Description :- Introduction to WEKA

WEKA (Waikato Environment for knowledge Analysis) is a popular suite of machine learning software written in Java. developed at the University of Waikato. New Zealand. WEKA is a free software available under the GNU General public licence.

Weka is an open source software provides tools for data preprocessing, implementation of several Machine learning algorithms, and visualisation techniques and apply them to real world data mining problems.

History of WEKA:

In 1993, the university of waikato in Newzealand began development of the original version of WEKA. which became a mix of TCL/Tk, C and make files. In 1997, the decision was made to redevelop weka from scratch in modelling algorithms.

The developers are Ian H. witten, Eibe Frank investors are pentaho Inc.

In 2006, pentaho corp. creates of the most popular Business Intelligence.

Features of WEKA

* preprocess: The preprocessing of data is a crical ask in data mining.
* Classify: Classification is one of the essential function in machine learning, where we assign classes or categories to items.
* Cluster: In cluster, a dataset is arranged in different groups/cluster based on some similarities.

Exp No: 8

Date :

Page No:

* Associative : Association rules highlight all the assocations and correlation between items of a dataset

* selective attributes : Every dataset contains a lot of attributes but several of them may not be significantly valuable.

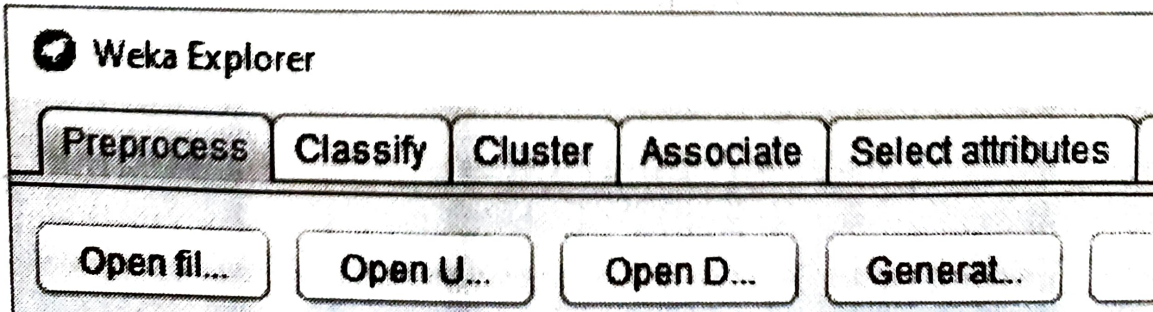
* Visualize : In visualize tab, different plot matrix and graphs are available to show the trends and errors identified by model.

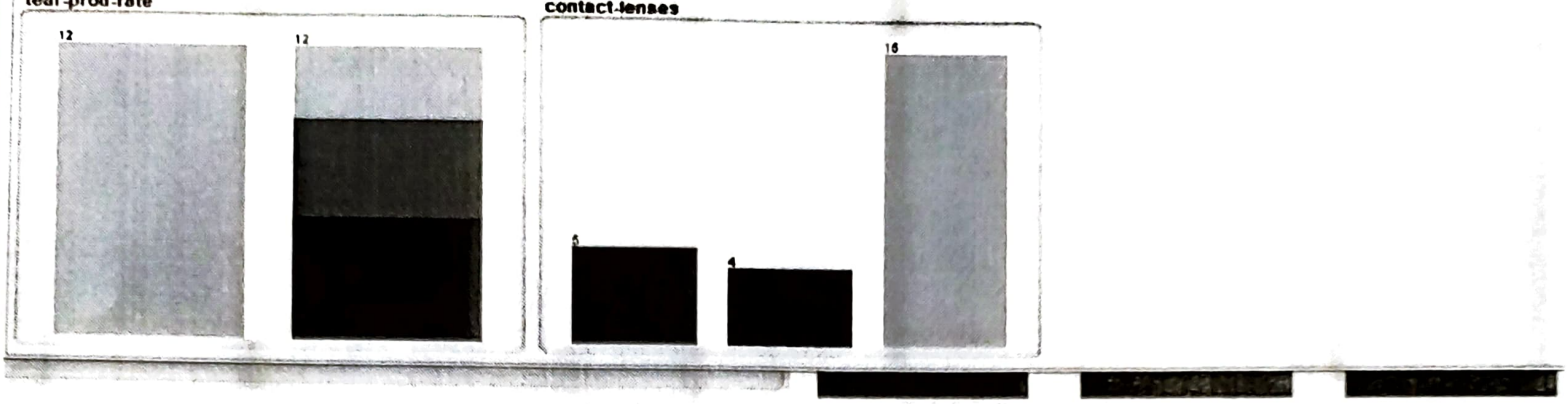
* Latest version : These are two versions of weka
Weka 3.8 is the latest stable version
Weka 3.9 is the development version.

Python Program:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
data=pd.read_csv("/Iris.csv")
print(data.head(10))

data.info()
plt.figure(figsize=(10,7))
x=data["SepalLengthCm"]
plt.hist(x,bins=20,color="green")
plt.title("Sepal length in cm")
plt.xlabel("Sepal_Lengh_cm")
plt.ylabel("Count")
```

IRIS.ARFF:





program:

numpy:

NumPy is a Python library used for working with arrays.
It also has functions for working in domain of linear algebra, fourier transform, and matrices.
NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.
NumPy stands for Numerical Python.

pandas:

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labeled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python

matplotlib:

Matplotlib is a cross-platform, data visualization and graphical plotting library for Python and its numerical extension NumPy. As such, it offers a viable open source alternative to MATLAB. Developers can also use matplotlib's APIs (Application Programming Interfaces) to embed plots in GUI applications

Aim: Perform following data preprocessing tasks using Python.

Normalisation:

Normalisation is used to scale the data of an attribute so that it falls in a smaller range, such as -1.0 to 1.0 or 0.0 to 1.0. It is generally used for classification algorithms

Min-Max Normalisation:

In this technique of knowledge normalisation, a linear transformation is performed on first data.

$$v' = v - \min(A) / \max(A) - \min(A) (\text{newmax}(A) - \text{new min}(A)) + \text{new min}(A)$$

where, A is the Attribute data

min(A) = minimum absolute of A

max(A) = maximum absolute of A

v' = new value of each attribute

v = old value of each attribute

newmax & new min is the max & min value within the range.

Binarize Data:

Sklearn. Preprocessing

Binarize() is a method belongs to a preprocessing module. It plays a key role in the discretization of continous feature values.

Example:

A continous data of pixels values of an 8 bit gray scale image have values ranging between 0 (black) and 255 (white) and one needs it to be black and white. So, using Binarizer() One can set a threshold converting pixel values from 0-127 to 0 and 128-255 as 1

Exp No: 3
Date :

Page No:

Syntax:

sklearn. preprocessing .Binarizer (threshold, copy)

Standardize Data:

Data standardization is the process of rescaling the attributes so that they have mean as 0 and variance as 1. The ultimate goal to perform standarization is to bring down all the features to a common scale without distoring the difference in the range of values.

Hence, the normalized values of

PROGRAM:

```
from numpy import asarray
from sklearn.preprocessing import MinMaxScaler
#define data
data=asarray([[100,0.001],
              [8,0.05],
              [50,0.005],
              [88,0.07],
              [4,0.1]])
print(data)
#define min max scaler
scaler=MinMaxScaler()
#transform data
scaled=scaler.fit_transform(data)
print(scaled)
```

OUTPUT:

```
[[1.0e+02 1.0e-03]
 [8.0e+00 5.0e-02]
[5.0e+01 5.0e-03]
 [8.8e+01 7.0e-02]
 [4.0e+00 1.0e-01]]
[[1.         0.        ]
 [0.04166667 0.49494949]
 [0.47916667 0.04040404]
[0.875      0.6969697 ]
 [0.         1.        ]]
```

PROGRAM

```
[1] from sklearn.preprocessing import Binarizer
    import pandas
    import numpy as np
    url="https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.csv"

[8] colnames=['preg','plas','pres','skin','test','mass','pedi','age',
    'class']

[10] print(colnames)

    ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']

[9] data=pandas.read_csv(url,names=colnames)

[6] print(data)

        preg  plas  pres  skin  test  mass   pedi  age  class
    0      6   148    72    35     0  33.6  0.627   50      1
    1      1    85    66    29     0  26.6  0.351   31      0
    2      8   183    64     0     0  23.3  0.672   32      1
    3      1    89    66    23    94  28.1  0.167   21      0
    4      0   137    40    35   168  43.1  2.288   33      1
    ..   ...   ...   ...   ...   ...   ...    ...  ...    ...
    763   10   101    76    48   180  32.9  0.171   63      0
    764    2   122    70    27     0  36.8  0.340   27      0
    765    5   121    72    23   112  26.2  0.245   30      0
    766    1   126    60     0     0  30.1  0.349   47      1
    767    1    93    70    31     0  30.4  0.315   23      0

    [768 rows x 9 columns]

[11] array=data.values

[12] array

    array([[  6.   , 148.   ,  72.   , ...,  0.627,  50.   ,   1.   ],
           [  1.   ,  85.   ,  66.   , ...,  0.351,  31.   ,   0.   ],
           [  8.   , 183.   ,  64.   , ...,  0.672,  32.   ,   1.   ],
           ...,
           [  5.   , 121.   ,  72.   , ...,  0.245,  30.   ,   0.   ],
           [  1.   , 126.   ,  60.   , ...,  0.349,  47.   ,   1.   ],
           [  1.   ,  93.   ,  70.   , ...,  0.315,  23.   ,   0.   ]])

[13] X=array[:,0:8]
     Y=array[:,8]

     print(X)

    [[  6.   148.    72.   ...  33.6   0.627  50.  ]
     [  1.    85.    66.   ...  26.6   0.351  31.  ]
     [  8.   183.    64.   ...  23.3   0.672  32.  ]
     ...
     [  5.   121.    72.   ...  26.2   0.245  30.  ]
     [  1.   126.    60.   ...  30.1   0.349  47.  ]
     [  1.    93.    70.   ...  30.4   0.315  23.  ]]
```

```
[17]  binarizer=Binarizer(threshold=0.0).fit(X)
      binaryX=binarizer.transform(X)
```

```
print(binaryX[0:10,:])
```

```
[[1. 1. 1. 1. 0. 1. 1. 1.]
 [1. 1. 1. 1. 0. 1. 1. 1.]
 [1. 1. 1. 0. 0. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1.]
 [0. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 0. 0. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 0. 0. 0. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 0. 0. 0. 1. 1.]]
```

Data standardization is **the process of rescaling the attribute**
they have mean as 0 and variance as 1. The ultimate goal to p
standardization is to bring down all the features to a common sc
distorting the differences in the range of the values.

```
[2] from sklearn.preprocessing import StandardScaler
    import pandas
    import numpy as np
    url="https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indi
```

```
[3] colnames=['preg','plas','pres','skin','test','mass','pedi','age',
    'class']
```

```
[4] print(colnames)

    ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
```

```
[5] data=pandas.read_csv(url,names=colnames)
```

```
[6] print(data)
```

	preg	plas	pres	skin	test	mass	pedi	age	class
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
..	...	...	...	...	...	...	...	...	...
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

```
[13] scaler=StandardScaler().fit(X)
     rescaledX=scaler.transform(X)
```

```
print(rescaledX)
```

```
[[ 0.63994726  0.84832379  0.14964075 ...  0.2046
   1.4259954 ]
 [-0.84488505 -1.12339636 -0.16054575 ... -0.6844
  -0.19067191]
 [ 1.23388019  1.94372388 -0.26394125 ... -1.1032
  -0.10558415]
 ...
 [ 0.3429808   0.00330087  0.14964075 ... -0.7351
  -0.27575966]
 [-0.84488505  0.1597866  -0.47073225 ... -0.2402
   1.17073215]
 [-0.84488505 -0.8730192   0.04624525 ... -0.2021
  -0.87137393]]
```

Date :

Page No:

4)

Aim: To write a program to calculate chi-square value using python.

A Chisquare test is used to help determine if observation results are in life with expected results. and to rule out that observations are due to chance. A chisquare test is appropriate for this when data being analyted are from a random sample and when the variable in question in a categori-cal variable.

A chisquare test is one of the statistical test. we can use to decide whether there is a correlation between categorical values.

The formula is $x^2 = \frac{\Sigma (Oi - Ei)^2}{Ei}$

x^2 = chi-square

Oi = Observed value

Ei = Expected value.

```
import seaborn as sns
import pandas as pd
import numpy as np
dataset = sns.load_dataset('tips')
dataset.head()
```

Output:

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3

```
dataset_table = pd.Crosstab(dataset['sex'], dataset
                            ['smoker'])
print(dataset_table)
```

Output:

smoker	Yes	No
sex		
Male	60	97
Female	33	54

```
Observed_values = dataset_table.values
print("Observed values :-\n", Observed_values)
```

Output: Observed values:

```
[[60 97]
 [33 54]]
```

```
from seaborn.distributions import stats
val = stats.chi2_contingency(dataset_table)
print(val)
```

(0.0, 1.0, 1, array([[59.8401, 97.1598], [33.159, 53.840]]))

```
no_of_rows = len(dataset_table.iloc[0:2, 0])
no:_of_columns = len(dataset_table.iloc[0, 0:2])
ddof = (no_of_rows - 1) * (no_of_columns - 1)
print("degree of: freedom", ddof).
```

Output:

degree of freedom : 1

5) Demonstrate performing classification on data sets load each dataset into weka and run Id3, J48 classification algorithm. Study the classifier output. Compute entropy values kappa statistic.

Aim: To demonstrate performing classification on data sets.

Description:

ID3: ID3 stands for Iterative Dichotomiser 3 and is named such because the algorithm iteratively dichotomizes features into two or more groups at each step. It uses top-down approach to build a decision tree.

J48: J48 is based on a topdown strategy, a recursive divide and Conquer strategy. You select which attribute to split on at the root node. that splits the instances into subsets, one for each branch that extends from root.

Eneropy Values: Entropy is measured between 0 and 1. Depending on the no of classes in your dataset, entropy can be greater than 1 but it means the same thing

Entropy is nothing but the measure of disorter.

$$E(S) = \sum_{i=1} -P_i \log_2 P_i.$$

Kappa Statistics:

The kappa statistic is ametric that compares an observed Accuracy with an Expected Accuracy. The kappa statistic is used not only to evaluate a single classifer, but also to evaluate classifiers amongst themselves.

Confusion Matrix;

A confusion matrix is a table that is greater used to describe the performance of a classification model on a set of test for which the true values are known.

K- nearest neighbours algorithm:

In statistics, the k-nearest neighbours algorithms is a non-parametric supervised learning method. The k-nearest neighbours also known as kNN, which uses proximity to make classifications or predictions about the group of an individual data point. While it can be used for either regression or classification problems, it is typically used as a classification algorithm, working off the assumption that similar points can be found near one another.

Naive Bayers Classifier algorithm:-

* Naive Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classific-ation problems.

* It is mainly used in text classification that includes a high dimensional training dataset.

* Naive Bayes classifier is one of the simple and most-effective classification algorithms which helps in building the fast machine learing models that can make quick predictions.

* Some popular Examples of Naive Bayes Algorithm are Spam filtration, sentiment analysis, and Classification articles.

```
Mean absolute error                      0.3559
Root mean squared error                  0.4243
Relative absolute error                  0.4609
Root relative squared error             70.9526 %
Total Number of Instances               97.3291 %
                                         461

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure   MCC      ROC Area  PRC Area  Class
                 0.807     0.458     0.777       0.807   0.792       0.357    0.733     0.816     tested_negati
                 0.542     0.193     0.597       0.542   0.564       0.357    0.733     0.531     tested_positi
Weighted Avg.    0.718     0.369     0.713       0.718   0.715       0.357    0.733     0.720

=== Confusion Matrix ===

   a   b   <-- classified as
 247  59 |   a = tested_negative
  71  84 |   b = tested_positive
```

Kappa statistic:

Cohen's kappa statistic measures interrater reliability (sometimes called interobserver agreement). Interrater reliability, or precision, happens when yourdata raters (or collectors) give the same score to the same data item.
This statistic should only be calculated when:
Two raters each rate one trial on each sample,
or.One rater rates two trials on each sample.

$$k = \frac{p0 - pe}{1 - pe} = 1 - \frac{1 - p0}{1 - pe}$$

Mean Absolute Error
The Mean Absolute Error(MAE) is the average of all absolute errors.The formula is:

$$\mathbf{MAE} = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$

Where:

n = the number of errors,
Σ = summation symbol (which means "add them all up"),
$|x_i - x|$ = the absolute errors.

Root mean squared error:

Root mean square error or root mean square deviation is one of the most commonly used measures for evaluating the quality of predictions. It shows how far predictions fall from measured true values using Euclidean distance.

Root mean square error can be expressed as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

where N is the number of data points, y(i) is the i-th measurement, and y (i) isits corresponding prediction.

Relative absolute error:

It is a way to measure the performance of a predictive model. It's primarily used in machine learning, data mining, and operations management. RAE is notto be confused with relative error, which is a general measure of precision or accuracy for instruments like clocks, rulers, or scales.

$$\frac{|p_1 - a_1| + \ldots + |p_n - a_n|}{|\bar{a} - a_1| + \ldots + |\bar{a} - a_n|}$$

Root relative squared error:

The Root Relative Squared Error (RRSE) is defined as the square root of the sum of squared errors of a predictive model normalized by the sum of squarederrors of a simple model.

the root relative squared error E_i of an individual model i is evaluated by theequation:

$$E_i = \sqrt{\frac{\sum_{j=1}^{n} (P_{(ij)} - T_j)^2}{\sum_{j=1}^{n} (T_j - \bar{T})^2}}$$

where $P_{(ij)}$ is the value predicted by the individual model i for record j (out of n records); T_j is the target value for record j; and $\bar{T}$ is given by the formula:

$$\bar{T} = \frac{1}{n} \sum_{j=1}^{n} T_j$$

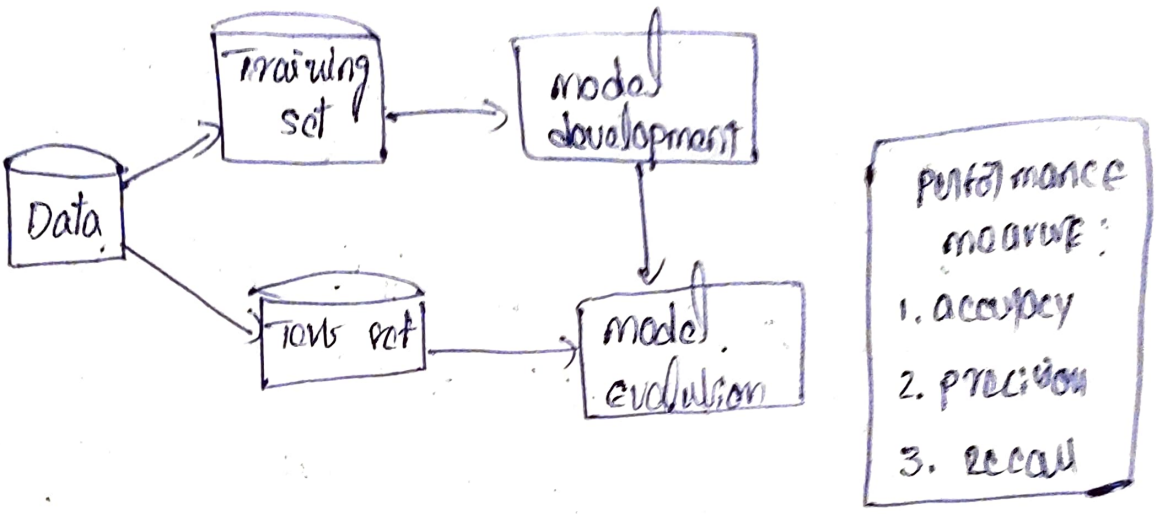
For a perfect fit, the numerator is equal to 0 and $E_i = 0$. So, the E_i index rangesfrom 0 to infinity, with 0 corresponding to the ideal.

, where:

n: represents the number of observations

Experiment - 7

Naive Bayer Classification



~~P(h/D~~

Naive Bayer is a statistical classification technique based on Bayer.

$$P(h/D) = \frac{P(D/h)\,P(h)}{P(D)}$$

* P(h): The probability of hypothesis h being true. This is known as the Prior probability of h.
* P(D): The probability of the data This is known as the prior Probability.
* P(h/D): The probability of hypothesis his given the data D This is known as

Week – 7

Write a program of Naive Bayesian classification usi programming language.

AIM: To write a program of Naive Bayesian classification using P programming language.

DESCRIPTION:

Naive Bayesian is a statistical classification technique based on

bayes theorem. It is fast, accurate and liable algorithm.

Naive Bayesian has classifiers have high accuracy and speed on l datasets.

The Naive Bayes classification algorithm is **a probabilistic classi** based on probability models that incorporate strong independence assumptions.

PROGRAM:

```
from sklearn import datasets
from sklearn import metrics
from sklearn.naive_bayes import GaussianNB
dataset = datasets.load_iris()
print(datasets)
model = GaussianNB()
model.fit(dataset.data,dataset.target)
print(model)
expected = dataset.target
predicted = model.predict(dataset.data)
print(metrics.classification_report(expected,predicted))
print(metrics.confusion_matrix(expected,predicted))
```

OUTPUT:

```
<module 'sklearn.datasets' from '/usr/local/lib/python3.7/dist-packages/sklearn/datasets/__init__.py'>
GaussianNB()
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.94      0.94      0.94        50
           2       0.94      0.94      0.94        50

    accuracy                           0.96       150
   macro avg       0.96      0.96      0.96       150
weighted avg       0.96      0.96      0.96       150

[[50  0  0]
 [ 0 47  3]
 [ 0  3 47]]
```

Week – 10

Write a Python program to generate frequent item sets / association rules using Apriori algorithm.

AIM: To write a Python program to generate frequent item sets / association rules using Apriori algorithm.

DECRIPTION:

The algorithm is used to finding frequent itemset in a dataset for boolean association rule.

Name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. We apply an iterative approach or level-wise search where k-frequent itemsets are used to find k+1 itemsets.

To improve the efficiency of level-wise generation of frequent itemsets, an important property is used called **Apriori property** which helps by reducing the search space.

Apriori Property: All non-empty subset of frequent itemset must be frequent. The key concept of Apriori algorithm is its anti-monotonicity of support measure.

Frequent Itemset: The itemset that occurs frequently is called frequent itemset.

PROGRAM:

```
pip install apyori
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5975 sha256=15a6583e7917bfeb62bedbd6532e8c18b988a87105cfc6715453fe7cc4ac0198
  Stored in directory: /root/.cache/pip/wheels/cb/f6/e1/57973c631d27efd1a2f375bd6a83b2a616c4021f24aeb84080
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
import numpy as np
import pandas as pd
from apyori import apriori
```

```
data=pd.read_csv('/apri.csv')
```

```
print(data)
```

```
    Wine  Chips  Bread  Butter  Milk  Apple
0   Wine  Chips  Bread  Butter  Milk  Apple
1    NaN    NaN  Bread  Butter  Milk    NaN
2    NaN  Chips    NaN     NaN   NaN  Apple
3   Wine  Chips  Bread  Butter  Milk  Apple
4   Wine  Chips    NaN     NaN  Milk    NaN
5   Wine  Chips  Bread  Butter   NaN  Apple
6   Wine  Chips    NaN     NaN  Milk    NaN
7   Wine    NaN  Bread     NaN   NaN  Apple
8   Wine    NaN  Bread  Butter  Milk    NaN
9    NaN  Chips  Bread  Butter   NaN  Apple
10  Wine    NaN    NaN  Butter  Milk  Apple
11  Wine  Chips  Bread  Butter  Milk    NaN
12  Wine    NaN  Bread     NaN  Milk  Apple
13  Wine    NaN  Bread  Butter  Milk  Apple
14  Wine  Chips  Bread  Butter  Milk  Apple
15   NaN  Chips  Bread  Butter  Milk  Apple
16   NaN  Chips    NaN  Butter  Milk  Apple
17  Wine  Chips  Bread  Butter  Milk  Apple
18  Wine    NaN  Bread  Butter  Milk  Apple
```

```
data.shape
```

```
(19, 6)
```

```
records=[]
for i in range(0,19):
  records.append([str(data.values[i,j]) for j in range(0,
```

```
ass_rules=apriori(records,min_support=0.5,confidence=0.7)
result=list(ass_rules)
```

```
print(len(result))
```

```
21
```

Week – 11

Write a Python program to generate frequent item sets / association rules using FP-growth Tree algorithm.

AIM: To write a Python program to generate frequent item sets / association rules using FP-growth Tree algorithm.

DECRIPTION:

Frequent Itemset: The itemset that occurs frequently is called frequent itemset.

The FP-Growth Algorithm is an alternative way to find frequent item sets without using candidate generations. For so much, it uses a divide-and-conquer strategy. The core of this method is the usage of a special data structure named frequent-pattern tree (FP-tree), which retains the item set association information.

FP-Tree: The frequent-pattern tree (FP-tree) is a compact data structure that stores quantitative information about frequent patterns in a database. Each transaction is read and then mapped onto a path in the FP-tree. This is done until all transactions have been read.

PROGRAM:

```
pip install pyfpgrowth
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pyfpgrowth
  Downloading pyfpgrowth-1.0.tar.gz (1.6 MB)
     |████████████████████████████████| 1.6 MB 7.6 MB/s
Building wheels for collected packages: pyfpgrowth
  Building wheel for pyfpgrowth (setup.py) ... done
  Created wheel for pyfpgrowth: filename=pyfpgrowth-1.0-py2.py3-none-any.whl size=5503 sha256=28a93758a505c3977be0eeba4b30884297f184253480...
  Stored in directory: /root/.cache/pip/wheels/73/97/4b/f12ac994f6bbb9959739625543582 4c73ad3916be1e678be55
Successfully built pyfpgrowth
Installing collected packages: pyfpgrowth
Successfully installed pyfpgrowth-1.0
```

```
#sample code to do FP- growth in python
import pyfpgrowth
#creating Sample Transactions
transactions = [
    ['Milk', 'Bread','Saffron'],
    ['Peanuts','Milk'],
    ['Honey','Coconut','Water'],
    ['Orange','Jam']
]
```

```
#finding the frequent patterns with min support threshold=0.5
FrequentPatterns = pyfpgrowth.find_frequent_patterns(transactions = transactions,support_threshold = 0.5)
print(FrequentPatterns)
```

OUTPUT:

{('Bread',): 1, ('Bread', 'Milk'): 1, ('Saffron',): 1, ('Bread', 'Saffron'): 1, ('Milk', 'Saffron'): 1, ('Bread', 'Milk', 'Saffron'): 1, ('Peanuts',): 1, ('Milk', 'Peanuts'): 1, ('Honey',): 1, ('Coconut',): 1, ('Coconut', 'Honey'): 1, ('Water',): 1, ('Coconut', 'Water'): 1, ('Honey', 'Water'): 1, ('Coconut', 'Honey', 'Water'): 1, ('Orange',): 1, ('Jam',): 1, ('Jam', 'Orange'): 1, ('Milk',): 2}

```
#generating rules with min confidence threshold=0.5
Rules = pyfpgrowth.generate_association_rules(patterns = FrequentPatterns,confidence_threshold=0.5)
print(Rules)
```

OUTPUT:

{('Bread',): (('Milk', 'Saffron'), 1.0), ('Milk',): (('Peanuts',), 0.5), ('Saffron',): (('Bread', 'Milk'), 1.0), ('Bread', 'Milk'): (('Saffron',), 1.0), ('Bread', 'Saffron'): (('Milk',), 1.0), ('Milk', 'Saffron'): (('Bread',), 1.0), ('Peanuts',): (('Milk',), 1.0), ('Coconut',): (('Honey', 'Water'), 1.0), ('Honey',): (('Coconut', 'Water'), 1.0), ('Water',): (('Coconut', 'Honey'), 1.0), ('Coconut', 'Honey'): (('Water',), 1.0), ('Coconut', 'Water'): (('Honey',), 1.0), ('Honey', 'Water'): (('Coconut',), 1.0), ('Jam',): (('Orange',), 1.0), ('Orange',): (('Jam',), 1.0)}

Aim: To visualize the datasets using matplotlib in python (Histogram, Barplot, Bar chart, pie chart etc)

Description:

* Matplotlib is a low level graph plotting library in python that serves as a visualization utility.
* Matplotlib was created by John. D. Hunter
* Matplotlib is open source and we can use it freely
* Matplotlib is mostly written in python, a few segments are written in C, objective -C and Is for platform compatibility

* Histogram:

- A Histogram is a graph showing frequency distributions.
- It is a graph showing the number of observations with in each given interval.
- In Matplotlib, we use the hist() function to create histograms
- The hist() function will use an array of numbers to create a histogram, the array is sent to the function as an argument.

Boxplot:

A Boxplot is also known as whisker plot is created to display the summary of the set of data values having properties like minimum, first quartile, median, third quartile and maximum. In the box plot, a box plot is created from the first quartile to the third quartile, a vertical line is also there which goes through the box at the median.

Bar Charts:

Bar charts can be made with matplotlib. matplotlib is a python module that lets you plot all kinds of charts Bar chart is one of the type of charts it can be plot. There are many different variations of bar chart. the method bar() creates a bar chart.

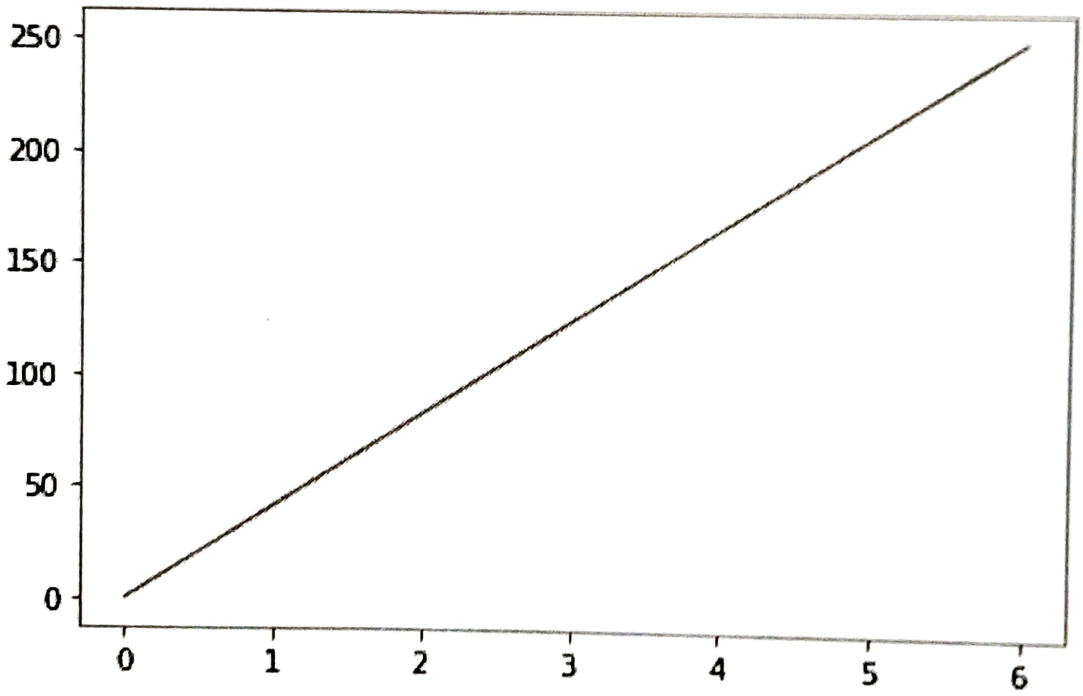
pie chart

With pyplot, you can use the pie() function to draw pie charts. A pie chart is a circular statistical plot that can display only one series of data. that are of the Chart is total percentage of given data. The area of slices of pie represents percentage of the parts of the data.

Week – 12

Visualize the datasets using matplotlib in python.(Histogra
Bar chart, Pie chart etc.,)

```
import matplotlib.pyplot as plt
import numpy as np

xpoints=np.array([0,6])
ypoints=np.array([0,250])

plt.plot(xpoints,ypoints)
plt.show()
```

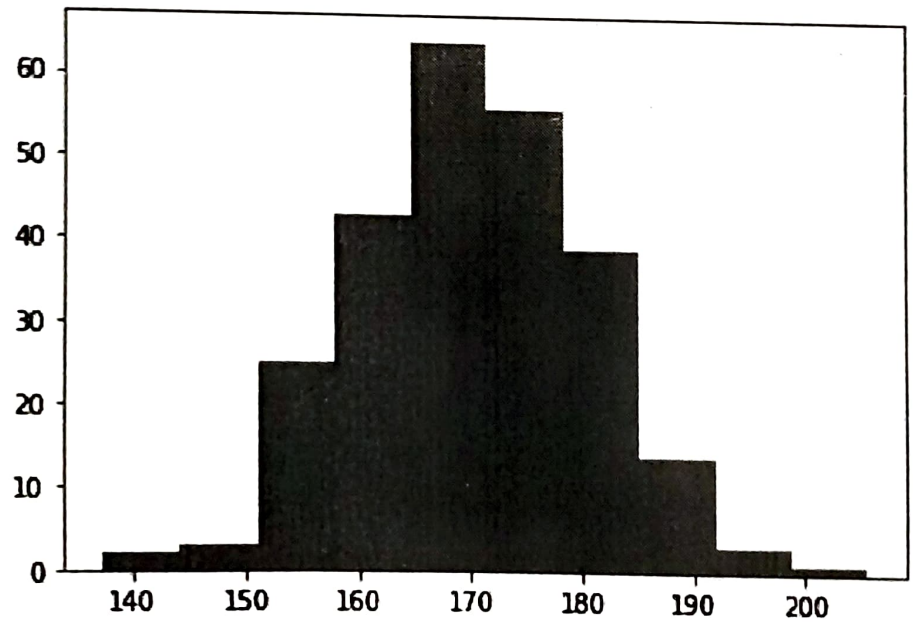


The plot() function is used to draw points (markers)in a diagram B

```
#A normal data distribution by numpy
import numpy as np

x=np.random.normal(170,10,250)

print(x)
```

```
[162.06219045 168.89655805 178.02696963 173.43754038 168.10975633
 167.257875   160.94525091 171.33103894 174.07918171 170.8098853
 176.02079345 182.46538275 178.28976614 175.821064   171.2425441
 180.34654865 167.67320374 179.9604204  163.11327908 167.64301519
 156.65948856 181.27839547 162.84200082 171.8238321  175.71627964
 164.50983203 170.4828843  154.55699173 169.39975546 163.27211912
 168.03416565 149.03720244 167.31908601 161.96142092 173.05782436
 172.27035122 170.52218219 183.26859979 151.07846246 168.80611226
 163.90645598 180.21534805 166.31669544 176.74855211 177.53949826
 177.9365034  166.76187359 168.26793957 165.03540585 161.48454623
 187.65374431 183.84429794 190.83578551 168.30900982 150.95911974
```

```
#The hist() function will read the array and produce a histogram:
import matplotlib.pyplot as plt
import numpy as np

x=np.random.normal(170,10,250)

plt.hist(x)
plt.show()
```



Box Plot: It is a type of chart that depicts a group of numerical data through their quartiles. It is a simple way to visualize the shape of our data. It makes comparing characteristics of data between categories very easy

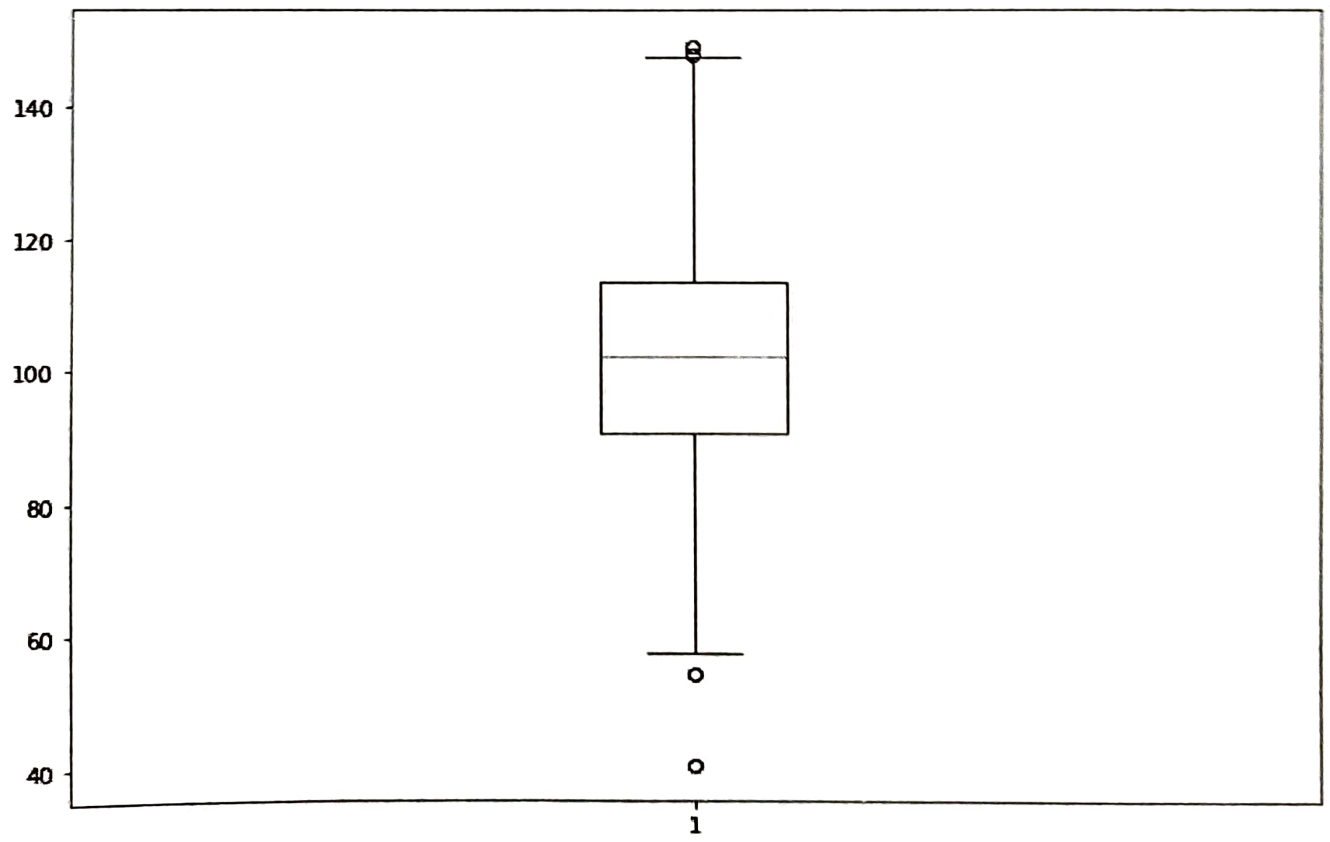
Uses of a Box Plot

Box plots provide a visual summary of the data with which we can quickly identify the average value of the data, how dispersed the data is, whether the data is skewed or not (skewness).

The Median gives you the average value of the data.

Box Plots shows Skewness of the data-

```
import numpy as np
#creating dataset
np.random.seed(10)
data=np.random.normal(100,20,200)

fig=plt.figure(figsize=(10,7))

#creating plot

plt.boxplot(data)

#show plot
plt.show()
```



Bar Chart:

A Bar graph or a Histogram is the diagr... statistics. In bar graphs or histograms, the use of graphs, ... tabular data makes it very easy to understand the concept and relationships among data.

The pictorial representation of data in groups, either in horizontal o... vertical bars where the length of the bar represents the value of t... present on axis. They (bar graphs) are usually used to display or ... the information belonging to 'categorical data' i.e; data that fit in som... category.

```
import matplotlib.pyplot as plt
fig=plt.figure()
ax=fig.add_axes([0,0,1,1])
langs=['C','C++','Java','Python','PHP']
students=[23,17,35,29,12]
ax.bar(langs,students)
plt.show()
```