

MACHINE LEARNING LAB

ETCS-454

Faculty Name: Mr. Anupam Kumar

Name: Varun Negi

Roll No: 13314802719

Semester: 8th

Group: 8C7



Maharaja Agrasen Institute of Technology, PSP area,
Sector — 22, Rohini, New Delhi — 110085

(Affiliated to Guru Gobind Singh Indraprastha University,
New Delhi)



MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

VISION

To nurture young minds in a learning environment of high academic value and imbibe spiritual and ethical values with technological and management competence

MISSION

The Institute shall endeavor to incorporate the following basic missions in the teaching methodology:

Engineering Hardware – Software Symbiosis

Practical exercises in all Engineering and Management disciplines shall be carried out by Hardware equipment as well as the related software enabling deeper understanding of basic concepts and encouraging inquisitive nature.

Life – Long Learning

The Institute strives to match technological advancements and encourage students to keep updating their knowledge for enhancing their skills and inculcating their habit of continuous learning.

Liberalization and Globalization

The Institute endeavors to enhance technical and management skills of students so that they are intellectually capable and competent professionals with Industrial Aptitude to face the challenges of globalization.

Diversification

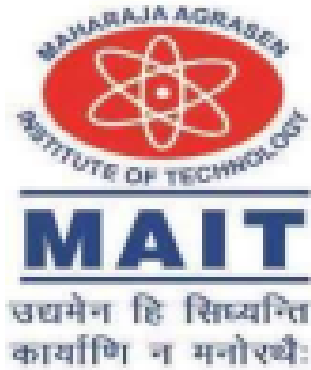
The Engineering, Technology and Management disciplines have diverse fields of studies with different attributes. The aim is to create a synergy of the above attributes by encouraging analytical thinking.

Digitization of Learning Processes

The Institute provides seamless opportunities for innovative learning in all Engineering and Management disciplines through digitization of learning processes using analysis, synthesis, simulation, graphics, tutorials and related tools to create a platform for multi-disciplinary approach.

Entrepreneurship

The Institute strives to develop potential Engineers and Managers by enhancing their skills and research capabilities so that they become successful entrepreneurs and responsible citizens



MAHARAJA AGRASEN INSTITUTE OF TECHNOLOGY

COMPUTER SCIENCE & ENGINEERING DEPARTMENT

VISION

“To be centre of excellence in education, research and technology transfer in the field of computer engineering and promote entrepreneurship and ethical values.”

MISSION

“To foster an open, multidisciplinary and highly collaborative research environment to produce world-class engineers capable of providing innovative solutions to real life problems and fulfill societal needs.”

Department of Computer Science and Engineering

Rubrics for Lab Assessment

Rubrics		0	1	2	3
		Missing	Inadequate	Needs Improvement	Adequate
R1	Is able to identify the problem to be solved and define the objectives of the experiment.	No mention is made of the problem to be solved.	An attempt is made to identify the problem to be solved but it is described in a confusing manner, objectives are not relevant, objectives contain technical/ conceptual errors or objectives are not measurable.	The problem to be solved is described but there are minor omissions or vague details. Objectives are conceptually correct and measurable but may be incomplete in scope or have linguistic errors.	The problem to be solved is clearly stated. Objectives are complete, specific, concise, and measurable. They are written using correct technical terminology and are free from linguistic errors.
R2	Is able to design a reliable experiment that solves the problem.	The experiment does not solve the problem.	The experiment attempts to solve the problem but due to the nature of the design the data will not lead to a reliable solution.	The experiment attempts to solve the problem but due to the nature of the design there is a moderate chance the data will not lead to a reliable solution.	The experiment solves the problem and has a high likelihood of producing data that will lead to a reliable solution.
R3	Is able to communicate the details of an experimental procedure clearly and completely.	Diagrams are missing and/or experimental procedure is missing or extremely vague.	Diagrams are present but unclear and/or experimental procedure is present but important details are missing.	Diagrams and/or experimental procedure are present but with minor omissions or vague details.	Diagrams and/or experimental procedure are clear and complete.
R4	Is able to record and represent data in a meaningful way.	Data are either absent or incomprehensible.	Some important data are absent or incomprehensible.	All important data are present, but recorded in a way that requires some effort to comprehend.	All important data are present, organized and recorded clearly.
R5	Is able to make a judgment about the results of the experiment.	No discussion is presented about the results of the experiment.	A judgment is made about the results, but it is not reasonable or coherent.	An acceptable judgment is made about the result, but the reasoning is flawed or incomplete.	An acceptable judgment is made about the result, with clear reasoning. The effects of assumptions and experimental uncertainties are considered.

MACHINE LEARNING LAB PRACTICAL RECORD

PAPER CODE: **ETCS-454**

NAME OF STUDENT: **Varun Negi**

UNIVERSITY ROLL NO.: **13314802719**

BRANCH: **CSE**

GROUP: **8C7**

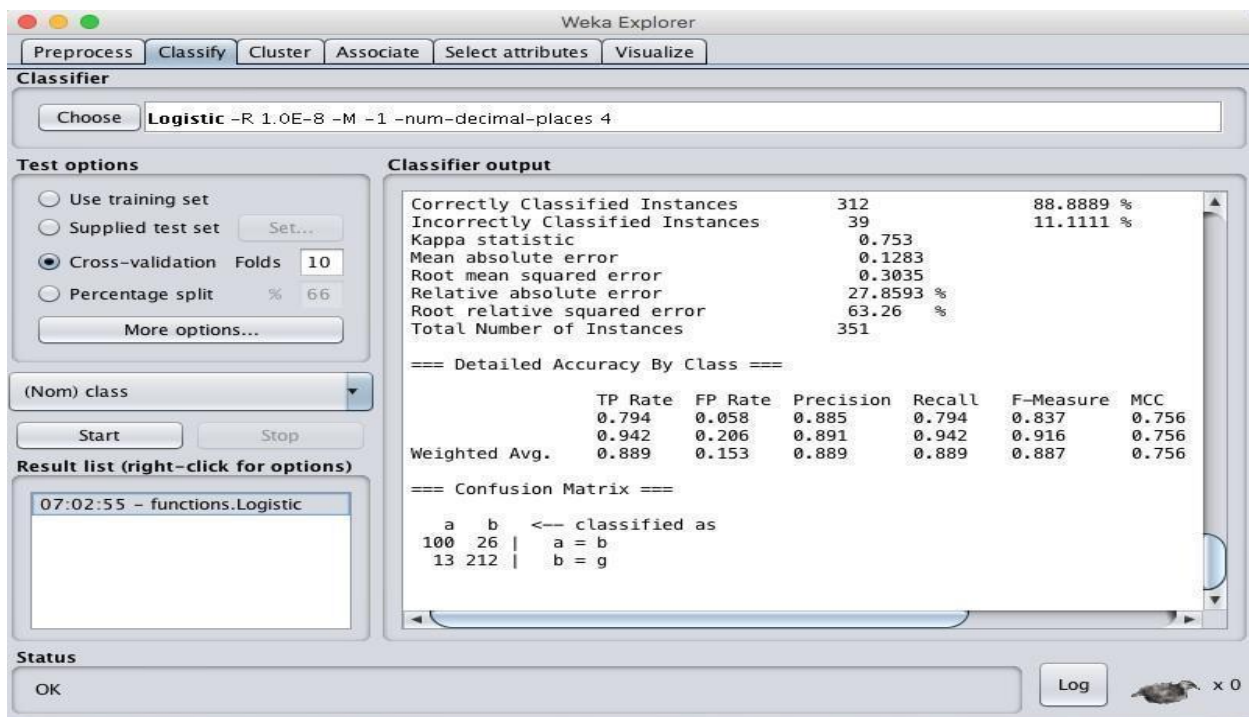
Ex p. No .	Experiment Name	Date	Marks					Total Marks	Signature with Date
			R1	R2	R3	R4	R5		
1.	Introduction to Machine learning lab with tools (hands-on Weka)								
2.	Understanding of Machine learning algorithms List of Databases								
3.	Implement K means Algorithm using WEKA Tool. Implement the K-means algorithm and apply it to the data you selected. Evaluate performance by measuring the sum of Euclidean distance of each example from its class Center. Test the performance of the algorithm as a function of the parameter k.								
4.	Study of Databases and understanding attributes evaluation in regard to problem description								
5.	Working of Major Classifiers, a) Naïve Bayes b) Decision								

	Tree c)CART d) ARIMA (using (e) linear and logistics regression (f) Support vector machine (g) KNN (datasets can be: Breast Cancer data file or Reuters data set).								
6.	Design a prediction model for Analysis of round trip Time of Flight measurement from a supermarket using the random forest, Naïve Bayes, etc.								
7.	Implement supervised learning (KNN classification) Estimate the accuracy of using 5-fold cross-validation. Choose the appropriate options for missing values.								
8.	Introduction to R. Be aware of the basics of machine learning methods in R.								
9.	Build and develop a model in R for a particular classifier (random Forest).								
10.	Develop a machine learning method using Neural Networks in python to Predict stock prices based on past price variation.								
11.	Case Study of RMS Titanic Database to predict survival on the basis of the decision tree, Logistic Regression, KNN or k-Nearest Neighbors, Support Vector Machines								
12.	Understanding of Indian education in Rural villages to predict whether a girl child will be sent to school or not?.								
13.	Understanding of dataset of contact patterns among students collected in the National University of Singapore.								

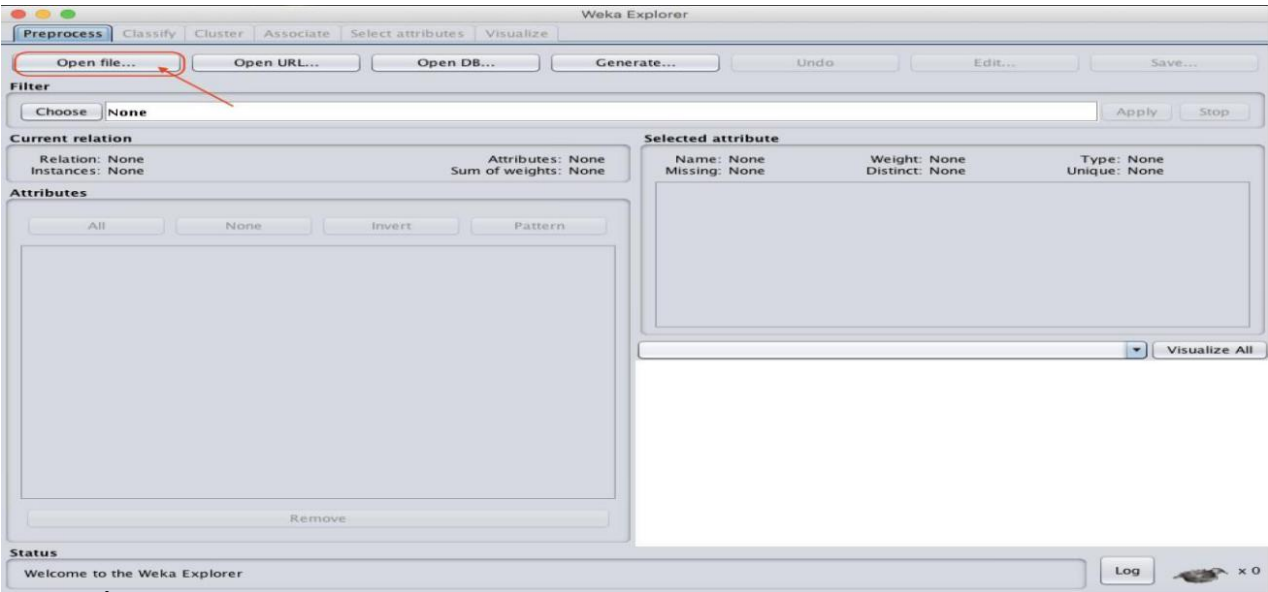
EXPERIMENT-1

Aim: Introduction to machine learning lab with tools (hands on Weka).

Weka is a data mining/machine learning application and is being developed by Waikato University in New Zealand. The purpose of this article is to teach you how to use the Weka Explorer, classify a dataset with Weka, and visualize the results.



1. **KnowledgeFlow** is a Java-Beans based interface for tuning and machine learning experiments.
2. **Smple CLI** is a simple command line interface provided to run Weka functions directly.
3. **Explorer** is an environment to discover the data.
4. **Experimenter** is an environment to make experiments and statistical tests between learning schemes.



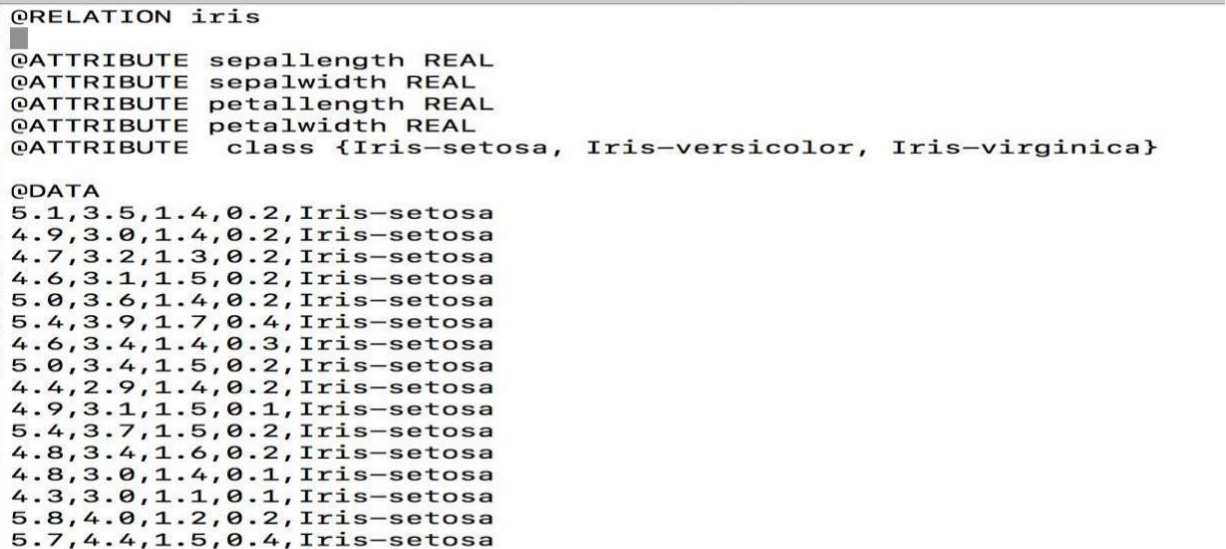
Pre-processing :

Most of the time, the data wouldn't be perfect, and we would need to do pre-processing before applying machine learning algorithms on it. Doing pre-processing is easy in Weka. You can simply click the "Open file" button and load your file as certain file types: **Arff, CSV, C4.5, binary, LIBSVM, XRFF**; you can also load **SQL db** file via the URL and then you can apply filters to it. However, we won't need to do pre-processing for this post since we'll use a dataset that Weka provides for us.

H11							
	A	B	C	D	E	F	G
1	5,10	3,5	1,4	0,2	Iris-setosa		
2	4,9	3	1,4	0,2	Iris-setosa		
3	4,7	3,2	1,3	0,2	Iris-setosa		
4	4,6	3,1	1,5	0,2	Iris-setosa		
5	5,00	3,6	1,4	0,2	Iris-setosa		
6	5,4	3,9	1,7	0,4	Iris-setosa		
7	4,6	3,4	1,4	0,3	Iris-setosa		
8	5	3,4	1,5	0,2	Iris-setosa		
9	4,4	2,9	1,4	0,2	Iris-setosa		
10	4,9	3,1	1,5	0,1	Iris-setosa		
11	5,4	3,7	1,5	0,2	Iris-setosa		
12	4,8	3,4	1,6	0,2	Iris-setosa		
13	4,8	3	1,4	0,1	Iris-setosa		
14	4,3	3	1,1	0,1	Iris-setosa		
15	5,8	4	1,2	0,2	Iris-setosa		
16	5,7	4,4	1,5	0,4	Iris-setosa		
17	5,4	3,9	1,3	0,4	Iris-setosa		
18	5,1	3,5	1,4	0,3	Iris-setosa		
19	5,7	3,8	1,7	0,3	Iris-setosa		
20	5,1	3,8	1,5	0,3	Iris-setosa		
21	5,4	3,4	1,7	0,2	Iris-setosa		
22	5,1	3,7	1,5	0,4	Iris-setosa		
23	4,6	3,6	1	0,2	Iris-setosa		
24	5,1	3,3	1,7	0,5	Iris-setosa		
25	4,8	3,4	1,9	0,2	Iris-setosa		
26	5	3	1,6	0,2	Iris-setosa		
27	5	3,4	1,6	0,4	Iris-setosa		
28	5,2	3,5	1,5	0,2	Iris-setosa		
29	5,2	3,4	1,4	0,2	Iris-setosa		
30	4,7	3,2	1,6	0,2	Iris-setosa		
31	4,8	3,1	1,6	0,2	Iris-setosa		
32	5,4	3,4	1,5	0,4	Iris-setosa		
33	5,2	4,1	1,5	0,1	Iris-setosa		
34	5,5	4,2	1,4	0,2	Iris-setosa		
35	4,9	3,1	1,5	0,1	Iris-setosa		
36	5	3,2	1,2	0,2	Iris-setosa		

If your data type is in xls format like in previous image, you have to convert the file. I'll use the **Iris** dataset to illustrate the conversion:

1. Convert your **.xls** to **.csv** format
2. Open your CSV file in any text editor and first add **@RELATION** database_name to the first row of the CSV file
3. Add attributes by using the following definition: **@ATTRIBUTE** attr_name attr_type. If attr_type is numeric you should define it as **REAL**, otherwise you have to add values between curly parentheses. Sample images are below.
4. At last, add a **@DATA** tag just above on your data rows. Then save your file with **.arff** extension. You can see the illustration in next figure.

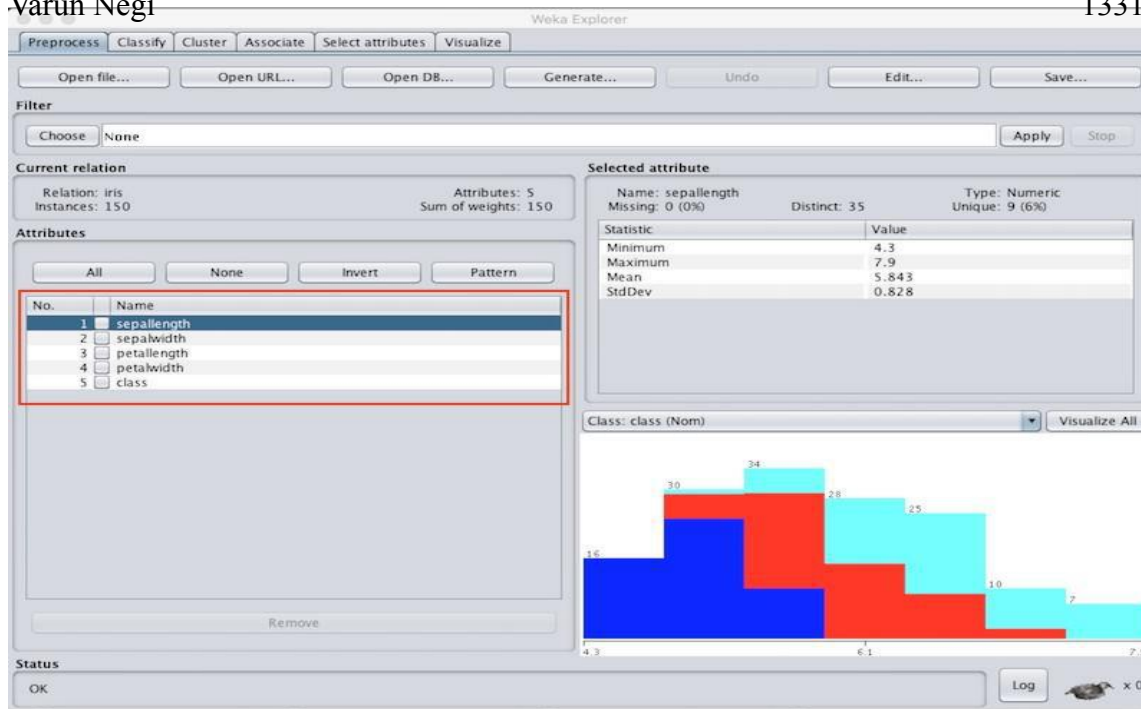


```
@RELATION iris
@ATTRIBUTE sepalength REAL
@ATTRIBUTE sepalwidth REAL
@ATTRIBUTE petallength REAL
@ATTRIBUTE petalwidth REAL
@ATTRIBUTE class {Iris-setosa, Iris-versicolor, Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
5.0,3.6,1.4,0.2,Iris-setosa
5.4,3.9,1.7,0.4,Iris-setosa
4.6,3.4,1.4,0.3,Iris-setosa
5.0,3.4,1.5,0.2,Iris-setosa
4.4,2.9,1.4,0.2,Iris-setosa
4.9,3.1,1.5,0.1,Iris-setosa
5.4,3.7,1.5,0.2,Iris-setosa
4.8,3.4,1.6,0.2,Iris-setosa
4.8,3.0,1.4,0.1,Iris-setosa
4.3,3.0,1.1,0.1,Iris-setosa
5.8,4.0,1.2,0.2,Iris-setosa
5.7,4.4,1.5,0.4,Iris-setosa
```

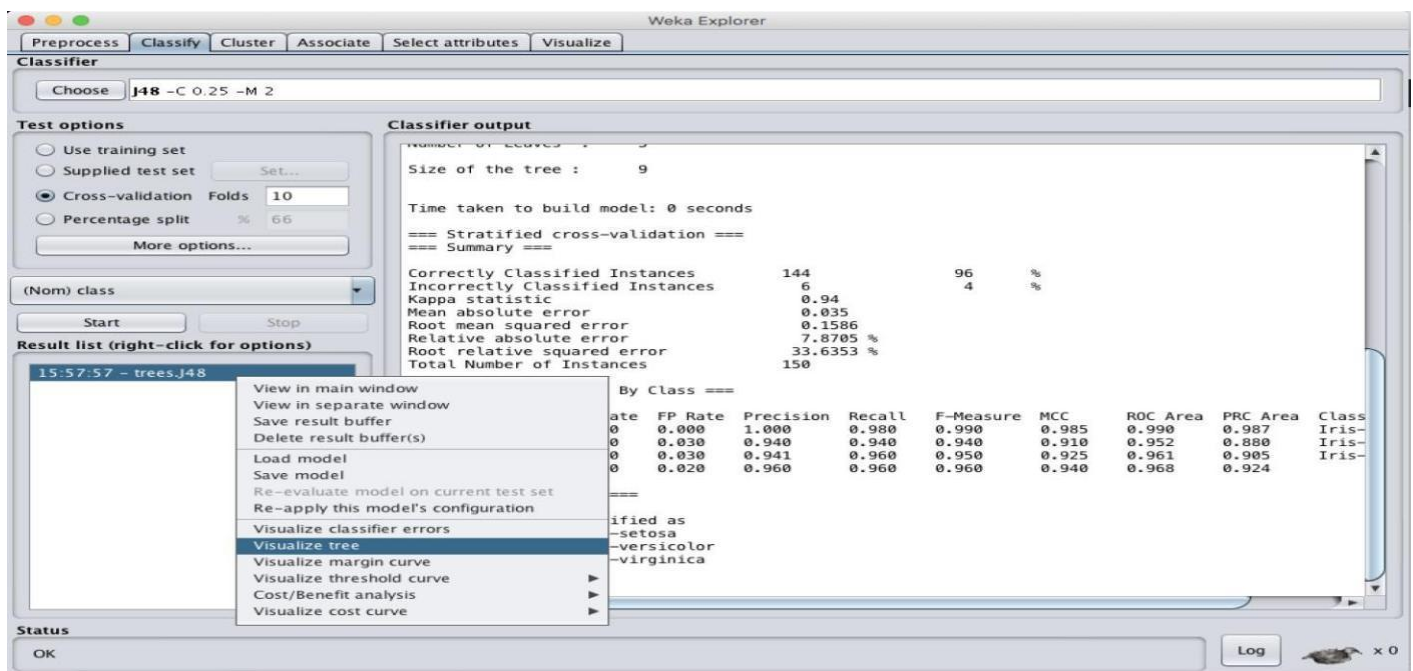
Load Your Data

Click the “Open file” button from the Pre-process section and load your **.arff** file from your local file system. If you couldn’t convert your **.csv** to **.arff**, don’t worry, because **Weka** will do that instead of you.



If you could follow all the steps so far, you can load your data set successfully and you'll see attribute names (it is illustrated at the red area on above images). The preprocess stage is named as Filter in Weka, you can click the 'Choose' button from Filter and apply any filter you want. For example, if you would like to use *Association Rule Mining* as a training model, you have to dissociate numeric and continuous attributes. To be able to do that you can follow the path: **Choose -> Filter -> Supervised -> Attribute -> Discretize**.

Visualizing the Result : If you'd like to visualize this results you can use graphic presentations as you can see in below Figure.



VIVA VOICE QUESTIONS

Q. What is a logistic function? What is the range of values of a logistic function?

Ans. Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability.

Q. Why is logistic regression important ?

Ans. Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign. Logistic regression transforms its output using the logistic sigmoid function to return a probability value.

Q. What is the formula for the logistic regression function?

Ans. $f(z) = 1/(1+e^{-(\alpha+1X_1+2X_2+\dots+kX_k)})$

Q. Why can't linear regression be used in place of logistic regression for binary classification?

Ans. Predicted value is continuous, not probabilistic, sensitive to imbalance the data.

Q: What are the assumptions of logistic regression?

Ans: The assumptions of logistic regression include:

The outcome variable should be binary or categorical.

The relationship between predictors and the log odds of the outcome should be linear.

There should be little or no multicollinearity among the predictor variables.

The observations should be independent of each other.

EXPERIMENT-2

Aim: Understanding of Machine learning algorithm

Machine learning algorithms are programs that can learn from data and improve from experience, without human intervention. Learning tasks may include learning the function that maps the input to the output, learning the hidden structure in unlabeled data; or ‘instance-based learning’, where a class label is produced for a new instance by comparing the new instance (row) to instances from the training data, which were stored in memory.

Types of Machine Learning Algorithms

There are 3 types of machine learning (ML) algorithms:

1. Supervised Learning Algorithms:

Supervised learning uses labeled training data to learn the mapping function that turns input variables (X) into the output variable (Y). In other words, it solves for f in the following equation: $Y = f(X)$

This allows us to accurately generate outputs when given new inputs.

We'll talk about two types of supervised learning: classification and regression.

Classification is used to predict the outcome of a given sample when the output variable is in the form of categories. A classification model might look at the input data and try to predict labels like “sick” or “healthy.”

Regression is used to predict the outcome of a given sample when the output variable is in the form of real values. For example, a regression model might process input data to predict the amount of rainfall, the height of a person, etc.

The first 5 algorithms that we cover in this blog – Linear Regression, Logistic Regression, CART, Naïve-Bayes, and K-Nearest Neighbors (KNN) — are examples of supervised learning.

Ensembling is another type of supervised learning. It means combining the predictions of multiple machine learning models that are individually weak to produce a more accurate prediction on a new sample. Algorithms 9 and 10 of this article — Bagging with Random Forests, Boosting with XGBoost — are examples of ensemble techniques.

2. Unsupervised Learning Algorithms:

Unsupervised learning models are used when we only have the input variables (X) and no corresponding output variables. They use unlabeled training data to model the underlying structure of the data.

We'll talk about three types of unsupervised learning:

Association is used to discover the probability of the co-occurrence of items in a collection. It is extensively used in market-basket analysis. For example, an association model might be used to discover that if a customer purchases bread, s/he is 80% likely to also purchase eggs.

Clustering is used to group samples such that objects within the same cluster are more similar to each other than to the objects from another cluster.

Dimensionality Reduction is used to reduce the number of variables of a data set while ensuring that important information is still conveyed. Dimensionality Reduction can be done using Feature Extraction methods and Feature Selection methods. Feature Selection selects a subset of the original variables. Feature Extraction performs data transformation from a high-dimensional space to a low-dimensional space. Example: PCA algorithm is a Feature Extraction approach.

Algorithms 6-8 that we cover here — Apriori, K-means, PCA — are examples of unsupervised learning.

3. Reinforcement learning:

Reinforcement learning is a type of machine learning algorithm that allows an agent to decide the best next action based on its current state by learning behaviors that will maximize a reward.

Reinforcement algorithms usually learn optimal actions through trial and error. Imagine, for example, a video game in which the player needs to move to certain places at certain times to earn points. A reinforcement algorithm playing that game would start by moving randomly but, over time through trial and error, it would learn where and when it needed to move the in-game character to maximize its point total.

VIVA QUESTIONS

Q. Why is accuracy not a good measure for classification problems?

Ans. Accuracy can be a useful measure if we have the same amount of samples per class but if we have an imbalanced set of samples accuracy isn't useful at all. Even more so, a test can have a high accuracy but actually perform worse than a test with a lower accuracy.

Q. What are false positives and false negatives?

Ans. A **false positive** is an error in [data reporting](#) in which a test result improperly indicates presence of a condition, such as a disease (the result is *positive*), when in reality it is not present, while a **false negative** is an error in which a test result improperly indicates no presence of a condition (the result is *negative*), when in reality it is present.

Q. What are the true positive rate (TPR), true negative rate (TNR), false positive rate (FPR), and false negative rate (FNR)?

Ans. TPR refers to the ratio of positives correctly predicted from all the true labels. In simple words, it is the frequency of correctly predicted true labels. $TPR = TP / (TP + FN)$ TNR refers to the ratio of negatives correctly predicted from all the false labels. It is the frequency of correctly predicted false labels. $TNR = TN / (TN + FP)$ FPR refers to the ratio of positives incorrectly predicted from all the true labels. It is the frequency of incorrectly predicted false labels. $FPR = FP / (TN + FP)$ FNR refers to the ratio of negatives incorrectly predicted from all the false labels. It is the frequency of incorrectly predicted true labels. $FNR = FN / (TP + FN)$

Q. What are precision and recall?

Ans. Precision means the percentage of your results which are relevant. On the other hand, recall refers to the percentage of total relevant results correctly classified by your algorithm.

Q: What is the difference between supervised and unsupervised learning?

Ans: Supervised learning is a type of machine learning where the algorithm learns from labeled data, where the input data is accompanied by the corresponding output labels. The algorithm learns to predict the output labels for new, unseen data based on the patterns observed in the labeled training data. In contrast, unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data. The algorithm explores the patterns and structures within the data without any specific output labels.

Q: What is the bias-variance trade-off in machine learning?

Ans: The bias-variance trade-off refers to the trade-off between the bias and variance of a machine learning model. Bias represents the error introduced by approximating a real-world problem with a simplified model. High bias can cause underfitting, where the model fails to capture the underlying patterns in the data. Variance, on the other hand, represents the model's sensitivity to the fluctuations in the training data. High variance can cause overfitting, where the model performs well on the training data but fails to generalize to new, unseen data. The goal is to strike a balance between bias and variance to achieve a model that performs well on both the training and test data.

EXPERIMENT – 3

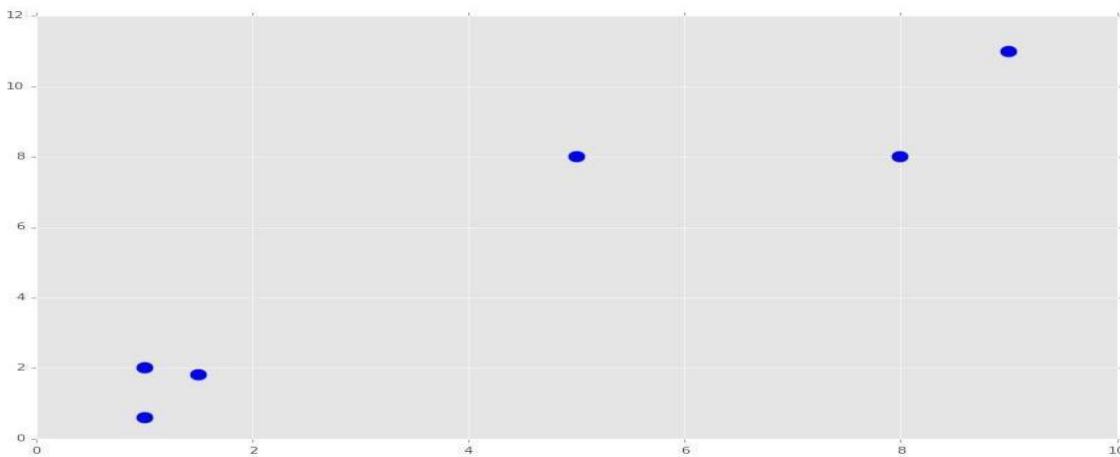
Aim: Implement K mean Algorithm using Python. Evaluate performance by measuring the sum of Euclidean distance of each example from its class centre. Test the performance of the algorithm as a function of the parameter k.

Python Code :

First we plot the data:

```
import matplotlib.pyplot as plt
import numpy as np
from matplotlib import style
style.use('ggplot')
X = np.array([[1, 2],
              [1.5, 1.8],
              [5, 8],
              [8, 8],
              [1, 0.6],
              [9, 11]])

plt.scatter(X[:,0], X[:,1], s=150)
plt.show()
```



KMeans.py :

```
class K_Means: def __init__(self, k=2,
tol=0.001, max_iter=300):
self.k = k
self.tol = tol
self.max_iter = max_iter

def fit(self, data):
self.centroids = {}
```



```

for i in range(self.k):
    self.centroids[i] = data[i]

for i in range(self.max_iter):
    self.classifications = {}

    for i in range(self.k):
        self.classifications[i] = []

    for featureset in data:
        distances = [np.linalg.norm(featureset-self.centroids[centroid]) for centroid in
self.centroids] classification = distances.index(min(distances))
        self.classifications[classification].append(featureset)

    prev_centroids = dict(self.centroids)

    for classification in self.classifications: self.centroids[classification]
    = np.average(self.classifications[classification],axis=0)

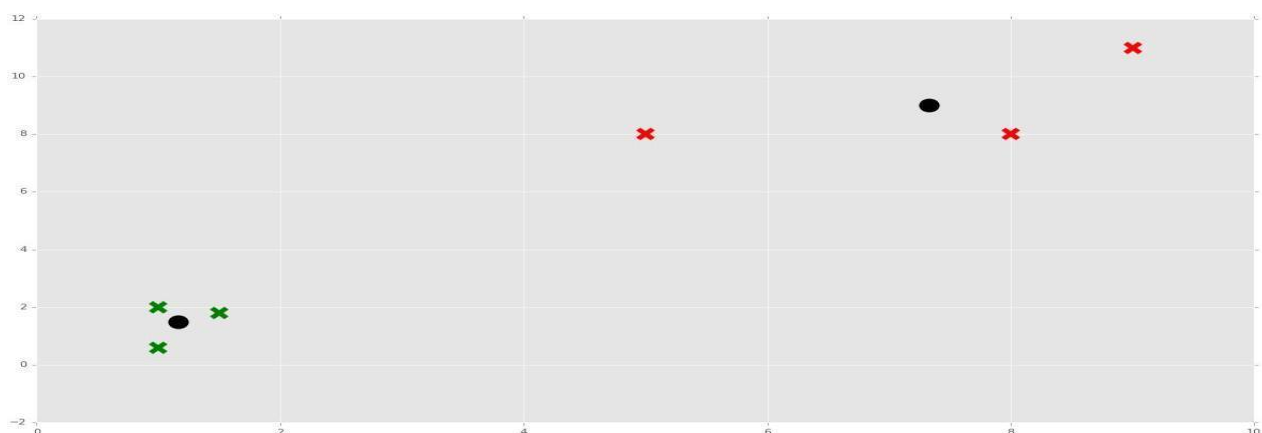
    optimized = True

    for c in self.centroids:
        original_centroid = prev_centroids[c] current_centroid = self.centroids[c] if
        np.sum((current_centroid-original_centroid)/original_centroid*100.0) >
        self.tol:
            print(np.sum((current_centroid-original_centroid)/original_centroid*100.0))
            optimized = False

    if optimized:
        break

def predict(self,data):
    distances = [np.linalg.norm(data-self.centroids[centroid]) for centroid
in self.centroids] classification = distances.index(min(distances)) return
classification

```

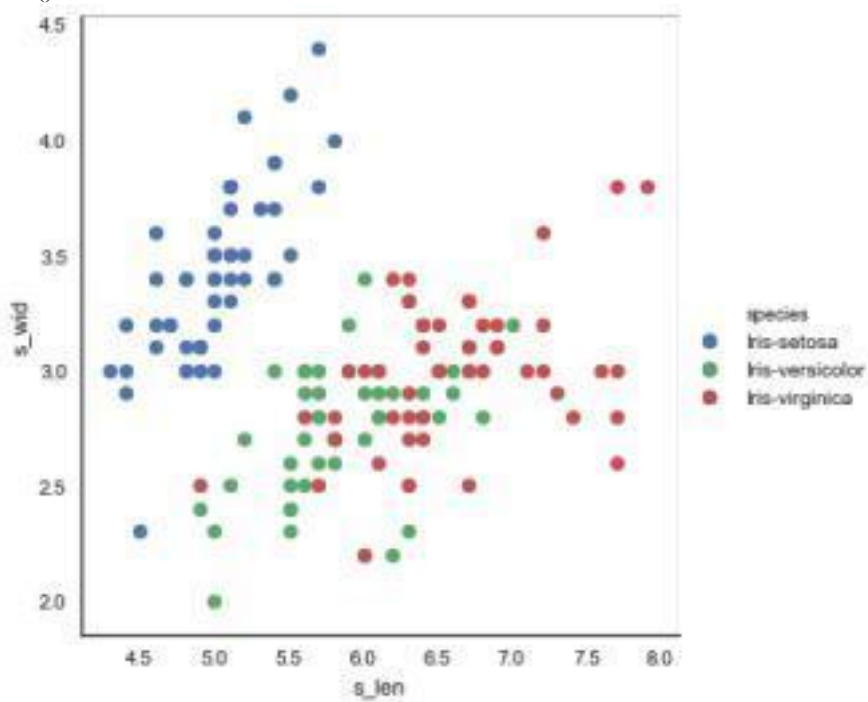


```
model = K_Means()  
model.fit(X)
```

```
for centroid in model.centroids:  
plt.scatter(model.centroids[centroid][0], model.centroids[centroid][1],  
marker="o", color="k", s=150, linewidths=5)
```

```
for classification in model.classifications:  
color = colors[classification] for featureset in  
model.classifications[classification]:  
plt.scatter(featureset[0], featureset[1], marker="x", color=color, s=150, linewidths=5)
```

```
plt.show()
```



VIVA QUESTIONS

Q. Why “Naïve” Bayes is Naïve ?

Ans. A subtle issue with Naive-Bayes is that if you have no occurrences of a class label and a certain attribute value together (e.g. class="nice", shape="sphere") then the frequency-based probability estimate will be zero. Given NaiveBayes' conditional independence assumption, when all the probabilities are multiplied you will get zero and this will affect the posterior probability estimate.

Q. What is SVM ?

Ans. Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate.

Q. What is perceptron ?

Ans. The **perceptron** is an algorithm for supervised learning of binary classifiers. A binary classifier is a function which can decide whether or not an input, represented by a vector of numbers, belongs to some specific class.^[1] It is a type of linear classifier, i.e. a classification algorithm that makes its predictions based on a linear predictor function combining a set of weights with the feature vector.

Q: How does the K-means algorithm initialize centroids?

Ans: The K-means algorithm initializes centroids by randomly selecting K data points from the dataset as the initial centroids. Alternatively, other initialization methods like k-means++ can be used to improve the efficiency and quality of the final clustering.

Q: How does the K-means algorithm assign data points to clusters?

Ans: The K-means algorithm assigns data points to clusters based on their proximity to the centroids. It computes the Euclidean distance (or other distance measures) between each data point and each centroid, and assigns the data point to the cluster associated with the nearest centroid.

Q: What is the objective function in K-means algorithm?

Ans: The objective function in the K-means algorithm is to minimize the within-cluster sum of squares (WCSS) or the sum of squared distances between each data point and its centroid within its assigned cluster. The algorithm iteratively updates the centroids to minimize this objective function.

EXPERIMENT – 4

Aim : Study of databases and understanding attributes evaluation in regard to problem description.

Many data files have store the data or problem description and also their attributes in different formats such as

- a) Arff data
- b) .xlms data
- c) .csv data

a) Abalone data set :

I am using the Abalone data set for understanding the concepts of databases and it's attributes,

Abstract: Predict the age of abalone from physical measurements



Data Set Characteris tics:	Multivariate	Number of Instances:	4177	Area:	Life	
---	--------------	---------------------------------	------	--------------	------	--

Attribute Characteristics:	Categorical, Integer, Real	Number of Attributes:	8	Date Donated	1995-12-01
Associated Tasks:	Classification	Missing Values?	No	Number of Web Hits:	934219

Data Set Information:

Predicting the age of abalone from physical measurements. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope -- a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age. Further information, such as weather patterns and location (hence food availability) may be required to solve the problem.

From the original data examples with missing values were removed (the majority having the predicted value missing), and the ranges of the continuous values have been scaled for use with an ANN (by dividing by 200).

Attribute Information:

Given is the attribute name, attribute type, the measurement unit and a brief description. The number of rings is the value to predict: either as a continuous value or as a classification problem.

Name / Data Type / Measurement Unit / Description

Sex / nominal / -- / M, F, and I (infant)

Length / continuous / mm / Longest shell measurement

Diameter / continuous / mm / perpendicular to length

Height / continuous / mm / with meat in shell

Whole weight / continuous / grams / whole abalone

Shucked weight / continuous / grams / weight of meat

Viscera weight / continuous / grams / gut weight (after bleeding)

Shell weight / continuous / grams / after being dried

Rings / integer / -- / +1.5 gives the age in years

b) Annealing data set:

Abstract: Steel annealing data

Data Set Characteristics:	Multivariate	Number of Instances:	798	Area:	Physical
----------------------------------	--------------	-----------------------------	-----	--------------	----------

Attribute Characteristics:	Categorical, Integer, Real	Number of Attributes:	38	Date Donated	N/A
Associated Tasks:	Classification	Missing Values?	Yes	Number of Web Hits:	16252

Data Set Information:

N/A

Attribute Information:

Attribute Listing:

1. family: --,GB,GK,GS,TN,ZA,ZF,ZH,ZM,ZS
2. product-type: C, H, G
3. steel: -,R,A,U,K,M,S,W,V
4. carbon: continuous
5. hardness: continuous
6. temper_rolling: -,T
7. condition: -,S,A,X
8. formability: -,1,2,3,4,5
9. strength: continuous
10. non-ageing: -,N
11. surface-finish: P,M,- 12. surface-quality: -,D,E,F,G
13. enamelability: -,1,2,3,4,5
14. bc: Y,-
15. bf: Y,- 16. bt: Y,-
17. bw/me: B,M,- 18. bl: Y,-
19. m: Y,-
20. chrom: C,-
21. phos: P,-
22. cbond: Y,-
23. marvi: Y,-
24. exptl: Y,-
25. ferro: Y,- 26. corr: Y,-
27. blue/bright/varn/clean: B,R,V,C,-
28. lustre: Y,- 29. jurofm: Y,-
30. s: Y,- 31. p: Y,-
32. shape: COIL, SHEET
33. thick: continuous

34. width: continuous

35. len: continuous

36. oil: -,Y,N

37. bore: 0000,0500,0600,0760 38. packing: -,1,2,3 classes: 1,2,3,4,5,U

-- The '-' values are actually 'not_applicable' values rather than 'missing_values' (and so can be treated as legal discrete values rather than as showing the absence of a discrete value).

VIVA QUESTIONS

Q1: What is a database?

A1: A database is a structured collection of data that is organized and stored in a way that allows for efficient storage, retrieval, and management of data. It provides a systematic way to store and manage large amounts of structured information.

Q2: What is the purpose of a primary key in a database table?

A2: A primary key is a unique identifier for each record (row) in a database table. It ensures that each record has a unique identity and allows for efficient retrieval and referencing of specific records. The primary key constraint enforces the uniqueness and integrity of the data.

Q3: In machine learning, what are attributes?

A3: In machine learning, attributes refer to the features or characteristics of a data instance. They represent the different properties or variables associated with each data point. Attributes can be numerical (e.g., age, income) or categorical (e.g., gender, color), and they are used as inputs to machine learning algorithms to make predictions or classifications.

Q4: What is the role of data normalization in machine learning?

A4: Data normalization is a preprocessing step in machine learning that scales the attribute values to a common range. It ensures that all attributes contribute equally to the learning process by preventing attributes with larger value ranges from dominating the algorithm. Normalization helps improve the performance and convergence of machine learning models.

Q5: What is the difference between structured and unstructured data?

A5: Structured data refers to data that is organized and follows a predefined schema or data model, such as data stored in databases with well-defined tables and relationships. Unstructured data, on the other hand, does not have a predefined structure and lacks a fixed schema. Examples of unstructured data include text documents, images, audio files, and social media posts. Machine learning techniques often require structured data or need preprocessing steps to handle unstructured data.

EXPERIMENT – 5

Aim : Working of Major Classifiers :

- a) Naïve Bayes
- b) Decision Tree
- c) CART
- d) ARIMA
- e) Linear and Logistic regression

a) Naïve Bayes Algorithm :

I am using the Python library scikit-learn to build the Naive Bayes algorithm.

```
>>> from sklearn.naive_bayes import GaussianNB
>>> from sklearn.naive_bayes import MultinomialNB
>>> from sklearn import datasets
>>> from sklearn.metrics import confusion_matrix

>>> iris = datasets.load_iris()

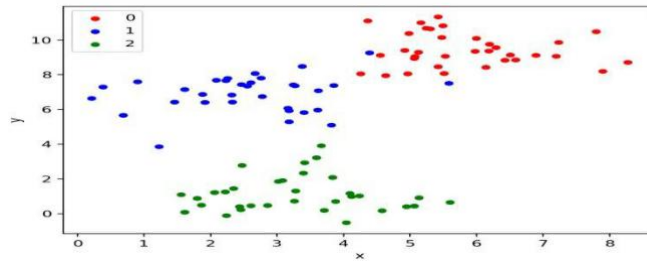
>>> gnb = GaussianNB()
>>> mnb = MultinomialNB()

>>> y_pred_gnb = gnb.fit(iris.data, iris.target).predict(iris.data)
>>> cnf_matrix_gnb = confusion_matrix(iris.target, y_pred_gnb)

>>> print(cnf_matrix_gnb)
[[50 0 0]
 [0 47 3]
 [0 3 47]]

>>> y_pred_mnb = mnb.fit(iris.data, iris.target).predict(iris.data)
>>> cnf_matrix_mnb = confusion_matrix(iris.target, y_pred_mnb)

>>> print(cnf_matrix_mnb)
[[50 0 0]
 [0 46 4]
 [0 3 47]]
```



Output :

b) Decision Tree

```
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

# Function importing
def importdata():
    balance_data = pd.read_csv(
https://archive.ics.uci.edu/ml/machine-learning-
    + 'databases/balance-scale/balance-scale.data',
    sep=',', header=None)

    # Printing the dataset shape
    print("Dataset Length: ", len(balance_data))
    print("Dataset Shape: ", balance_data.shape)

    # Printing the dataset observations
    print("Dataset: ", balance_data.head())
    return balance_data

# Function to split the dataset
def splitdataset(balance_data):

    # Separating the target variable
    X = balance_data.values[:, 1:5]
    Y = balance_data.values[:, 0]

    # Splitting the dataset into train and test
    X_train, X_test, y_train, y_test = train_test_split(
        X, Y, test_size=0.3, random_state=100)

    return X, Y, X_train, X_test, y_train, y_test

# Function to perform training with giniIndex
def train_using_gini(X_train, X_test, y_train):

    # Creating the classifier object
    clf_gini = DecisionTreeClassifier(criterion="gini",
        random_state=100, max_depth=3, min_samples_leaf=5)
```

```

# Performing training
clf_gini.fit(X_train, y_train)
return clf_gini

# Function to perform training with
entropy. def tarin_using_entropy(X_train,
X_test, y_train):

    # Decision tree with entropy
    DecisionTreeClassifier(
"entropy", random_state = 100,
max_depth = 3, min_samples_leaf = 5)

    # Performing training
clf_entropy.fit(X_train, y_train)    return
clf_entropy

# Function to make
predictions def prediction(X_test,
clf_object):

    # Predicton on test with
giniIndex y_pred =
clf_object.predict(X_test)
print("Predicted values:")    print(y_pred)
    return y_pred

# Function to calculate
accuracy def cal_accuracy(y_test,
y_pred):

    print("Confusion Matrix: ",
confusion_matrix(y_test, y_pred))

    print ("Accuracy : ",
accuracy_score(y_test,y_pred)*100)

    print("Report : ",
classification_report(y_test, y_pred))

# Driver
code def main():

    # Building
Phase    data    =
importdata()
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

    # Operational Phase
print("Results Using Gini Index:")

    # Prediction using gini    y_pred_gini =
prediction(X_test, clf_gini)
cal_accuracy(y_test, y_pred_gini)

```

```

print("Results Using Entropy:")
# Prediction using entropy
y_pred_entropy = prediction(X_test, clf_entropy)
cal_accuracy(y_test, y_pred_entropy)

# Calling main
function if
__name__=="__main__":
main()

```

Predicted values:

```

['R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L'
 'L' 'R' 'L' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L'
 'L' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L'
 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'L' 'R' 'L' 'L' 'L' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'R' 'R' 'L' 'R' 'L'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'L' 'L' 'L' 'R' 'L' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L'
 'L' 'L' 'L' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'L' 'R'
 'L' 'R' 'R' 'L' 'L' 'R' 'L' 'R' 'R' 'R' 'R' 'R' 'L' 'R' 'R' 'R' 'R' 'R'
 'R' 'L' 'R' 'L' 'R' 'R' 'L' 'R' 'L' 'R' 'L' 'R' 'L' 'L' 'L' 'L' 'L' 'R'
 'R' 'R' 'L' 'L' 'L' 'R' 'R' 'R']

```

Confusion Matrix: [[0 6 7]

[06322]

[0 20 70]]

Accuracy : 70.7446808511

Report :

	precision	recall	f1-score	support
B	0.00	0.00	0.00	13
L	0.71	0.74	0.72	85
0.71	0.78	0.74	90	avg / total
0.66	0.71	0.68	188	

Output :

c) CART

```

# CART on the Bank Note
dataset from random

```

```
random import randrange from
csv import reader
```

```
# Load a CSV file
def load_csv(filename):
    file = open(filename, "rt")
    lines = reader(file)
    dataset = list(lines)
    return dataset
```

```
# Convert string column to float
def str_column_to_float(dataset, column):
    for row in dataset:
        row[column] = float(row[column].strip())
```

```
# Split a dataset into k folds
def cross_validation_split(dataset, n_folds):
    dataset_split = list()
    dataset_copy = list(dataset)
    fold_size = int(len(dataset) / n_folds)
    for i in range(n_folds):
        fold = list()
        while len(fold) < fold_size:
            index = randrange(len(dataset_copy))
            fold.append(dataset_copy.pop(index))
        dataset_split.append(fold)
    return dataset_split
```

```
# Calculate accuracy
def percentage_def accuracy_metric(actual, predicted):
    correct = 0
    for i in range(len(actual)):
        if actual[i] == predicted[i]:
            correct += 1
    return correct / float(len(actual)) * 100.0
```

```
# Evaluate an algorithm using a cross validation
def evaluate_algorithm(dataset, algorithm, n_folds, *args):
    folds = cross_validation_split(dataset, n_folds)
    scores = list()
    for fold in folds:
        train_set = list(folds)
        train_set.remove(fold)
        train_set = sum(train_set, [])
        test_set = list()
        for row in fold:
            row_copy = list(row)
            test_set.append(row_copy)
            row_copy[-1] = None
        predicted = algorithm(train_set, test_set, *args)
        actual = [row[-1] for row in fold]
        accuracy = accuracy_metric(actual, predicted)
        scores.append(accuracy)
    return scores
```

```
# Split a dataset based on an attribute and an attribute value
def test_split(index, value, dataset):
    left, right = list(), list()
    for row in dataset:
        if row[index] < value:
            left.append(row)
        else:
            right.append(row)
    return left, right
```

```
# Calculate the Gini index for a split dataset

def gini_index(groups, classes):

# count all samples at split point

n_instances = float(sum([len(group) for group in groups]))
```

d) ARIMA

```
import pandas as pd
data = pd.read_csv("Electric_Production.csv", index_col=0)
data.head()
data.index = pd.to_datetime(data.index)
data.columns = ['Energy Production']
```

```
import plotly.plotly as ply
import cufflinks as cf
data.iplot(title="Energy Production Jan 1985--Jan 2018")
```

```
from plotly.plotly import plot_mpl
from statsmodels.tsa.seasonal import seasonal_decompose
result = seasonal_decompose(data,
model='multiplicative')
fig = result.plot()
plot_mpl(fig)
```

```
from pyramid.arima import auto_arima
stepwise_model = auto_arima(data, start_p=1,
start_q=1, max_p=3, max_q=3, m=12, start_P=0, seasonal=True, d=1, D=1, trace=True,
error_action='ignore', suppress_warnings=True,
stepwise=True)
print(stepwise_model.aic())
```

Running the example prints a summary of the fit model. This summarizes the coefficient values used as well as the skill of the fit on the on the in-sample observations.

ARIMA Model Results

=====	
1	
2	
3	
5	


```

Dep. Variable:          D.Sales No. Observations:          35
Model:                ARIMA(5, 1, 0)  Log Likelihood          -196.170
Method:                css-mle S.D. of innovations          64.241
Date:                  Mon, 12 Dec 2016          AIC
406.340

Time:                  11:09:13          BIC
417.227

Sample:                02-01-1901 HQIC
410.098

- 12-01-1903

=====
=====

Int.]          coef      std err          z      P>|z|      [95.0% Conf.

const          12.0649      3.652      3.304      0.003      4.908
19.222
ar.L1.D.Sales - -1.1082      0.183     -6.063      0.000     -1.466
0.750
ar.L2.D.Sales  -0.6203      0.282     -2.203      0.036     -1.172
-0.068

```

e) Linear Regression :

```

import numpy as np
import matplotlib.pyplot as plt

```

```

def estimate_coef(x, y):
    #      number of
    observations/points n =
    np.size(x)

    # mean of x and y vector
    m_x, m_y = np.mean(x), np.mean(y)

    #      calculating cross-deviation and deviation
    about x SS_xy = np.sum(y*x) - n*m_y*m_x
    SS_xx = np.sum(x*x) - n*m_x*m_x

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x

```

```

    return(b_0, b_1)

def plot_regression_line(x, y, b):    #
    plotting the actual points as scatter plot
    plt.scatter(x, y, color = "m",
                marker = "o", s = 30)    # predicted
    response vector
    y_pred = b[0] + b[1]*x

    # plotting the
    regression line
    plt.plot(x, y_pred,
             color = "g")

    # putting labels
    plt.xlabel('x')    plt.ylabel('y')

    #      functi
    on to show
    plot
    plt.show()

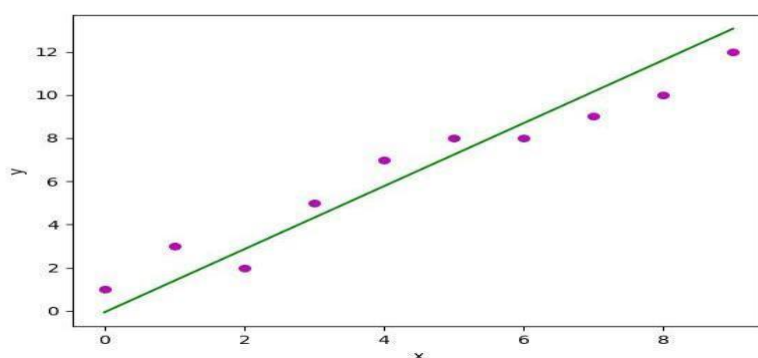
def main():    # observations    x =
    np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
    y = np.array([1, 3, 2, 5, 7, 8, 8, 9, 10, 12])

    # estimating coefficients    b =
    estimate_coef(x, y)    print("Estimated
    coefficients:\nb_0 = {} \
        \nb_1 = {}".format(b[0], b[1]))

    #      plotting
    regression line
    plot_regression_lin
    e(x, y, b)

if __name__ == "__main__":
    main()

```



VIVA QUESTIONS

Q1: What is a decision tree classifier?

A1: A decision tree classifier is a supervised machine learning algorithm that uses a tree-like model to make predictions. It splits the input space into regions based on the feature values and creates a tree structure where each internal node represents a decision based on a feature, and each leaf node represents a class label or outcome. Decision trees are easy to interpret and can handle both categorical and numerical data.

Q2: What is a support vector machine (SVM) classifier?

A2: A support vector machine classifier is a supervised machine learning algorithm used for binary classification and can be extended to handle multi-class problems. SVM constructs a hyperplane or set of hyperplanes in a high-dimensional space that separates the different classes with the maximum margin. It aims to find the optimal decision boundary that maximizes the distance between the support vectors and the hyperplane.

Q3: What is a random forest classifier?

A3: A random forest classifier is an ensemble learning method that combines multiple decision trees to make predictions. It works by creating a set of decision trees on randomly sampled subsets of the training data and aggregating the predictions of individual trees to make the final prediction. Random forests are robust, handle high-dimensional data well, and can capture complex relationships between features.

Q4: What is a logistic regression classifier?

A4: Logistic regression is a supervised machine learning algorithm used for binary classification. Despite its name, logistic regression is a classification algorithm, not a regression algorithm. It models the relationship between the input features and the probability of the binary outcome using a logistic function. Logistic regression is widely used due to its simplicity, interpretability, and effectiveness in a variety of applications.

Q5: What is a k-nearest neighbors (KNN) classifier?

A5: The k-nearest neighbors classifier is a supervised machine learning algorithm used for both classification and regression tasks. It classifies a new data point based on the majority class label of its k nearest neighbors in the feature space. KNN does not build an explicit model but relies on the training instances for classification. It is simple to implement and can handle both numerical and categorical data.

EXPERIMENT 6

AIM - Design a prediction model for Analysis of round-trip Time of Flight measurement from a supermarket using random forest, naïve bayes etc.

CMU-SuperMarket Data Collection and Preprocessing for round-trip time of Flight(RToF) .

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as mse
data = pd.read_csv("cmu_supermarket.csv")
data = data.iloc[:, 0:5]
data.columns = ['X', 'Y', 'Z', 'Mag', 'RToF']
data = data.dropna()
data.head()
```

	X	Y	Z	Mag	RToF
0	5	0	0	3167.0	17.44
1	5	0	0	3218.0	18.49
4	5	0	0	3183.0	15.02
6	5	0	0	3160.0	16.74
7	5	0	0	3161.0	14.94

```
X = data.iloc[:, 0:4]
```

```
Y = data.iloc[:, 4]
```

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.10, random_state=33)
```

Algorithms:

3. Naïve Bayes

```
from sklearn.linear_model import
BayesianRidge
nb_model = BayesianRidge()
nb_model = nb_model.fit(X_train, Y_train)
prediction = nb_model.predict(X_test)
rmse = (mse(y_true = Y_test, y_pred = prediction))**1/2
print("RMSE : ", rmse)
```

```
RMSE : 9.127520132505145
```

```

4. Decision Tree from sklearn.tree import
DecisionTreeRegressor dt_model =
DecisionTreeRegressor() dt_model =
dt_model.fit(X_train , Y_train) prediction =
dt_model.predict(X_test) rmse = (mse(y_true = Y_test ,
y_pred = prediction))**1/2 print("RMSE : " , rmse)

```

```

RMSE : 2.214809487951807

```

```

5. Random Forest from sklearn.ensemble import
RandomForestRegressor rf_model=
RandomForestRegressor() rf_model=
rf_model.fit(X_train , Y_train) prediction =
rf_model.predict(X_test) rmse = (mse(y_true = Y_test ,
y_pred = prediction))**1/2 print("RMSE : " , rmse)

```

```

RMSE : 1.4735218540718842

```

6. SVM

```

from sklearn.svm import SVR svm_model
= SVR()
svm_model = svm_model.fit(X_train , Y_train)
prediction = svm_model.predict(X_test) rmse =
(mse(y_true = Y_test , y_pred =
prediction))**1/2 print("RMSE : " , rmse)

```

```

RMSE : 14.82720182239393

```

7. KNN

```

from sklearn.neighbors import KNeighborsRegressor
knn_model = KNeighborsRegressor(n_neighbors=5)
knn_model = knn_model.fit(X_train , Y_train)
prediction = knn_model.predict(X_test)
rmse = (mse(y_true = Y_test , y_pred =
prediction))**1/2 print("RMSE : " , rmse)

```

```

RMSE : 2.428723668674699

```

Result:

By performing the regression using the above five algorithms, we observe that 'Random Forest' is most accurate with least Root Mean Absolute Error.

Therefore, we conclude that out of the above algorithms, Random Forest performs best.

VIVA QUESTIONS

Q. What is ensemble learning ?

Ans. Ensemble learning is the process by which multiple models, such as classifiers or experts, are strategically generated and combined to solve a particular computational intelligence problem.

Q. Why ensemble learning is used ?

Ans. Ensemble learning helps improve **machine learning** results by combining several models. ... **Ensemble methods** are meta-algorithms that combine several **machine learning** techniques into one predictive model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking).

Q. When to use ensemble learning ?

Ans. Ensemble learning is usually used to average the predictions of different models to get a better prediction. Bagging consists of running multiple different models, each on a different set of input samples and then taking the average of those predictions.

Q. What are the two paradigms of ensemble methods?

The two paradigms of ensemble methods are

- a) Sequential ensemble methods
- b) Parallel ensemble methods

Q: What is a decision tree classifier?

Ans: A decision tree classifier is a supervised machine learning algorithm that uses a tree-like model to make predictions. It splits the input space into regions based on the feature values and creates a tree structure where each internal node represents a decision based on a feature, and each leaf node represents a class label or outcome. Decision trees are easy to interpret and can handle both categorical and numerical data.

Q: What is a support vector machine (SVM) classifier?

Ans: A support vector machine classifier is a supervised machine learning algorithm used for binary classification and can be extended to handle multi-class problems. SVM constructs a hyperplane or set of hyperplanes in a high-dimensional space that separates the different classes with the maximum margin. It aims to find the optimal decision boundary that maximizes the distance between the support vectors and the hyperplane.

EXPERIMENT - 7

Aim: Implement Supervised Learning (KNN Classification Algorithm).

Supervised Learning :

It is the learning where the value or result that we want to predict is within the training data (labeled data) and the value which is in data that we want to study is known as Target or Dependent Variable or *Response Variable*.

All the other columns in the dataset are known as the Feature or Predictor Variable or Independent Variable.

Supervised Learning is classified into two categories:

1. **Clarification:** Here our target variable consists of the categories.
2. **Regression:** Here our target variable is continuous and we usually try to find out the line of the curve.

K-Nearest Neighbor algorithm:

This algorithm is used to solve the classification model problems. K-nearest neighbor or K-NN algorithm basically creates an imaginary boundary to classify the data. When new data points come in, the algorithm will try to predict that to the nearest of the boundary line.

Therefore, larger k value means smother curves of separation resulting in less complex models. Whereas, smaller k value tends to overfit the data and resulting in complex models.

```
# Import necessary modules from
sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import
train_test_split from sklearn.datasets import
load_iris

# Loading data
irisData = load_iris()

# Create feature and target
arrays X = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.2, random_state=42)

knn = KNeighborsClassifier(n_neighbors=7)
```



```

knn.fit(X_train, y_train)

# Predict on dataset which model has not seen before print(knn.predict(X_test))

import numpy as np
import matplotlib.pyplot as plt

irisData = load_iris()

# Create feature and target
arrays X = irisData.data
y = irisData.target

# Split into training and test set
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size = 0.2, random_state=42)

neighbors = np.arange(1, 9) train_accuracy
= np.empty(len(neighbors))

test_accuracy = np.empty(len(neighbors))

# Loop over K values for i, k in
enumerate(neighbors): knn =
KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train)

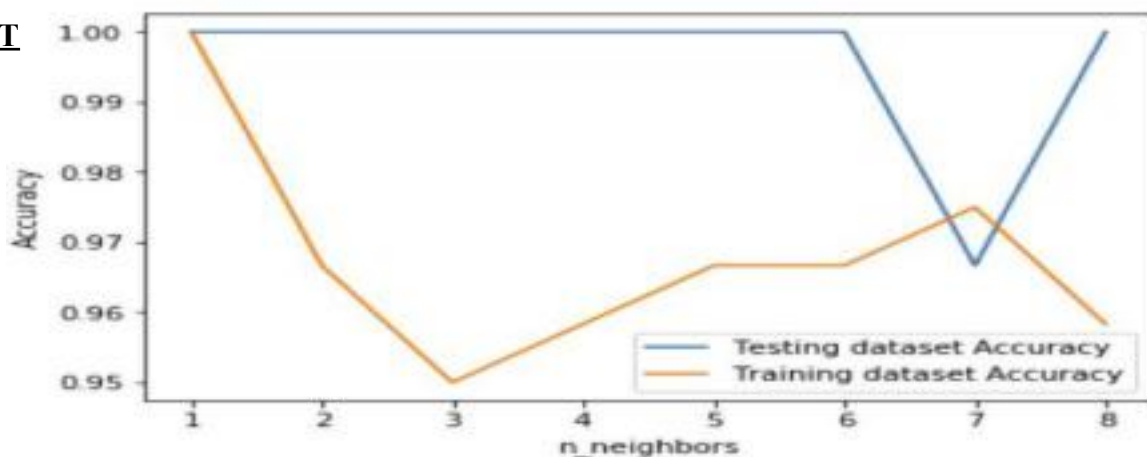
# Compute training and test data accuracy
train_accuracy[i] = knn.score(X_train, y_train)
test_accuracy[i] = knn.score(X_test, y_test)

# Generate plot plt.plot(neighbors, test_accuracy, label = 'Testing dataset
Accuracy') plt.plot(neighbors, train_accuracy, label = 'Training dataset
Accuracy')

plt.legend()
plt.xlabel('n_neighbors')
plt.ylabel('Accuracy') plt.show()

```

OUTPUT



VIVA QUESTIONS

Q1: What is supervised learning?

A1: Supervised learning is a machine learning technique where the algorithm learns from labeled training data. It involves training a model on input-output pairs, where the input data is accompanied by the corresponding output labels. The model learns to predict the output labels for new, unseen data based on the patterns observed in the labeled training data.

Q2: What is the K-Nearest Neighbors (KNN) classification algorithm?

A2: The K-Nearest Neighbors (KNN) classification algorithm is a supervised machine learning algorithm used for classification tasks. It classifies a new data point based on the majority class label of its K nearest neighbors in the feature space. The value of K determines the number of neighbors to consider. KNN is a non-parametric algorithm, meaning it does not make any assumptions about the underlying data distribution.

Q3: How does the KNN algorithm determine the class label of a new data point?

A3: The KNN algorithm determines the class label of a new data point by considering the class labels of its K nearest neighbors. It computes the distance between the new data point and all the training data points using a distance metric (e.g., Euclidean distance). The K nearest neighbors with the shortest distances are identified, and the majority class label among them is assigned to the new data point.

Q4: What are the advantages of the KNN algorithm?

A4: Some advantages of the KNN algorithm include:

- Simplicity and ease of implementation.
- No assumptions about the underlying data distribution.
- Ability to handle both numerical and categorical data.
- Flexibility in choosing the value of K to balance between bias and variance.
- Capability to adapt to new data points and handle incremental learning.

Q5: What are the limitations of the KNN algorithm?

A5: Some limitations of the KNN algorithm include:

- Computational complexity increases as the size of the training data grows.
- Sensitivity to the choice of distance metric and the value of K.
- The need for proper feature scaling to ensure all features contribute equally.
- Difficulty in handling datasets with imbalanced class distributions.
- Inefficiency in high-dimensional feature spaces (curse of dimensionality)

EXPERIMENT - 8

Aim: Understanding R and its Basics

R is a powerful programming language and software environment widely used for statistical computing and data analysis. This document aims to provide an introduction to R and its basics, including data types, variables, operators, and basic data manipulation operations. By the end of this tutorial, you will have a solid foundation in R programming and be ready to explore more advanced concepts.

R Environment:

The R programming environment provides an interactive interface for executing R code. It consists of a console, where commands are entered and executed, and an editor for writing and saving scripts. Familiarize yourself with the R environment and its various components to get started.

Variables and Data Types:

In R, variables are used to store data. R supports various data types, including numeric, character, logical, and factor. Learn how to assign values to variables using the assignment operator, and perform basic arithmetic operations using variables.

Data Input and Output:

Data can be read into R from various sources, such as CSV files or databases. Explore the `read.csv()` function to load data into R. Create a sample dataset manually or load an existing dataset to understand its structure and contents. Practice saving datasets in different formats using suitable R functions.

Data Manipulation:

Data manipulation is a crucial aspect of data analysis. Learn how to subset, filter, and sort data using R's functions. Familiarize yourself with operations on data frames, such as subsetting with logical conditions, filtering rows based on specific criteria, and sorting data by specific columns. Perform basic data transformations, such as adding new columns or modifying existing ones.

Summary Statistics and Visualization:

R provides various functions for calculating summary statistics, such as mean, median, minimum, maximum, and summary. Explore these functions to analyze numeric variables in your dataset. Additionally, R offers built-in plotting functions for creating visualizations. Practice creating simple plots, histograms, and boxplots to gain insights into your data.

Machine Learning Methods in R

R has a vast array of machine learning algorithms and packages for various tasks. Gain familiarity with popular machine learning methods available in R, such as:

1. Linear Regression:
 - Fit a linear model to predict numeric outcomes based on a set of predictor variables.
 - Use the `lm()` function to perform linear regression analysis.
2. Decision Trees:
 - Construct tree-based models for classification and regression tasks using packages like `rpart` or `randomForest`.
 - Use the `rpart()` function to build a decision tree model.
3. Logistic Regression:
 - Perform binary or multinomial classification using the logistic regression algorithm.
 - Use the `glm()` function with the family argument set to "binomial" for logistic regression.
4. k-Nearest Neighbors (KNN):
 - Utilize the KNN algorithm for classification or regression by identifying the nearest neighbors in the feature space.
 - Use the `knn()` function from the `class` package to implement KNN.
5. Support Vector Machines (SVM):
 - Apply SVM algorithms for classification or regression tasks, separating data points with hyperplanes.
 - Use the `svm()` function from the `e1071` package to build SVM models.
6. Random Forests:
 - Build an ensemble of decision trees to perform classification or regression using the `randomForest` package.
 - Use the `randomForest()` function to create random forest models.
7. Naive Bayes:
 - Implement the Naive Bayes algorithm for classification tasks, assuming independence among features.
 - Use the `naiveBayes()` function from the `e1071` package for Naive Bayes classification.

VIVA QUESTIONS

Q. What is Regularization ?

Ans. This is a form of regression, that constrains/ regularizes or shrinks the coefficient estimates towards zero. In other words, *this technique discourages learning a more complex or flexible model, so as to avoid the risk of overfitting*. A simple relation for linear regression looks like this. Here Y represents the learned relation and β represents the coefficient estimates for different variables or predictors(X).

Q. What is Ensemble Learning ?

Ans. Ensemble learning is the use of algorithms and tools in machine learning and other disciplines, to form a collaborative whole where multiple methods are more effective than a single learning method. Ensemble learning can be used in many different types of research, for flexibility and enhanced results.

Q. What is Bagging ?

Ans. A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction.

Q. What is Boosting ?

Ans. Boosting is a general ensemble method that creates a strong classifier from a number of weak classifiers. This is done by building a model from the training data, then creating a second model that attempts to correct the errors from the first model. Models are added until the training set is predicted perfectly or a maximum number of models are added.

Q: What is R and how is it used in machine learning?

Ans: R is a programming language and software environment primarily used for statistical computing, data analysis, and machine learning. It provides a wide range of tools and packages specifically designed for statistical modeling and data manipulation. In machine learning, R allows us to implement various algorithms and techniques for tasks such as classification, regression, and clustering.

EXPERIMENT-9

Aim: Build and develop a model in R for a particular classifier (random Forest).

Random Forests is a popular ensemble learning algorithm used for both classification and regression tasks. It combines multiple decision trees to make predictions, where each tree is trained on a random subset of the training data and features. In this tutorial, I will guide you through building and developing a Random Forest classifier model in R.

Step 1: Install and Load Required Packages

To get started, make sure you have the necessary packages installed. You can install them using the `install.packages()` function. We'll be using the `randomForest` package for building the Random Forest classifier.

```
install.packages("randomForest")  
library(randomForest)
```

Step 2: Prepare the Data

Next, you need to prepare your data for training the Random Forest model. Ensure your dataset is properly formatted and split into training and testing sets.

```
# Load the dataset (replace "dataset.csv" with your dataset file)  
dataset <- read.csv("dataset.csv")  
  
# Split the data into training and testing sets (adjust the split ratio as needed)  
set.seed(123) # Set a seed for reproducibility  
train_indices <- sample(nrow(dataset), nrow(dataset) * 0.7) # 70% for training  
train_data <- dataset[train_indices, ]  
test_data <- dataset[-train_indices, ]
```

Step 3: Train the Random Forest Model

Now, it's time to train the Random Forest classifier using the training data.

```
# Define the target variable column index (replace 1 with the appropriate index)
target_column <- 1

# Train the Random Forest classifier
rf_model <- randomForest(
  dataset[, target_column] ~ ., # Use all other columns as predictors
  data = train_data,
  ntree = 100, # Number of trees in the forest (adjust as needed)
  mtry = sqrt(ncol(train_data)), # Number of variables to consider at each split
  importance = TRUE # Compute variable importance measures
)
```

Step 4: Make Predictions

Once the Random Forest model is trained, you can use it to make predictions on new, unseen data.

```
# Make predictions on the test data
predictions <- predict(rf_model, test_data)
```

Step 5: Evaluate the Model

To assess the performance of the Random Forest model, you can compare the predicted labels with the actual labels in the test data. There are several evaluation metrics you can use, such as accuracy, precision, recall, or the confusion matrix.

```
# Calculate accuracy
accuracy <- sum(predictions == test_data[, target_column]) / nrow(test_data)
```

Step 6: Further Model Development

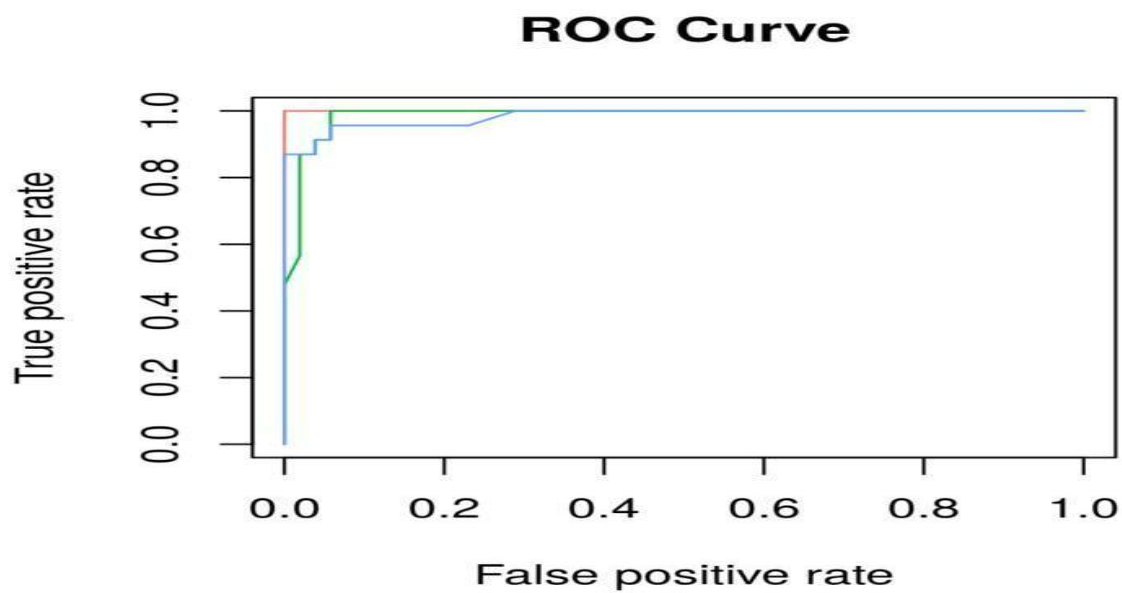
Random Forest models provide additional features for model development, such as variable importance measures and tuning parameters. Here are a few examples:

```
# Get variable importance measures
var_importance <- importance(rf_model)

# Plot variable importance
varImpPlot(rf_model)

# Tune the Random Forest model
tuned_rf_model <- tuneRF(
  train_data[, -target_column], # Use all predictors except the target variable
  train_data[, target_column],
  ntreeTry = c(50, 100, 150), # Test different numbers of trees
  mtryStart = sqrt(ncol(train_data)), # Start with the default value
  stepFactor = 1.5 # Adjust the search step factor
)
```

Output:



VIVA QUESTIONS

Q1: What is the Random Forest classifier?

A1: The Random Forest classifier is an ensemble learning method that combines multiple decision trees to make predictions. It randomly selects subsets of the training data and features for each tree and aggregates the predictions to determine the final outcome. Random Forest is robust, handles high-dimensional data well, and can capture complex relationships between features.

Q2: How does the Random Forest classifier handle overfitting?

A2: Random Forest combats overfitting by using random subsets of the training data and features for each tree. This randomness introduces diversity among the trees, reducing the correlation and overfitting. Additionally, Random Forest performs feature randomization, considering a random subset of features at each split, which further helps in reducing overfitting.

Q3: How does the Random Forest classifier handle missing data?

A3: Random Forest can handle missing data by utilizing the concept of surrogate splits. During the training process, if a value is missing for a particular feature, the algorithm calculates an alternative split using surrogate variables correlated with the missing feature. This ensures that missing data does not significantly impact the classification performance.

Q4: What are the advantages of the Random Forest classifier?

A4: Some advantages of the Random Forest classifier include:

- Ability to handle high-dimensional data and large feature spaces.
- Robustness against overfitting and resistance to noisy data.
- Interpretability through feature importance measures.
- Good performance even without extensive hyperparameter tuning.
- Capability to handle both classification and regression tasks.

Q5: Can the Random Forest classifier handle categorical variables?

A5: Yes, the Random Forest classifier can handle categorical variables. It can work with both categorical and numerical features. For categorical variables, the algorithm uses techniques like one-hot encoding or ordinal encoding to convert them into numerical representations that can be used for building decision trees.

EXPERIMENT - 10

Aim: Develop a machine learning method using Neural Networks in python to Predict stock prices based on past price variation.

Program :

#Import the libraries

```
import math
import pandas_datareader as web
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from keras.models import Sequential
from keras.layers import Dense, LSTM
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
```

#Get the stock quote

```
df = web.DataReader('AAPL', data_source='yahoo',
start='2012-01-01', end='2019-12-17')
```

#Show the data

```
df
```

#Visualize the closing price history

```
plt.figure(figsize=(16,8))
plt.title('Close Price History')
plt.plot(df['Close'])
plt.xlabel('Date', fontsize=18)
plt.ylabel('Close Price USD ($)', fontsize=18)
plt.show()
```

#Create a new dataframe with only the 'Close' column

```
data = df.filter(['Close'])#Converting the dataframe to a numpy
array dataset = data.values#Get /Compute the number of rows to
train the model on training_data_len = math.ceil( len(dataset) *.8)
```

#Scale the all of the data to be values between 0 and 1

```
scaler = MinMaxScaler(feature_range=(0, 1)) scaled_data = scaler.fit_transform(dataset)
```

#Create the scaled training data set

```
train_data = scaled_data[0:training_data_len, :] #Split the data into x_train and y_train data sets x_train=[] y_train=[] for i in range(60,len(train_data)): x_train.append(train_data[i-60:i,0]) y_train.append(train_data[i,0])
```

#Convert x_train and y_train to numpy arrays

```
x_train, y_train = np.array(x_train), np.array(y_train)
```

#Build the LSTM network model

```
model = Sequential() model.add(LSTM(units=50, return_sequences=True, input_shape=(x_train.shape[1], 1))) model.add(LSTM(units=50, return_sequences=False)) model.add(Dense(units=25)) model.add(Dense(units=1))
```

#Compile the model

```
model.compile(optimizer='adam', loss='mean_squared_error')
```

#Train the model

```
model.fit(x_train, y_train, batch_size=1, epochs=1)
```

#Test data set

```
test_data = scaled_data[training_data_len - 60: , :] #Create the x_test and y_test data sets x_test = [] y_test = dataset[training_data_len: , :] #Get all of the rows from index 1603 to the rest and all of the columns (in this case it's only column 'Close'), so 2003 - 1603 = 400 rows of data for i in range(60,len(test_data)): x_test.append(test_data[i-60:i,0])
```



VIVA QUESTIONS

Q. What are the three stages to build the hypotheses or model in machine learning?

Ans.

- a) Model building
- b) Model testing
- c) Applying the model

Q. What is 'Training set' and 'Test set'?

Ans. In various areas of information science like machine learning, a set of data is used to discover the potentially predictive relationship known as 'Training Set'. Training set is an examples given to the learner, while Test set is used to test the accuracy of the hypotheses generated by the learner, and it is the set of example held back from the learner. Training set are distinct from Test set.

Q. What is the difference between artificial learning and machine learning?

Ans. Designing and developing algorithms according to the behaviours based on empirical data are known as Machine Learning. While artificial intelligence in addition to machine learning, it also covers other aspects like knowledge representation, natural language processing, planning, robotics etc.

Q. What is the general principle of an ensemble method and what is bagging and boosting in ensemble method?

The general principle of an ensemble method is to combine the predictions of several models built with a given learning algorithm in order to improve robustness over a single model.

Bagging is a method in ensemble for improving unstable estimation or classification schemes. While boosting method are used sequentially to reduce the bias of the combined model. Boosting and Bagging both can reduce errors by reducing the variance term.

Q. How do you choose an algorithm for a classification problem?

Ans. The answer depends on the degree of accuracy needed and the size of the training set. If you have a small training set, you can use a low variance/high bias classifier. If your training set is large, you will want to choose a high variance/low bias classifier.

EXPERIMENT-11

Aim: Understanding of RMS Titanic Dataset to predict survival by training a model and predict the required solution.

The sinking of the **RMS Titanic** is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships. One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

	Survived	Pclass	Sex	Age	Fare	Embarked	Title	IsAlone	Age*Class
0	0	3	0	1	0	0	1	0	3
1	1	1	1	2	3	1	3	0	2
2	1	3	1	1	1	0	2	1	3
3	1	1	1	2	3	0	3	0	2
4	0	3	0	2	1	0	1	1	6
5	0	3	0	1	1	2	1	1	3
6	0	1	0	3	3	0	1	1	3
7	0	3	0	0	2	0	4	0	0
8	1	3	1	1	1	0	3	0	3
9	1	2	1	0	2	1	3	0	0

Survived (Target Variable) - Binary categorical variable where 0 represents not survived and 1 represents survived.

Pclass - Categorical variable. It is passenger class.

● **Sex** - Binary Variable representing the gender the of passenger

● **Age** - Feature engineered

variable. It is divided into 4 classes.

Fare - Feature engineered variable. It is divided into 4 classes.

Embarked - Categorical Variable. It tells the Port of embarkation.

Title - New feature created from names. The title of names is classified into 4 different classes.

isAlone - Binary Variable. It tells whether the passenger is travelling alone or not.

Age*Class - Feature engineered variabl

Model, predict and solve

Now we are ready to train a model and predict the required solution. There are 60+ predictive modelling algorithms to choose from. We must understand the type of problem and solution requirement to narrow down to a select few models which we can evaluate. Our problem is a classification and regression problem. We want to identify relationship between output (Survived or not) with other variables or features (Gender, Age, Port...). We are also performing a category of machine learning which is called supervised learning as we are training our model with a given dataset. With these two criteria - Supervised Learning plus Classification and Regression, we can narrow down our choice of models to a few. These include:

Logistic Regression

KNN or k-Nearest Neighbours

Support Vector Machines

Naive Bayes classifier

Decision Tree

```
X_train = train_df.drop("Survived", axis=1)
Y_train = train_df["Survived"]
X_test = test_df.drop("PassengerId", axis=1).copy()
X_train.shape, Y_train.shape, X_test.shape
```

```
((891, 8), (891,), (418, 8))
```

Size of the training and testing dataset

Logistic Regression is a useful model to run early in the workflow. Logistic regression measures the relationship between the categorical dependent variable (feature) and one or more independent variables (features) by estimating probabilities using a logistic function, which is the cumulative logistic distribution.

```
# Logistic Regression

logreg = LogisticRegression()
logreg.fit(X_train, Y_train)
Y_pred = logreg.predict(X_test)
acc_log = round(logreg.score(X_train, Y_train) * 100, 2)
acc_log
```

```
80.35999999999999
```

Note the confidence score generated by the model based on our training dataset.

In pattern recognition, the k-Nearest Neighbours algorithm (or k-NN for short) is a nonparametric method used for classification and regression. A sample is classified by a majority vote of its neighbours, with the sample being assigned to the class most common among its k nearest neighbours (k is a positive integer, typically small). If k = 1, then the object is simply assigned to the class of that single nearest neighbour.

```
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, Y_train)
Y_pred = knn.predict(X_test)
acc_knn = round(knn.score(X_train, Y_train) * 100, 2)
acc_knn
```

84.739999999999995

KNN confidence score is better than Logistics Regression but worse than SVM.

Next we model using Support Vector Machines which are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training samples, each marked as belonging to one or the other of **two categories**, an SVM training algorithm builds a model that assigns new test samples to one category or the other, making it a non-probabilistic binary linear classifier.

Note that the model generates a confidence score which is higher than Logistics Regression Model.

```
# Support Vector Machines

svc = SVC()
svc.fit(X_train, Y_train)
Y_pred = svc.predict(X_test)
acc_svc = round(svc.score(X_train, Y_train) * 100, 2)
acc_svc
```

83.840000000000003

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features) in a learning problem.

```
# Gaussian Naive Bayes

gaussian = GaussianNB()
gaussian.fit(X_train, Y_train)
Y_pred = gaussian.predict(X_test)
acc_gaussian = round(gaussian.score(X_train, Y_train) * 100, 2)
acc_gaussian
```

72.280000000000001

The model generated confidence score is the lowest among the models evaluated so far.

This model uses a decision tree as a predictive model which maps features (tree branches) to conclusions about the target value (tree leaves). Tree models where the target variable can take a finite set of values are called classification trees; in these tree structures, leaves represent class labels and branches represent conjunctions of features that lead to those class labels. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees. The model confidence score is the highest among

models evaluated so far.

```
# Decision Tree

decision_tree = DecisionTreeClassifier()
decision_tree.fit(X_train, Y_train)
Y_pred = decision_tree.predict(X_test)
acc_decision_tree = round(decision_tree.score(X_train, Y_train) * 100, 2)
acc_decision_tree
```

86.760000000000005

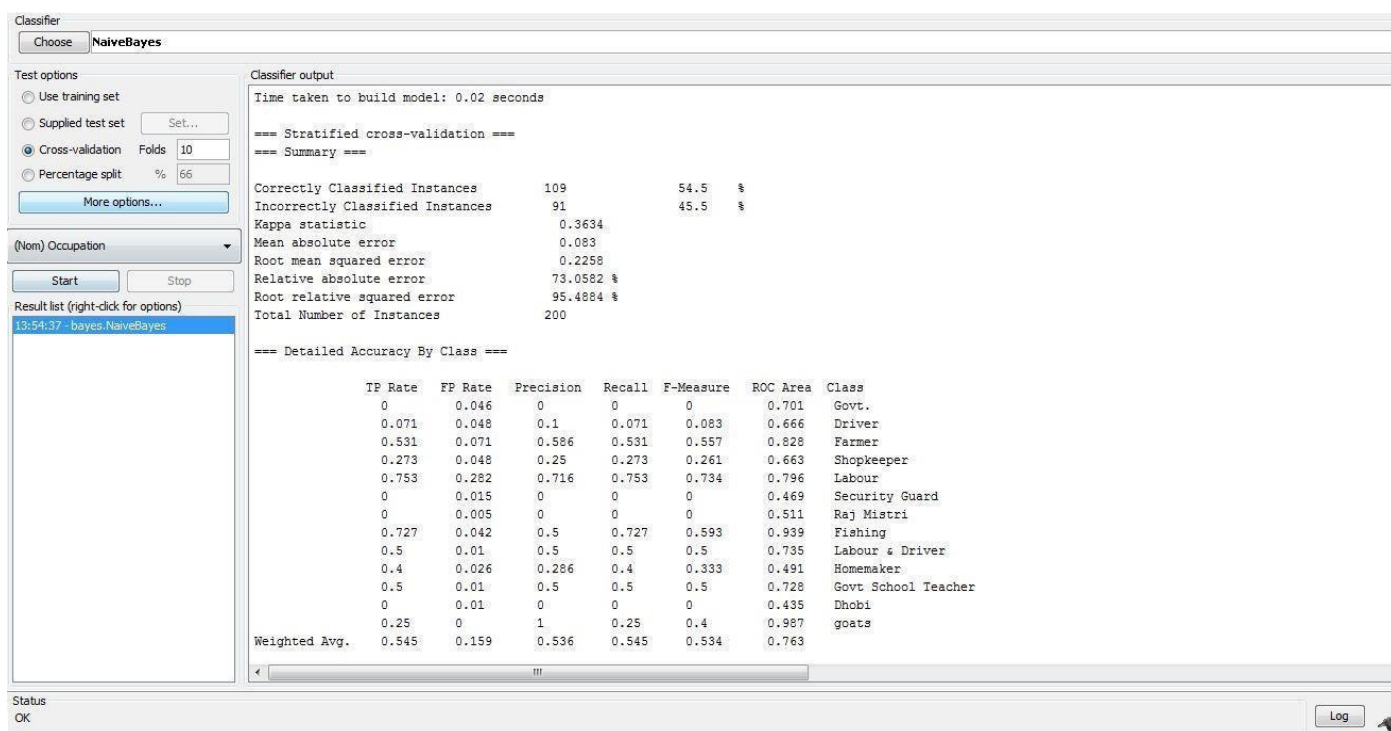
EXPERIMENT-12

Aim: Understanding of Indian education in Rural villages to predict whether girl child will be sent to school or not?

The data is focused on rural India. It primarily looks into the fact whether the villagers are willing to send the girl children to school or not and if they are not sending their daughters to school the reasons have also been mentioned. The district is Gwalior. Various details of the villagers such as village, gender, age, education, occupation, category, caste, religion, land etc have also been collected.

1 Naïve Bayes Classifier

The algorithm was run with 10-fold cross-validation: this means it was given an opportunity to



make a prediction for each instance of the dataset (with different training folds) and the presented result is a summary of those predictions. Firstly, I noted the Classification Accuracy. The model achieved a result of 109/200 correct or 54.5%.

=== Confusion Matrix ===

```
a b c d e f g h i j k l m <-- classified as
0 0 1 1 0 1 0 2 0 0 0 0 0 | a = Govt.
2 1 1 1 8 0 0 0 0 1 0 0 0 | b = Driver
2 0 1 7 2 9 0 0 2 0 0 0 0 | c = Farmer
0 0 4 3 2 0 1 0 0 1 0 0 0 | d = Shopkeeper
```

1 0 0 0 1 1 0 8 0 0 0 0 0 | h = Fishing
0 0 2 0 0 0 0 0 2 0 0 0 0 | i = Labour & Driver
0 0 2 0 1 0 0 0 0 2 0 0 0 | j = Homemaker
0 0 0 0 1 0 0 0 0 2 0 0 0 | k = Govt School Teacher
0 0 0 1 4 0 0 0 0 0 0 0 0 | l = Dhobi
1. 0 0 0 3 0 0 0 0 0 0 0 0 1 | m = goats

Now when we have model, we need to load our test data we've created before. For this, select Supplied test set and click button Set. Click More Options, where in new window, choose PlainText from Output predictions. Then click left mouse button on recently created model on result list and select Re-evaluate model on current test set. After re-evaluation

```

Classifier output
    200    5:Labour    5:Labour    0    0.014    0.267    0.017    *0.529    0    0    0    0.029    0.091    0    0.054    0

=== Summary ===

Correctly Classified Instances    151    75.5    %
Incorrectly Classified Instances    49    24.5    %
Kappa statistic    0.6634
Mean absolute error    0.0554
Root mean squared error    0.1649
Total Number of Instances    200

=== Detailed Accuracy By Class ===

    TP Rate    FP Rate    Precision    Recall    F-Measure    ROC Area    Class
    0.8    0.021    0.5    0.8    0.615    0.994    Govt.
    0.5    0.032    0.538    0.5    0.519    0.915    Driver
    0.625    0.012    0.909    0.625    0.741    0.945    Farmer
    0.636    0.026    0.583    0.636    0.609    0.929    Shopkeeper
    0.825    0.175    0.816    0.825    0.821    0.901    Labour
    0.5    0    1    0.5    0.667    0.999    Security Guard
    1    0.005    0.8    1    0.889    1    Raj Mistri
    1    0.037    0.611    1    0.759    0.999    Fishing
    1    0    1    1    1    1    Labour & Driver
    0.8    0.015    0.571    0.8    0.667    0.944    Homemaker
    1    0.015    0.571    1    0.727    0.997    Govt School Teacher
    0    0    0    0    0    0.981    Dhobi
    1    0    1    1    1    1    goats
Weighted Avg.    0.755    0.094    0.759    0.755    0.746    0.931

```

Now the Classification Accuracy is 151/200 correct or 75.5%.

TP = true positives: number of examples predicted positive that are actually positive

FP = false positives: number of examples predicted positive that are actually negative

TN = true negatives: number of examples predicted negative that are actually negative

FN = false negatives: number of examples predicted negative that are actually positive

Recall is the TP rate (also referred to as sensitivity) what fraction of those that are actually

positive were predicted positive? : TP / actual positives Precision is TP / predicted Positive what fraction of those predicted positive are actually positive? **precision** is also referred to as Positive predictive value (PPV); Other related measures used in classification include True Negative Rate and Accuracy: True Negative Rate is also called **Specificity**. (TN / actual negatives) 1-specificity is x-axis of ROC curve: this is the same as the FP rate (FP / actual negatives)

F-measure A measure that combines precision and recall is the harmonic mean of precision and recall, the traditional F-measure or balanced F-score:

$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$

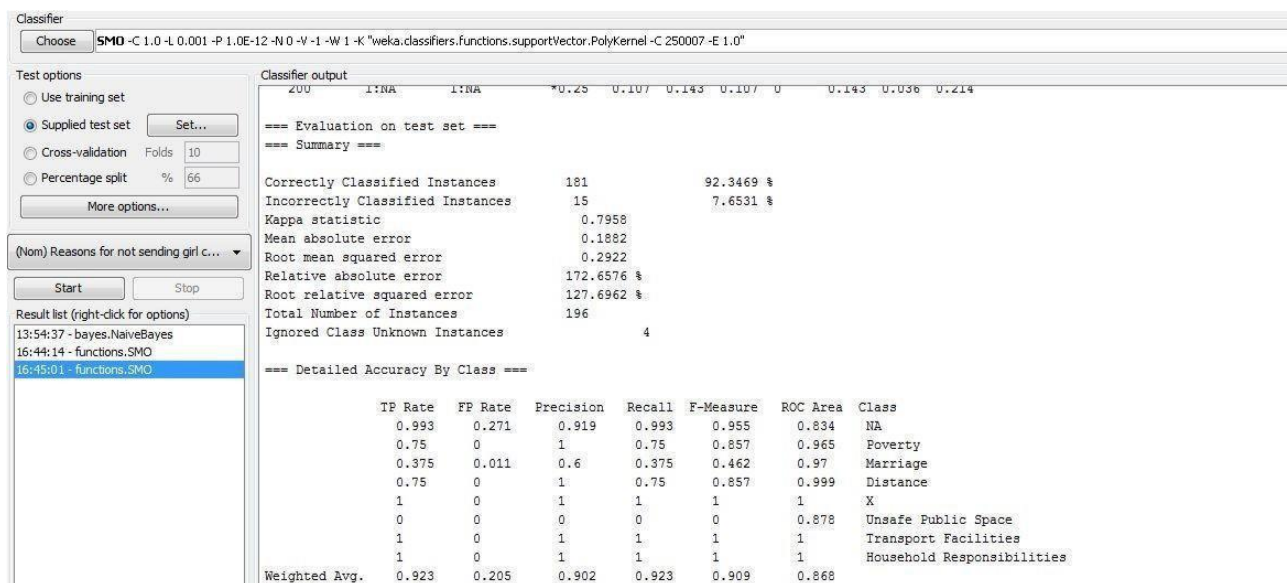
Mean absolute error (MAE)

The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. The equation is given in the library references. Expressed in words, the MAE is the average over the verification sample of the absolute values of the differences between forecast and the corresponding observation. The MAE is a linear score which means that all the individual differences are weighted equally in the average;

Root mean squared error (RMSE)

The RMSE is a quadratic scoring rule which measures the average magnitude of the error. The equation for the RMSE is given in both of the references. Expressing the formula in words, the difference between forecast and corresponding observed values are each squared and then averaged over the sample. Finally, the square root of the average is taken. Since the errors are squared before they are averaged, the RMSE gives a relatively high weight to large errors. This means the RMSE is most useful when large errors are particularly undesirable.

2 Support Vector Machine



The screenshot shows the Weka Classifier window with the SMO classifier selected. The test options are set to 'Supplied test set' with a 'Set...' button. The classifier output window displays the following results:

```

Classifier
Choose SMO -C 1.0 -L 0.001 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E 1.0"

Test options
Use training set
Supplied test set Set...
Cross-validation Folds 10
Percentage split % 66
More options...

(Nom) Reasons for not sending girl c...
Start Stop

Result list (right-click for options)
13:54:37 - bayes.NaiveBayes
16:44:14 - functions.SMO
16:45:01 - functions.SMO

Classifier output
200 1:NA 1:NA *0.25 0.107 0.143 0.107 0 0.143 0.036 0.214

=== Evaluation on test set ===
=== Summary ===

Correctly Classified Instances 181 92.3469 %
Incorrectly Classified Instances 15 7.6531 %
Kappa statistic 0.7958
Mean absolute error 0.1882
Root mean squared error 0.2922
Relative absolute error 172.6576 %
Root relative squared error 127.6962 %
Total Number of Instances 196
Ignored Class Unknown Instances 4

=== Detailed Accuracy By Class ===

TP Rate FP Rate Precision Recall F-Measure ROC Area Class
0.993 0.271 0.919 0.993 0.955 0.834 NA
0.75 0 1 0.75 0.857 0.965 Poverty
0.375 0.011 0.6 0.375 0.462 0.97 Marriage
0.75 0 1 0.75 0.857 0.999 Distance
1 0 1 1 1 1 X
0 0 0 0 0 0.878 Unsafe Public Space
1 0 1 1 1 1 Transport Facilities
1 0 1 1 1 1 Household Responsibilities
Weighted Avg. 0.923 0.205 0.902 0.923 0.909 0.868

```

The model achieved a result of 181/200 correct or 92.3469%.

We have classified the dataset on the basis the reasons why the villagers are unwilling to send girl children to schools in Gwalior village. The different classes are NA, Poverty, Marriage, Distance, X, Unsafe Public Space, Transport Facilities, and Household Responsibilities. The weighted average true positive rate is 0.923 that is nearly all the predicted positive values are actually positive. The weighted average false positive rate is 0.205 that is few of them are predicted as positive values but are actually negative. The precision in 0.902 that is the algorithm is nearly accurate.

=== Confusion Matrix ===

```

a b c d e f g h <-- classified as
147 0 1 0 0 0 0 0 | a = NA
 4 12 0 0 0 0 0 0 | b = Poverty
 5  0 3 0 0 0 0 0 | c = Marriage
 0 0 1 3 0 0 0 0 | d = Distance
 0 0 0 0 8 0 0 0 | e = X
 4 0 0 0 0 0 0 0 | f = Unsafe Public Space
 0 0 0 0 0 0 4 0 | g = Transport Facilities
 1 0 0 0 0 0 0 4 | h = Household Responsibilities

```

The confusion matrix shows that majority of the reasons were not available and out of the reasons which were available people did not send their daughters to school because of poverty and very few of them considered Distance as a major factor for not sending their girl children to school.

3 Random Forest

The screenshot shows the RStudio Classifier window for a Random Forest model. The 'Test options' section on the left shows 'Supplied test set' selected. The 'Classifier output' section on the right displays the following summary:

```

=== Evaluation on test set ===
=== Summary ===
Correctly Classified Instances      200      100 %
Incorrectly Classified Instances    0        0 %
Kappa statistic                    1
Mean absolute error                 0.028
Root mean squared error             0.0725
Relative absolute error             24.6886 %
Root relative squared error         30.6797 %
Total Number of Instances          200

```

Below the summary, a 'Detailed Accuracy By Class' table is shown:

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
1	0	1	1	1	1	1	Govt.
1	0	1	1	1	1	1	Driver
1	0	1	1	1	1	1	Farmer
1	0	1	1	1	1	1	Shopkeeper
1	0	1	1	1	1	1	Labour
1	0	1	1	1	1	1	Security Guard
1	0	1	1	1	1	1	Raj Mistri
1	0	1	1	1	1	1	Fishing
1	0	1	1	1	1	1	Labour & Driver
1	0	1	1	1	1	1	Homemaker
1	0	1	1	1	1	1	Govt School Teacher
1	0	1	1	1	1	1	Dhobi
1	0	1	1	1	1	1	goats
Weighted Avg.	1	0	1	1	1	1	

The accuracy of this algorithm is 100% that is 200/200 have been correctly classified.

==== Confusion Matrix ====

```

a b c d e f g h i j k l m <-- classified as
50000000000000 | a = Govt.
01400000000000 | b = Driver
00320000000000 | c = Farmer
00011000000000 | d = Shopkeeper
00009700000000 | e = labour
000004000000 0 | f = Security Guard
000000400000 0 | g = Raj Mistri
0 0 0 0 0 0 11 0 0 0 0 0 | h = Fishing
0000000040000 | i = Labour & Driver
0000000005000 | j = Homemaker
0000000000400 | k = Govt School Teacher
0000000000050 | l = Dhobi
1. 0 0 0 0 0 0 0 0 0 0 0 4 | m = goats

```

There is no observation which has been misclassified. Maximum number of villagers are laborers.

4 Random Tree

Classifier

Choose RandomTree -K 0 -M 1.0 -S 1

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 10

☐ Percentage split % 66

More options...

(Nom) Reasons for not sending girl c...

Start Stop

Result list (right-click for options)

- 13:54:37 - bayes.NaiveBayes
- 16:44:14 - functions.SMO
- 16:45:01 - functions.SMO
- 17:11:40 - trees.RandomForest
- 17:12:02 - trees.RandomForest
- 17:17:15 - trees.RandomTree
- 17:17:27 - trees.RandomTree
- 17:17:39 - trees.RandomTree

Classifier output

```

20      ?      1:NA      + *1      0      0      0      0      0      0      0

```

==== Stratified cross-validation ====

==== Summary ====

Correctly Classified Instances	149	76.0204 %
Incorrectly Classified Instances	47	23.9796 %
Kappa statistic	0.4265	
Mean absolute error	0.0617	
Root mean squared error	0.2386	
Relative absolute error	56.273 %	
Root relative squared error	104.1535 %	
Total Number of Instances	196	
Ignored Class Unknown Instances	4	

==== Detailed Accuracy By Class ====

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.851	0.458	0.851	0.851	0.851	0.708	NA
	0.5	0.05	0.471	0.5	0.485	0.757	Poverty
	0.375	0.021	0.429	0.375	0.4	0.733	Marriage
	0.75	0.005	0.75	0.75	0.75	0.872	Distance
	0.75	0	1	0.75	0.857	0.874	X
	0	0.042	0	0	0	0.439	Unsafe Public Space
	0	0.016	0	0	0	0.732	Transport Facilities
	0.75	0	1	0.75	0.857	0.874	Household Responsibilities
Weighted Avg.	0.76	0.352	0.775	0.76	0.767	0.722	

The classification accuracy is 76.0204% that is 149/200 have been classified correctly.

The false positive rate is 0.352 that is highest of all the four algorithms applied above. Here 35.2% of the values which should have been classified negatively have been assigned a positive value.

==== Confusion Matrix ====

a	b	c	d	e	f	g	h	<-- classified as
126	7	3	1	0	8	3	0	a = NA
7	8	1	0	0	0	0	0	b = Poverty
4	1	3	0	0	0	0	0	c = Marriage
1	0	0	3	0	0	0	0	d = Distance
2	0	0	0	6	0	0	0	e = X
4	0	0	0	0	0	0	0	f = Unsafe Public Space
3	1	0	0	0	0	0	0	g = Transport Facilities
1	0	0	0	0	0	0	3	h = Household Responsibilities

22 NA , 8 Poverty , 5 Marriage, 1 Distance, 2 X, 4 Unsafe Public Space, 4 Transport Facilities and 1 Household Responsibilities class values have been misclassified.

The best algorithm out of the above algorithms is Random Forest with 100% accuracy rate and the worst is Naïve Bayes algorithm with 75.5% accuracy rate.

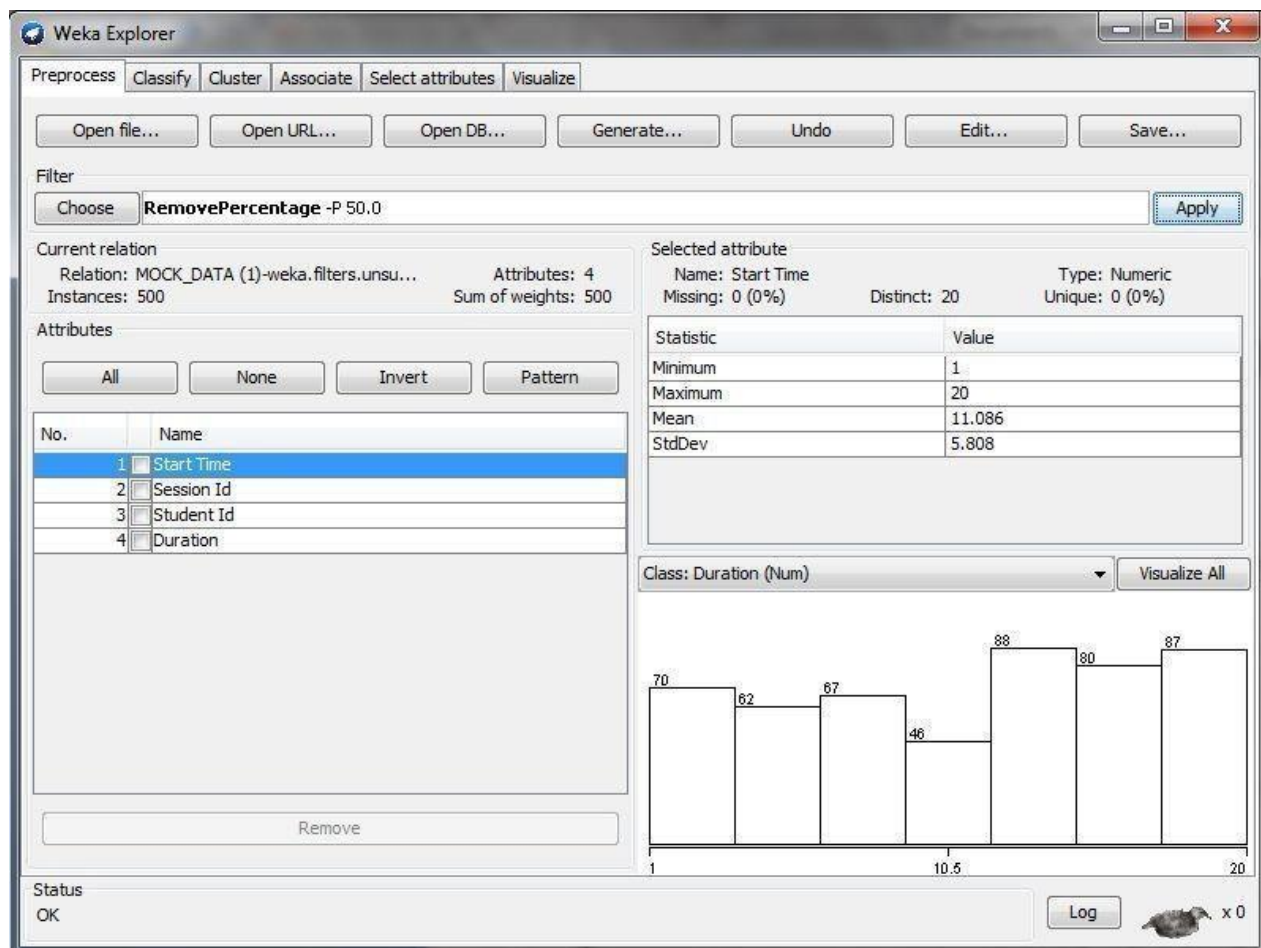
EXPERIMENT-13

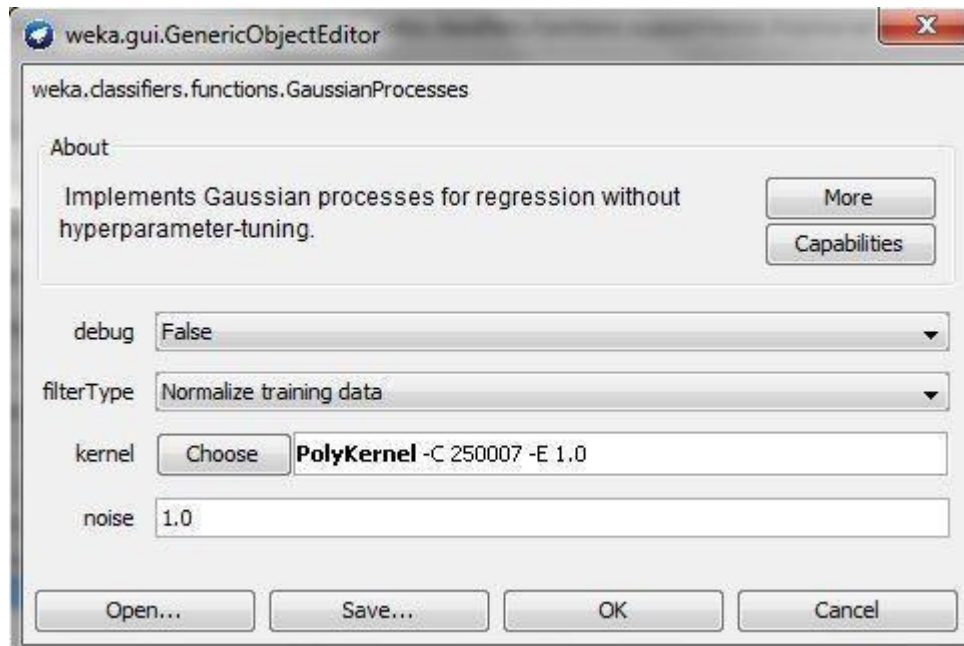
Aim: Understanding of Dataset of contact patterns among students collected in National University of Singapore.

This is dataset collected from contact patterns among students collected during the spring semester 2006 in National University of Singapore

Using RemovePercentage filter, instances have been reduced to: 500

This data has been taken and saved as training data set and then used for further classification.



ALGORITHM-1 : GaussianProcesses

=== Run information ===

Scheme: weka.classifiers.functions.SimpleLinearRegression

Relation: MOCK_DATA (1)-weka.filters.unsupervised.instance.RemovePercentage-P50.0 Instances: 500

Attributes: 4

Start Time

Session Id

Student Id

Duration

Test mode: evaluate on training data

=== Classifier model (full training set) ===

Linear regression on Session Id

$0.03 * \text{Session Id} + 10.38$

Time taken to build model: 0 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correlation coefficient 0.0677

Mean absolute error 4.9869

Root mean squared error 5.7893

Relative absolute error	99.7326 %
Root relative squared error	99.7708 %
Total Number of Instances	500

ALGORITHM 2: Linear Regression



Linear Regression Model

Start Time =

$$0.0274 * \text{Session Id} + 10.3846$$

Time taken to build model: 0.01 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

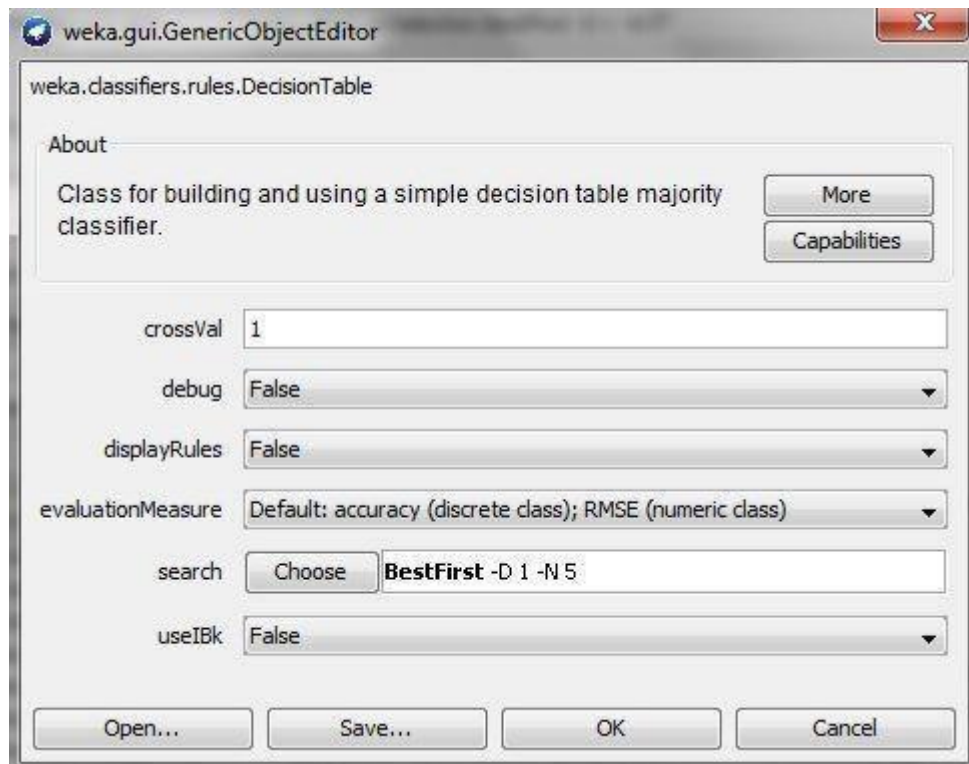
=== Summary ===

Correlation coefficient	0.0677
Mean absolute error	4.9869
Root mean squared error	5.7893

Relative absolute error 99.7326 %
 Root relative squared error 99.7708 %
 Total Number of Instances 500

Algorithm 3: Decision Table

Algorithm 6: DecisionTable



Merit of best subset found: 5.814

Evaluation (for feature selection): CV (leave one out) Feature set: 1

Time taken to build model: 0.02 seconds

=== Evaluation on training set ===

Time taken to test model on training data: 0 seconds

=== Summary ===

Correlation coefficient 0
 Mean absolute error 5.0003
 Root mean squared error 5.8026
 Relative absolute error 100 %
 Root relative squared error 100 %
 Total Number of Instances 500

CONCLUSION:

Six algorithms have been used to measure the best classifier. Depending on various attributes, performance of various algorithms can be measured via mean absolute error and correlation coefficient.

Depending on the results above, worst correlation has been found by DecisionTable and best correlation has been found by Decision Stump

=== Run information ===

Scheme: weka.classifiers.rules.DecisionTable -X 1 -S "weka.attributeSelection.BestFirst -D 1-N5"

Relation: MOCK_DATA (1)-weka.filters.unsupervised.instance.RemovePercentage-P50.0 Instances: 500

Attributes: 4

Start Time

Session Id

Student Id

Duration

Test mode: evaluate on training data

=== Classifier model (full training set) ===

Decision Table:

Number of training instances: 500

Number of Rules : 1

Non matches covered by Majority class.

Best first.

Start set: no attributes

Search direction: forward

Stale search after 5 node expansions

Total number of subsets evaluated: 9