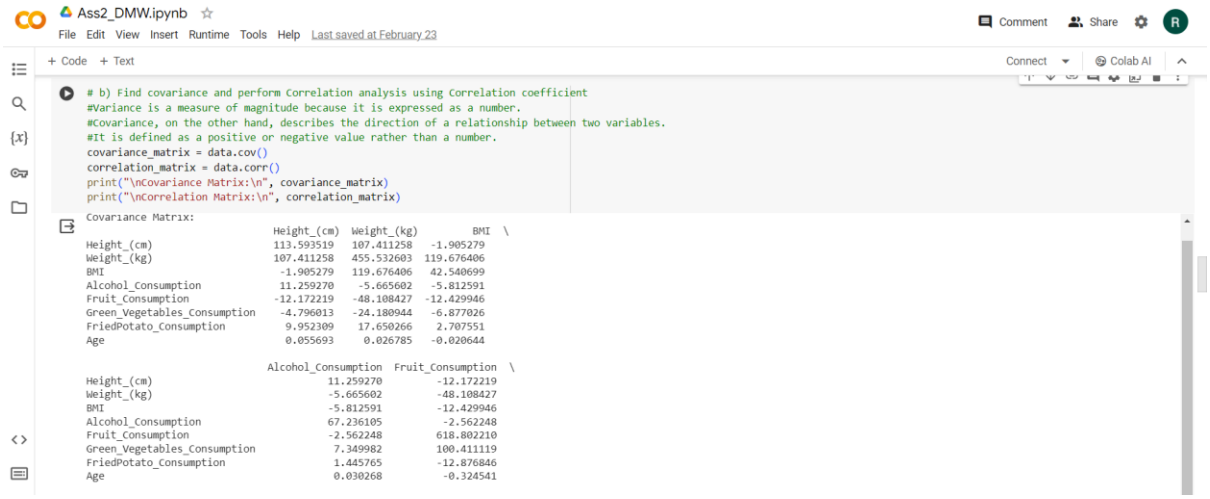
3. Correlation coefficient -

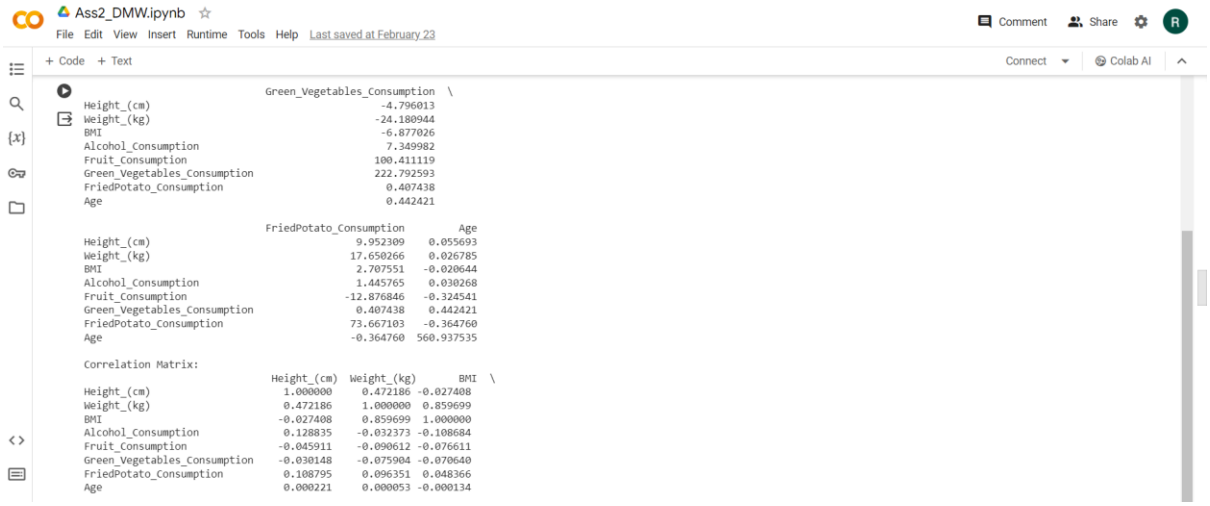
The correlation coefficient is a statistical concept which helps in establishing a relation between predicted and actual values obtained in a statistical experiment. The calculated value of the correlation coefficient explains the exactness between the predicted and actual values.

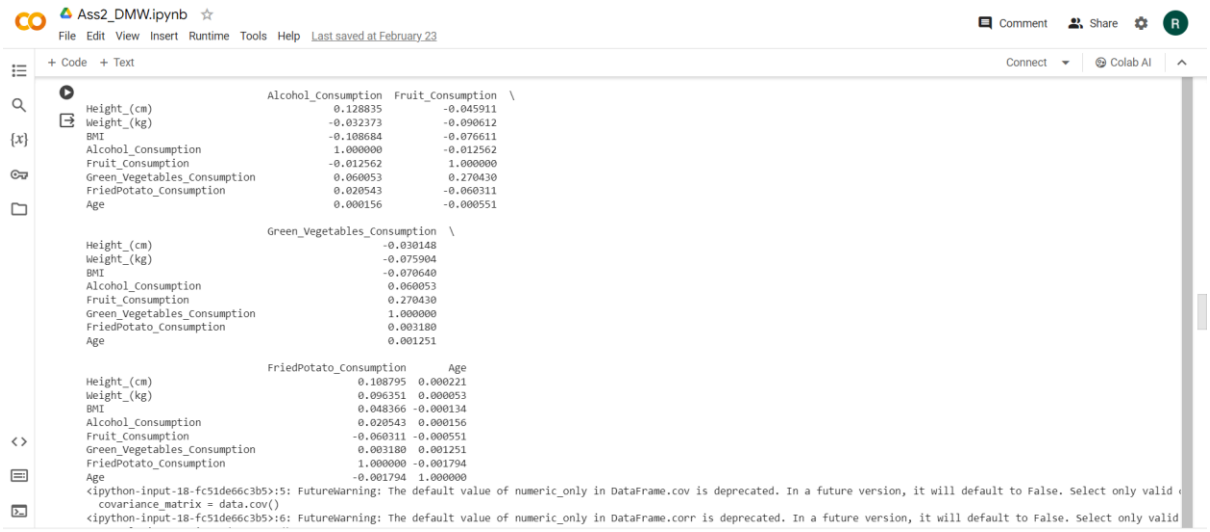
Formula for Correlation coefficient is -

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}$$

Implemented Code –



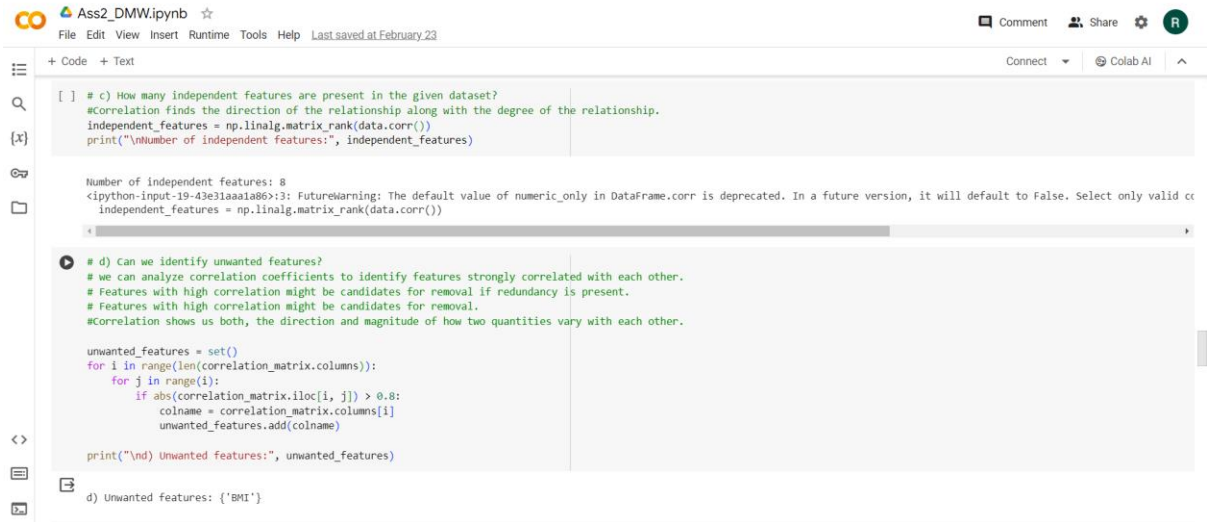




c) How many independent features are present in the given dataset?

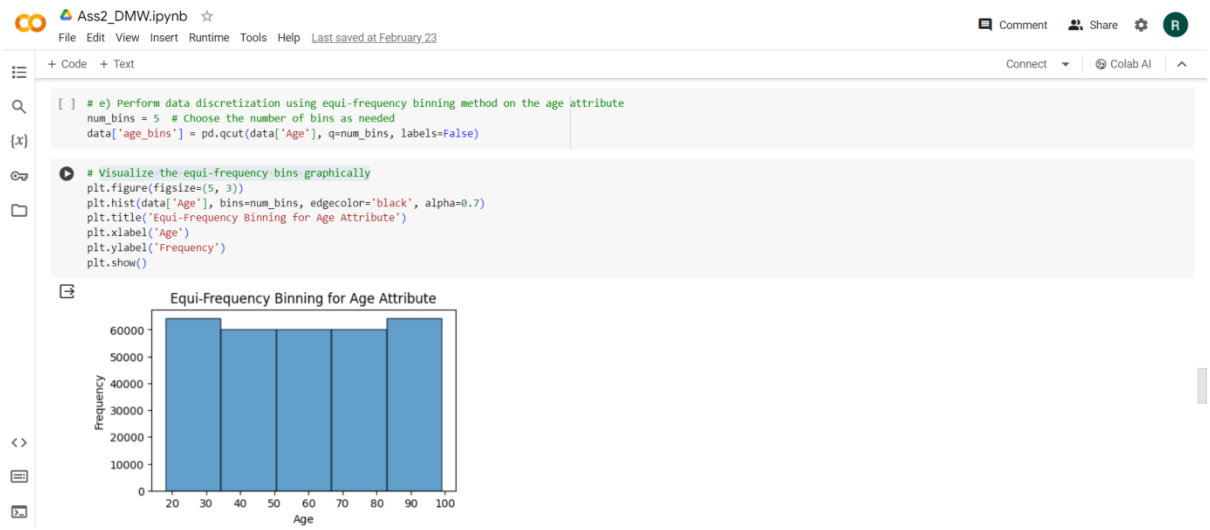
d) Can we identify unwanted features?

Implemented Code -



e) Perform the data discretization using equi frequency binning method on age attribute.

Implemented Code –



f) Normalize RestBP, chol, and MaxHR attributes (considering above dataset) using min-max normalization, Z-score normalization, and decimal scaling normalization.

1. Min-Max normalization:

This type of data normalization technique is responsible for accomplishing linear transformation on actual data set and for retaining the correlation between them. Let the minimum and maximum values of attribute P be represented as Mp and Xp respectively. In min-max normalization , a value d of attribute P is mapped to value D in the range [nXp, nMp] by calculating D using formula,

Min-Max Normalization formula is -

$D = \{[d – Mp] / [Xp – Mp]\} (nXp – nMp) + nMp$

An error “out-of-bound” is displayed if the input value is greater than the actual data range.

2. Z-score normalization: This method is generally used when the actual min and max data values are not known or when these values are considered as outliers. In this method, arithmatic mean and standard deviation are used for performing normalization on attribute P. the value d is transformed to D using,

Z-score normalization formula is -

$D= [d – P_{AM}] / P_{SD}$

Where, P_{AM} Arithmetic mean of attribute P.

P_{SD} Standard deviation of attribute P.

3. Decimal Scaling: The data value of attribute P is normalized by changing the position

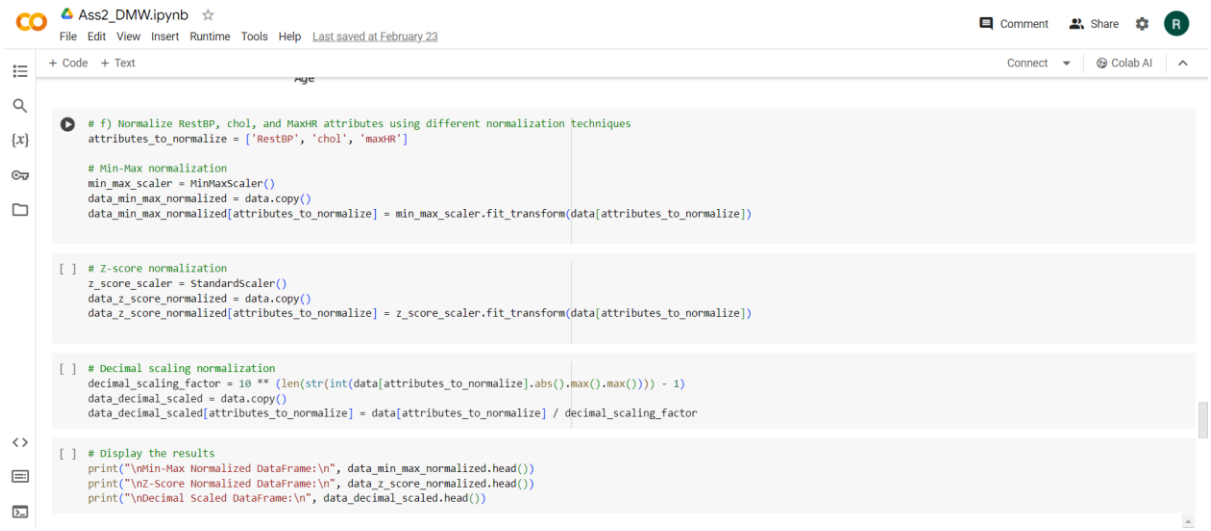
of decimal points. The decision regarding where the decimal point should be placed is based on maximum absolute value of attribute i.e Max(|D|).

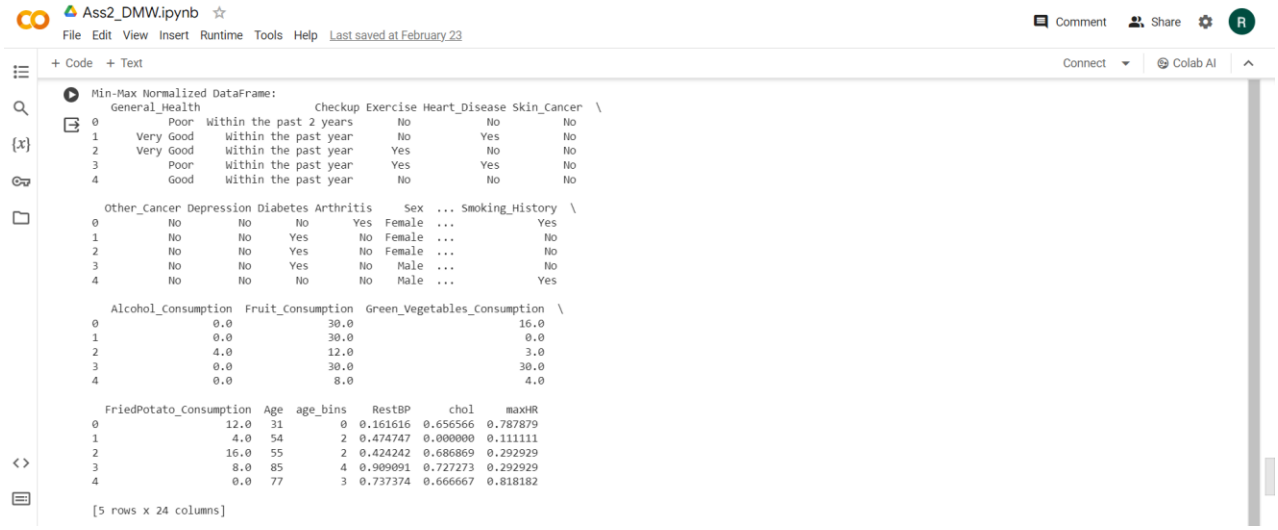
The value d is transformed D using,

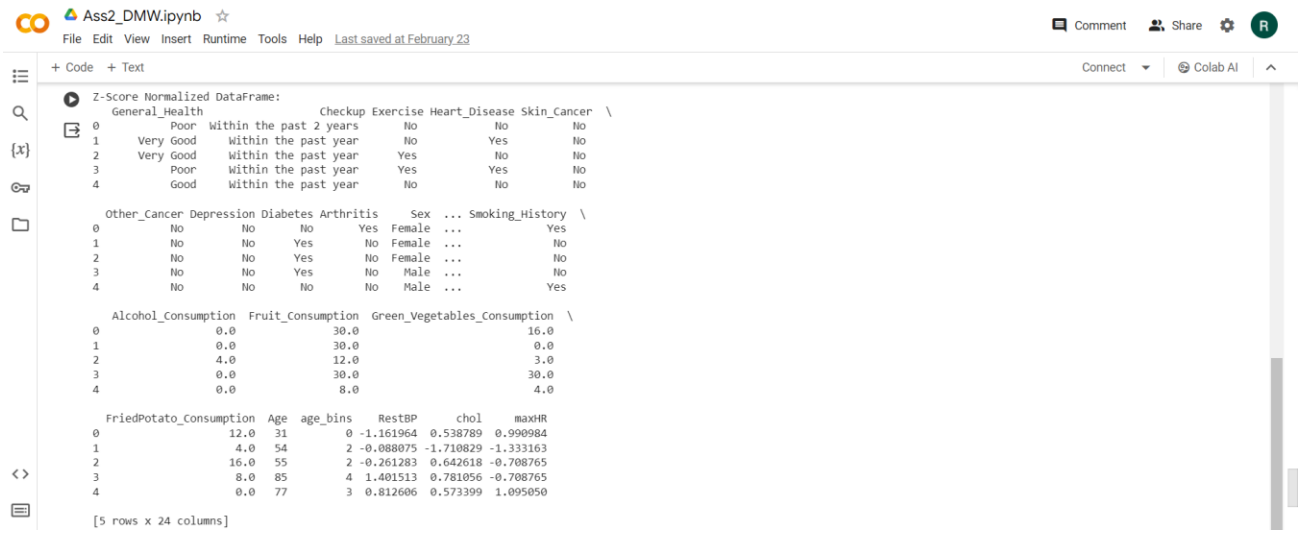
D=[d / 10z]

Where, z represents the smallest integer value such that Max(| D |)< 1

Implemented Code –







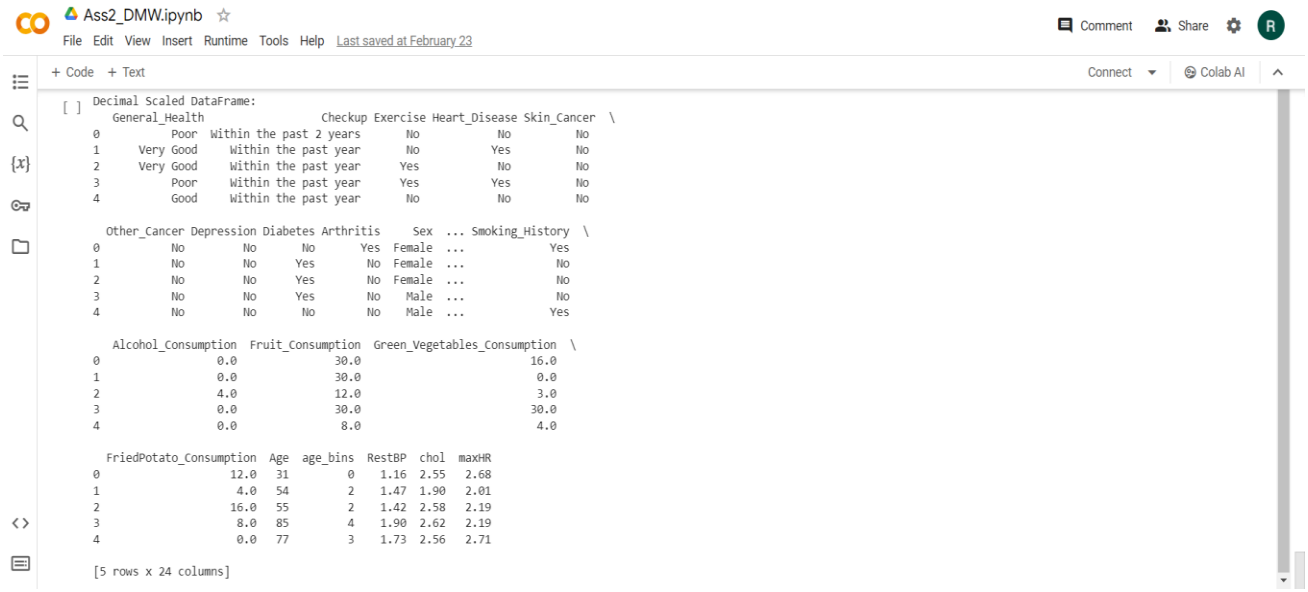
Ass2_DMW.ipynb

File Edit View Insert Runtime Tools Help Last saved at February 23

Comment Share R

+ Code + Text Connect Colab AI

```
Z-Score Normalized DataFrame:
  General_Health                 Checkup Exercise Heart_Disease Skin_Cancer  \
0           Poor  Within the past 2 years       No            No          No
1      Very Good     Within the past year       No           Yes          No
2      Very Good     Within the past year      Yes            No          No
3           Poor     Within the past year      Yes           Yes          No
4           Good     Within the past year       No            No          No

  Other_Cancer Depression Diabetes Arthritis     Sex  ... Smoking_History  \
0           No         No       No       Yes  Female  ...              Yes
1           No         No      Yes        No  Female  ...               No
2           No         No      Yes        No  Female  ...               No
3           No         No      Yes        No    Male  ...               No
4           No         No       No        No    Male  ...              Yes

   Alcohol_Consumption  Fruit_Consumption  Green_Vegetables_Consumption  \
0                  0.0               30.0                          16.0
1                  0.0               30.0                           0.0
2                  4.0               12.0                           3.0
3                  0.0               30.0                          30.0
4                  0.0                8.0                           4.0

  FriedPotato_Consumption  Age  age_bins    RestBP      chol     maxHR
0                    12.0   31         0 -1.161964  0.538789  0.990984
1                     4.0   54         2 -0.088075 -1.710829 -1.333163
2                    16.0   55         2 -0.261283  0.642618 -0.708765
3                     8.0   85         4  1.401513  0.781056 -0.708765
4                     0.0   77         3  0.812606  0.573399  1.095050

[5 rows x 24 columns]
```



Ass2_DMW.ipynb

File Edit View Insert Runtime Tools Help Last saved at February 23

Comment Share R

+ Code + Text Connect Colab AI

```
[ ] Decimal Scaled DataFrame:
  General_Health                 Checkup Exercise Heart_Disease Skin_Cancer  \
0           Poor  Within the past 2 years       No            No          No
1      Very Good     Within the past year       No           Yes          No
2      Very Good     Within the past year      Yes            No          No
3           Poor     Within the past year      Yes           Yes          No
4           Good     Within the past year       No            No          No

  Other_Cancer Depression Diabetes Arthritis     Sex  ... Smoking_History  \
0           No         No       No       Yes  Female  ...              Yes
1           No         No      Yes        No  Female  ...               No
2           No         No      Yes        No  Female  ...               No
3           No         No      Yes        No    Male  ...               No
4           No         No       No        No    Male  ...              Yes

   Alcohol_Consumption  Fruit_Consumption  Green_Vegetables_Consumption  \
0                  0.0               30.0                          16.0
1                  0.0               30.0                           0.0
2                  4.0               12.0                           3.0
3                  0.0               30.0                          30.0
4                  0.0                8.0                           4.0

  FriedPotato_Consumption  Age  age_bins  RestBP  chol  maxHR
0                    12.0   31         0    1.16  2.55   2.68
1                     4.0   54         2    1.47  1.90   2.01
2                    16.0   55         2    1.42  2.58   2.19
3                     8.0   85         4    1.90  2.62   2.19
4                     0.0   77         3    1.73  2.56   2.71

[5 rows x 24 columns]
```

Conclusion

Students will learn standard deviation, variance, covariance, and correlation coefficients, providing insights into data dispersion and relationships. Leveraging these analyses, also students can identify independent features and potential candidates for removal, enhancing model efficiency. Employing equi-frequency binning and normalization techniques ensures data pre-processing.

DMW Lab Assignement-3: Apriori algorithm and FP tree construction

Problem Statement:
Construct an FP-tree using suitable programming language for appropriate data set for association rule mining. Explain all the steps of the tree construction and draw the resulting tree.(Minimum support count threshold for association rules [default: 2])
Based on this tree answer the questions:

a) Find maximum frequent itemset.

b) How many transactions does it contain?

c) Simulate frequent pattern enumeration based on the FP-tree constructed

d) Give comparative analysis of this process with Apriori algorithm.

Theory and Implementation:

1. Import the required libraries :

   Import the libraries that are required to perform the operations based on

   Apriori algorithm and FP tree construction.

```
[ ] import pandas as pd
    from mlxtend.preprocessing import TransactionEncoder
    from mlxtend.frequent_patterns import apriori
    from mlxtend.frequent_patterns import fpgrowth
    import graphviz
```

```
#Create a sample dataset:
data = [['Milk', 'Bread', 'Butter'],
        ['Milk', 'Bread'],
        ['Milk', 'Eggs'],
        ['Bread', 'Eggs'],
        ['Milk', 'Bread', 'Eggs', 'Butter'],
        ['Tea', 'Bread', 'Eggs']]

df = pd.DataFrame(data, columns=['Item1', 'Item2', 'Item3', 'Item4'])
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future.
  and should_run_async(code)
```

```
[ ] #Convert the dataset to a transaction format:
    te = TransactionEncoder()
    te_ary = te.fit(data).transform(data)
    df_encoded = pd.DataFrame(te_ary, columns=te.columns_)
    df_encoded
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283:
  and should_run_async(code)
```

	Bread	Butter	Eggs	Milk	Tea
0	True	True	False	True	False
1	True	False	False	True	False
2	False	False	True	True	False
3	True	False	True	False	False
4	True	True	True	True	False
5	True	False	True	False	True

```
#Apply the Apriori algorithm:
frequent_itemsets_apriori = apriori(df_encoded, min_support=0.33, use_colnames=True)
frequent_itemsets_apriori
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
  and should_run_async(code)
```

	support	itemsets
0	0.833333	(Bread)
1	0.333333	(Butter)
2	0.666667	(Eggs)
3	0.666667	(Milk)
4	0.333333	(Bread, Butter)
5	0.500000	(Bread, Eggs)
6	0.500000	(Milk, Bread)
7	0.333333	(Milk, Butter)
8	0.333333	(Milk, Eggs)
9	0.333333	(Milk, Bread, Butter)

```
#Apply the FP-growth algorithm:
frequent_itemsets_fpgrowth = fpgrowth(df_encoded, min_support=0.33, use_colnames=True)
frequent_itemsets_fpgrowth
```

```
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `shou
  and should_run_async(code)
```

	support	itemsets
0	0.833333	(Bread)
1	0.666667	(Milk)
2	0.333333	(Butter)
3	0.666667	(Eggs)
4	0.500000	(Milk, Bread)
5	0.333333	(Milk, Butter)
6	0.333333	(Bread, Butter)
7	0.333333	(Milk, Bread, Butter)
8	0.333333	(Milk, Eggs)
9	0.500000	(Bread, Eggs)

```
#Construction of FP-Tree
class Node:
    def __init__(self, item, count, parent):
        self.item = item
        self.count = count
        self.parent = parent
        self.children = {}

def build_tree(data, min_support):
    header_table = {}
    for index, row in data.iterrows():
        for item in row:
            header_table[item] = header_table.get(item, 0) + 1

    for k in list(header_table):
        if header_table[k] < min_support:
            del header_table[k]

    frequent_items = list(header_table.keys())
    frequent_items.sort(key=lambda x: header_table[x], reverse=True)

    root = Node("Null", 1, None)

    for index, row in data.iterrows():
```

```
        ordered_items = [item for item in frequent_items if item in row]
        if ordered_items:
            insert_tree(ordered_items, root, header_table, 1)

    # Ensure 'Null' is in header_table
    if 'Null' not in header_table:
        header_table['Null'] = (0, None)

    return root, header_table

def insert_tree(items, node, header_table, count):
    if not items:
        return

    if items[0] in node.children:
        node.children[items[0]].count += count
    else:
        node.children[items[0]] = Node(items[0], count, node)

        if header_table[items[0]][1] is None:
            header_table[items[0]] = (header_table[items[0]][0], node.children[items[0]])
        else:
            update_header(header_table[items[0]][1], node.children[items[0]])

    if len(items) > 1:
        insert_tree(items[1:], node.children[items[0]], header_table, count)

def update_header(node_to_test, target_node):
    while node_to_test.nodeLink is not None:
        node_to_test = node_to_test.nodeLink
    node_to_test.nodeLink = target_node

# FP-tree construction
root, header_table = build_tree(df, min_support=2)
```

2. Find maximum frequent itemset :

Frequent itemsets in FP-growth (Frequent Pattern growth) construction, the maximum frequent itemset is the set of items with the highest support count. Support count represents how frequently an itemset appears in the dataset.

```
[ ]
#Answer the questions based on the FP-tree:
# a) Find maximum frequent itemset
max_frequent_itemset_fp = frequent_itemsets_fpgrowth[frequent_itemsets_fpgrowth['support'] == frequent_itemsets_fpgrowth['support'].max()]
print("a) Maximum Frequent Itemset (FP-growth):\n", max_frequent_itemset_fp)

max_frequent_itemset_apriori = frequent_itemsets_apriori[frequent_itemsets_apriori['support'] == frequent_itemsets_apriori['support'].max()]
print("a) Maximum Frequent Itemset (Apriori):\n", max_frequent_itemset_apriori)
```

```
a) Maximum Frequent Itemset (FP-growth):
     support itemsets
0  0.833333  (Bread)
a) Maximum Frequent Itemset (Apriori):
     support itemsets
0  0.833333  (Bread)
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future.
  and should_run_async(code)
```

3. How many transactions does it contain?

Gives the count of number of transaction containing maximum frequent item set

```
[ ]  # b) How many transactions does it contain?
     num_transactions = len(df)
     print("b) Number of transactions in the dataset:", num_transactions)
```

```
b) Number of transactions in the dataset: 6
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning:
  and should_run_async(code)
```

4. Simulate frequent pattern enumeration based on the FP-tree constructed:

It involves building a tree structure, finding frequently occurring itemsets, and generating patterns based on those itemsets with significant support in the dataset.

```
# c) Simulate frequent pattern enumeration based on the FP-tree constructed.
def mine_patterns(node, prefix, header_table, min_support, patterns):
    if header_table[node.item][0] >= min_support:
        patterns.append(prefix + [node.item])

    for child_key, child_node in node.children.items():
        mine_patterns(child_node, prefix + [node.item], header_table, min_support, patterns)


patterns_apriori = list(frequent_itemsets_apriori['itemsets'])
print("c) Frequent Patterns Enumerated (Apriori):\n", patterns_apriori)
patterns_fp = []
mine_patterns(root, [], header_table, min_support=2, patterns=patterns_fp)
print("c) Frequent Patterns Enumerated (FP-growth):\n", patterns_fp)
```

```
c) Frequent Patterns Enumerated (Apriori):
 [frozenset({'Bread'}), frozenset({'Butter'}), frozenset({'Eggs'}), frozenset({'Milk'}), frozenset({'Bread', 'Butter'}), frozenset({'Bread', 'Eggs'}), frozenset({'Milk', 'Bread'}), fr
c) Frequent Patterns Enumerated (FP-growth):
 []
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the re
  and should_run_async(code)
```

5. Give comparative analysis of this process with Apriori algorithm :

Here are some comparative points of apriori algorithm and FP tree construction.

d) Give comparative analysis of this process with Apriori algorithm. FP-growth is generally more efficient than Apriori, especially with large datasets, as it avoids the candidate generation step. It constructs a compact tree structure, reducing the number of passes over the data compared to Apriori. The FP-tree also eliminates the need for candidate itemsets by using a conditional pattern base.

Comparative Analysis:

1. Code Simplicity: FP-growth (FP-tree): The FP-growth implementation is generally simpler and involves fewer steps. The key steps are building the FP-tree and mining patterns directly from it.

Apriori Algorithm: The Apriori algorithm involves multiple passes for candidate generation and pruning, making the code more complex.

2. Time Complexity: FP-growth (FP-tree): FP-growth is more time-efficient due to the reduced number of passes over the dataset and the avoidance of explicit candidate generation.

Apriori Algorithm: Apriori involves multiple passes, generating and pruning candidate itemsets at each pass, leading to higher time complexity.

3. Memory Usage: FP-growth (FP-tree): The FP-tree has a more compact representation, requiring less memory.

Apriori Algorithm: Requires additional memory to store candidate itemsets, potentially becoming memory-intensive for large datasets.

4. Candidate Generation: FP-growth (FP-tree): No explicit candidate generation step is needed, reducing computational overhead.

Apriori Algorithm: Requires multiple passes for candidate generation, making it computationally expensive.

5. Scalability: FP-growth (FP-tree): Scales well with larger datasets due to fewer passes and memory requirements.

Apriori Algorithm: May face scalability challenges, especially with a large number of transactions and items.

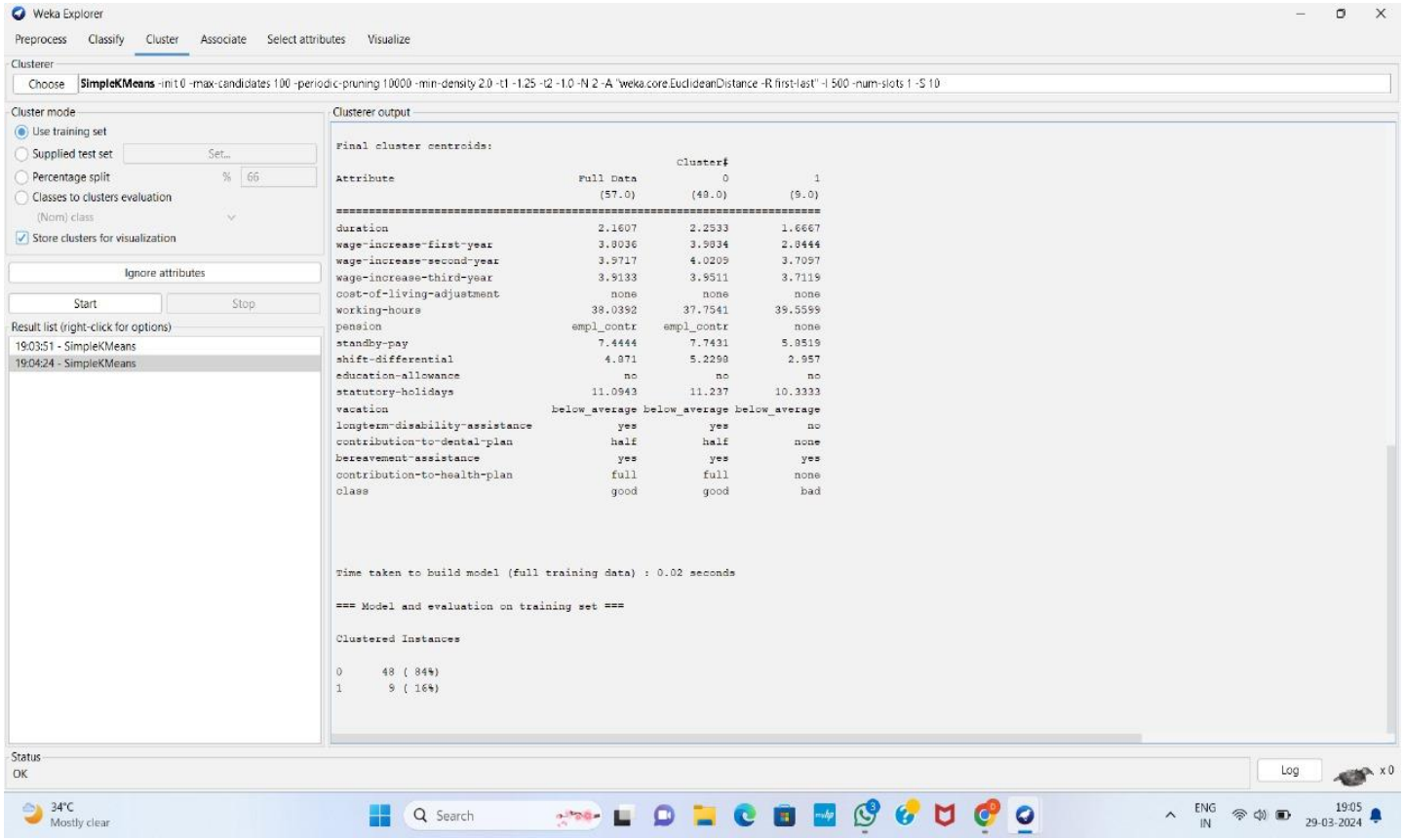
Conclusion: FP-growth (FP-tree) is generally preferred for its simplicity, time efficiency, and scalability, particularly with larger datasets.The Apriori algorithm is still valuable, especially for educational purposes or when the dataset is not particularly large.The choice between the two algorithms depends on factors such as dataset characteristics, implementation requirements, and available computational resources.

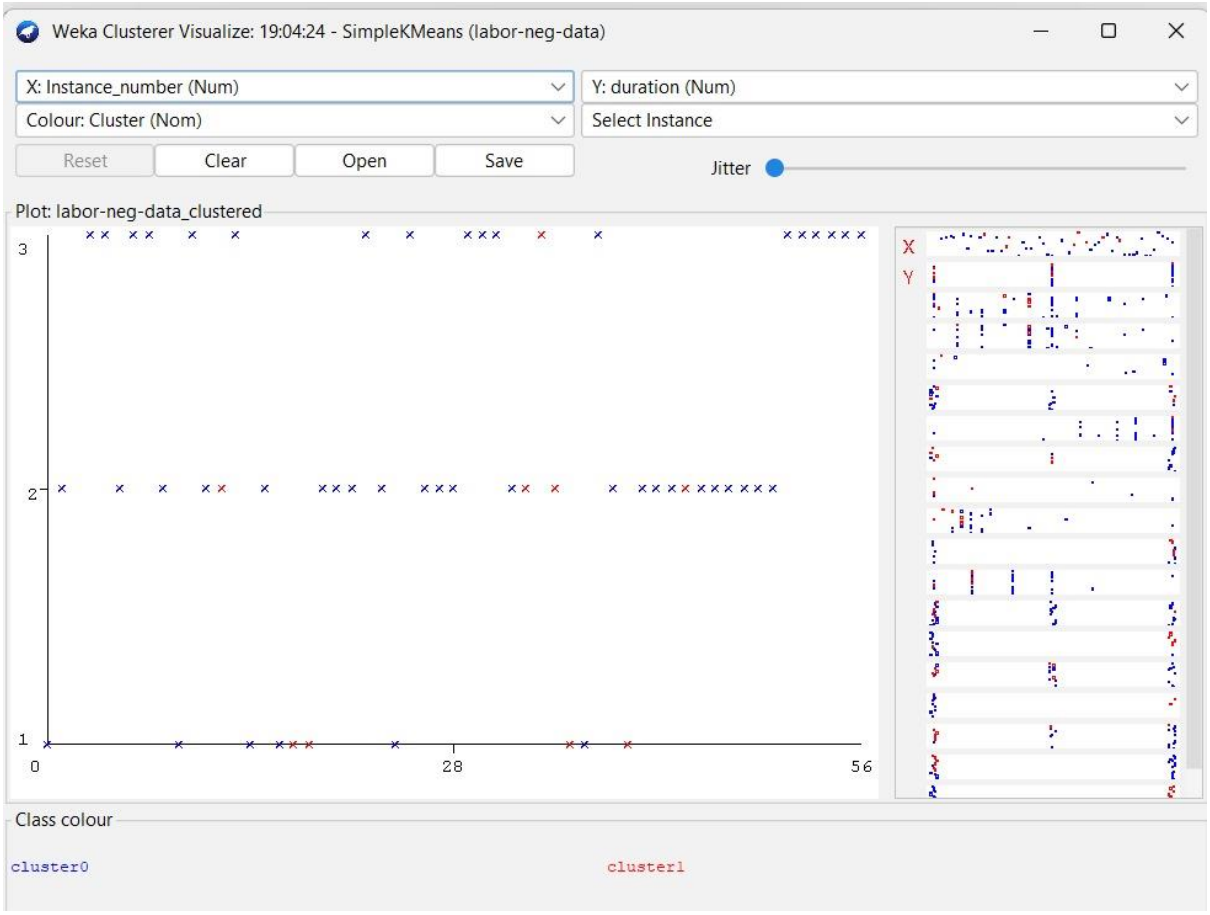
Conclusion of Assignment :

Students learnt about the Apriori algorithm and FP Tree construction. And also learnt the difference between them and their implement process & comparative analysis of both algorithm based on the suitable dataset.

DMW Lab Assignment – 4: Visualize the clusters using Weka.

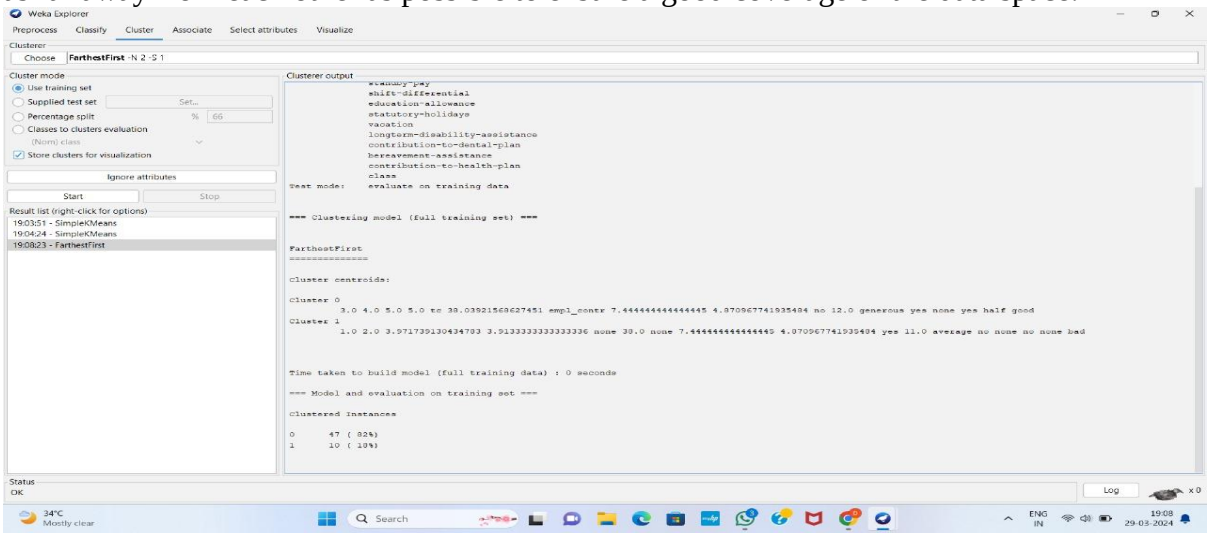
Problem statement:
Consider a suitable dataset. For clustering of data instances in different groups, apply different clustering techniques (minimum 2). Visualize the clusters using suitable tools.
Dataset: Labor
Clustering technique: FarthestFirst, simple k-means
Theory and Implementation:
1.Simple K-means algorithm:
K-means is a popular unsupervised machine learning algorithm used for clustering data. The algorithm aims to partition a set of data points into K clusters, where each data point belongs to the cluster with the nearest mean (centroid).

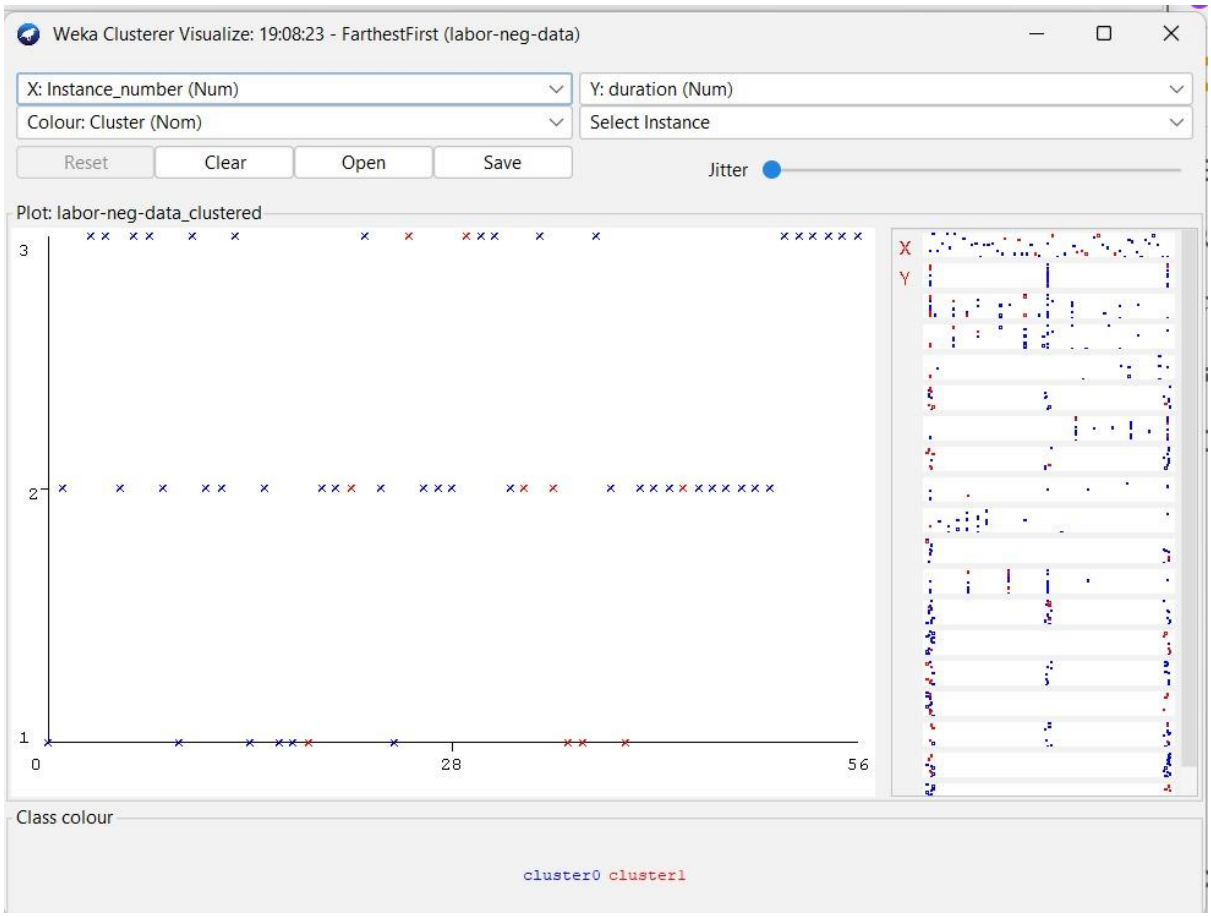




2. FarthestFirst algorithm:

The Farthest First algorithm, also known as the Farthest First Traversal algorithm, is a simple heuristic algorithm used for seeding initial centroids in K-means clustering. The idea is to select the initial centroids as far away from each other as possible to ensure a good coverage of the data space.





Conclusion:

From above assignment we learnt about the Weka tool. We also learnt to apply different clustering techniques to visualize the clusters using Weka tool. We applied simple k-means and FarthestFirst algorithm for clustering.

DMW Lab Assignment: 5 Classify text documents

Aim:

Classify text documents and evaluate precision, recall.

Problem Definition/Objective:

Consider a suitable text dataset. Remove stop words, apply stemming and feature selection techniques to represent documents as vectors. Classify documents and evaluate precision, recall

Input:CSV Dataset

Output: Classification of text documents

Prerequisites:

1. Need java installed in the system.

Relevant Theory:

The filters are found when click the "Choose " button under "Filter" section. This button opens a window with root weka. From there selecte filters and the unsupervised folder to after select attribute and finally select StringToWordVector.
StringToWordVector filter can configured its attributes with language processing techniques. To edit this filter is only necessary to click on the filter name. it will open a that show the following options. They were generated a set of optimal options from different combinations of options applied to the same training data . Each resulting model was calculated its F-measurement which describes the proportion of its predicted instances erroneously. The options that generated the greatest number of instances predicted correctly are as follows:

1. a) wordsToKeep: Standing with 1000 since it defines the word limit per class to maintain. Where doNotOperateOnPerClassBasis flag: as "False" to base wordsToKeep in all classes.

1. b) TFTransform as "True", DFTransform as "True" outputWordCounts as "True" and normalizeDocLength: is set to "No normalization".

The values are not normalized to the filter papers find more interrelated and count how often a word is in the document and not only consider whether the term is in the document. *OutputWordCounts* is the flag that describes whether a word exist or not in the document and *normalizeDocLength* couts a word with its actual value from tf-idf result of that word in the document, no matter how small or longer the document is.

1. c) lowerCaseTokens: as "True" to convert all to lowercase words before being added to the record and analyze the same word in lowercase and uppercase separately.

1. d) Stemmer: selects the algorithm to elimination the morpheme in a given language in order to reduce the word to its root. Select no stemmer as the classification of texts is multilingual and it will only aply stemming for one language. No stemmer is configured when click on the “Select” button menu is deployed and “NullStemmer” is selected.

Procedure:

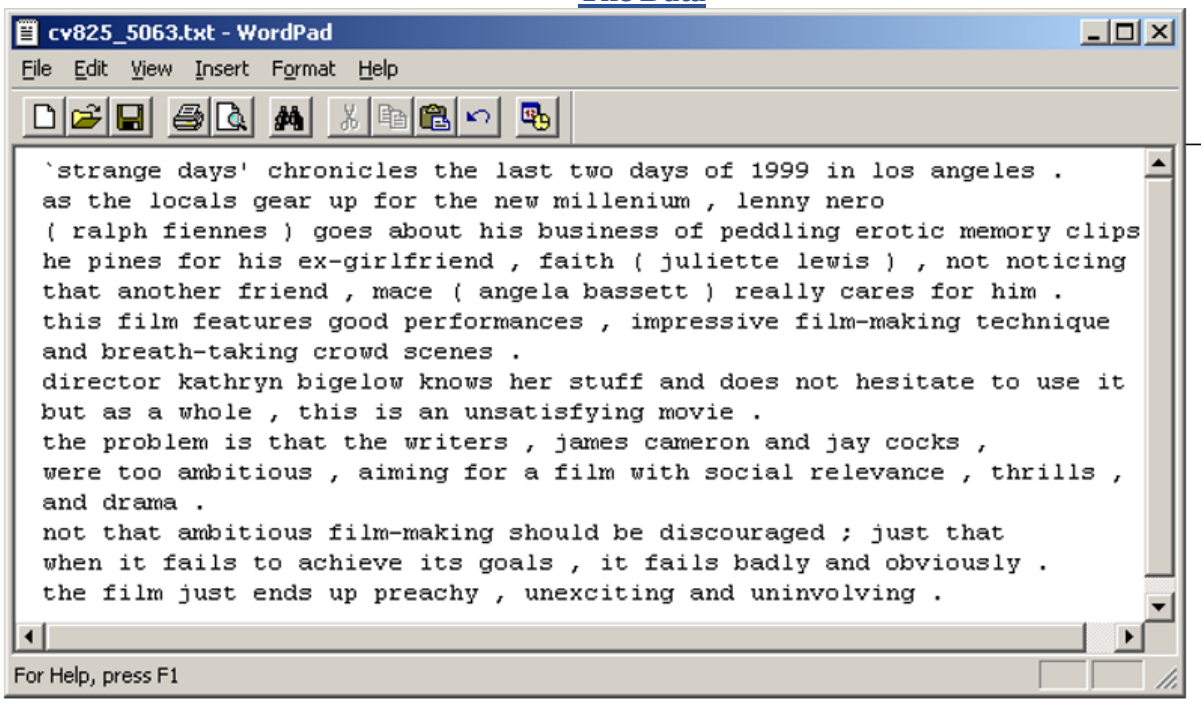
Implementation of Task: Building a model for movies reviews in English for classifying it into positive or negative.

Sentiment Polarity Dataset Version 2.0

1000 positive movie review and 1000 negative review texts from: Thumbs up? Sentiment Classification using Machine Learning Techniques. Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Proceedings of EMNLP, pp. 79--86, 2002.

From: **http://www.cs.cornell.edu/people/pabo/movie-review-data/**

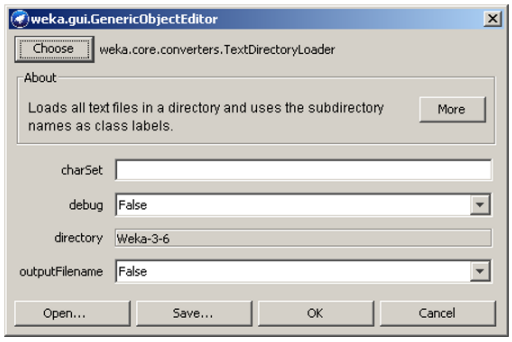
The Data

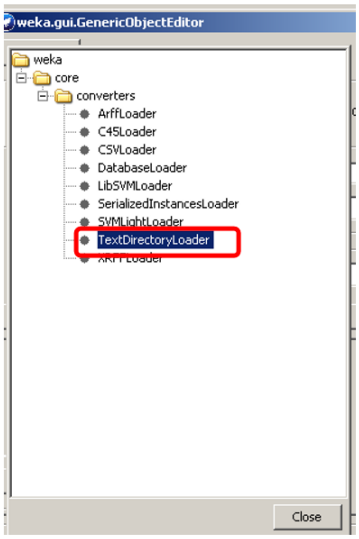


Open WEKA

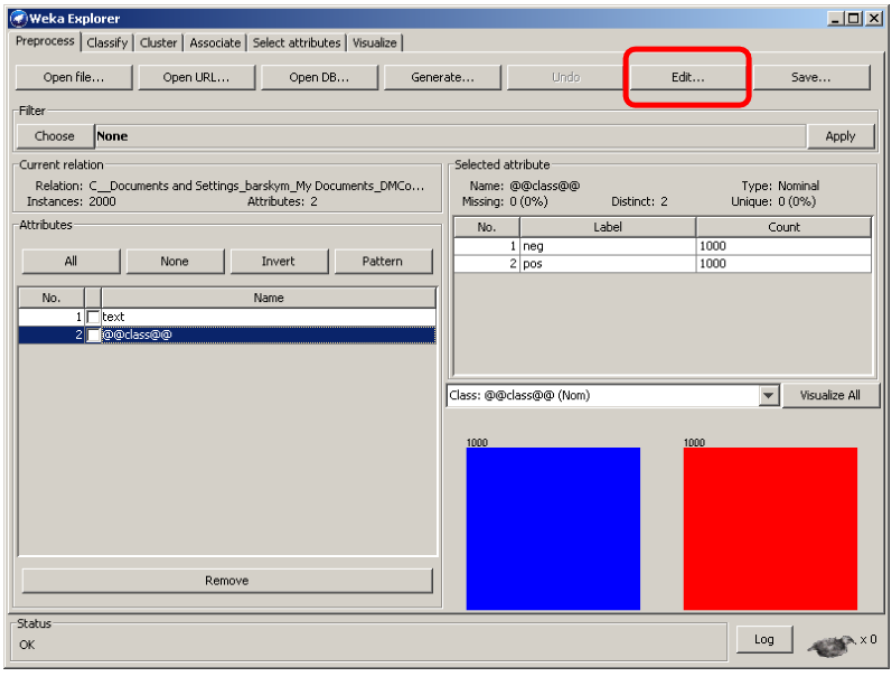
• Convert file to .arff format

Choose converter

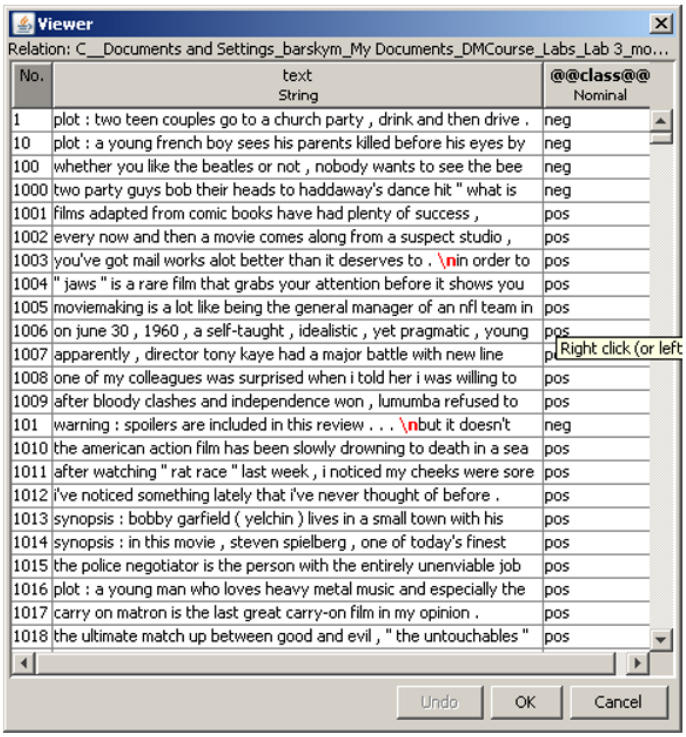




Class



Edit->View



Save converted file in movoiereviews.arff format

Convert text field into word vectors

• Filter-> • Unsupervised-> • Attribute -> • StringToWordVector

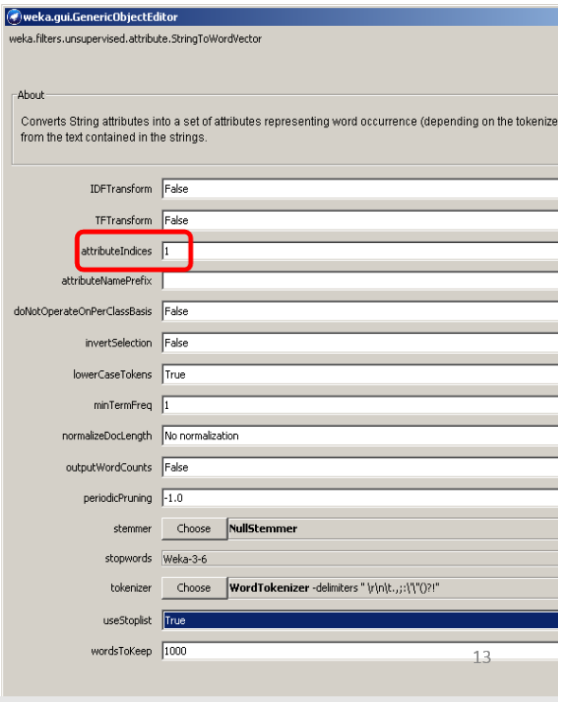
• This will convert each word in string field into a numeric attribute

• The name of the attribute would be the word itself • The value of the attribute would be 0 (absent) or 1 (present) in the current document.

Right-click for options

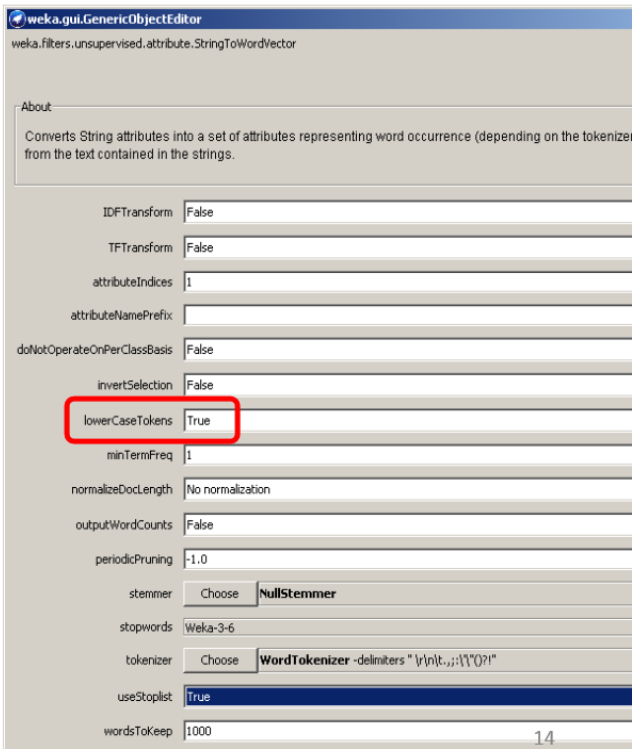
- Select attribute

to convert



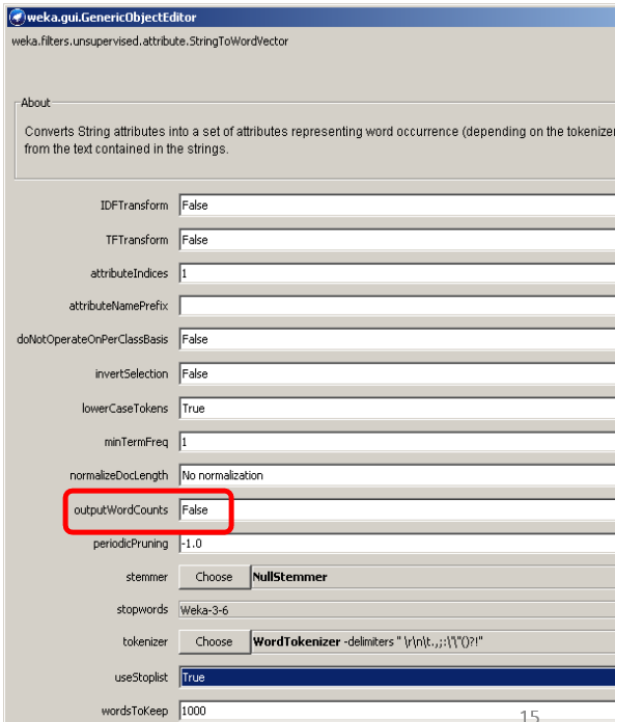
Options

- Can convert all words to lower case – preferred



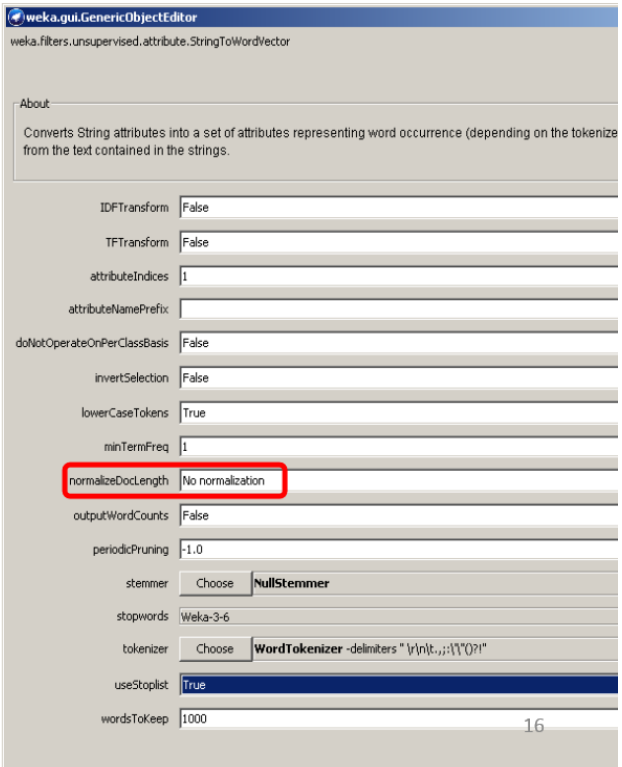
Options

- Can output the words counts in each document, instead of just occurrence.
- We will use the boolean 'present-absent' for this lab



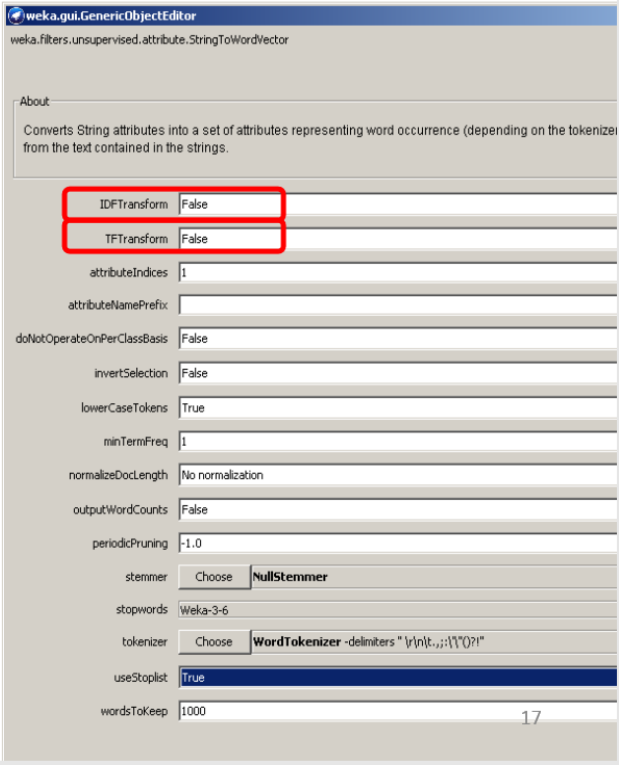
Options

- If 'outputWordsCounts' is selected, can normalize word counts by dividing to the total number of words in the document – thus creating the normalized word vector



Options

- If ‘outputWordsCounts’ is selected, can do TF or IDF-transform or both for each document:



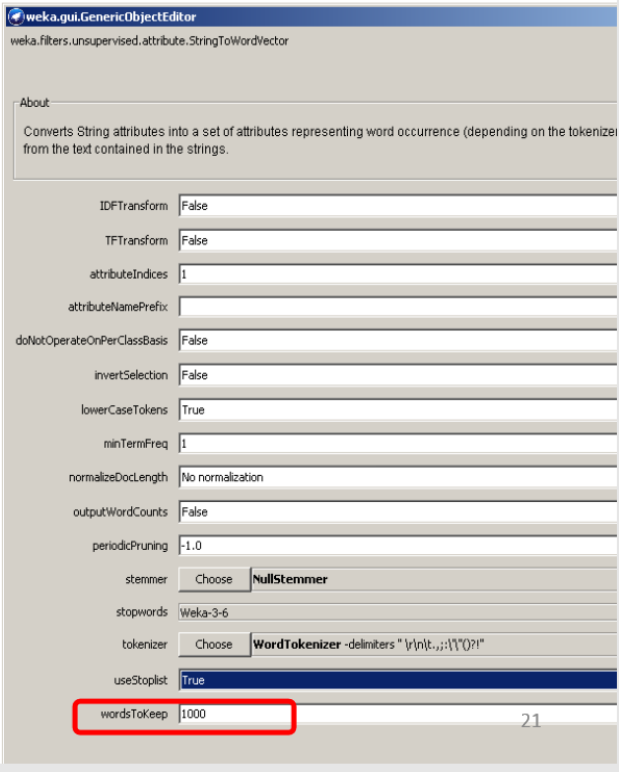
Term Frequency –Document Frequency

Tf (wordi,d)=frequency (word i)/[total words in d] DF(wordi) = number of documents containing wordi/total number of documents

The bigger TF the more discriminative is wordi The smaller DF the more discriminative is wordi

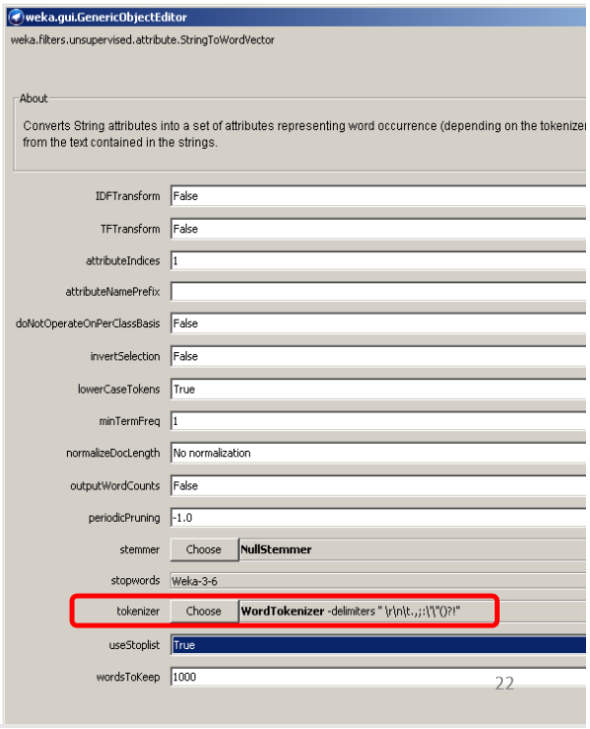
Options

- Keeps 1000 most frequent words for each class. Sometimes a little more, if there is a tie
- Less frequent words are discarded



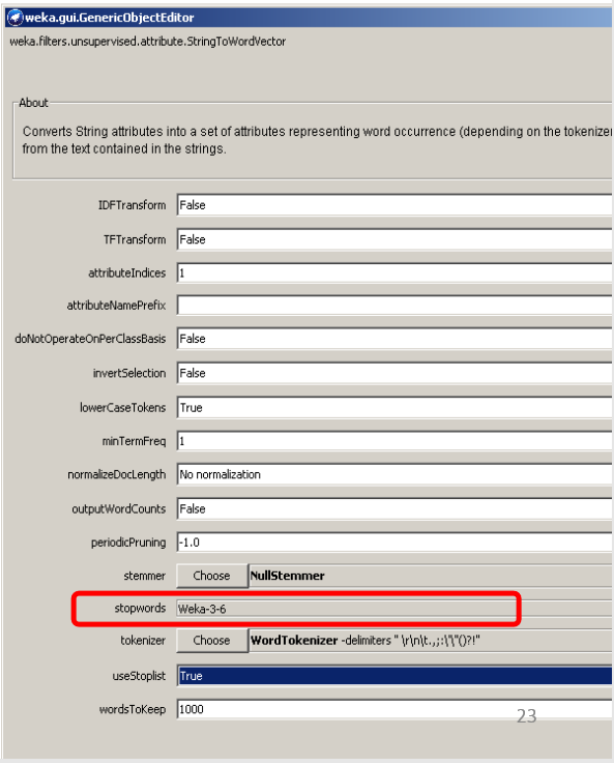
How the words are extracted

- The tokenizer is supplied with the list of characters which are considered as delimiters. The extracted word is the trimmed string between two delimiters
- You can provide your own application-dependent tokenizer



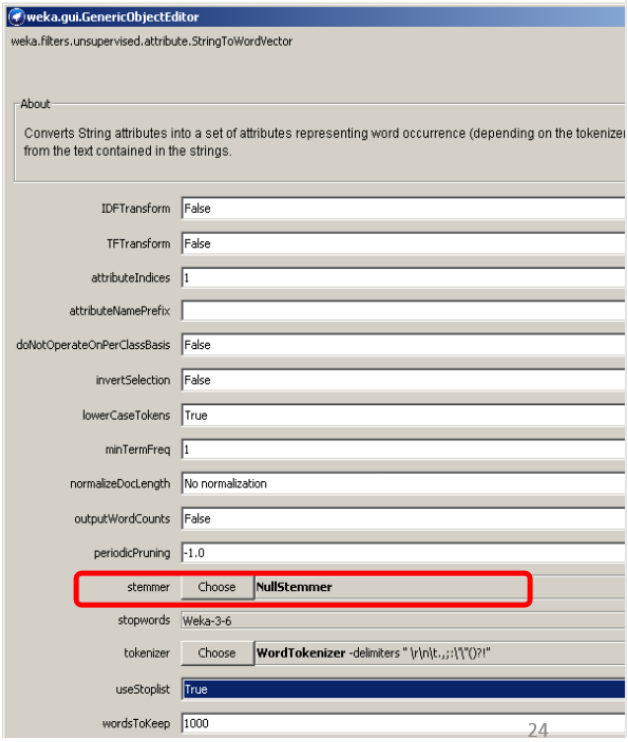
How the words are extracted

- Words such as “the”, “in”, “of” are removed – they occur in each document
- You can replace the default list with your own stop word list such as supplied stopwords_google.txt

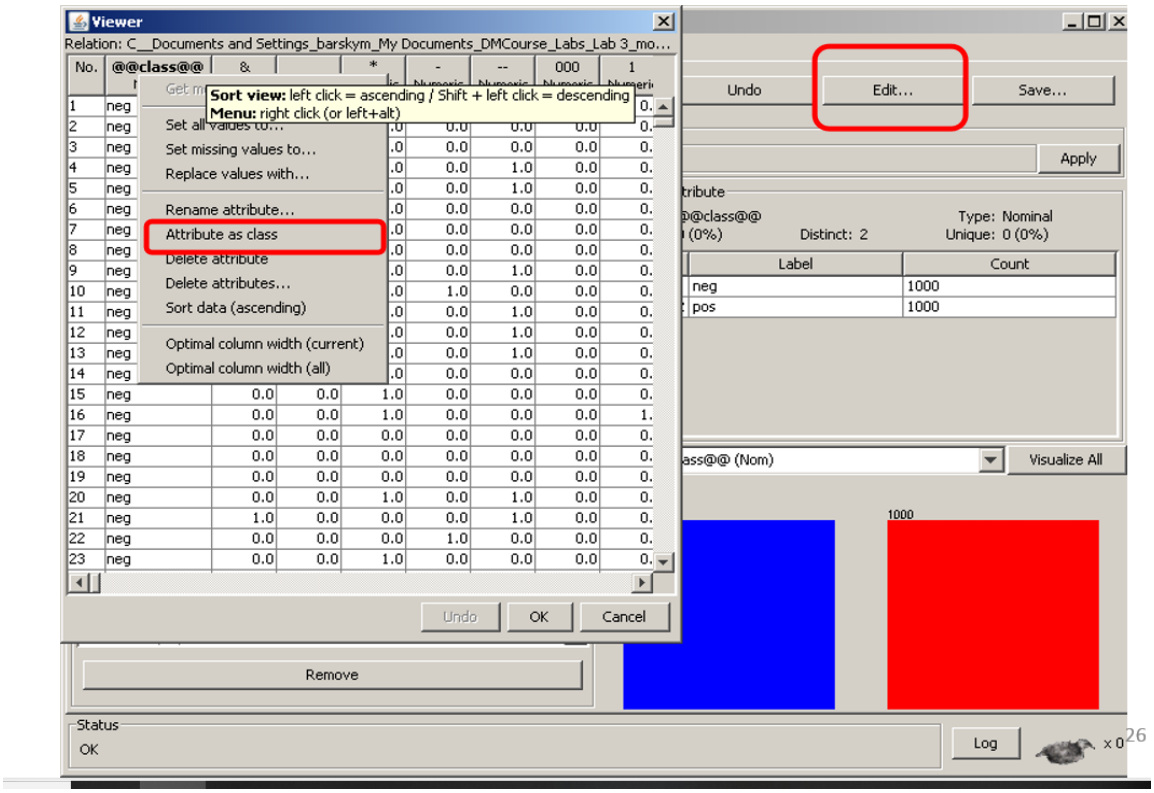


How the words are extracted

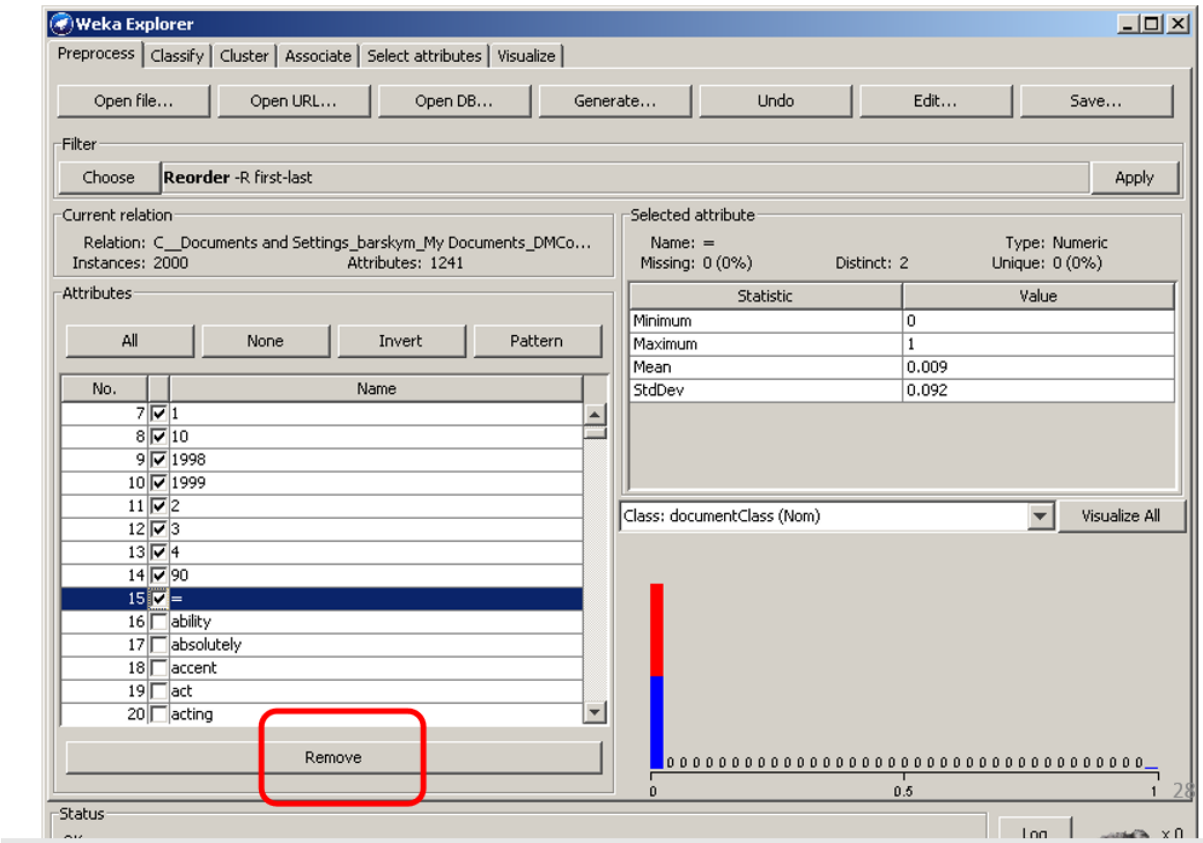
- Stemmer – identifies words which have the same root, for example: “cat”, “cats”, “catlike”, “catty” have similar meaning related to the root “cat” and stemmer treats them all as the same word. No stemmer by default



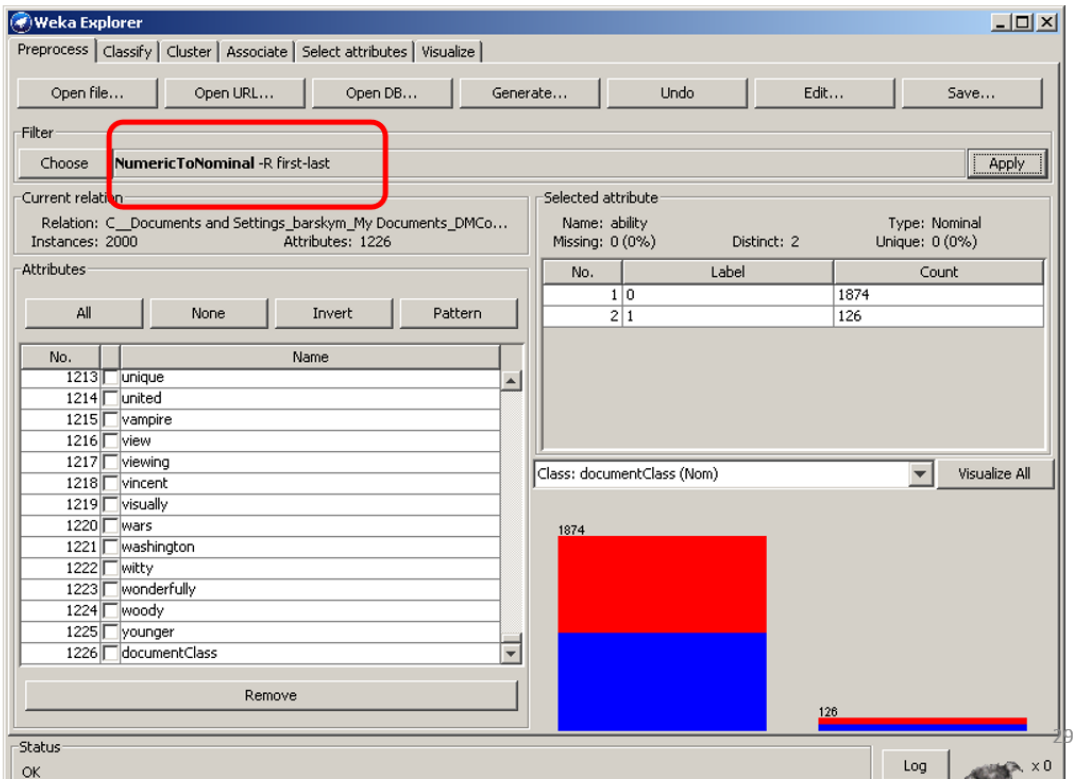
Move class attribute to the end of vector



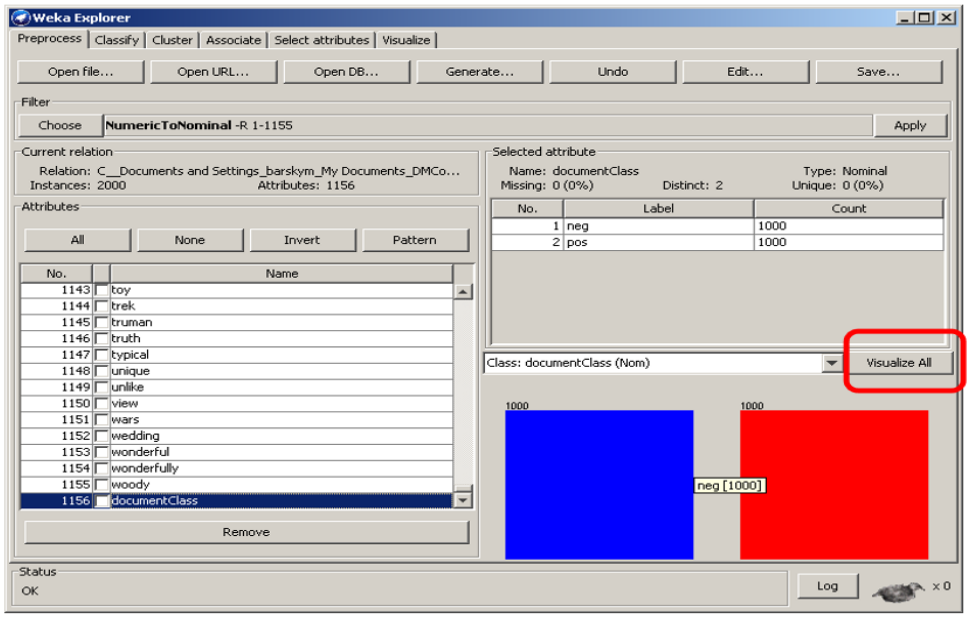
Clean some junk words



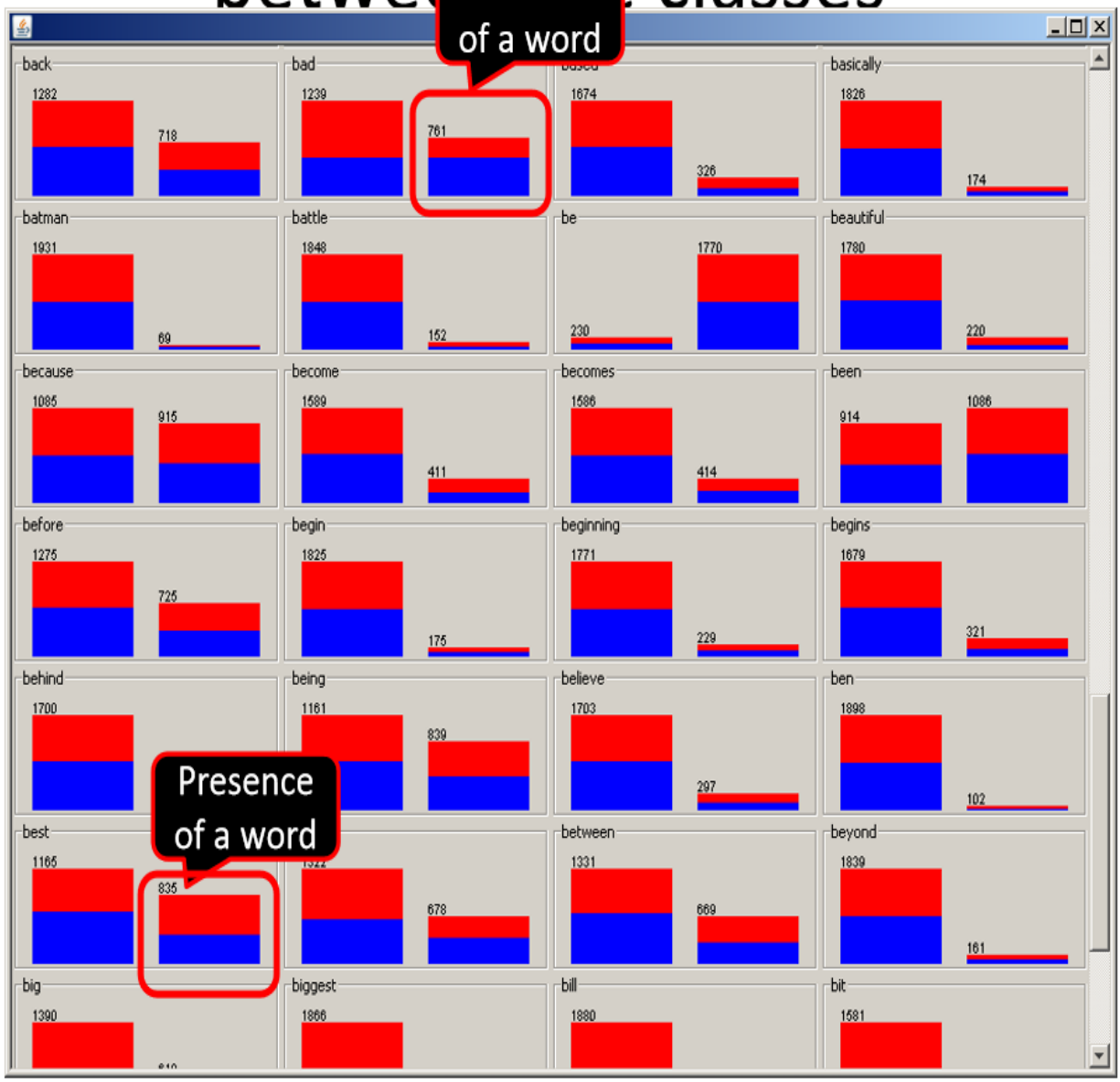
Convert all numeric to nominal (boolean)



Visualize all attributes



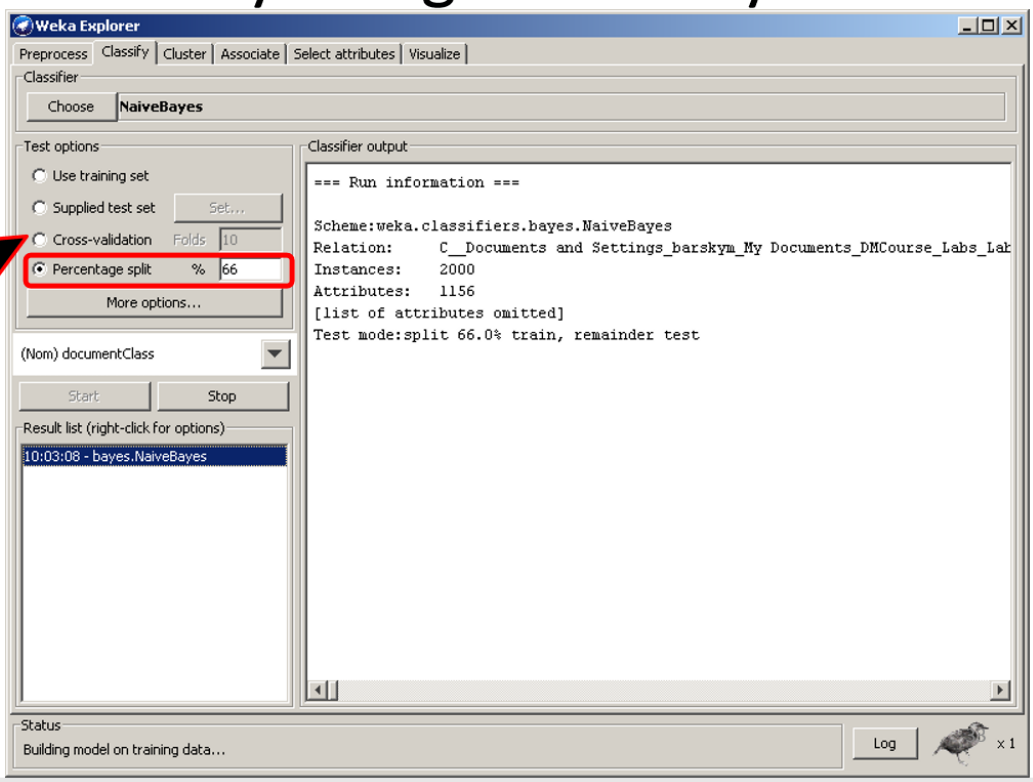
Presence of a word discriminates between the classes

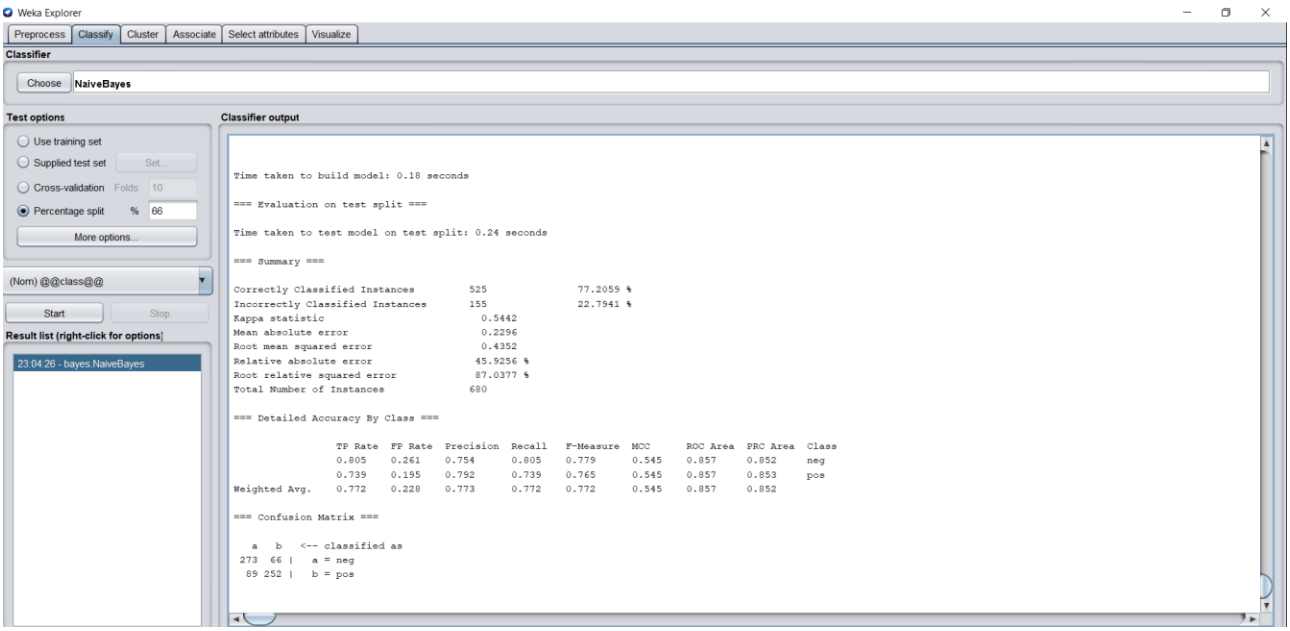


Absence of a word discriminates between the classes

Classify using Naïve Bayes

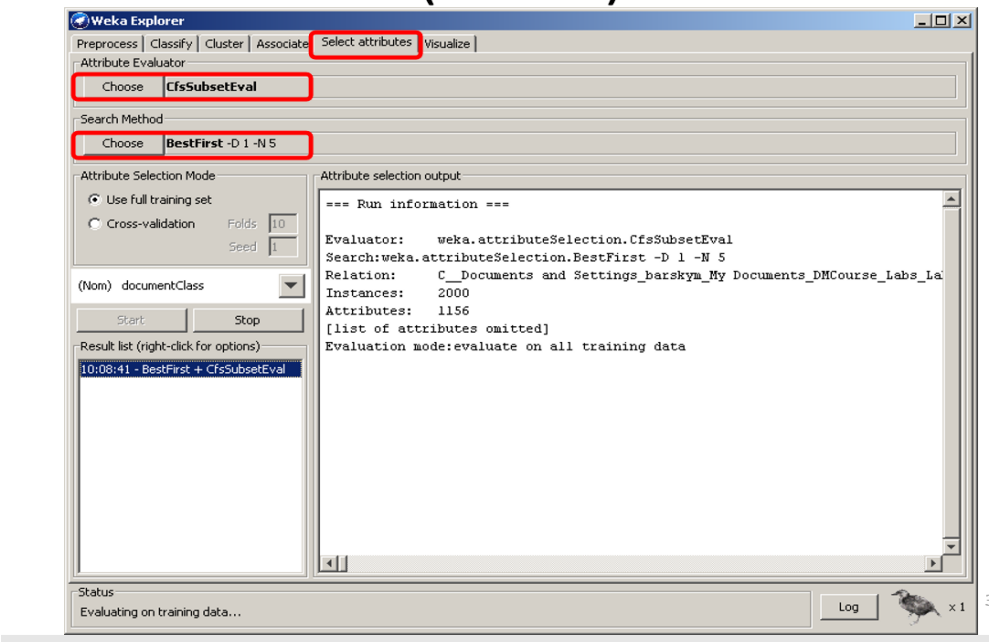
Don't use cross-validation – takes too much time

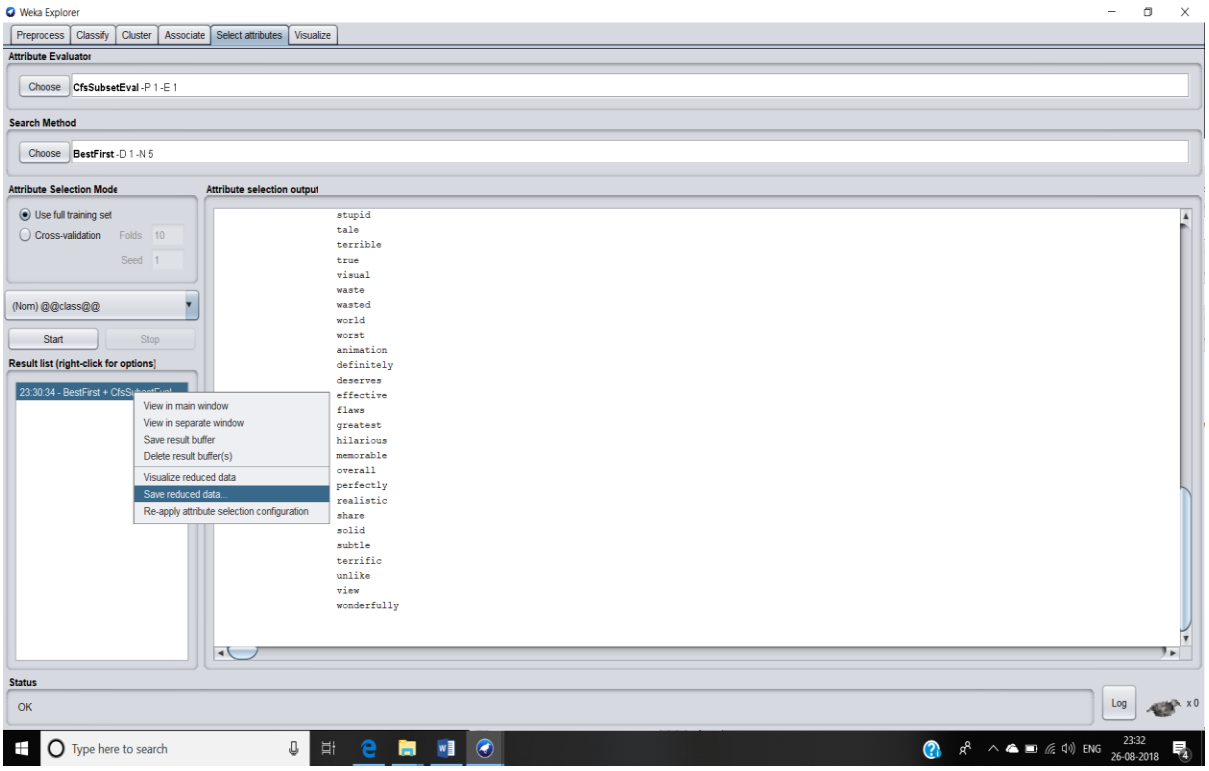




Feature (discriminative words) selection • WEKA class CfsSubsetEval evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. Subsets of features that are highly correlated with the class while having low inter-correlation are preferred.To find such attributes WEKA uses BestFirst search: Searches the space of attribute subsets by greedy hillclimbing augmented with a backtracking facility. Setting the number of consecutive nonimproving nodes allowed controls the level of backtracking done. Best first may start with the empty set of attributes and search forward, or start with the full set of attributes and search backward, or start at any point and search in both directions (by considering all possible single attribute additions and deletions at a given point

Attribute (words) selection





Selected discriminative attributes (words): 51

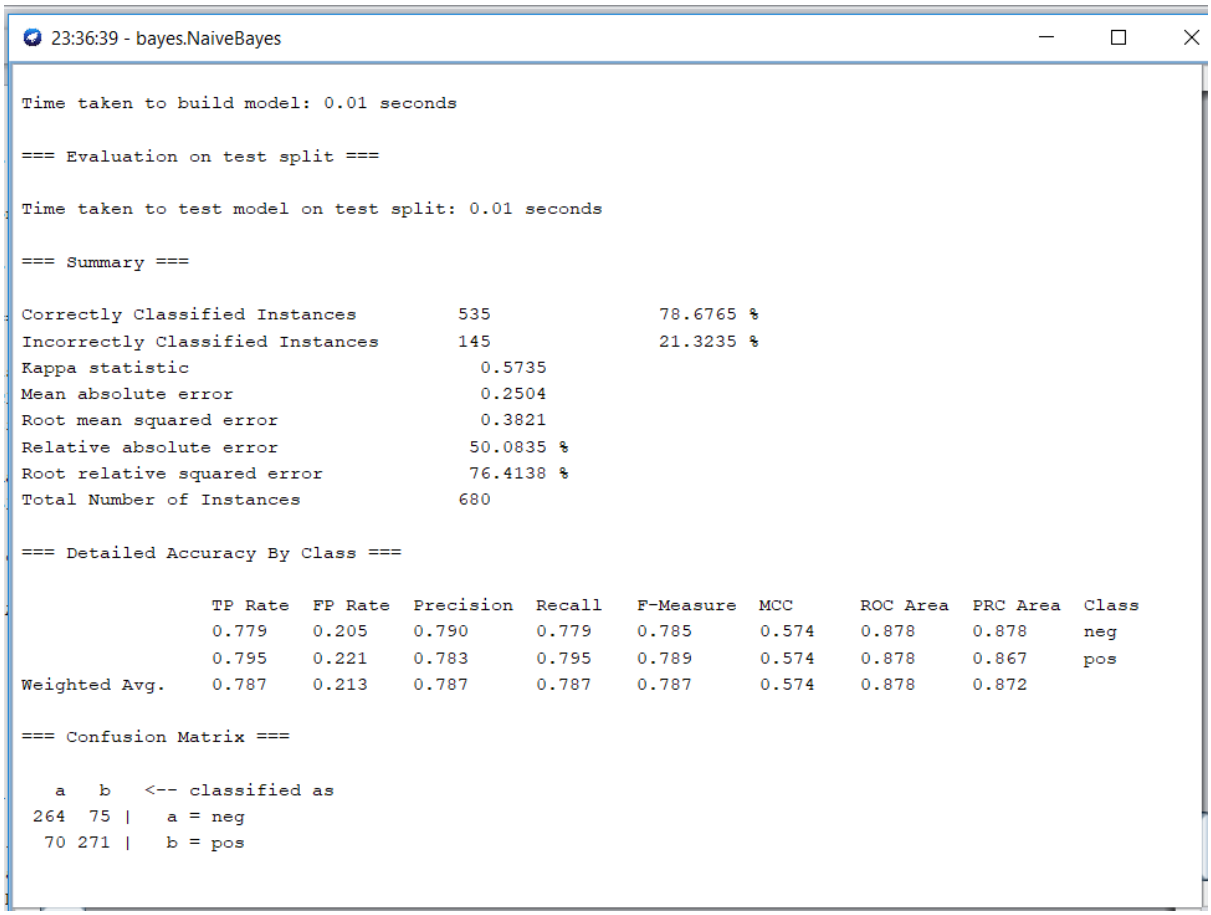
- Right-click result -> save reduced data as Movies_reviews_reduced.arff

also	world
awful	worst
bad	animation
boring	definitely
both	deserves
dull	effective
fails	flaws
great	greatest
joke	hilarious
lame	memorable
life	overall
many	perfectly
maybe	realistic
mess	share
nothing	solid
others	subtle

← Some of selected words

Classify again

• Accuracy **78.67** – little improvement, **but only 51 words instead of 1155**



Conclusion:

Text Documents were classified.

DMW Lab Assignement-6: Classification of dataset based on suitable dataset

Dataset: Heart.csv

Problem Statement: Classify the dataset using different algorithms ,use suitable dataset and classify the dataset using WEKA tool.

Theory and Implementation:

1. Import Required Libraries

Import pandas for data handling, train_test_split for data splitting, various classifiers for classification tasks (DecisionTreeClassifier, SVC, KNeighborsClassifier), accuracy_score for evaluating model performance, and matplotlib for plotting.

```
[ ] import pandas as pd
    from sklearn.model_selection import train_test_split
    from sklearn.tree import DecisionTreeClassifier
    from sklearn.svm import SVC
    from sklearn.neighbors import KNeighborsClassifier
    from sklearn.metrics import accuracy_score
```

```
import pandas as pd

# Importing dataset with specified encoding
data = pd.read_csv('Hdata.csv')
data
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1020	59	1	1	140	221	0	1	164	1	0.0	2	0	2	1
1021	60	1	0	125	258	0	0	141	1	2.8	1	1	3	0
1022	47	1	0	110	275	0	0	118	1	1.0	1	1	2	0
1023	50	0	0	110	254	0	0	159	0	0.0	2	0	2	1
1024	54	1	0	120	188	0	1	113	0	1.4	1	1	3	0

2. Data Preprocessing and Splitting

Split the dataset into features (X) and target variable (y) to prepare for model training and testing using train_test_split from sklearn.

```
[ ]  # Step 3: Data preprocessing
     # Split the dataset into features (X) and target variable (y)
     X = data.drop(columns=['thal'])
     y = data['thal']
```

```
[ ]  # Split the data into training and testing sets
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

3. Choose Classification Techniques

Initialize classifiers for Decision Trees, SVM, and KNN using the respective classifier classes from sklearn.

```
# Step 4: Choose Classification Techniques
# Initialize classifiers for Decision Trees, SVM, and KNN
dt_classifier = DecisionTreeClassifier()
svm_classifier = SVC()
knn_classifier = KNeighborsClassifier()
```

4. Train and Evaluate Models

Train each classifier using the training data (X_train, y_train) and make predictions on the testing data (X_test) to evaluate model accuracy.

```
# Step 5: Train and Evaluate Models
# Train each model using the training data
dt_classifier.fit(X_train, y_train)
svm_classifier.fit(X_train, y_train)
knn_classifier.fit(X_train, y_train)

# Make predictions on the testing data
dt_predictions = dt_classifier.predict(X_test)
svm_predictions = svm_classifier.predict(X_test)
knn_predictions = knn_classifier.predict(X_test)

# Calculate accuracy for each model using accuracy_score()
dt_accuracy = accuracy_score(y_test, dt_predictions)
svm_accuracy = accuracy_score(y_test, svm_predictions)
knn_accuracy = accuracy_score(y_test, knn_predictions)
```

5. Compare Results and Plot Accuracy

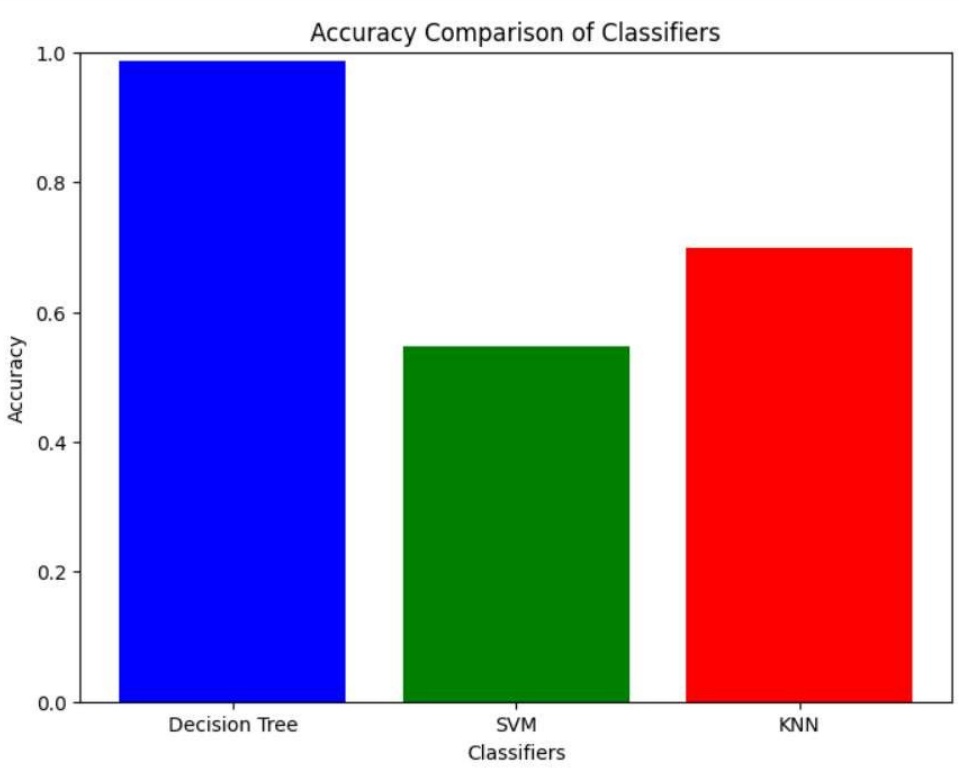
Calculate accuracy scores for each classifier using accuracy_score() and compare the performance of Decision Tree, SVM, and KNN classifiers.

Finally, plot the accuracy comparison using matplotlib.

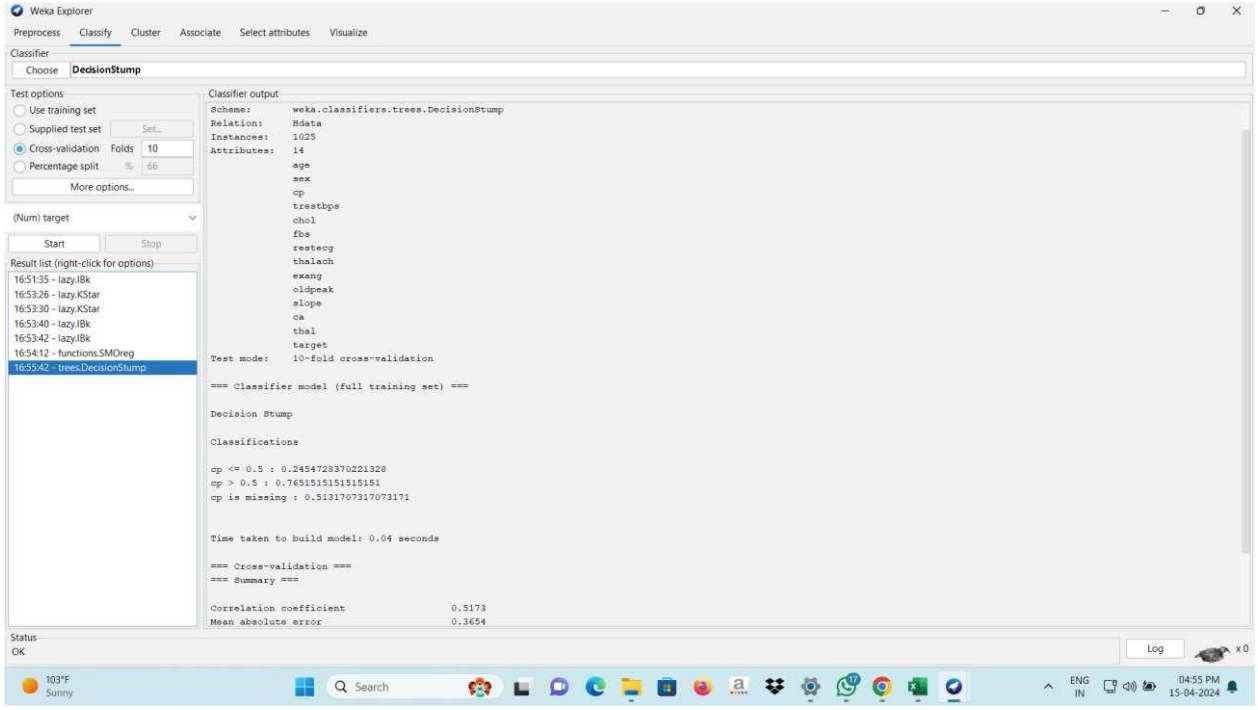
```
[ ] # Step 6: Compare Results
    print("Decision Tree Accuracy:", dt_accuracy)
    print("SVM Accuracy:", svm_accuracy)
    print("KNN Accuracy:", knn_accuracy)
```

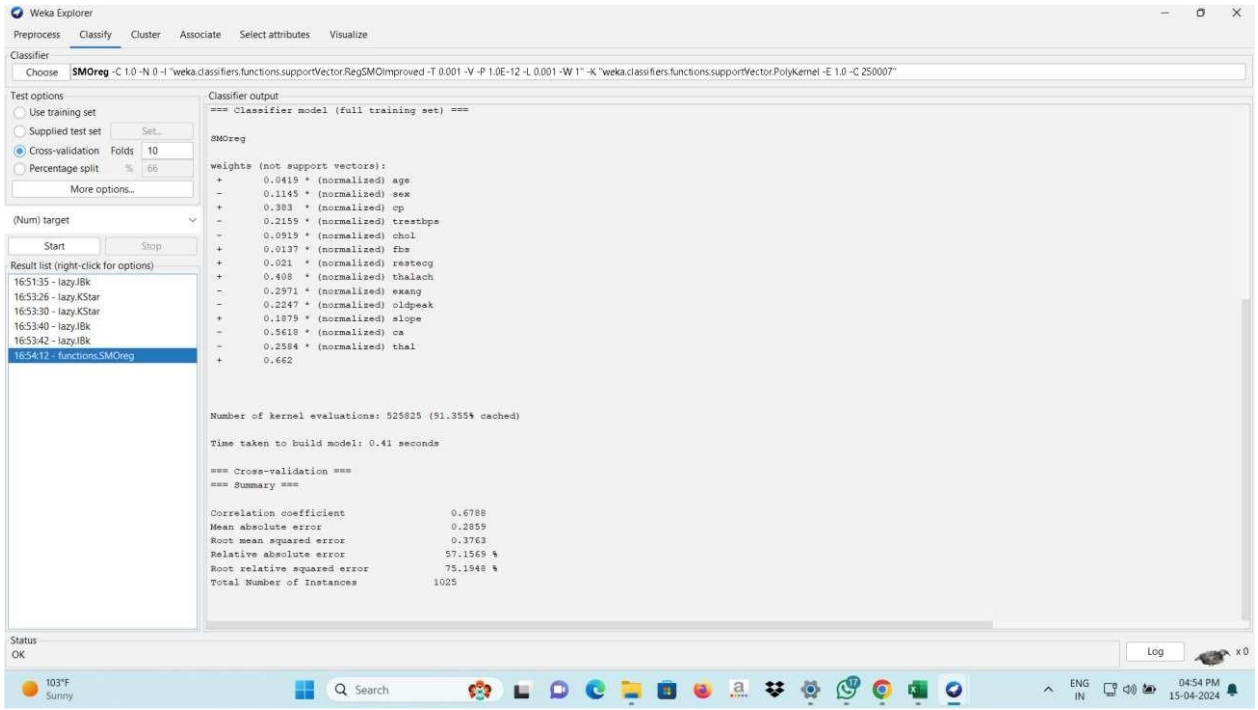
```
Decision Tree Accuracy: 0.9853658536585366
SVM Accuracy: 0.5463414634146342
KNN Accuracy: 0.697560975609756
```

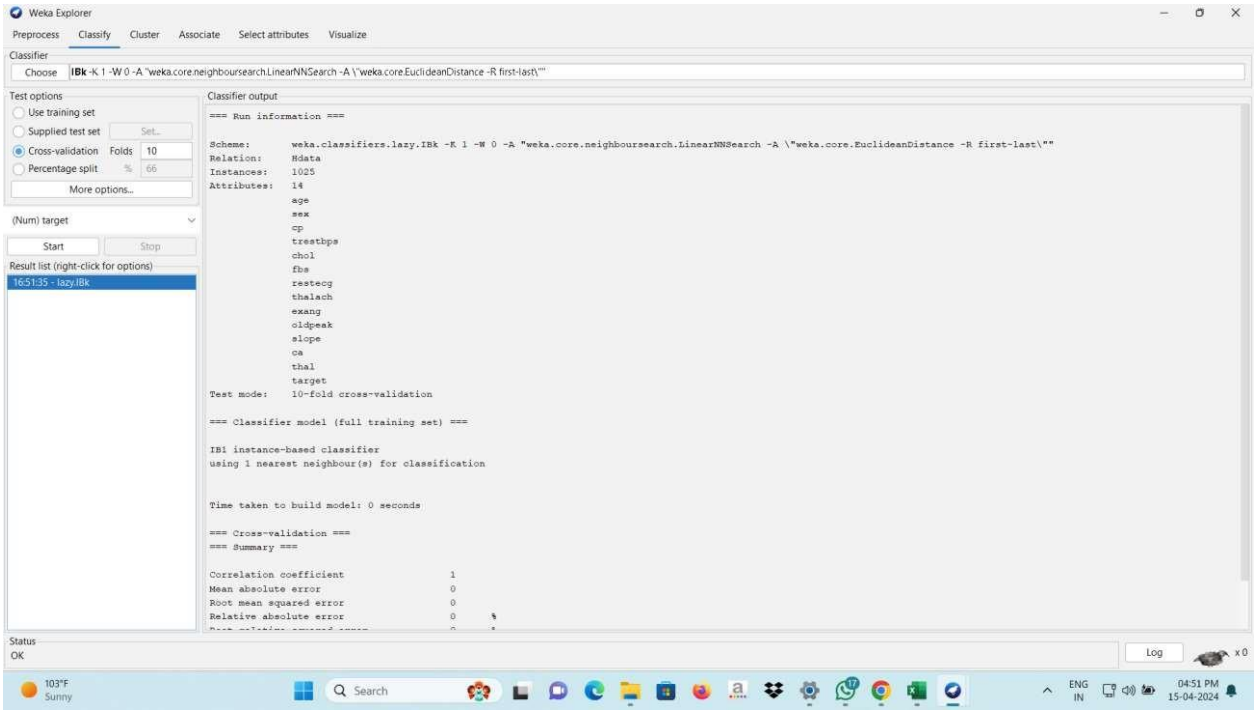
```
import matplotlib.pyplot as plt

# Accuracy scores obtained
accuracies = [dt_accuracy, svm_accuracy, knn_accuracy]
classifiers = ['Decision Tree', 'SVM', 'KNN']

# Plotting the accuracies
plt.figure(figsize=(8, 6))
plt.bar(classifiers, accuracies, color=['blue', 'green', 'red'])
plt.xlabel('Classifiers')
plt.ylabel('Accuracy')
plt.title('Accuracy Comparison of Classifiers')
plt.ylim(0, 1)  # Set y-axis limit from 0 to 1 for accuracy percentage
plt.show()
```



Weka Implementation







Conclusion:

Students will learn & apply the classification algorithms.

T. Bhaskar.

Prepared by:

Dr.T.Bhaskar

Course Coordinator

Approved by:

Dr. D.B. Kshirsagar

Head, Computer Engineering