

	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

01/04/24

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	PO1 0	PO1 1	PO1 2	PSO 1	PSO 2	PSO 3
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 and Francis 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: 978-0321321367 ISBN-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 Decision Making", Wiley Publications, ISBN: 9780470753866

DMW Lab Assignment-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-qmzFSqN0xp8'}) # replace the id with id of file you want to access
downloaded.GetContentFile('Heart.csv')
# https://drive.google.com/file/d/1KfsZrC6uz42aw0vvK0qh-qmzFSqN0xp8/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	reversible	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	reversible	Yes
299	300	68	1	asymptomatic	144	193	1	0	141	0	3.4	2	2.0	reversible	Yes
300	301	57	1	asymptomatic	130	131	0	0	115	1	1.2	2	1.0	reversible	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 x 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:

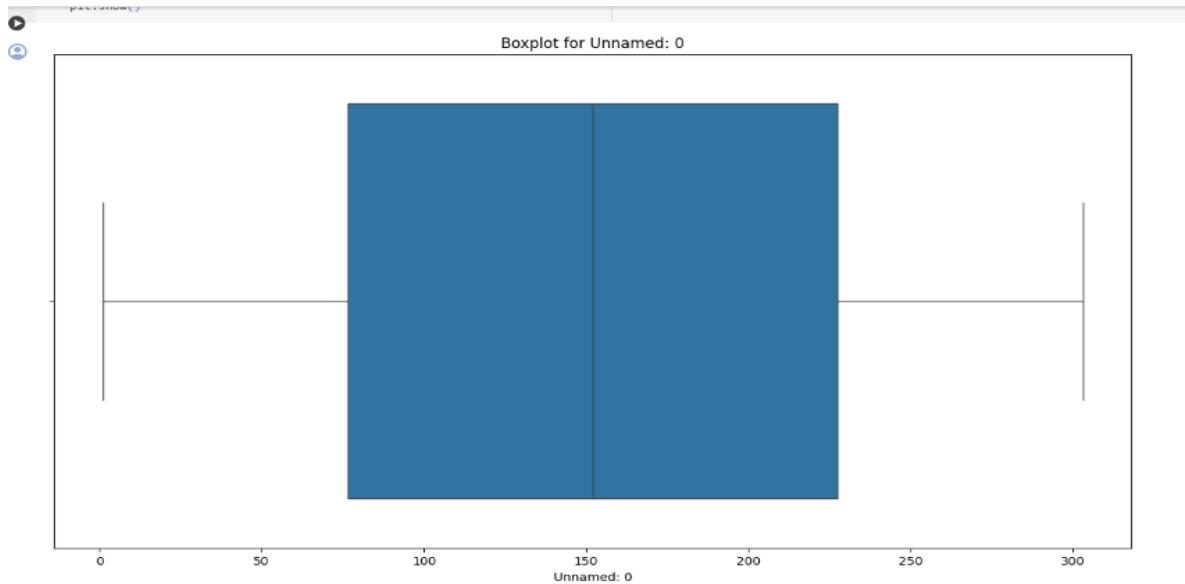
There were no inconsistencies found in the dataset that needed to be removed.

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

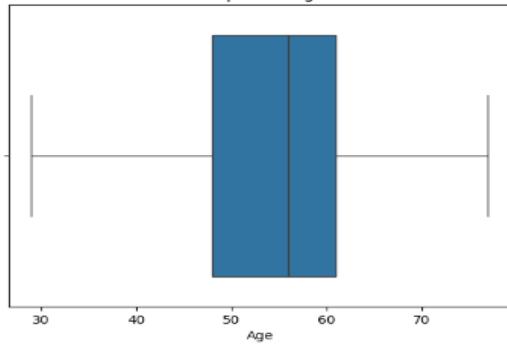
4. Boxplot Analysis:

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.

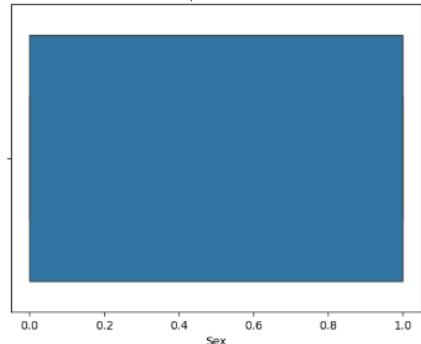
```
[ ] # c) Boxplot analysis for each numerical attribute and find outliers
numerical_attributes = data.select_dtypes(include=['float64', 'int64']).columns
plt.figure(figsize=(10, 5))
for col in numerical_attributes:
    sns.boxplot(x=data[col])
    plt.title(f'Boxplot for {col}')
    plt.show()
```



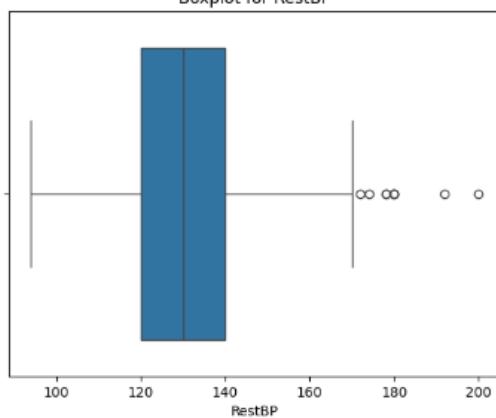
Boxplot for Age



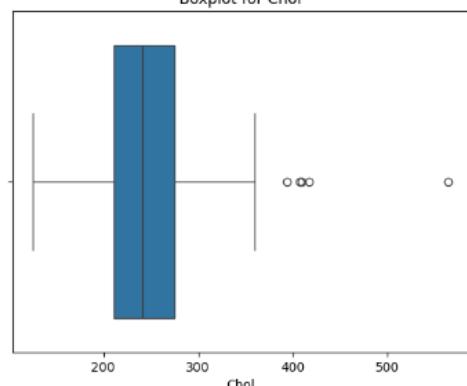
Boxplot for Sex



Boxplot for RestBP



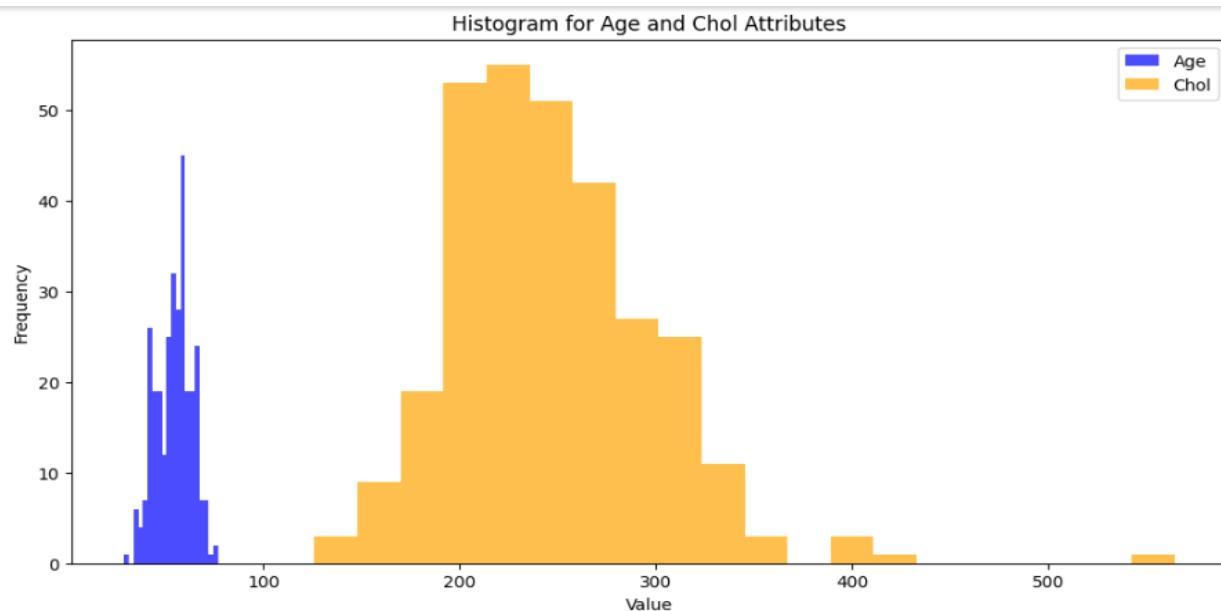
Boxplot for Chol



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 -

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

Implemented Code –

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

(x)
print("Standard Deviation of each numerical attribute:")
print(std_dev)
print("Variance of each numerical attribute:")
print(variance)

(c)
<ipython-input-17-997f16fb0043>:2: FutureWarning: The default value of numeric_only in DataFrame.std is deprecated. In a future version, it will default to False. In addition, specifying
std.dev = data.std()
Standard Deviation of each numerical attribute:
Height_(cm)           10.658026
Weight_(kg)            21.343210
BMI                  4.540322
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      618.002010
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 –

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

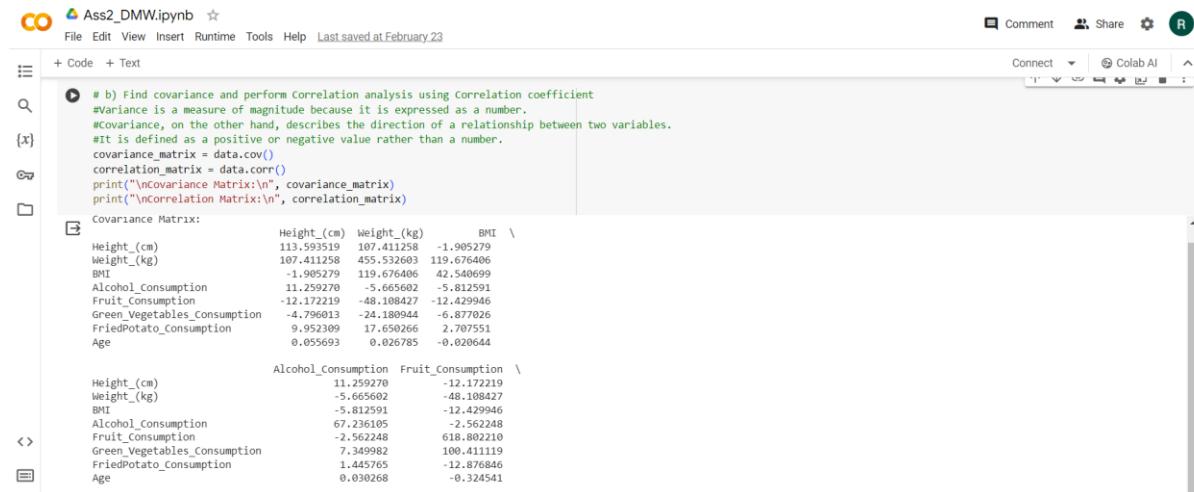
3. Correlation coefficient -

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 –

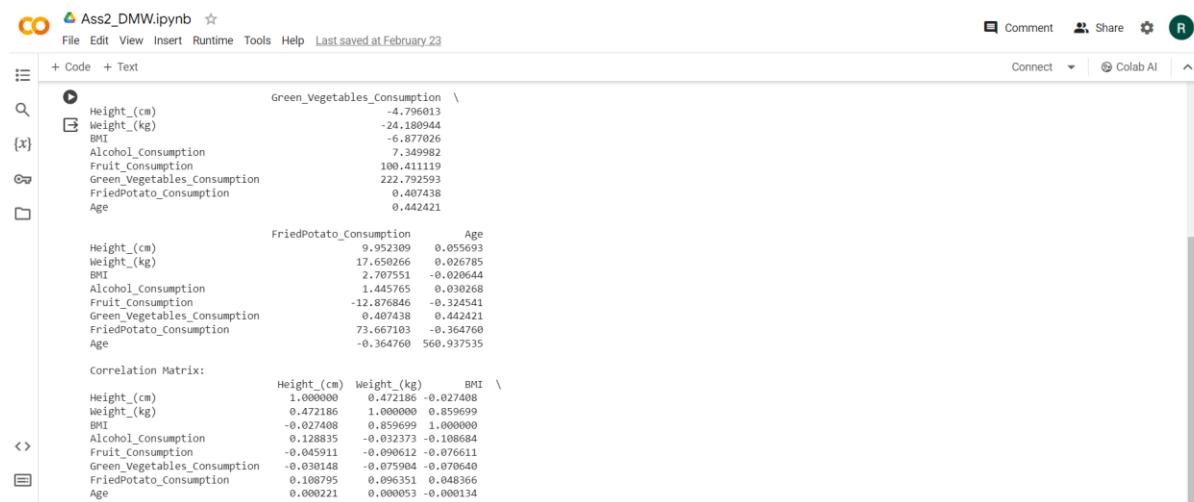


```
# b) Find covariance and perform correlation analysis using Correlation coefficient
#Variance is a measure of magnitude because it is expressed as a number.
#Covariance, on the other hand, describes the direction of a relationship between two variables.
#It is defined as a positive or negative value rather than a number.
covariance_matrix = data.cov()
correlation_matrix = data.corr()
print("\nCovariance Matrix:\n", covariance_matrix)
print("\nCorrelation Matrix:\n", correlation_matrix)
```

Covariance Matrix:

	Height_(cm)	Weight_(kg)	BMI
Height_(cm)	113.59510	187.411258	-1.005279
Weight_(kg)	107.411258	455.526003	110.676406
BMI	-1.005279	119.676406	42.540699
Alcohol_Consumption	11.259278	-5.665602	-5.812591
Fruit_Consumption	-12.172219	-48.108427	-12.429946
Green_Vegetables_Consumption	-4.796013	-24.180944	-6.877026
FriedPotato_Consumption	9.052309	17.650266	2.707551
Age	0.055693	0.026785	-0.020644

	Alcohol_Consumption	Fruit_Consumption	\
Height_(cm)	11.259278	-12.172219	
Weight_(kg)	-5.665602	-48.108427	
BMI	-5.812591	-12.429946	
Alcohol_Consumption	67.236105	-2.562248	
Fruit_Consumption	-2.562248	618.802210	
Green_Vegetables_Consumption	7.349982	100.411119	
FriedPotato_Consumption	1.445765	-12.876846	
Age	0.030268	-0.324541	



	Green_Vegetables_Consumption	\
Height_(cm)	-4.796013	
Weight_(kg)	-24.180944	
BMI	-6.877026	
Alcohol_Consumption	7.349982	
Fruit_Consumption	100.411119	
Green_Vegetables_Consumption	222.792593	
FriedPotato_Consumption	0.407438	
Age	0.442421	

	FriedPotato_Consumption	Age	\
Height_(cm)	9.052309	0.055693	
Weight_(kg)	17.650266	0.026785	
BMI	2.707551	-0.020644	
Alcohol_Consumption	1.445765	0.030268	
Fruit_Consumption	-12.876846	-0.324541	
Green_Vegetables_Consumption	0.407438	0.442421	
FriedPotato_Consumption	73.667108	-0.364760	
Age	-0.364760	560.937535	

	Height_(cm)	Weight_(kg)	BMI	\
Height_(cm)	1.000000	0.472186	-0.027488	
Weight_(kg)	0.472186	1.000000	0.859699	
BMI	-0.027488	0.859699	1.000000	
Alcohol_Consumption	0.128835	-0.032373	-0.108684	
Fruit_Consumption	-0.045911	-0.090612	-0.076611	
Green_Vegetables_Consumption	-0.030148	-0.075904	-0.070640	
FriedPotato_Consumption	0.108795	0.096351	0.048366	
Age	0.000221	0.000053	-0.000134	

```

Alcohol_Consumption  Fruit_Consumption \
Height_(cm)          0.128835   -0.045911
Weight_(kg)           -0.032373   -0.090612
BMI                  -0.108684   -0.076611
Alcohol_Consumption  1.000000
Fruit_Consumption    -0.012562   1.000000
Green_Vegetables_Consumption  0.060053   0.270430
FriedPotato_Consumption  0.020543   -0.060311
Age                  0.000156   -0.000551

Green_Vegetables_Consumption \
Height_(cm)          -0.030148
Weight_(kg)           0.07694
BMI                  -0.070640
Alcohol_Consumption  0.060053
Fruit_Consumption    0.270430
Green_Vegetables_Consumption  1.000000
FriedPotato_Consumption  0.003188
Age                  0.001251

FriedPotato_Consumption     Age
Height_(cm)              0.108795  0.000221
Weight_(kg)               0.096351  0.000053
BMI                      0.048366  -0.000134
Alcohol_Consumption       0.020543  0.000156
Fruit_Consumption         -0.060311  -0.000551
Green_Vegetables_Consumption  0.003188  0.001251
FriedPotato_Consumption  1.000000  -0.001794
Age                      -0.001794  1.000000

```

`cipython-input-18-fc51de66c3b5>:5: FutureWarning: The default value of numeric_only in DataFrame.cov is deprecated. In a future version, it will default to False. Select only valid columns.`

`cipython-input-18-fc51de66c3b5>:6: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns.`

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

d) Can we identify unwanted features?

Implemented Code -

```

[ ] # c) How many independent features are present in the given dataset?
#Correlation finds the direction of the relationship along with the degree of the relationship.
independent_features = np.linalg.matrix_rank(data.corr())
print("\nNumber of independent features:", independent_features)

Number of independent features: 8

```

`cipython-input-19-43e1aaa1ab6>:3: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns.`

```

[ ] # d) Can we identify unwanted features?
# we can analyze correlation coefficients to identify features strongly correlated with each other.
# Features with high correlation might be candidates for removal if redundancy is present.
# Features with high correlation might be candidates for removal.
#Correlation shows us both, the direction and magnitude of how two quantities vary with each other.

unwanted_features = set()
for i in range(len(correlation_matrix.columns)):
    for j in range(i):
        if abs(correlation_matrix.iloc[i, j]) > 0.8:
            colname = correlation_matrix.columns[i]
            unwanted_features.add(colname)

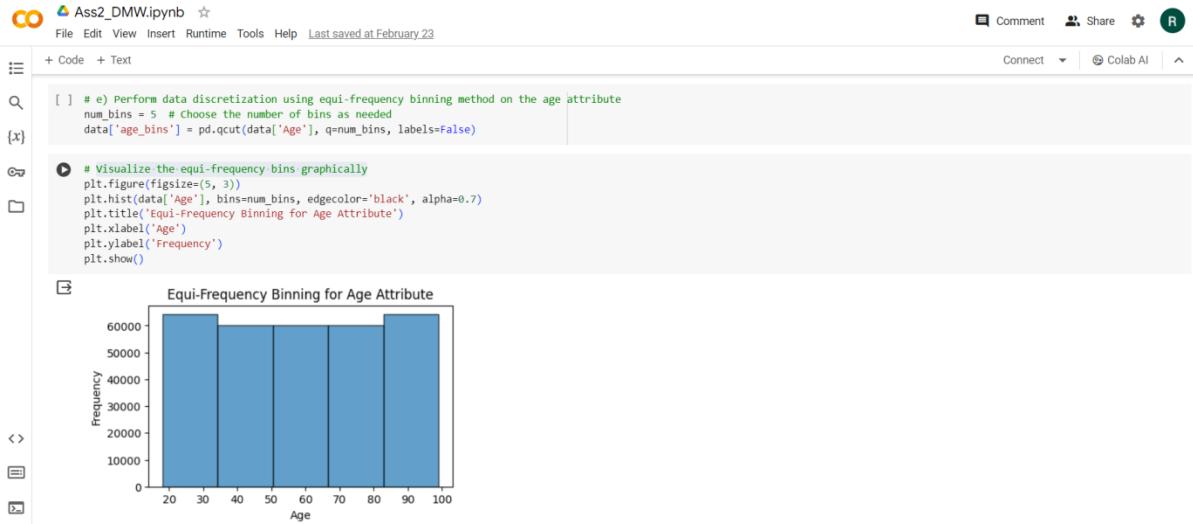
print("\nUnwanted features:", unwanted_features)

```

d) Unwanted features: {'BMI'}

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 M_p and X_p respectively. In min-max normalization , a value d of attribute P is mapped to value D in the range $[nX_p, nM_p]$ by calculating D using formula,

Min-Max Normalization formula is -

$$D = \{[d - M_p] / [X_p - M_p]\} (nX_p - nM_p) + nM_p$$

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, arithmetic 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 $\text{Max}(|D|)$.

The value d is transformed D using,

$$D = [d / 10^z]$$

Where, z represents the smallest integer value such that $\text{Max}(|D|) < 1$

Implemented Code –



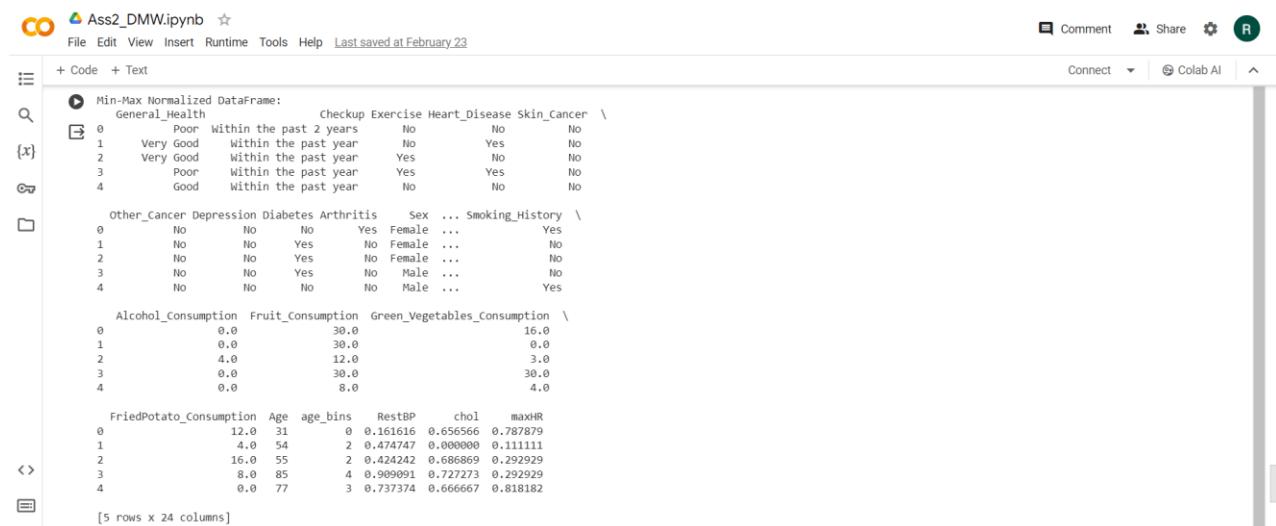
```
# f) Normalize RestBP, chol, and MaxHR attributes using different normalization techniques
attributes_to_normalize = ['RestBP', 'chol', 'maxHR']

# Min-Max normalization
min_max_scaler = MinMaxScaler()
data_min_max_normalized = data.copy()
data_min_max_normalized[attributes_to_normalize] = min_max_scaler.fit_transform(data[attributes_to_normalize])

# Z-score normalization
z_score_scaler = StandardScaler()
data_z_score_normalized = data.copy()
data_z_score_normalized[attributes_to_normalize] = z_score_scaler.fit_transform(data[attributes_to_normalize])

# Decimal scaling normalization
decimal_scaling_factor = 10 ** ((len(str(int(data[attributes_to_normalize].abs().max()).max(0)))) - 1)
data_decimal_scaled = data.copy()
data_decimal_scaled[attributes_to_normalize] = data[attributes_to_normalize] / decimal_scaling_factor

# Display the results
print("\nMin-Max Normalized DataFrame:\n", data_min_max_normalized.head())
print("\nZ-Score Normalized DataFrame:\n", data_z_score_normalized.head())
print("\nDecimal Scaled DataFrame:\n", data_decimal_scaled.head())
```



	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	0.161616	0.656566	0.787879
1	4.0	54	2	0.474747	0.000000	0.111111
2	16.0	55	2	0.424242	0.686869	0.292929
3	8.0	85	4	0.909091	0.727273	0.292929
4	0.0	77	3	0.737374	0.666667	0.818182

[5 rows x 24 columns]

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

```
+ Code + Text
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]
```

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

```
+ Code + Text
[ ] 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 Assignment-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_fpgrwth = fpgrwth(df_encoded, min_support=0.33, use_colnames=True)
frequent_itemsets_fpgrwth

→ /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_fpgrrowth[frequent_itemsets_fpgrrowth['support'] == frequent_itemsets_fpgrrowth['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'})]
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).

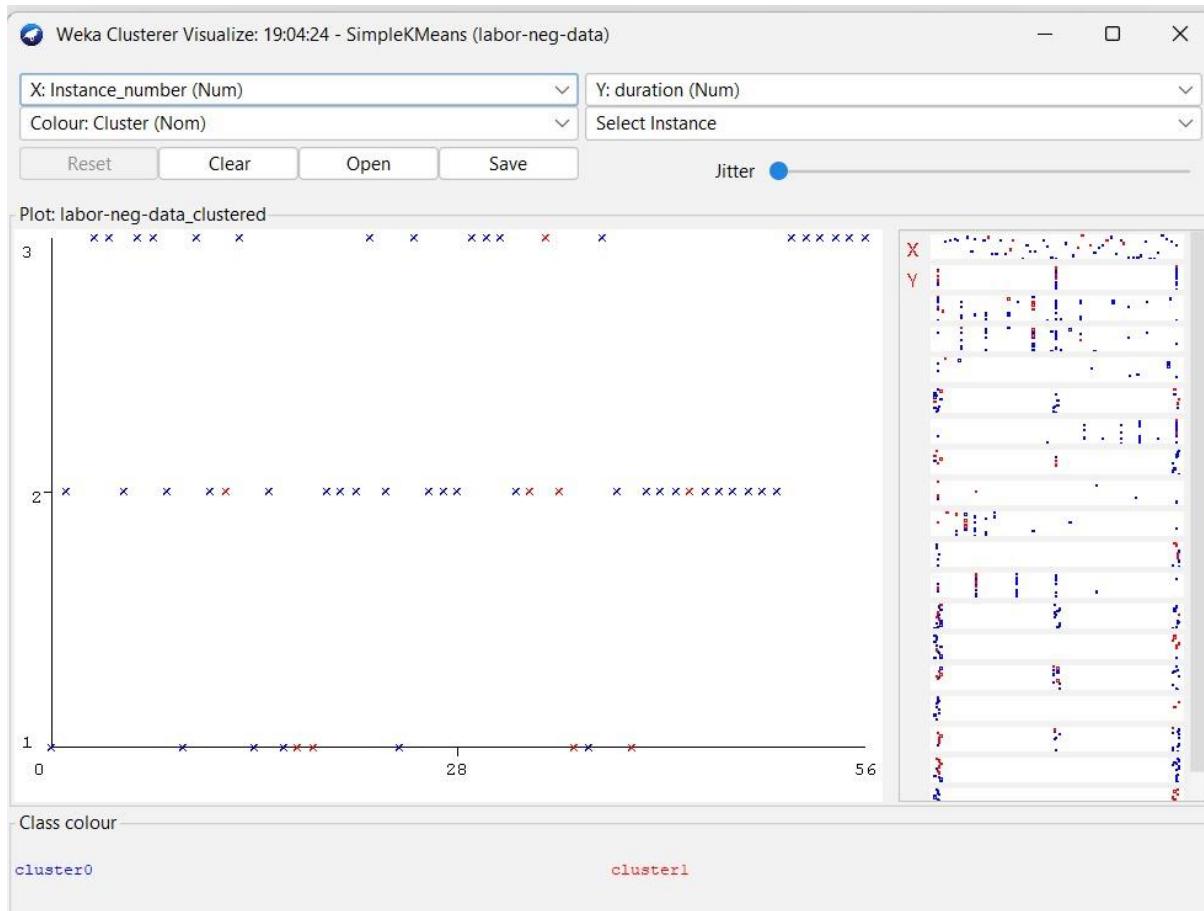
The screenshot shows the Weka Explorer interface with the 'Cluster' tab selected. In the 'Clusterer' section, 'SimpleKMeans' is chosen with the following command-line options: '-init 0 -max-candidates 1000 -periodic-pruning 10000 -min-density 2.0 -t1 1.25 -t2 1.0 -N 2 -A "weka.core.EuclideanDistance" -R first-last" -l 500 -num-slots 1 -S 10'. The 'Cluster mode' section has 'Use training set' checked. The 'Cluster output' panel displays the 'Final cluster centroids' table:

Attribute	Full Data (57.0)	Cluster#	
		0 (49.0)	1 (9.0)
duration	2.1607	2.2533	1.6667
wage-increase-first-year	3.0036	3.9934	2.9444
wage-increase-second-year	3.9717	4.0209	3.7097
wage-increase-third-year	3.9133	3.9511	3.7119
cost-of-living-adjustment	none	none	none
working-hours	38.0392	37.7541	39.5599
pension	empl_contr	empl_contr	none
standby-pay	7.4444	7.7431	5.8519
shift-differential	4.071	5.2290	2.957
education-allallowance	no	no	no
statutory-holidays	11.0943	11.237	10.3333
vacation	below_average	below_average	below_average
longterm-disability-assistance	yes	yes	no
contribution-to-dental-plan	half	half	none
bereavement-assistance	yes	yes	yes
contribution-to-health-plan	full	full	none
class	good	good	bad

Below the table, the message 'Time taken to build model (full training data) : 0.02 seconds' is displayed. The 'Model and evaluation on training set' section shows the clustered instances:

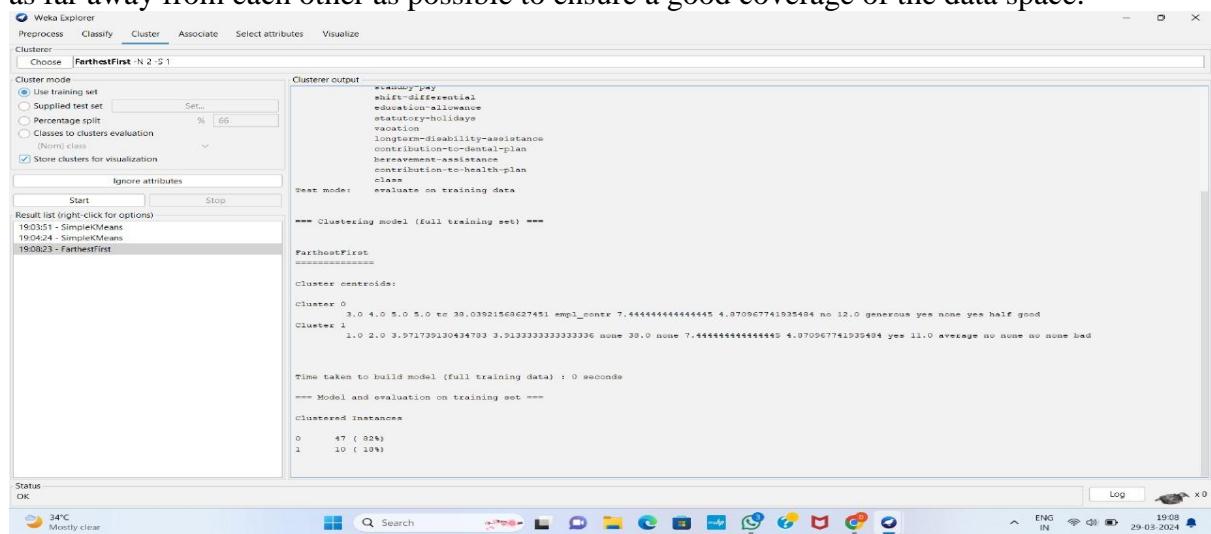
	0	1
Count	48	9
Percentage	(84%)	(16%)

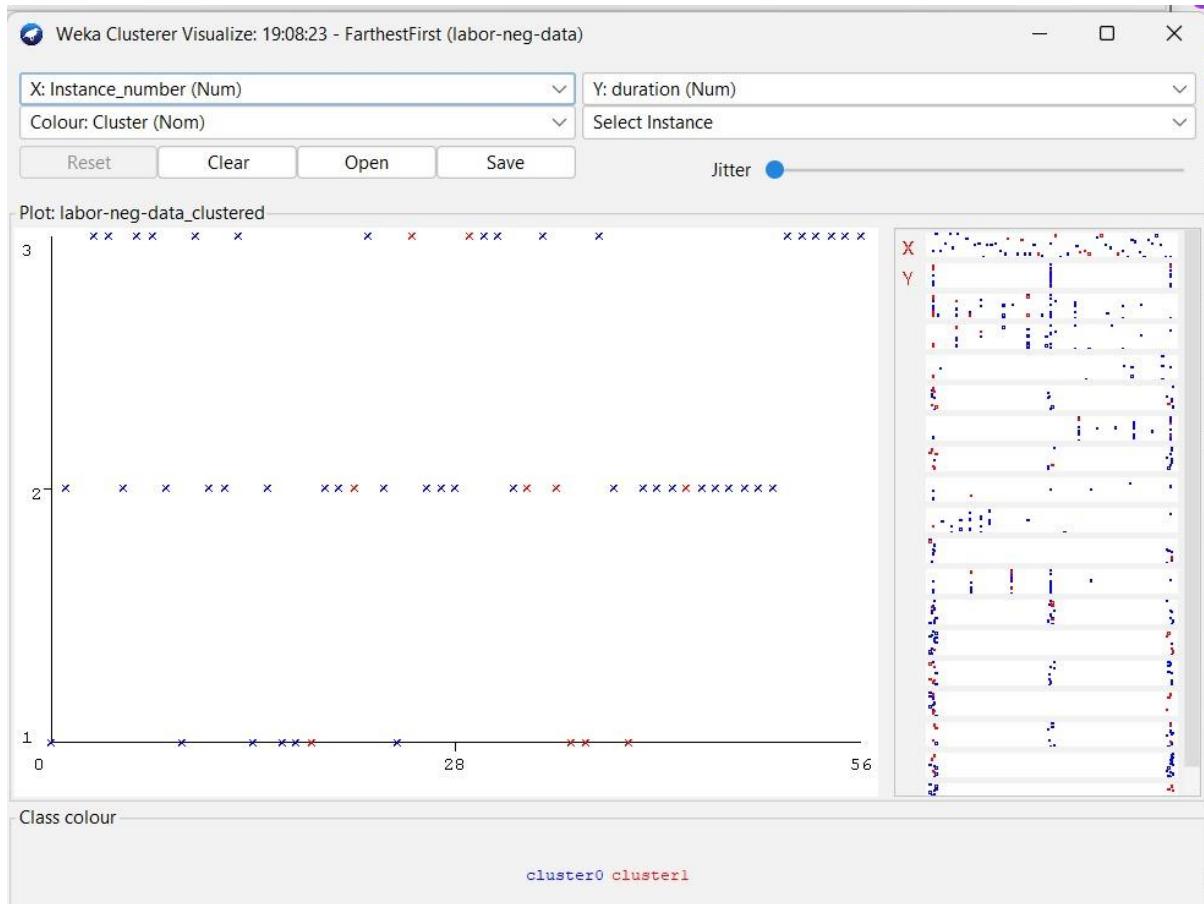
The status bar at the bottom indicates 'OK' and the system tray shows the date and time as 29-03-2024 19:05.



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.

- d) Stemmer: selects the algorithm to eliminate 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 apply stemming for one language. No stemmer is configured when click on the “Select” button menu is deployed and “NullStemmer” is selected.

Procedure:

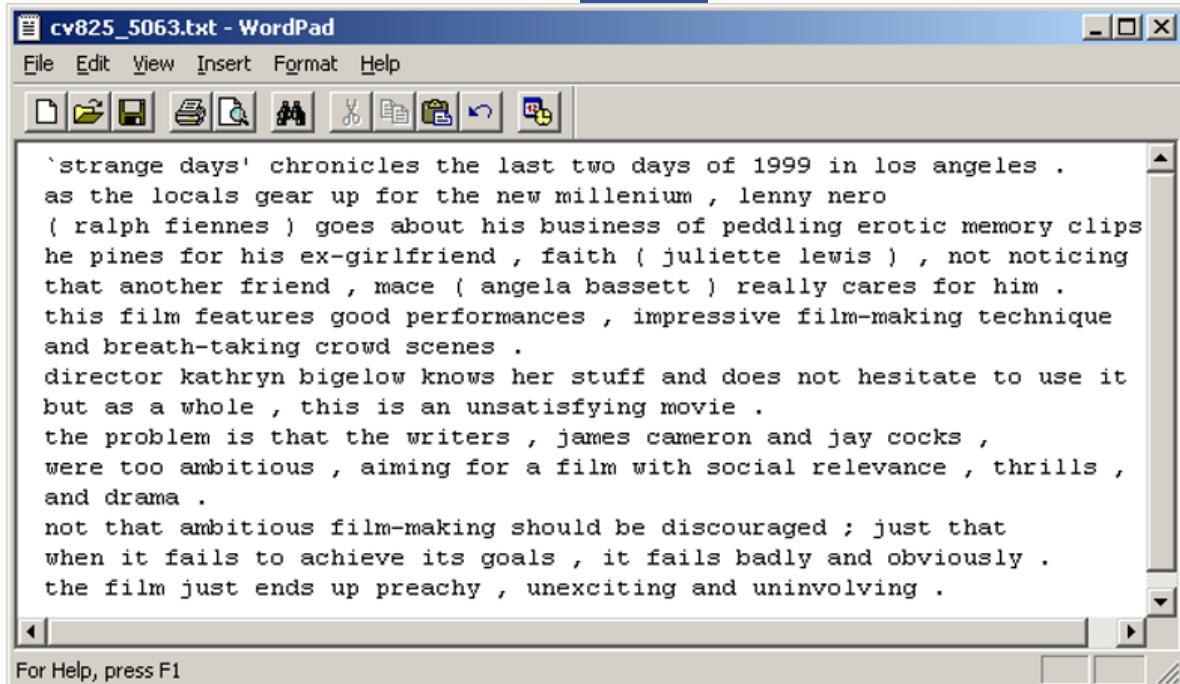
Implementation of Task: Building a model for movies reviews in English for classifying it into positive or negative.

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



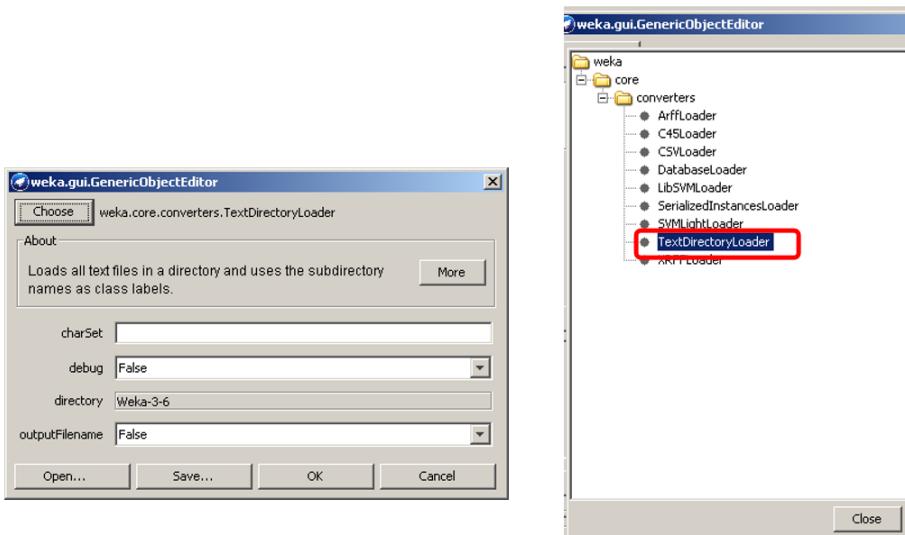
A screenshot of a Windows WordPad application window titled "cv825_5063.txt - WordPad". The window contains a single paragraph of text describing a movie review. The text discusses the movie 'strange days' and its director Kathryn Bigelow. It highlights the performances of actors like Ralph Fiennes, Lenny Nero, and Juliette Lewis, while noting the film's lack of social relevance and its preachy nature.

```
'strange days' chronicles the last two days of 1999 in los angeles .  
as the locals gear up for the new millenium , lenny nero  
( ralph fiennes ) goes about his business of peddling erotic memory clips  
he pines for his ex-girlfriend , faith ( juliette lewis ) , not noticing  
that another friend , mace ( angela bassett ) really cares for him .  
this film features good performances , impressive film-making technique  
and breath-taking crowd scenes .  
director kathryn bigelow knows her stuff and does not hesitate to use it  
but as a whole , this is an unsatisfying movie .  
the problem is that the writers , james cameron and jay cocks ,  
were too ambitious , aiming for a film with social relevance , thrills ,  
and drama .  
not that ambitious film-making should be discouraged ; just that  
when it fails to achieve its goals , it fails badly and obviously .  
the film just ends up preachy , unexciting and uninvolving .
```

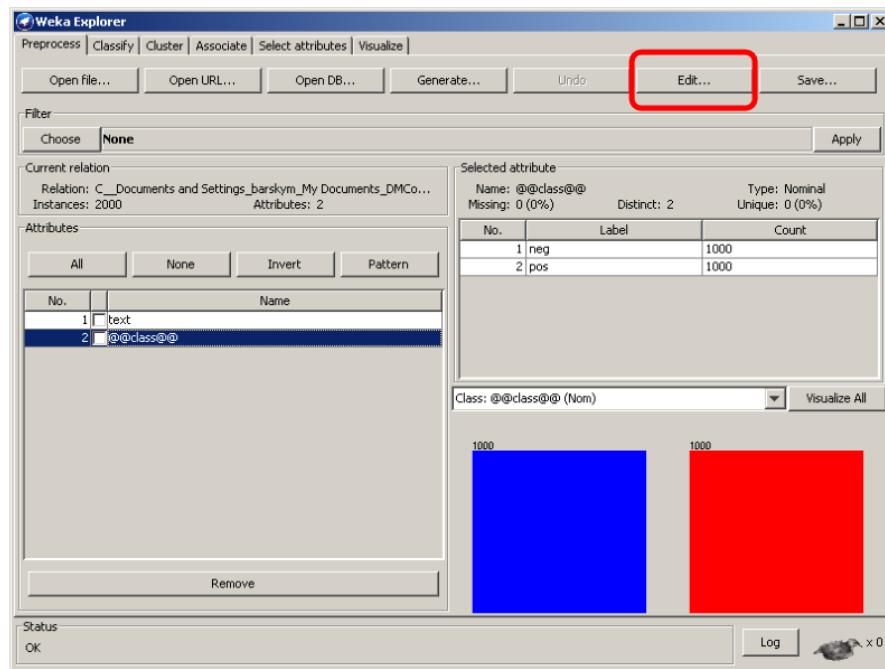
Open WEKA

- Convert file to .arff format

Choose converter



Class



Edit->View

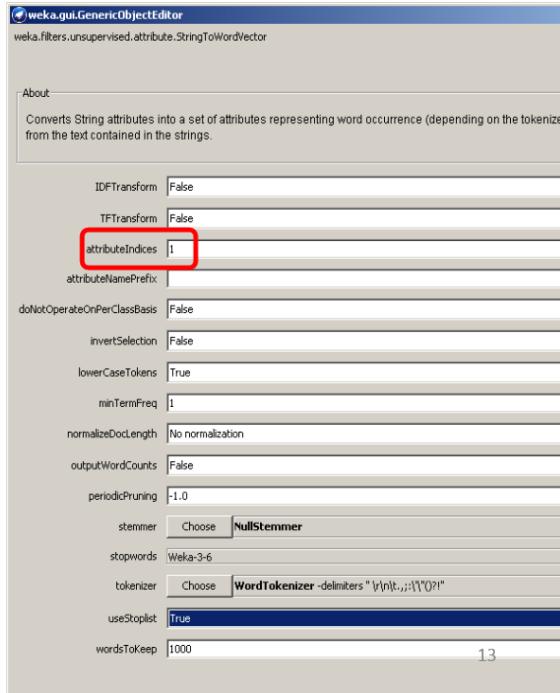
No.	text String	@@class@@ Nominal
1	plot : two teen couples go to a church party , drink and then drive .	neg
10	plot : a young french boy sees his parents killed before his eyes by	neg
100	whether you like the beatles or not , nobody wants to see the bee	neg
1000	two party guys bob their heads to haddaway's dance hit " what is	neg
1001	films adapted from comic books have had plenty of success ,	pos
1002	every now and then a movie comes along from a suspect studio ,	pos
1003	you've got mail works alot better than it deserves to . \n in order to	pos
1004	" jaws " is a rare film that grabs your attention before it shows you	pos
1005	moviemaking is a lot like being the general manager of an nfl team in	pos
1006	on june 30 , 1960 , a self-taught , idealistic , yet pragmatic , young	pos
1007	apparently , director tony kaye had a major battle with new line	pos
1008	one of my colleagues was surprised when i told her i was willing to	pos
1009	after bloody clashes and independence won , lumumba refused to	pos
101	warning : spoilers are included in this review . . . \but it doesn't	neg
1010	the american action film has been slowly drowning to death in a sea	pos
1011	after watching " rat race " last week , i noticed my cheeks were sore	pos
1012	i've noticed something lately that i've never thought of before .	pos
1013	synopsis : bobby garfield (yelchin) lives in a small town with his	pos
1014	synopsis : in this movie , steven spielberg , one of today's finest	pos
1015	the police negotiator is the person with the entirely unenviable job	pos
1016	plot : a young man who loves heavy metal music and especially the	pos
1017	carry on matron is the last great carry-on film in my opinion .	pos
1018	the ultimate match up between good and evil , " the untouchables "	pos

**Save converted file in moviereviews.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.

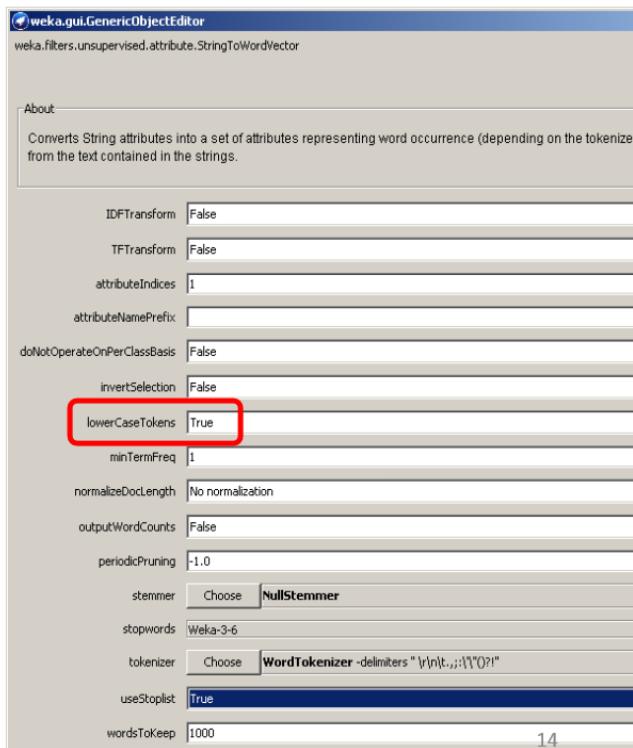
Right-click for options

- Select attribute to convert



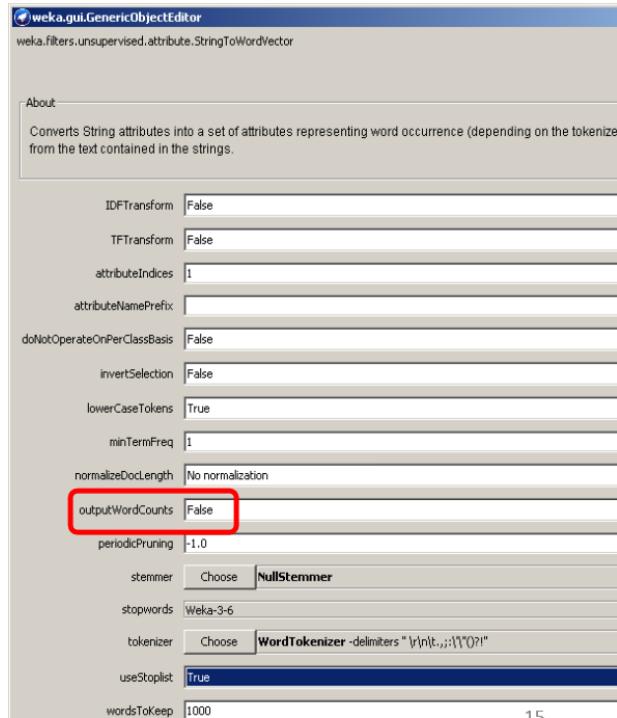
Options

- Can convert all words to lower case – preferred



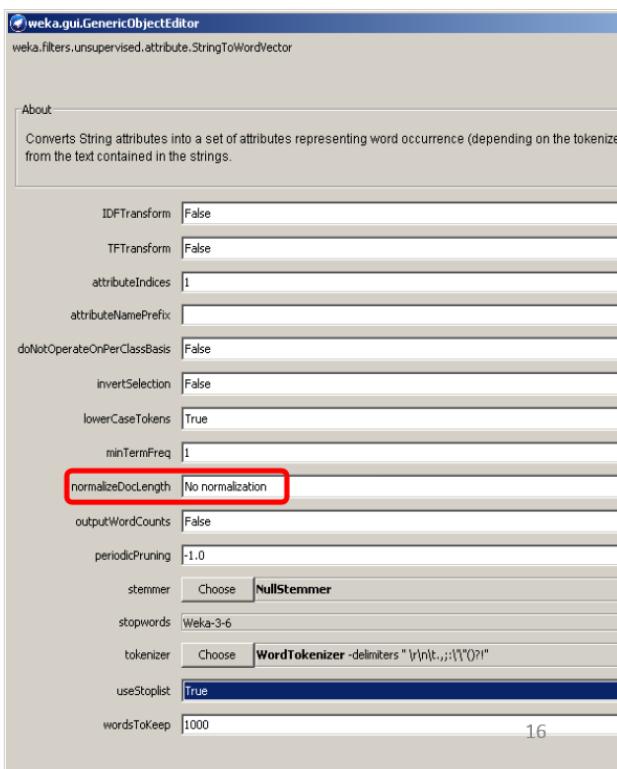
Options

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



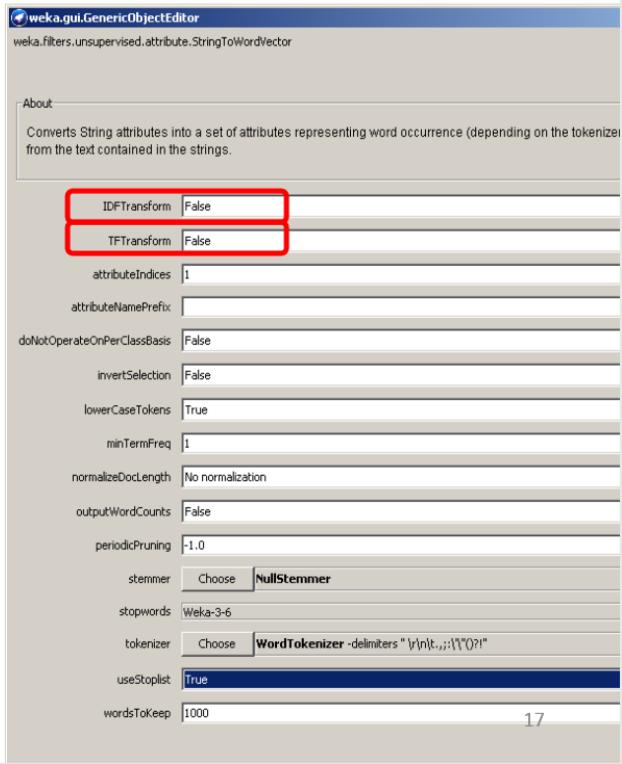
Options

- If ‘outputWordCounts’ 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:



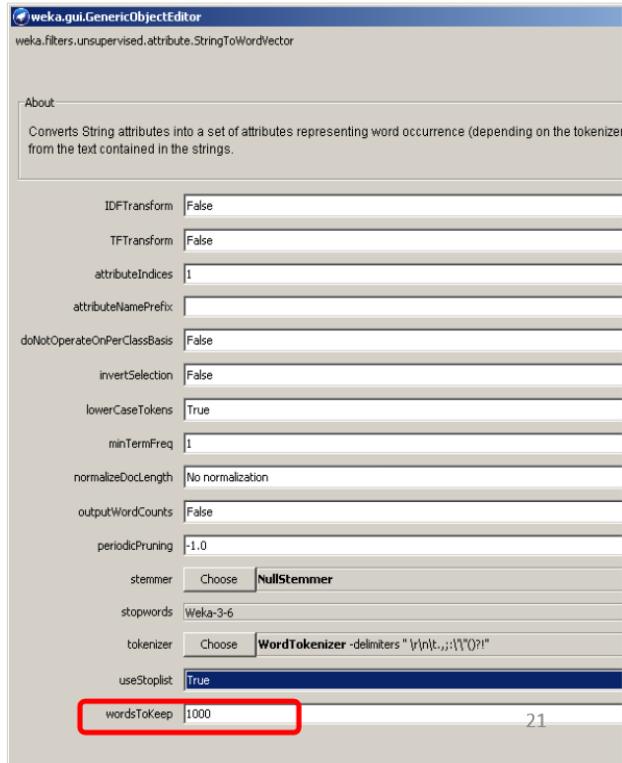
Term Frequency – Document Frequency

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

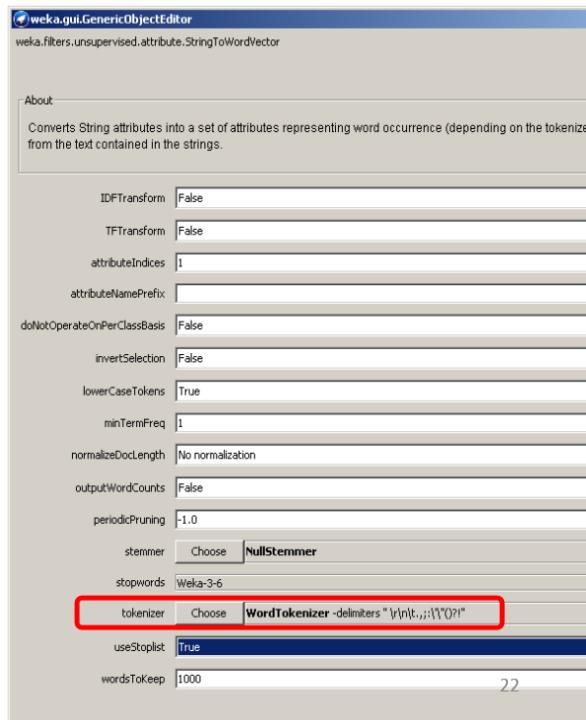
Options

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



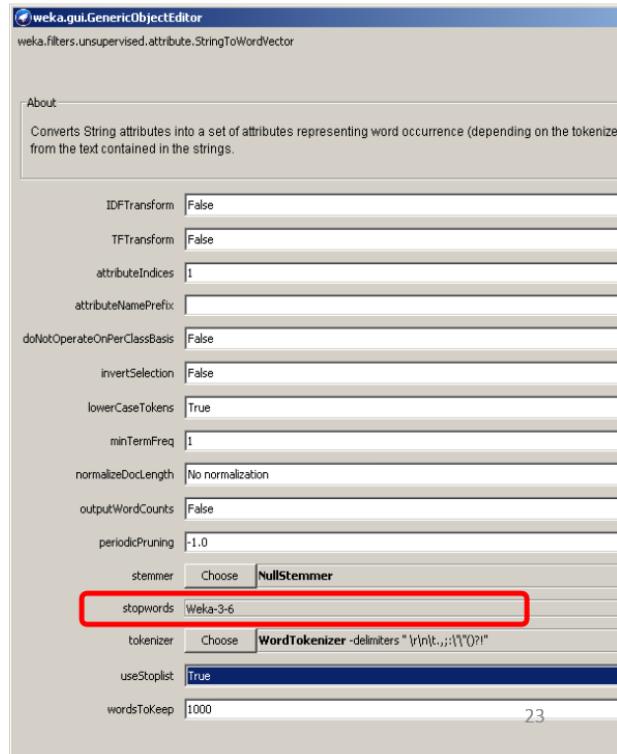
How the words are extracted

- 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



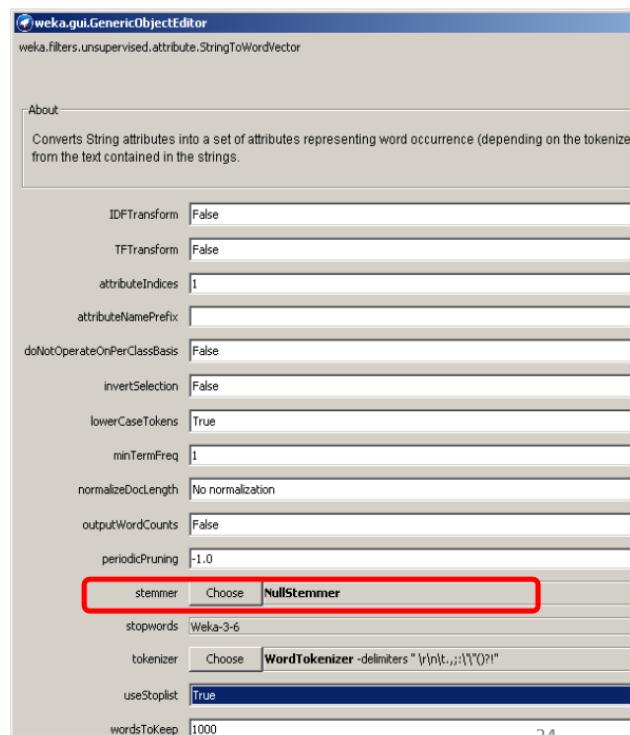
How the words are extracted

- 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`

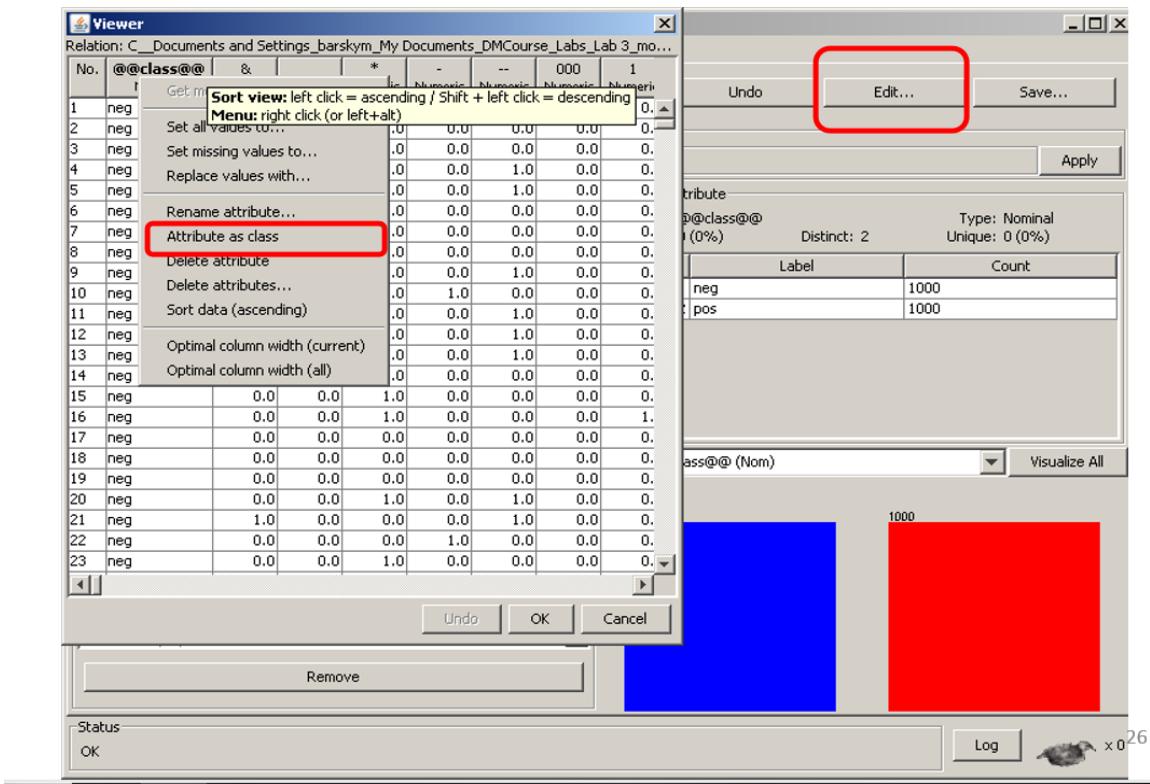


How the words are extracted

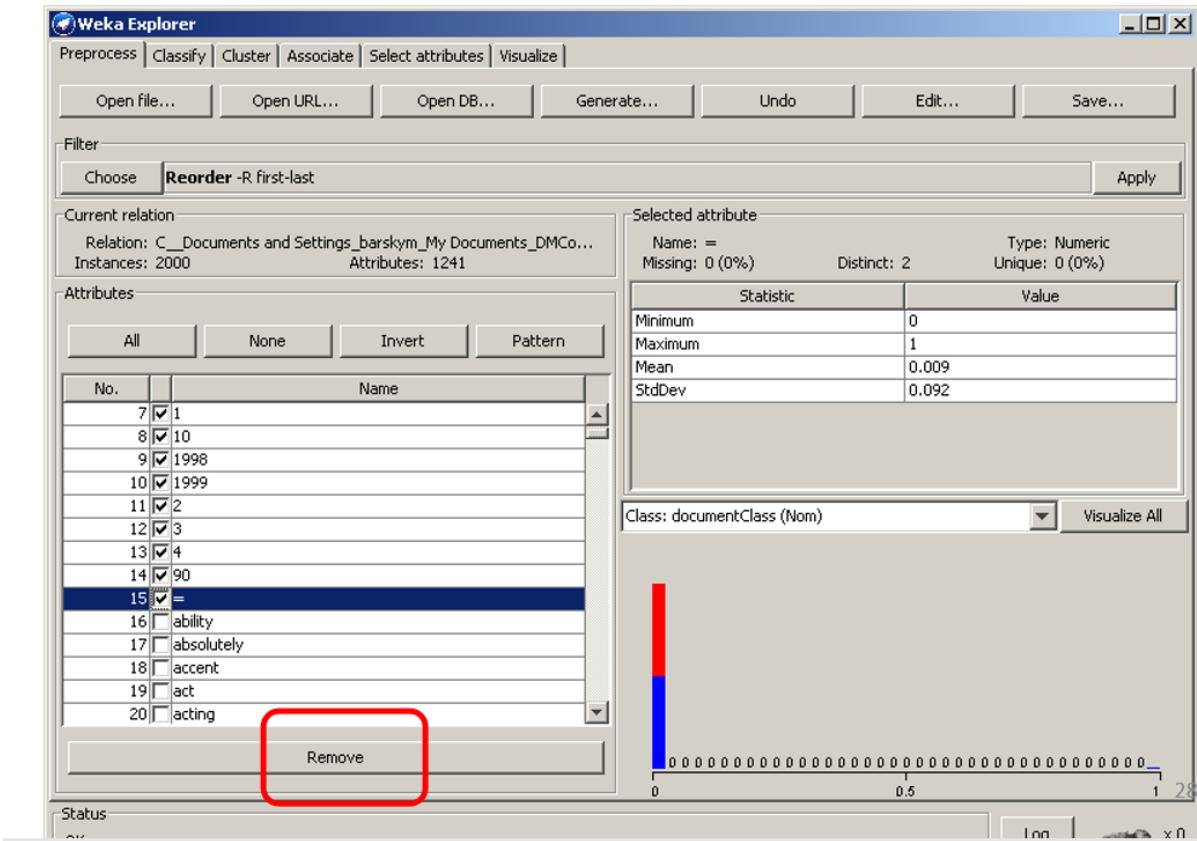
- Stemmer – identifies words which have the same root, for example: “cat”, “cats”, “catlike”, “catty” have similar meaning related to the root “cat” and stemmer treats them all as the same word. No stemmer by default



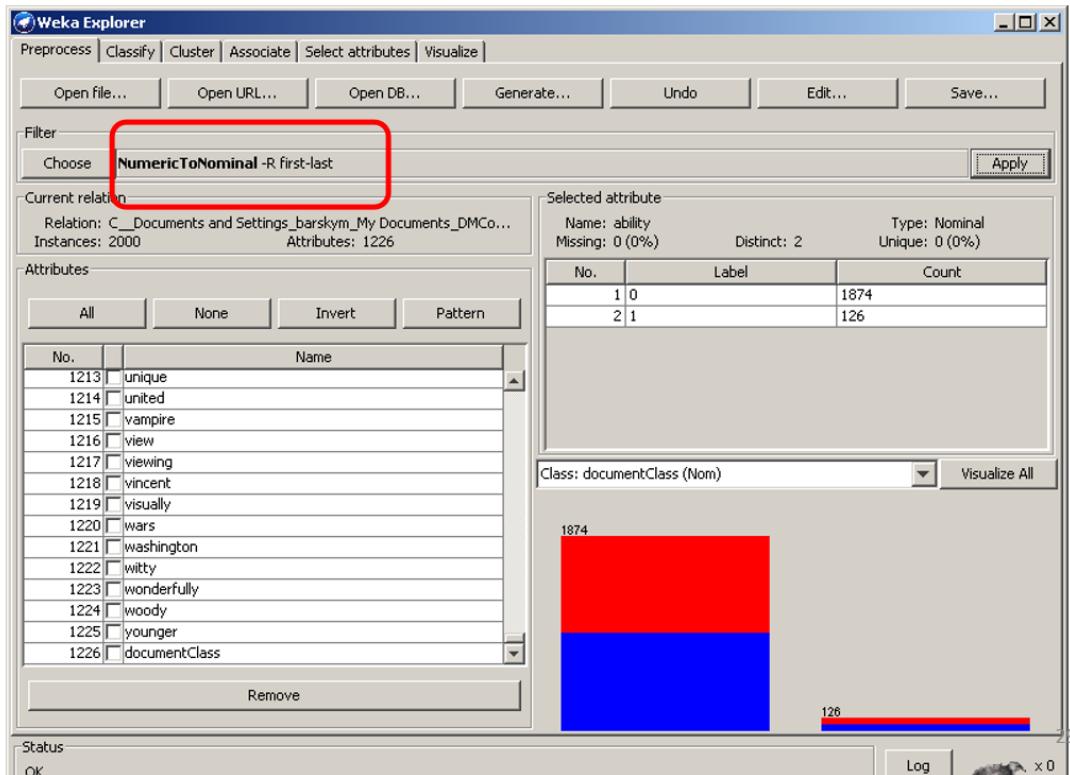
Move class attribute to the end of vector



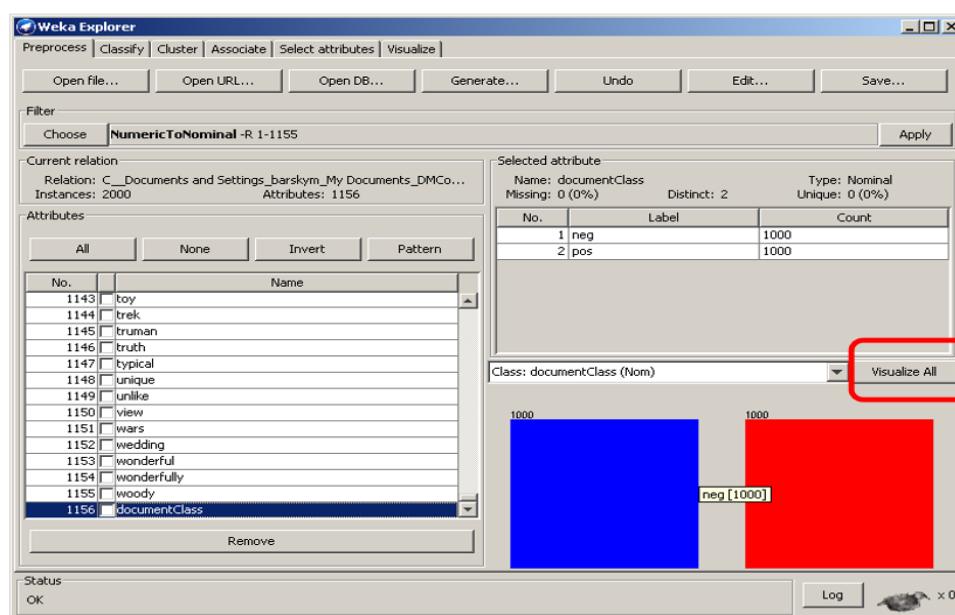
Clean some junk words



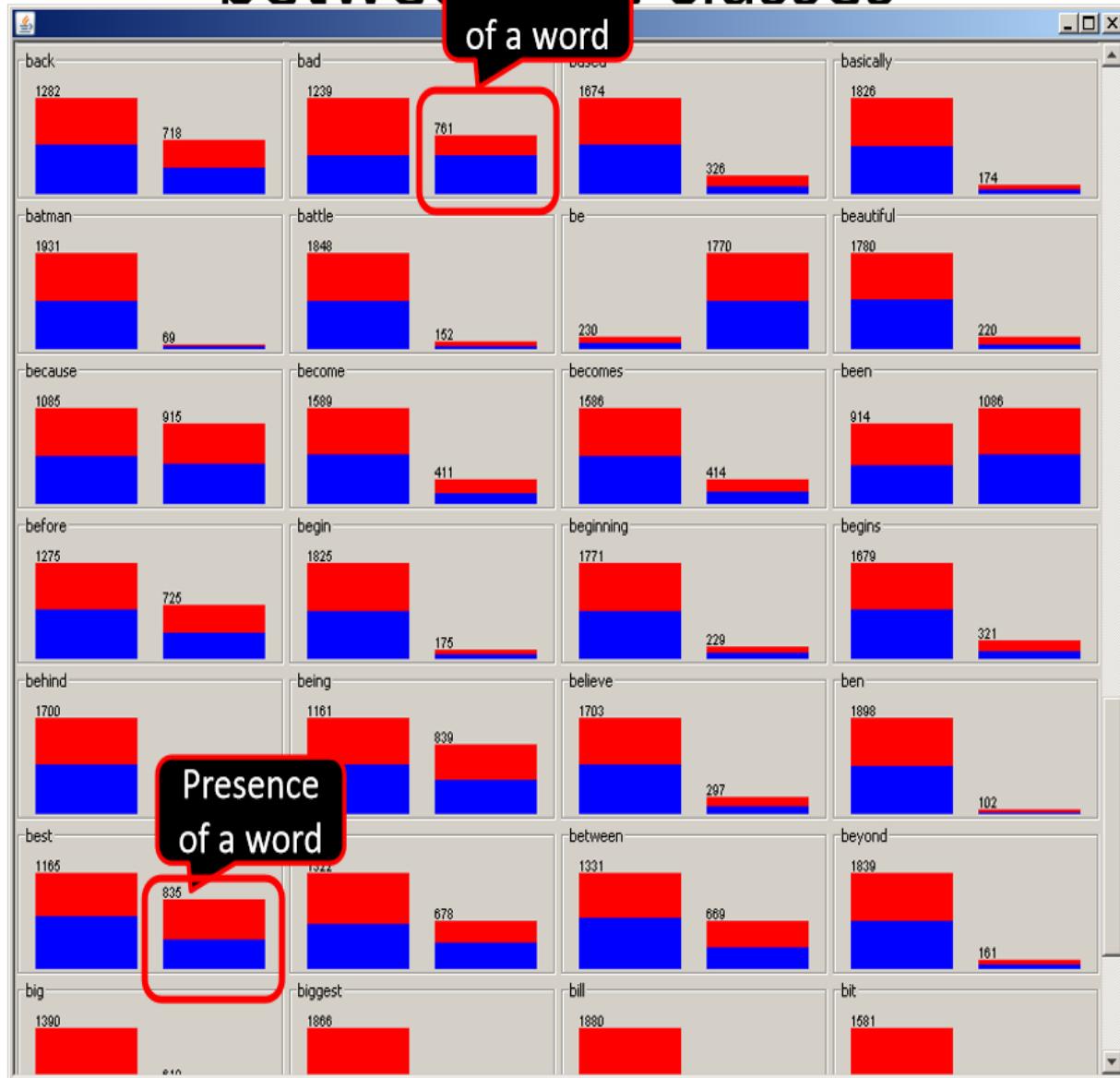
Convert all numeric to nominal (boolean)



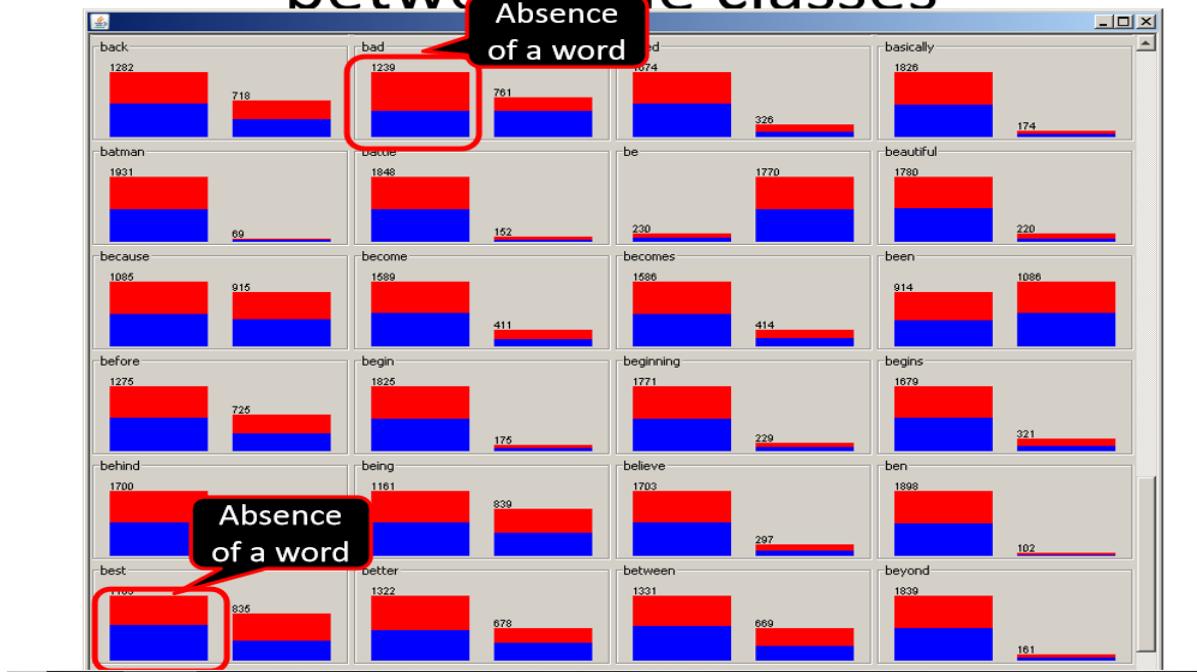
Visualize all attributes



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

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

Classifier

Choose NaiveBayes

Test options

Use training set

Supplied test set Set...

Cross-validation Folds 10

Percentage split % 66 More options...

Classifier output

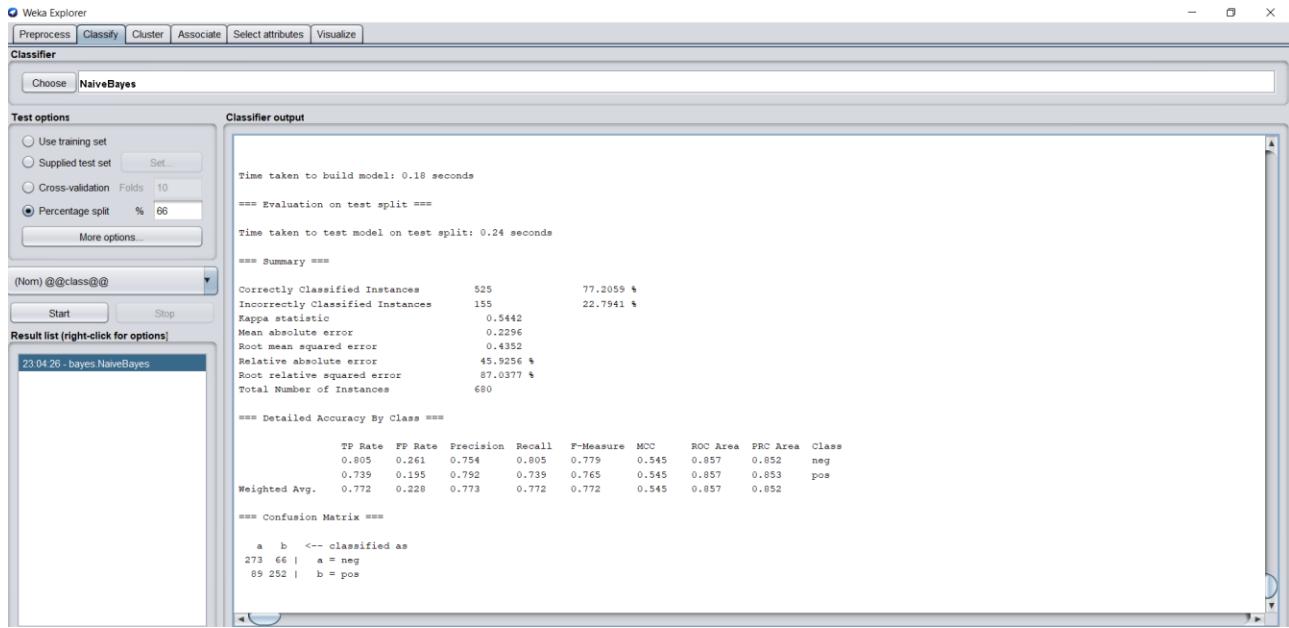
```
==== Run information ====
Scheme:weka.classifiers.bayes.NaiveBayes
Relation: C_Documents and Settings_barskym_My Documents_DMCourse_Labs_Lab
Instances: 2000
Attributes: 1156
[list of attributes omitted]
Test mode:split 66.0% train, remainder test
```

(Nom) documentClass

Start Stop

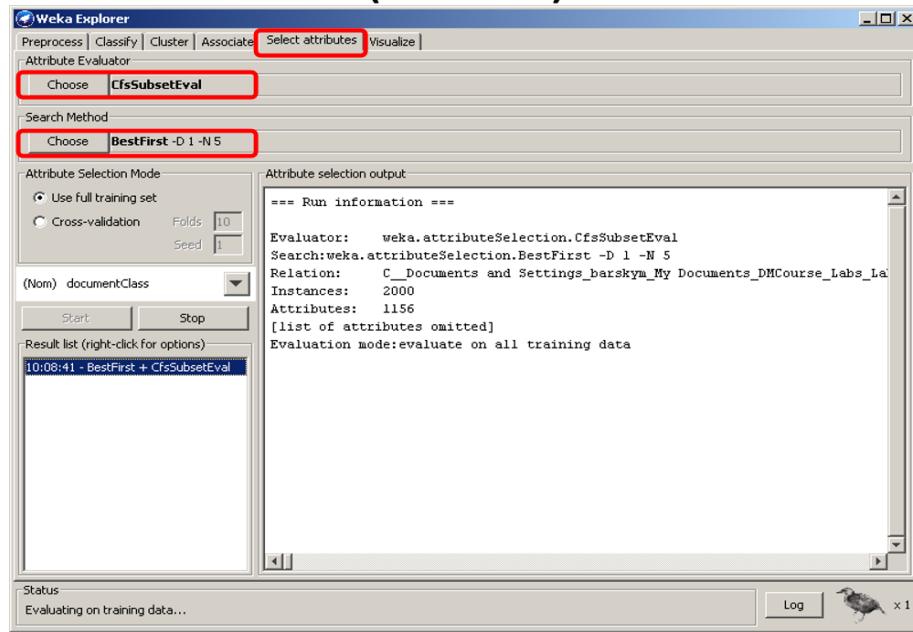
Result list (right-click for options)
10:03:08 - bayes.NaiveBayes

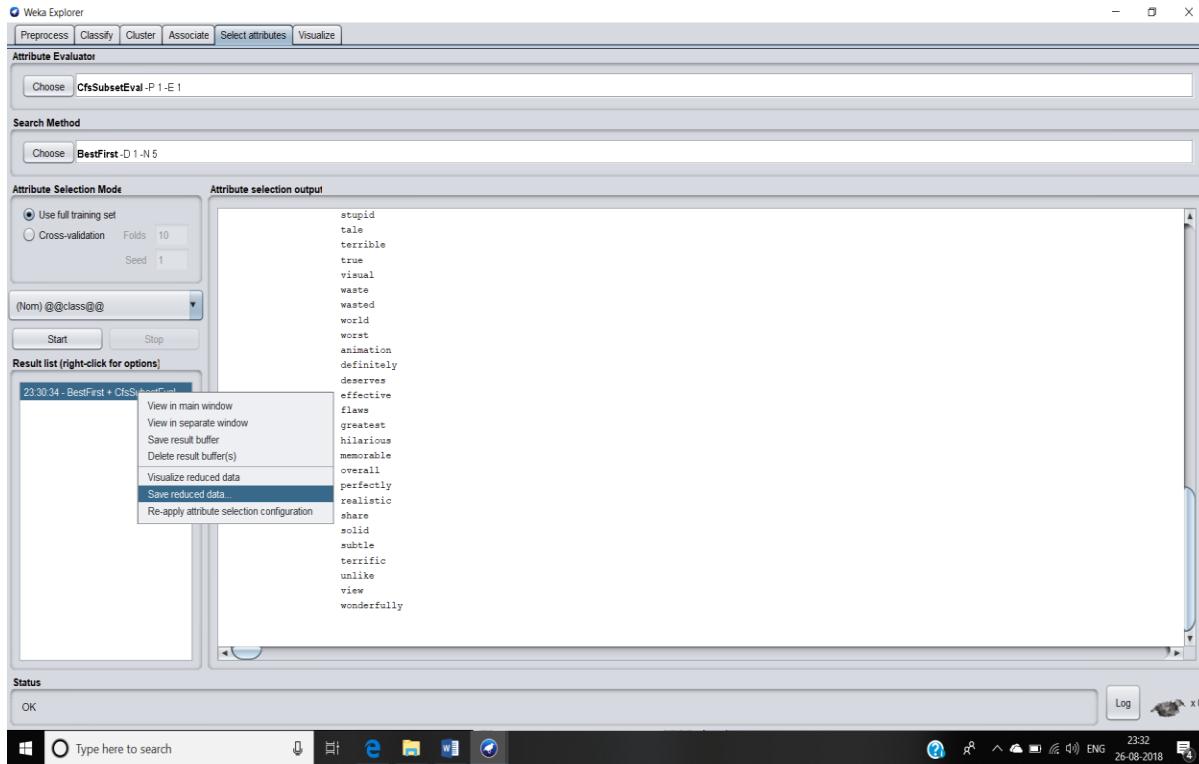
Status Building model on training data... Log x 1



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**

```
23:36:39 - bayes.NaiveBayes - □ ×

Time taken to build model: 0.01 seconds

==== Evaluation on test split ===

Time taken to test model on test split: 0.01 seconds

==== Summary ===

Correctly Classified Instances      535          78.6765 %
Incorrectly Classified Instances   145          21.3235 %
Kappa statistic                   0.5735
Mean absolute error               0.2504
Root mean squared error          0.3821
Relative absolute error           50.0835 %
Root relative squared error     76.4138 %
Total Number of Instances        680

==== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC    ROC Area  PRC Area  Class
      0.779    0.205    0.790     0.779    0.785     0.574   0.878    0.878    neg
      0.795    0.221    0.783     0.795    0.789     0.574   0.878    0.867    pos
Weighted Avg.      0.787    0.213    0.787     0.787    0.787     0.574   0.878    0.872

==== Confusion Matrix ===

  a   b  <-- classified as
264  75 |  a = neg
 70 271 |  b = pos
```

Conclusion:

Text Documents were classified.

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

Dataset: Heart.csv

Prablem 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	fbps	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`.

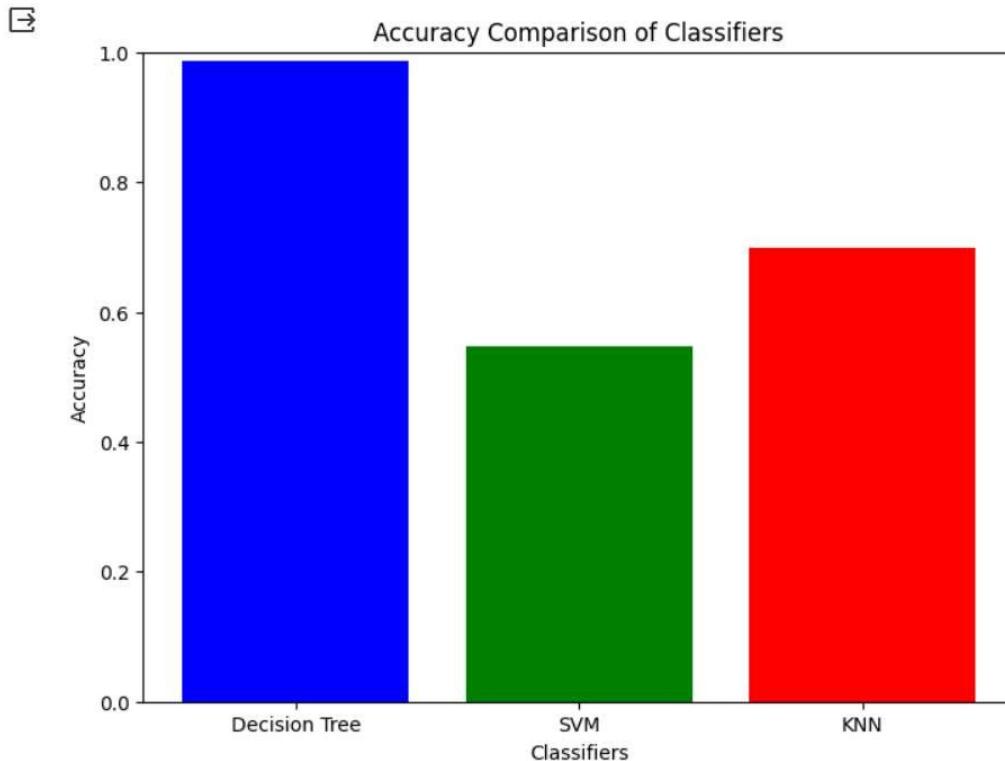
```
[ ] # Step 6: Compare Results
print("Decision Tree Accuracy:", dt_accuracy)
print("SVM Accuracy:", svm_accuracy)
print("KNN Accuracy:", knn_accuracy)

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

❸ 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

The image displays three separate windows of the Weka Explorer interface, each showing the results of a classification run on the same dataset. The top window shows the results for a DecisionStump classifier, the middle for SMOreg, and the bottom for a PolyKernel classifier. Each window includes a toolbar at the top, a classifier selection dropdown, and a main panel for viewing classifier output, test options, and result lists.

DecisionStump Classifier Output:

```
Scheme: weka.classifiers.trees.DecisionStump
Relation: Hdata
Instances: 1025
Attributes: 14
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
Test mode: 10-fold cross-validation

*** Classifier model (full training set) ***

Decision Stump

Classifications

cp <= 0.5 : 0.2454728370221328
cp > 0.5 : 0.7651515151515151
cp is missing : 0.5131707317073171

Time taken to build model: 0.04 seconds

*** Cross-validation ***
*** Summary ***

Correlation coefficient 0.5173
Mean absolute error 0.3654
```

SMOreg Classifier Output:

```
Scheme: weka.classifiers.functions.SMOreg
Relation: Hdata
Instances: 1025
Attributes: 14
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
Test mode: 10-fold cross-validation

*** Classifier model (full training set) ***

SMOreg

weights (no support vectors):
+
0.0419 * (normalized) age
+
0.1145 * (normalized) sex
+
0.303 * (normalized) cp
-
0.2159 * (normalized) trestbps
-
0.0919 * (normalized) chol
+
0.0137 * (normalized) fbs
+
0.021 * (normalized) restecg
+
0.408 * (normalized) thalach
-
0.2971 * (normalized) exang
-
0.2247 * (normalized) oldpeak
+
0.1879 * (normalized) slope
-
0.5618 * (normalized) ca
-
0.2584 * (normalized) thal
+
0.662

Number of kernel evaluations: 525825 (51.355% cached)

Time taken to build model: 0.41 seconds

*** Cross-validation ***
*** Summary ***

Correlation coefficient 0.6788
Mean absolute error 0.2859
Root mean squared error 0.3763
Relative absolute error 57.1569 %
Root relative squared error 75.1948 %
Total Number of Instances 1025
```

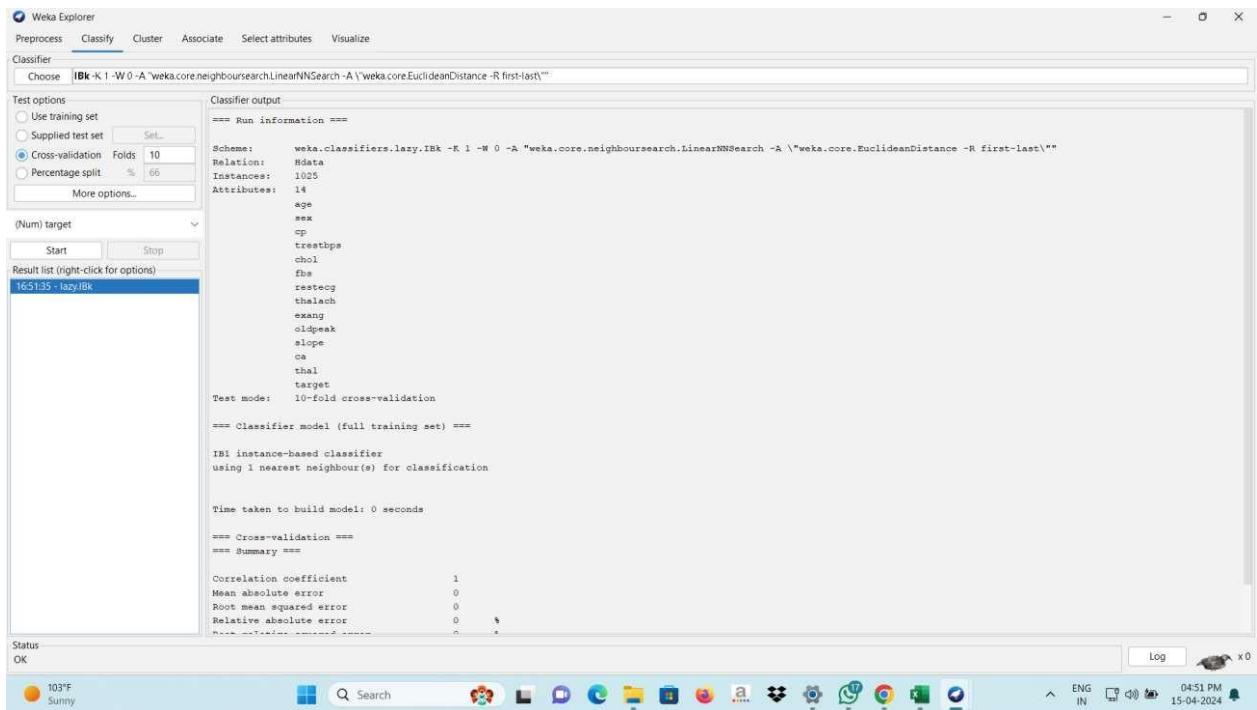
PolyKernel Classifier Output:

```
Scheme: weka.classifiers.functions.PolyKernel
Relation: Hdata
Instances: 1025
Attributes: 14
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
Test mode: 10-fold cross-validation

*** Classifier model (full training set) ***

PolyKernel

Correlation coefficient 0.6788
Mean absolute error 0.2859
Root mean squared error 0.3763
Relative absolute error 57.1569 %
Root relative squared error 75.1948 %
Total Number of Instances 1025
```



Conclusion:

Students will learn & apply the classification algorithms.

T.Bhaskar.

Prepared by:

Dr.T.Bhaskar

Course Coordinator

04/04/2024

Approved by:

Dr. D.B. Kshirsagar

Head, Computer Engineering