CHANDIGARH UNIVERSITY

UNIVERSITY INSTITUTE OF ENGINEERING

Lab Manual

On

Subject Name – Machine Learning Lab

Subject Code – 20CSP-317

Odd Semester- July-Dec, 2022

Prepared by:

Name: Dr. Rohit Kumar Singhal

Designation: Professor

Department of Computer Science & Engineering

CONTENTS

1. UNIVERSITY-VISION AND MISSION

VISION: To be globally recognized as a Centre of Excellence for Research, Innovation, Entrepreneurship and disseminating knowledge by providing inspirational learning to produce professional leaders for serving the society

MISSION:

- Providing world class infrastructure, renowned academicians and ideal environment for Research, Innovation, Consultancy and Entrepreneurship relevant to the society.
- Offering programs & courses in consonance with National policies for nation building and meeting global challenges.
- Designing Curriculum to match International standards needs of Industry, civil society and for inculcation of traits of Creative Thinking and Critical Analysis as well as Human and Ethical values.
- Ensuring students delight by meeting their aspirations through blended learning, corporate mentoring, professional grooming, flexible curriculum and healthy atmosphere based on co-curricular and extra-curricular activities.
- Creating a scientific, transparent and objective examination/evaluation system to ensure an ideal certification.
- Establishing strategic relationships with leading National and International corporates and universities for academic as well as research collaborations.
- Contributing for creation of healthy, vibrant and sustainable society by involving in Institutional Social Responsibility (ISR) activities like rural development, welfare of senior citizens, women empowerment, community service, health and hygiene awareness and environmental protection

2. DEPARTMENT-VISION AND MISSION

VISION:

To be recognized as a leading Computer Science and Engineering department through effective teaching practices and excellence in research and innovation for creating competent professionals with ethics, valuesand entrepreneurial attitude to deliver service to society and to meet the current industry standards at the global level.

MISSION:

M1: To provide practical knowledge using state-of-the-art technological support for the experientiallearning of our students.

M2: To provide industry recommended curriculum and transparent assessment for quality learningexperiences.

M3: To create global linkages for interdisciplinary collaborative learning and research.

M4: To nurture advanced learning platform for research and innovation for students ‘profound futuregrowth.

M5: To inculcate leadership qualities and strong ethical values through value based education.

3. PROGRAM EDUCATIONAL OBJECTIVES (PEO)

The statements of PEOs (revised from 2022) are given below:

PEO 1. Graduates of the Computer Science and Engineering can contribute to the Nation's growth through their ability to solve diverse and complex computer science & engineering problems across a broad range of application areas.

PEO 2. Graduates of the Computer Science and Engineering can be successful professionals, designing and implementing Products & Services of global standards in the field of Computer Science & Engineering, becoming entrepreneurs, pursuing higher studies & research.

PEO 3. Graduates of the Computer Science and Engineering Program can be able to adapt to changing scenario of dynamic technology with an ability to solve larger societal problems using logical and flexible approach in decision making.

Consistency of the PEOs with Mission of the Department Mission of the department –

PEO matrix

PEO statement	**M1**	**M2**	**M3**	**M4**	**M5**
PEO 1	H	M	L	H	L
PEO 2	M	M	H	M	L
PEO 3	L	L	M	L	H

4. PROGRAM OUTCOMES

Program Outcomes are adopted from the outcomes defined by the National Board of Accreditation of India, which is the permanent signatory of the Washington Accord. Program outcomes are defined to ensure the holistic development of students.

Engineering Graduates will be able to:

PO 1. Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO 2. Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO 3. Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO 4. Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO 5. Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO 6. The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO 7. Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO 8. Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO 9. Individual and team work: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO 10. Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO 11. Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO 12. Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

Student Outcomes

The Bachelor of Engineering is a programme offered by the Department of Computer Science & Engineering in accordance with the Student Outcome of Computing Accreditation Commission (CAC) and Engineering Accreditation Commission (EAC) of ABET. The Student Outcomes are as follows:

Student Outcomes according to Computing Accreditation Commission (CAC)

SO 1. Analyze a complex computing problem and apply principles of computing and other relevant disciplines to identify solutions.

SO 2. Design, implement and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

SO 3. Communicate effectively in a variety of professional contexts.

SO 4. Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.

SO 5. Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.

SO 6. Apply computer science theory and software development fundamentals to produce computing-based solutions.

Student Outcomes according to Engineering Accreditation Commission (EAC)

SO 1. An ability to identify, formulates, and solve complex engineering problems by applying principles of engineering, science, and mathematics

SO 2. An ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as a global, cultural, social, environmental, and economic factor

SO 3. An ability to communicate effectively with a range of audiences

SO 4. An ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts

SO 5. An ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives.

SO 6. An ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions.

SO 7. An ability to acquire and apply new knowledge as needed, using appropriate learning strategies.

5. PROGRAM SPECIFIC OUTCOMES (PSOS)

A Graduate of Computer Science and Engineering Program will be able:

PSO 1. To acquire proficiency in developing and implementing efficient solutions using emerging technologies, platforms and free and open-source software (FOSS).

PSO 2. To gain critical understanding of hardware and software tools catering to the contemporary needs of IT industry.

6. COURSE OBJECTIVE

MACHINE LEARNING LAB-CSP-317

	Course Objective
1	To understand the history and development of Machine Learning
2	To provide a comprehensive foundation to Machine Learning and Optimization methodology with applications t.
3	To study learning processes: supervised and unsupervised, deterministic and statistical knowledge of Machine learners, and ensemble learning
4	To understand modern techniques and practical trends of Machine learning

7. COURSE OUTCOMES

MACHINE LEARNING LAB-CSP-317

	Course Outcomes
1	Explore and analyse the data with different data pre-processing and visualization techniques.
2	Select and apply the appropriate machine learning algorithm to solve problems of moderate complexity.
3	Design and evaluate the machine learning models through python / R in built functions.
4	Evaluate the machine learning models pre-processed through various feature engineering algorithms by python/ R programming.
5	Optimize the models learned and report on the expected accuracy that can be attained by applying the algorithms to a real-world problem.

8. MAPPING OF COs/POs/PSOs

CO_PO_SO Mapping (Practical)

CO-PO-Mapping														
CO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12	PSO1	PSO2
CO-1	3	3		3	3									
CO-2	3		3	3	3								2	
CO-3		3	3	3	3								2	2
CO-4	3			3	3			2						
CO-5	3		3	3	3			2		2		2	2	
Overall Student's Outcomes(														
		CAC--SO1	CAC--SO2	CAC--SO3	CAC--SO4	CAC--SO5	CAC—SO6	EAC-SO1	EAC-SO2	EAC-SO3	EAC-SO4	EAC-SO5	EAC-SO6	EAC-SO7
		3	3				3	3	3				3	3

CO_SO_mapping (Practical)

CO	CAC-SO1	CAC-SO2	CAC-SO3	CAC-SO4	CAC-SO5	CAC-SO6	EAC-SO1	EAC-SO2	EAC-SO3	EAC-SO4	EAC-SO5	EAC-SO6	EAC-SO7
CO1	✓	✓					✓					✓	✓
CO2	✓	✓				✓	✓	✓				✓	✓
CO3		✓				✓	✓	✓				✓	✓
CO4	✓	✓				✓	✓	✓				✓	✓
CO5		✓				✓	✓	✓				✓	✓

9. SYLLABUS (AS APPROVED IN BOS)

Chandigarh University, Gharuan

<table>
<tr><td rowspan="4">Subject Code CSP-317</td><td>Machine Learning Lab</td><td>L</td><td>T</td><td>P</td><td>C</td></tr>
<tr><td>Total Contact Hours : 48Hours</td><td rowspan="2">0</td><td rowspan="2">0</td><td rowspan="2">2</td><td rowspan="2">1</td></tr>
<tr><td>Common to all Specializations of CSE 3rd Year</td></tr>
<tr><td colspan="5">Prerequisite:
Knowledge of basic computer science principles and skills, at a level sufficient to write a reasonably non-trivial computer program.</td></tr>
<tr><td colspan="6">Marks-100</td></tr>
<tr><td colspan="3">Internal-60</td><td colspan="3">External-40</td></tr>
<tr><td>Unit</td><td colspan="5">Course Outcome</td></tr>
<tr><td>1</td><td colspan="5">Classify fundamental of data analysis, machine learning algorithms as supervised learning or unsupervised learning.</td></tr>
<tr><td>2</td><td colspan="5">Select and apply the appropriate machine learning algorithm to solve problems of moderate complexity.</td></tr>
<tr><td>3</td><td colspan="5">Design and evaluate the unsupervised models through python / R in built functions.</td></tr>
<tr><td>4</td><td colspan="5">Evaluate the machine learning models pre-processed through various feature engineering algorithms by python/ R programming.</td></tr>
<tr><td>5</td><td colspan="5">Optimize the models learned and report on the expected accuracy that can be attained by applying the algorithms to a real-world problem.</td></tr>
</table>

List of Experiments

UNIT-I

1. Implement Exploratory Data Analysis on any data set.
2. Implement Data Visualization.
3. Implement Linear Regression on any data set.

UNIT-II

1. Implement Support Vector Machine on any data set and analyze the accuracy with Logistic regression
2. Implement Naïve Bayes on any dataset.
3. Implement K-Nearest Neighbor on any data set
4. Implement Decision Tree and compare the performance with Random Forest on any data set.

UNIT-III

1. Implement K-means clustering algorithm (cluster some sample data set into disjoint clusters using K-means).
2. Implement Principle Component Analysis.
3. Implement Association Rule Mining.

CO-PO-Mapping														
CO	**PO 1**	**PO 2**	**PO 3**	**PO 4**	**PO 5**	**PO 6**	**PO 7**	**PO 8**	**PO 9**	**PO 10**	**PO 11**	**PO 12**	**PSO1**	**PSO2**
CO-1	**3**	**3**												
CO-2			**3**	**3**	**3**								**2**	
CO-3		**3**			**3**								**2**	**2**
CO-4				**3**	**3**			**2**						
CO-5					**3**			**2**		**2**		**2**	**2**	
Student's Outcomes														
	CAC--SO 1	**CAC--SO 2**	**CAC--SO 3**	**CAC--SO 4**	**CAC--SO 5**	**EAC-SO 1**	**EAC-SO 2**	**EAC-SO 3**	**EAC-SO 4**	**EAC-SO 5**	**EAC-SO 6**	**EAC-SO 7**	**CAC--SO 1**	**CAC--SO 2**
	3	**3**				**3**	**3**	**3**				**3**	**3**	**3**

CO-SO Mapping

CO	CAC-SO1	CAC-SO2	CAC-SO3	CAC-SO4	CAC-SO5	CAC-SO6	EAC-SO1	EAC-SO2	EAC-SO3	EAC-SO4	EAC-SO5	EAC-SO6	EAC-SO7
CO1			✓	✓		✓					✓		✓
CO2			✓	✓		✓			✓		✓		✓
CO3		✓		✓		✓	✓		✓		✓		✓
CO4		✓		✓	✓	✓	✓		✓		✓	✓	✓
CO5		✓	✓	✓		✓		✓			✓	✓	✓

Relationship between the Course Outcomes (COs) and Program Outcomes (POs)

Mapping Between COs and POs		
SN	**Course Outcome (CO)**	**Mapped Programme Outcome (PO)**
1	To generate analytical and conceptual ability related to fundamentals of Java.	Use an integrated development environment to write, compile, run, and test simple object-oriented Java programs.
2	To understand the concepts of Web application development.	Read and make elementary modifications to Java programs that solve real-world problems.
3	To understand the concepts of Fundamentals of I/O , Database Connectivity	Designs will demonstrate the use of good object-oriented design principles including encapsulation and information hiding.
4	To Implement of the OOPS concepts using Eclipse Environment	The implementation will demonstrate the use of a variety of basic control structures including selection and repetition; classes and objects in a tiered architecture (user interface, controller, and application logic layers); primitive and reference data types including composition; basic AWT components; file-based I/O; and one-dimensional arrays.
5	To implement the concepts of Collections and able to access database through	Test plans will include test cases demonstrating Testing strategies.

10. LIST OF EXPERIMENTS (MAPPED WITH COS)

No.	Experiment Name	Mapped with CO Number(s)
1	Implement Exploratory Data Analysis on any data set.	**CO1,CO3,CO4,CO5**
2	Implement Data Visualization.	**CO1,CO3,CO4,CO5**
3	Implement Linear Regression on any data set.	**CO1,CO3,CO4**
4	Implement Support Vector Machine on any data set and analyze the accuracy with Logistic regression	**CO1,CO4**
5	Implement Naïve Bayes on any dataset.	**CO1**
6	Implement K-Nearest Neighbor on any data set	**CO1**
7	Implement Decision Tree and compare the performance with Random Forest on any data set	**CO1,CO2,CO4,CO5**
8	Implement K-means clustering algorithm (cluster some sample data set into disjoint clusters using K-means).	**CO4,CO5**
9	Implement Principle Component Analysis.	**CO2,CO5**
10	Implement Association Rule Mining.	**CO2,CO5**

11. MANUAL TO CONDUCT EACH EXPERIMENT

Experiment:1: Implement Exploratory Data Analysis on any data set.

Data Set: Titanic

Objectives:-
- **To Learn about Meta-data**
- **To learn About Different data exploratory analysis Techniques**

Requirement Analysis:
- **Goggle CoLab (Online Compiler)**
- **Jupyter Notebook (Offline)**

Hardware Requirement
- **Windows 10.**
- **Power Supply.**
- **RAM-4GB**

Problem statement:
The data set consists of samples from each Titanic Data Set. Features were measured from each sample. Given problem relates with introduction to machine learning functions and classification. To Implement Data Exploratory Analysis with insight view with the help of different functions.

Exploratory Data Analysis (EDA) is applied to **investigate** the data and **summarize** the key insights. It will give the basic understanding of data, it's **distribution**, null values and much more.
You can either explore data using graphs or through some python **functions.**
There will be two type of analysis. Univariate and Bivariat**e.** In the univariate, you will be analyzing a single attribute. But in the bivariate, you will be analyzing an attribute with the target attribute.
In the **non-graphical approach**, you will be using functions such as shape, summary, describe, isnull, info, datatypes and more.
In the **graphical approach**, you will be using plots such as scatter, box, bar, density and correlation plots.

Load the Data

Well, first things first. We will load the titanic dataset into python to perform EDA.

```
#Load the required libraries
import pandas as pd
import numpy as np
```

```
import seaborn as sns
#Load the data
df = pd.read_csv('titanic.csv')
#View the data
df.head()
```

Copy

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...	...
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

Our data is ready to be explored!

1. Basic information about data - EDA

The df.info() function will give us the basic information about the dataset. For any data, it is good to start by knowing its information. Let’s see how it works with our data.

```
#Basic information
df.info()
#Describe the data
df.describe()
```

Copy

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   PassengerId   891 non-null     int64
 1   Survived      891 non-null     int64
 2   Pclass        891 non-null     int64
 3   Name          891 non-null     object
 4   Sex           891 non-null     object
 5   Age           714 non-null     float64
 6   SibSp         891 non-null     int64
 7   Parch         891 non-null     int64
 8   Ticket        891 non-null     object
 9   Fare          891 non-null     float64
 10  Cabin         204 non-null     object
 11  Embarked      889 non-null     object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

Describe the data - Descriptive statistics.

	PassengerId	Survived	Pclass	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	8.000000	6.000000	512.329200

Using this function, you can see the number of null values, datatypes, and memory usage as shown in the above outputs along with descriptive statistics.

2. Duplicate values

You can use the df.duplicate.sum() function to the sum of duplicate value present if any. It will show the number of duplicate values if they are present in the data.

```
#Find the duplicates
df.duplicated().sum()
```

Copy 0

Well, the function returned ‘0’. This means, there is not a single duplicate value present in our dataset and it is a very good thing to know.

3. Unique values in the data

You can find the number of unique values in the particular column using unique() function in python.

```
#unique values
df['Pclass'].unique()
df['Survived'].unique()
df['Sex'].unique()
```

Copy

```
array([3, 1, 2], dtype=int64)
array([0, 1], dtype=int64)
array(['male', 'female'], dtype=object)
```

The unique() function has returned the unique values which are present in the data and it is pretty much cool!

Viva voce:

1. What are Outliers and how to deal with it?
2. What is feature engineering?
3. How can features transformed?
4. How null values are eliminated?
5. How missing values can be treated?
6. What is the Difference between Univariate, Bivariate, and Multivariate analysis?

Experiment:2: Implement Data Visualization.

Aim: To Implement Data Exploratory Analysis with insight view with the help of data Visualization.

Data Set: iris-flower-dataset

Objectives:-

- **To Learn about Meta-data**
- **To Learn About Visualization Tool and Techniques**

Requirement Analysis:

- **Goggle CoLab (Online Compiler)**
- **Jupyter Notebook (Offline)**

Hardware Requirement

- **Windows 10.**
- **Power Supply.**
- **RAM-4GB**

Problem statement:

The data set consists of 50 samples from each of three species of Iris (Iris Setosa, Iris virginica, and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Given problem is of classification. To Implement Exploratory Analysis with insight view with the help of data Visualization.

```
Import numpy as np
import pandas as pd
import seabornassns
import matplotlib.pyplot as plt
```

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

data.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety	Unnamed: 5
0	5.1	3.5	1.4	0.2	Setosa	NaN
1	4.9	3.0	1.4	0.2	Setosa	NaN
2	4.7	3.2	1.3	0.2	Setosa	NaN
3	4.6	3.1	1.5	0.2	Setosa	NaN
4	5.0	3.6	1.4	0.2	Setosa	NaN

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149 Data
```

```
columns (total 6 columns): sepal.length
           150 non-null float64
sepal.width   150 non-null float64
petal.length  150 non-null float64
petal.width   150 non-null float64
variety     150 non-null object Unnamed:
5          0 non-null float64 dtypes:
float64(5), object(1)
memory usage: 6.5+ KB
```

```
data.describe()
```

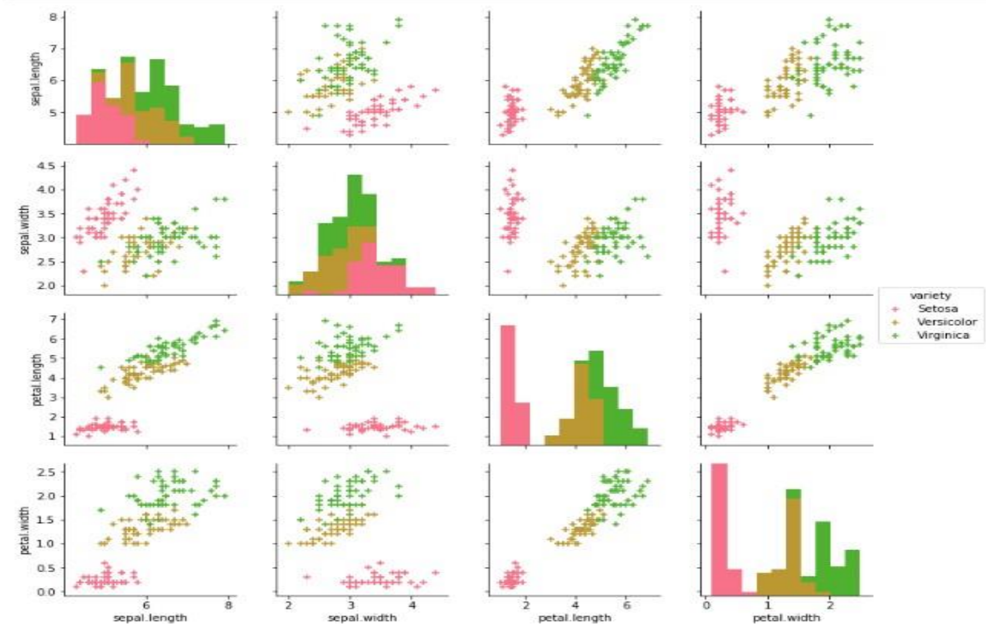
	sepal.length	sepal.width	petal.length	petal.width	Unnamed: 5
count	150.000000	150.000000	150.000000	150.000000	0.0
mean	5.843333	3.057333	3.758000	1.199333	NaN
std	0.828066	0.435866	1.765298	0.762238	NaN
min	4.300000	2.000000	1.000000	0.100000	NaN
25%	5.100000	2.800000	1.600000	0.300000	NaN
50%	5.800000	3.000000	4.350000	1.300000	NaN
75%	6.400000	3.300000	5.100000	1.800000	NaN
max	7.900000	4.400000	6.900000	2.500000	NaN

```
data['variety'].value_counts()
```

```
Setosa           50
Versicolor       50
Virginica        50
Name: variety, dtype: int64
```

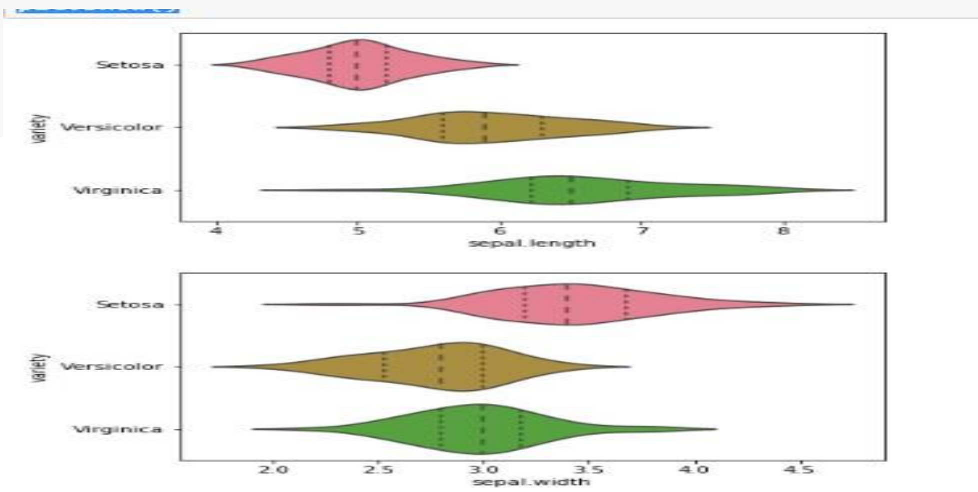
```
g=sns.pairplot(tmp,hue='variety', markers='+')
plt.show()
```

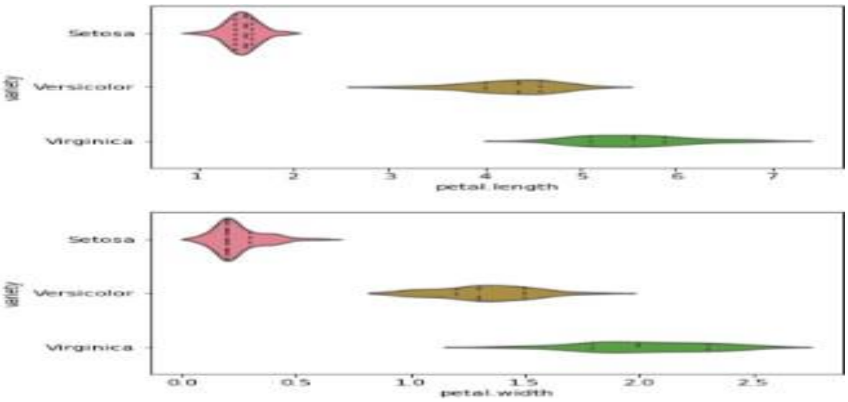
	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa




```
g=sns.violin plot(y='varie ty',x='sepal.l ength',data= data,inner='q uartile')
```

```
plt.show()
g=sns.violinplot(y='variety',x='sepal.width',data=data,inner='quartile')
plt.show()
g=sns.violinplot(y='variety',x='petal.length',data=data,inner='quartile')
plt.show()
g=sns.violinplot(y='variety',x='petal.width',data=data,inner='quartile')
plt.show()
```






```
X=data.drop(['Unnamed: 5','variety'],axis=1)
y=data['variety']
# print(X.head())
print(X.shape)
# print(y.head())
print(y.shape)
```

```
(150, 4)
(150,)
```

Viva voce:

1. What are the visualization techniques?
2. What should be done with suspected or missing data?
3. What process would you use to transform raw data into a visualization? ...
4. What is a scatter plot ?
5. Name some data validation techniques.
6. What are characteristics of a good data model?

Experiment:3 Implement Linear Regression on any data set.

Aim: To implement linear Regression on any Data set and justify the outcomes with parameters**.**

Link of Problem: Supermarket sales

Objectives:-

- To learn about different functions.
- To learn About Different Linear Regression Techniques
- To Learn about Linear Regression Model or algorithms

Requirement Analysis:

- Goggle CoLab (Online Compiler)
- Jupyter Notebook (Offline)

Hardware Requirement

- Windows 10.
- Power Supply.
- RAM-4GB

Problem statement:

The growth of supermarkets in most populated cities is increasing and market competitions are also high. The dataset is one of the historical sales of Supermarket Company which has recorded in 3 different branches for 3 months data. Predictive data analytics methods are easy to apply with this dataset.

Discussion:

Attribute information

Invoice id: Computer generated sales slip invoice identification number

Branch: Branch of supercenter (3 branches are available identified by A, B and C).

City: Location of supercenters

Customer type: Type of customers, recorded by Members for customers using member card and Normal for without member card.

Gender: Gender type of customer

Product line: General item categorization groups - Electronic accessories, Fashion accessories, Food and beverages, Health and beauty, Home and lifestyle, Sports and travel

Unit price: Price of each product in $

Quantity: Number of products purchased by customer

Tax: 5% tax fee for customer buying

Total: Total price including tax

Date: Date of purchase (Record available from January 2019 to March 2019)

Time: Purchase time (10am to 9pm)

Payment: Payment used by customer for purchase (3 methods are available – Cash, Credit card and Ewallet)

COGS: Cost of goods sold

Gross margin percentage: Gross margin percentage

Gross income: Gross income

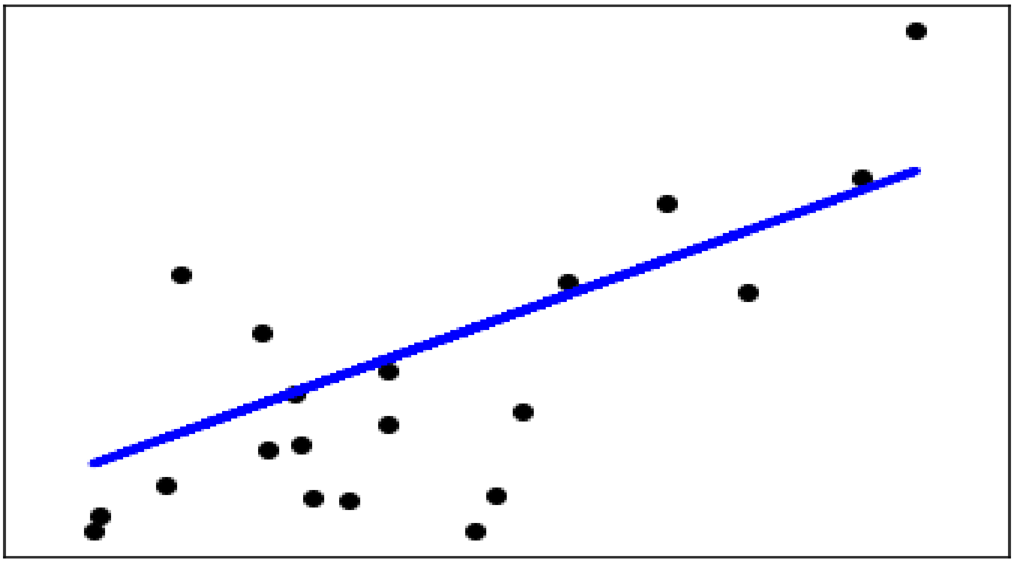
Rating: Customer stratification rating on their overall shopping experience (On a scale of 1 to 10)
This dataset can be used for predictive data analytics purpose.

Solution:

```
Import matplotlib.pyplot as plt import numpy as np
From sklearn import datasets, linear_model
From sklearn.metrics importmean_squared_error,  r2_score
# Load the diabetes dataset
diabetes=datasets.load_diabetes()


# Use only one feature
diabetes_X=diabetes.data[:, np.newaxis, 2]

# Split the data into training/testing sets diabetes_X_train=diabetes_X[:-20]
diabetes_X_test=diabetes_X[-20:]

# Split the targets into training/testing sets diabetes_y_train=diabetes.target[:-20]
diabetes_y_test=diabetes.target[-20:]
```

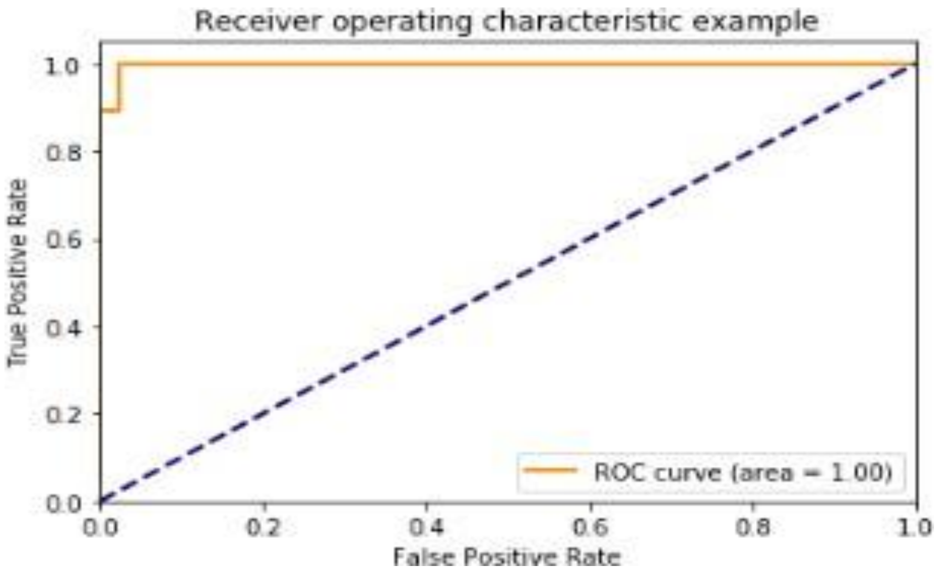
```
# Create linear regression object
regr=linear_model.LinearRegression()

# Train the model using the training sets
regr.fit(diabetes_X_train,  diabetes_y_train)

# Make predictions using the testing set
diabetes_y_pred=regr.predict(diabetes_X_test)

# The coefficients print('Coefficients: \n', regr.coef_) # The mean squared error print("Mean
squared error: %.2f"
%mean_squared_error(diabetes_y_test,  diabetes_y_pred))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f'% r2_score(diabetes_y_test, diabetes_y_pred))

# Plot outputs
plt.scatter(diabetes_X_test, diabetes_y_test,  color='black') plt.plot(diabetes_X_test,
diabetes_y_pred, color='blue', linewidth=3)

plt.xticks(())
plt.yticks(())

plt.show()
```

('Coefficients: \n', array([938.23786125])) Mean squared error: 2548.07
Variance score: 0.47



```
roc_auc=dict() foriinrange(n_classes):
fpr[i], tpr[i], _ =roc_curve(y1[:, i], y_score[:,

i]) roc_auc[i] =auc(fpr[i], tpr[i])
```

```
lw=2
plt.plot(fpr[2], tpr[2], color='darkorange',lw=lw, label='ROC curve (area =
%0.2f)'%roc_auc[2]) plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic example') plt.legend(loc="lower right")
plt.show()
```



Viva voce:

1. What is Learning Rate?
2. What are the Assumptions of Linear Regression?
3. What are the Different Types of Gradient Descent in Linear Regression?
4. What is Heteroscedasticity?
5. What is Multicollinearity and how can it Impact the Model?
6. What are the Loss Functions used in Linear Regression?

Experiment: 4: Implement Support Vector Machine on any data set and analyze the accuracy with Logistic regression

Aim: To Implement Support Vector Machine on Classification Problem and Junstify the outcome with relevant Parameters.

Data Set: **iris-flower-dataset**

Objectives:-

- To Learn about Meta-data
- To learn About Different Support Vector Machine Techniques
- To Learn about Support Vector Machine Model or algorithms

Requirement Analysis:

- Goggle CoLab (Online Compiler)
- Jupyter Notebook (Offline**)**

Hardware Requirement

- Windows 10.
- Power Supply.
- RAM-4GB

Problem statement:

The data set consists of 50 samples from each of three species of Iris (Iris Setosa, Iris virginica, and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Given problem is of classification. To Implement Data Exploratory Analysis with insight view with the help of data Visualization. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

This dataset became a typical test case for many statistical classification techniques in machine learning such as support vector machines

Content

The dataset contains a set of 150 records under 5 attributes - Petal Length, Petal Width, Sepal Length, Sepal width and Class(Species).

Discussion:

he Distribution of Sepal Length and Sepal Width is Normal, But the Distribution of Petal width and Petal Lenght is kind of Bi-Modal, After Analyzing, I came to the conclusion that it is due to the small petal width and length of iris Setosa, Iris Setosa is one of the species in iris dataset having least Petal Length and WIdth, which makes the Whole Petal Lenght and Width Bi-Modal

```
importnumpyasnp
importpandasaspd
importseabornassns
importmatplotlib.pyplotasplt

fromsklearn.model_selectionimporttrain_test_split
fromsklearn.metricsimportclassification_report,confusion_matrix
fromsklearn.metricsimportroc_curve,auc
fromsklearn.preprocessingimportlabel_binarize
fromsklearn.multiclassimportOneVsRestClassifier
fromsklearn.metricsimportprecision_recall_curve
fromsklearn.metricsimportroc_auc_score
fromsklearn.linear_modelimportLogisticRegression
fromsklearn.svmimportSVC
```

Solution:

```
data=pd.read_csv('Downloads/iris  data.csv')

data.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety	Unnamed: 5
0	5.1	3.5	1.4	0.2	Setosa	NaN
1	4.9	3.0	1.4	0.2	Setosa	NaN
2	4.7	3.2	1.3	0.2	Setosa	NaN
3	4.6	3.1	1.5	0.2	Setosa	NaN
4	5.0	3.6	1.4	0.2	Setosa	NaN

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
sepal.length    150 non-null float64
sepal.width     150 non-null float64
petal.length    150 non-null float64
petal.width     150 non-null float64
variety         150 non-null object
Unnamed: 5      0 non-null float64
dtypes: float64(5), object(1)
memory usage: 6.5+ KB
```

```
tmp=data.drop('Unnamed: 5',axis=1)
tmp.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

```
X=data.drop(['Unnamed: 5','variety'],axis=1)
y=data['variety']
# print(X.head())
print(X.shape)
# print(y.head())
print(y.shape)
```

```
lit(X,y,test_size=0.4,random_state=5)
print(X_train.shape) print(y_train.shape) print(X_test.shape) print(y_test.shape)
```

```
(150, 4)
(150,)
```

```
(90, 4)
(90,)
(60, 4)
(60,)
```

```
logreg.fit(X_train, y_train) y_pred=logreg.predict(X_test) print(metrics.accuracy_score(y_test, y_pred))
```

```
0.9333333333333333
```

```
logreg.predict([[6, 3, 4, 2]])
```

```
array(['Virginica'], dtype=object)
```

```
logreg.predict([[5, 3, 1, 0]]) sv=SVC()
sv.fit(X_train, y_train)
```

```
array(['Setosa'],
dtype=object)
```

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
```

```
pred=sv.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred))
```

```
0.9333333333333333
```

```
sv.predict([[6, 3, 4, 2]])
```

```
array(['Versicolor'], dtype=object)
```

sv.predict([[5, 3, 1, 0]])

```
array(['Setosa'], dtype=object)
```

Viva voce:

1. What are Support Vectors in SVMs?
2. What are hard Margin and Soft Margin SVMs?
3. What do you mean by Hinge loss?
4. What is the kernel trick?
5. What is the role of c-hyper Parameter in Svm?
6. Hyper Parameter affects the Bias /variance trade off?

5. **Implement Naïve Bayes on any dataset.**

Aim: To Implement Naïve Bayes algorithm and Justify the outcome with relevant Parameters.

Data Set: **iris-flower-dataset**

Objectives:-

- How to calculate the probabilities required by the Naive Bayes algorithm.
- How to implement the Naive Bayes algorithm from scratch.
- How to apply Naive Bayes to a real-world predictive modeling problem.

Requirement Analysis:

- Goggle CoLab (Online Compiler)
- Jupyter Notebook (Offline)

Hardware Requirement

- Windows 10.
- Power Supply.
- RAM-4GB

Problem statement:

The data set consists of 50 samples from each of three species of Iris (Iris Setosa, Iris virginica, and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Given problem is of classification. Bayes' Theorem provides a way that we can calculate the probability of a piece of data belonging to a given class, given our prior knowledge. Bayes' Theorem is stated as:

P(class|data) = (P(data|class) * P(class)) / P(data)
Where P(class|data) is the probability of class given the provided data.

An Introduction to Bayes Theorem for Machine Learning

Naive Bayes is a classification algorithm for binary (two-class) and multiclass classification problems. It is called Naive Bayes or idiot Bayes because the calculations of the probabilities for each class are simplified to make their calculations tractable.

Naive Bayes experiment

First we will develop each piece of the algorithm, then we will tie all of the elements together into a working implementation applied to a real dataset.
This Naive Bayes tutorial is broken down into 5 parts:
Step 1: Separate By Class.

Step 2: Summarize Dataset.
Step 3: Summarize Data By Class.
Step 4: Gaussian Probability Density Function.
Step 5: Class Probabilities.
These steps will provide the foundation that you need to implement Naive Bayes from scratch and apply it to your own predictive modeling problems.

Step 1: Separate By Class

We will need to calculate the probability of data by the class they belong to, the so-called base rate.

This means that we will first need to separate our training data by class. A relatively straightforward operation.

We can create a dictionary object where each key is the class value and then add a list of all the records as the value in the dictionary.

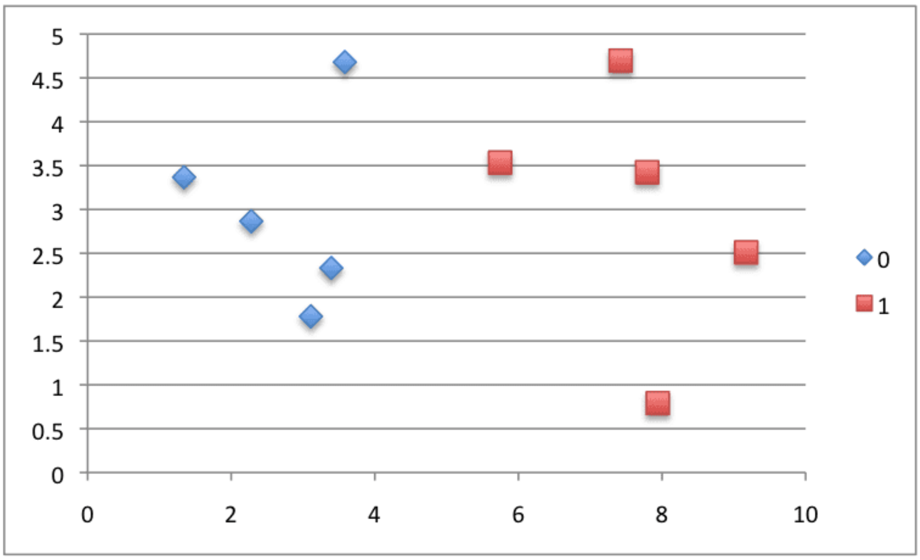
Below is a function named *separate_by_class()* that implements this approach. It assumes that the last column in each row is the class value.

```
# Split the dataset by class values, returns a dictionary
def separate_by_class(dataset):
	separated = dict()
	for i in range(len(dataset)):
		vector = dataset[i]
		class_value = vector[-1]
		if (class_value not in separated):
			separated[class_value] = list()
		separated[class_value].append(vector)
	return separated
```

We can contrive a small dataset to test out this function.

X1	X2	Y
3.393533211	2.331273381	0
3.110073483	1.781539638	0
1.343808831	3.368360954	0
3.582294042	4.67917911	0
2.280362439	2.866990263	0
7.423436942	4.696522875	1
5.745051997	3.533989803	1
9.172168622	2.511101045	1
7.792783481	3.424088941	1
7.939820817	0.791637231	1

We can plot this dataset and use separate colors for each class.



Scatter Plot of Small Contrived Dataset for Testing the Naive Bayes Algorithm

Putting this all together, we can test our *separate_by_class()* function on the contrived dataset.

```
# Example of separating data by class value

# Split the dataset by class values, returns a dictionary
def separate_by_class(dataset):
	separated = dict()
	for i in range(len(dataset)):
		vector = dataset[i]
		class_value = vector[-1]
		if (class_value not in separated):
			separated[class_value] = list()
		separated[class_value].append(vector)
	return separated

# Test separating data by class
dataset = [[3.393533211,2.331273381,0],
	[3.110073483,1.781539638,0],
	[1.343808831,3.368360954,0],
	[3.582294042,4.67917911,0],
	[2.280362439,2.866990263,0],
	[7.423436942,4.696522875,1],
	[5.745051997,3.533989803,1],
	[9.172168622,2.511101045,1],
```

```
            [7.792783481,3.424088941,1],
            [7.939820817,0.791637231,1]]
separated = separate_by_class(dataset)
for label in separated:
            print(label)
            for row in separated[label]:
                        print(row)
```

Running the example sorts observations in the dataset by their class value, then prints the class value followed by all identified records.

```
0
[3.393533211, 2.331273381, 0]
[3.110073483, 1.781539638, 0]
[1.343808831, 3.368360954, 0]
[3.582294042, 4.67917911, 0]
[2.280362439, 2.866990263, 0]
1
[7.423436942, 4.696522875, 1]
[5.745051997, 3.533989803, 1]
[9.172168622, 2.511101045, 1]
[7.792783481, 3.424088941, 1]
[7.939820817, 0.791637231, 1]
```

Next we can start to develop the functions needed to collect statistics.

Step 2: Summarize Dataset

We need two statistics from a given set of data.

We'll see how these statistics are used in the calculation of probabilities in a few steps. The two statistics we require from a given dataset are the mean and the standard deviation (average deviation from the mean).

The mean is the average value and can be calculated as:

mean = sum(x)/n * count(x)

Where *x* is the list of values or a column we are looking.

Below is a small function named *mean()* that calculates the mean of a list of numbers.

```
# Calculate the mean of a list of numbers
def mean(numbers):
            return sum(numbers)/float(len(numbers))
```

The sample standard deviation is calculated as the mean difference from the mean value. This can be calculated as:

standard deviation = sqrt((sum i to N (x_i – mean(x))^2) / N-1)

You can see that we square the difference between the mean and a given value, calculate the average squared difference from the mean, then take the square root to return the units back to their original value.

Below is a small function named *standard_deviation()* that calculates the standard deviation of a list of numbers. You will notice that it calculates the mean. It might be more efficient to calculate the mean of a list of numbers once and pass it to the *standard_deviation()* function as a parameter. You can explore this optimization if you're interested later.

```
from math import sqrt

# Calculate the standard deviation of a list of numbers
def stdev(numbers):
	avg = mean(numbers)
	variance = sum([(x-avg)**2 for x in numbers]) / float(len(numbers)-1)
	return sqrt(variance)
```

We require the mean and standard deviation statistics to be calculated for each input attribute or each column of our data.

We can do that by gathering all of the values for each column into a list and calculating the mean and standard deviation on that list. Once calculated, we can gather the statistics together into a list or tuple of statistics. Then, repeat this operation for each column in the dataset and return a list of tuples of statistics.

Below is a function named *summarize_dataset()* that implements this approach. It uses some Python tricks to cut down on the number of lines required.

```
# Calculate the mean, stdev and count for each column in a dataset
def summarize_dataset(dataset):
	summaries = [(mean(column), stdev(column), len(column)) for column in zip(*dataset)]
	del(summaries[-1])
	return summaries
```

The first trick is the use of the zip() function that will aggregate elements from each provided argument. We pass in the dataset to the *zip()* function with the * operator that separates the dataset (that is a list of lists) into separate lists for each row. The *zip()* function then iterates over each element of each row and returns a column from the dataset as a list of numbers. A clever little trick.

We then calculate the mean, standard deviation and count of rows in each column. A tuple is created from these 3 numbers and a list of these tuples is stored. We then remove the statistics for the class variable as we will not need these statistics.

Step 3: Summarize Data By Class

We require statistics from our training dataset organized by class.

Above, we have developed the *separate_by_class()* function to separate a dataset into rows by class. And we have developed *summarize_dataset()* function to calculate summary statistics for each column.

We can put all of this together and summarize the columns in the dataset organized by class values.

Below is a function named *summarize_by_class()* that implements this operation. The dataset is first split by class, then statistics are calculated on each subset. The results in the form of a list of tuples of statistics are then stored in a dictionary by their class value.

```
# Split dataset by class then calculate statistics for each row
def summarize_by_class(dataset):
	separated = separate_by_class(dataset)
	summaries = dict()
	for class_value, rows in separated.items():
		summaries[class_value] = summarize_dataset(rows)
	return summaries
```

Step 4: Gaussian Probability Density Function

Calculating the probability or likelihood of observing a given real-value like X1 is difficult.

One way we can do this is to assume that X1 values are drawn from a distribution, such as a bell curve or Gaussian distribution.

A Gaussian distribution can be summarized using only two numbers: the mean and the standard deviation. Therefore, with a little math, we can estimate the probability of a given value. This piece of math is called a Gaussian Probability Distribution Function (or Gaussian PDF) and can be calculated as:

f(x) = (1 / sqrt(2 * PI) * sigma) * exp(-((x-mean)^2 / (2 * sigma^2)))

Where *sigma* is the standard deviation for *x*, *mean* is the mean for *x* and *PI* is the value of pi.

Below is a function that implements this. I tried to split it up to make it more readable.

```
# Calculate the Gaussian probability distribution function for x
def calculate_probability(x, mean, stdev):
	exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
	return (1 / (sqrt(2 * pi) * stdev)) * exponent
```

Step 5: Class Probabilities

Now it is time to use the statistics calculated from our training data to calculate probabilities for new data.

Probabilities are calculated separately for each class. This means that we first calculate the probability that a new piece of data belongs to the first class, then calculate probabilities that it belongs to the second class, and so on for all the classes.

The probability that a piece of data belongs to a class is calculated as follows:

P(class|data) = P(X|class) * P(class)

You may note that this is different from the Bayes Theorem described above.
The division has been removed to simplify the calculation.
This means that the result is no longer strictly a probability of the data belonging to a class. The value is still maximized, meaning that the calculation for the class that results in the largest value is taken as the prediction. This is a common implementation simplification as we are often more interested in the class prediction rather than the probability.
The input variables are treated separately, giving the technique it's name "*naive*". For the above example where we have 2 input variables, the calculation of the probability that a row belongs to the first class 0 can be calculated as:
P(class=0|X1,X2) = P(X1|class=0) * P(X2|class=0) * P(class=0)
Now you can see why we need to separate the data by class value. The Gaussian Probability Density function in the previous step is how we calculate the probability of a real value like X1 and the statistics we prepared are used in this calculation.

Viva voce:

1. How would you use Naive Bayes classifier for categorical features?
2. What is the difference between Logistic Regression algorithms and Naïve Bayes algorithm?
3. What Bayes' Theorem (Bayes Rule) is all about?
4. What are some disadvantages of using Naive Bayes Algorithm?
5. What are some advantages of using Naive Bayes Algorithm?
6. What is the Central Limit Theorem (CLT)?

Experiment: 6 Implement K-Nearest Neighbor on any data set

Aim: To Implement K-nearest Neighbour on Classification Problem and Junstify the outcome with relevant Parameters.

Link of Problem: **https://www.kaggle.com/datasets/arshid/iris-flower-dataset**

Objectives:-

- To Learn about Meta-data and different clustering functions
- To learn About Different KNN Techniques
- To Learn about Cluster Model or algorithms

Requirement Analysis:

- Goggle CoLab (Online Compiler)
- Jupyter Notebook (Offline**)**

Hardware Requirement

- Windows 10.
- Power Supply.
- RAM-4GB

Problem statement:

The data set consists of 50 samples from each of three species of Iris (Iris Setosa, Iris virginica, and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Given problem is of classification. To Implement Data Exploratory Analysis with insight view with the help of data Visualization. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

This dataset became a typical test case for many statistical classification techniques in machine learning such as K-Nearest Neighbour.

Content

The dataset contains a set of 150 records under 5 attributes - Petal Length, Petal Width, Sepal Length, Sepal width and Class(Species).

Discussion:

he Distribution of Sepal Length and Sepal Width is Normal, But the Distribution of Petal width and Petal Lenght is kind of Bi-Modal, After Analyzing, I came to the conclusion that it is due to the small petal width and length of iris Setosa, Iris Setosa is one of the species in iris dataset having least Petal Length and WIdth, which makes the Whole Petal Lenght and Width Bi-Modal

Solution:

```
importnumpyasnp
importpandasaspd
importseabornassns
importmatplotlib.pyplotasplt

fromsklearn.model_selectionimporttrain_test_split
fromsklearn.metricsimportclassification_report,confusion_matrix
fromsklearn.metricsimportroc_curve,auc
fromsklearn.preprocessingimportlabel_binarize
fromsklearn.multiclassimportOneVsRestClassifier
fromsklearn.metricsimportprecision_recall_curve
fromsklearn.metricsimportroc_auc_score
fromsklearn.neighborsimportKNeighborsClassifier
```

```
data=pd.read_csv('Downloads/iris  data.csv')

data.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety	Unnamed: 5
0	5.1	3.5	1.4	0.2	Setosa	NaN
1	4.9	3.0	1.4	0.2	Setosa	NaN
2	4.7	3.2	1.3	0.2	Setosa	NaN
3	4.6	3.1	1.5	0.2	Setosa	NaN
4	5.0	3.6	1.4	0.2	Setosa	NaN

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
sepal.length    150 non-null float64
sepal.width     150 non-null float64
petal.length    150 non-null float64
petal.width     150 non-null float64
variety         150 non-null object
Unnamed: 5      0 non-null float64
dtypes: float64(5), object(1)
memory usage: 6.5+ KB
```

```
X=data.drop(['Unnamed: 5','variety'],axis=1) y=data['variety']
# print(X.head())
print(X.shape)
# print(y.head())
print(y.shape)
```

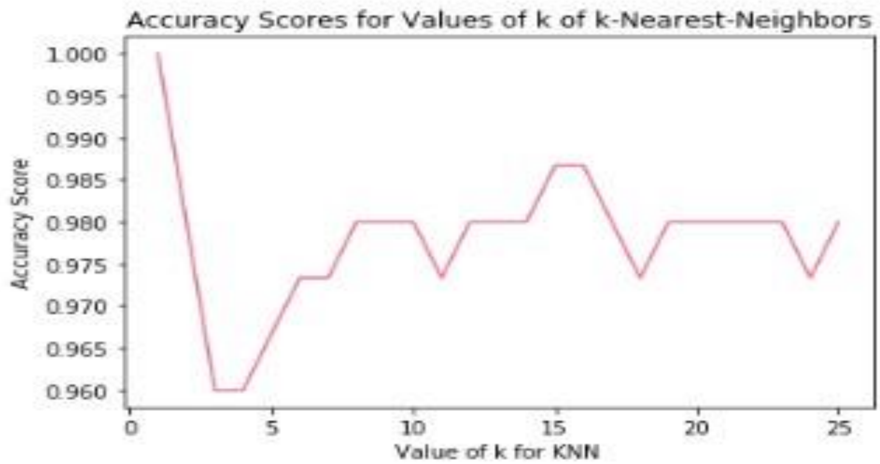
```
(150, 4)
(150,)
```

X_train,X_test,y_trai

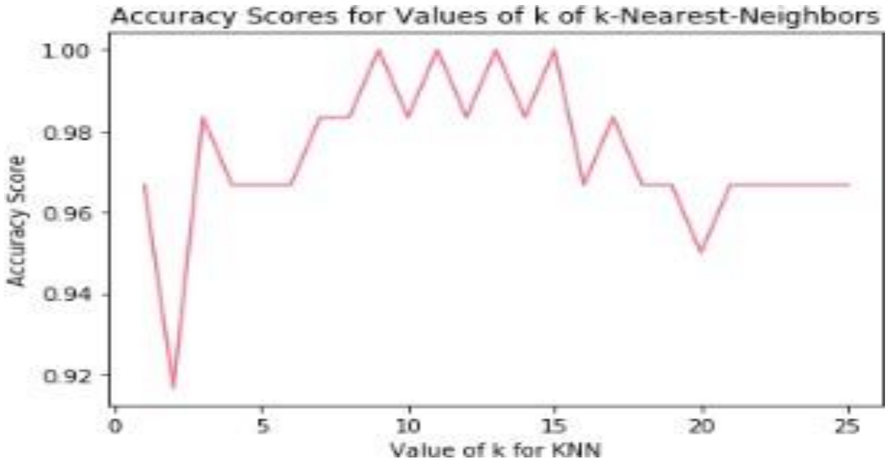
n,y_test=train_test_split(X,y,test_size=0.4,random_state=5)
print(X_train.shape) print(y_train.shape) print(X_test.shape) print(y_test.shape)

```
(90, 4)
(90,)
(60, 4)
(60,)
```

```
k_range=list(range(1,26))
scores=[]
forkink_range:
knn=KNeighborsClassifier(n_neighbors=k)
knn.fit(X,y)
y_pred=knn.predict(X)
scores.append(metrics.accuracy_score(y,y_pred))

plt.plot(k_range,scores)
plt.xlabel('Value of k for KNN')
plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores for Values of k of k-Nearest-Neighbors')
    plt.show()
```




```
k_range=list(range(1,26)) scores=[]
forkink_range: knn=KNeighborsClassifier(n_neighbors=k) knn.fit(X_train,y_train)
y_pred=knn.predict(X_test)
scores.append(metrics.accuracy_score(y_test,y_pred))

plt.plot(k_range,scores) plt.xlabel('Value of k for KNN') plt.ylabel('Accuracy Score')
plt.title('Accuracy Scores for Values of k of k-Nearest-Neighbors') plt.show()
```




```
knn=KNeighborsClassifier(n_neighbors=12) knn.fit(X, y)

# make a prediction for an example of an out-of-sample observation
knn.predict([[6, 3, 4, 2]])
```

```
array(['Versicolor'], dtype=object)
```

Viva voce:

7. What is “K” in KNN algorithm?
8. How do we decide the value of "K" in KNN algorithm?
9. Why is the odd value of “K” preferable in KNN algorithm?
10. What is the difference between Euclidean Distance and Manhattan distance? What is the formula of Euclidean distance and Manhattan distance?
11. Why is KNN algorithm called Lazy Learner?
12. Why should we not use KNN algorithm for large datasets?

Experiment: 7 Implement Decision Tree and compare the performance with Random Forest on any data set.

Aim: To Implement Decision tree on Classification Problem and Justify the outcome with relevant Parameters.

Link of Problem: **https://www.kaggle.com/datasets/arshid/iris-flower-dataset**

Objectives:-

- To Learn about Decision tree functions
- To learn About Different Decision Tree Techniques
- To Learn about Random Forest Model

Requirement Analysis:

- Goggle CoLab (Online Compiler)
- Jupyter Notebook (Offline)

Hardware Requirement

- Windows 10.
- Power Supply.
- RAM-4GB

Problem statement:

The data set consists of 50 samples from each of three species of Iris (Iris Setosa, Iris virginica, and Iris versicolor). Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters. Given problem is of classification. To Implement Data Exploratory Analysis with insight view with the help of data Visualization. Four features were measured from each sample: the length and the width of the sepals and petals, in centimeters.

This dataset became a typical test case for many statistical classification techniques in machine learning such as Decision Treee

Content

The dataset contains a set of 150 records under 5 attributes - Petal Length, Petal Width, Sepal Length, Sepal width and Class(Species).

Discussion:

he Distribution of Sepal Length and Sepal Width is Normal, But the Distribution of Petal width and Petal Lenght is kind of Bi-Modal, After Analyzing, I came to the conclusion that it is due to the small petal width and length of iris Setosa, Iris Setosa is one of the species in iris dataset having least Petal Length and WIdth, which makes the Whole Petal Lenght and Width Bi-Modal

we all know the importance of Exploratory Data Analysis in Machine Learning. The basic steps that one has to perform while doing EDA are Univariate, Bivariate & Multivariate Analysis. I have created a new notebook that contains the basic analysis on Iris dataset. Beginners can refer to this notebook as a starter for mastering EDA.

Solution:

```
importnumpyasnp
importpandasaspd
importseabornassns
importmatplotlib.pyplotasplt

fromsklearn.model_selectionimporttrain_test_split
fromsklearn.ensembleimportRandomForestClassifier
```

```
data=pd.read_csv('Downloads/iris data.csv')

data.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety	Unnamed: 5
0	5.1	3.5	1.4	0.2	Setosa	NaN
1	4.9	3.0	1.4	0.2	Setosa	NaN
2	4.7	3.2	1.3	0.2	Setosa	NaN
3	4.6	3.1	1.5	0.2	Setosa	NaN
4	5.0	3.6	1.4	0.2	Setosa	NaN

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
sepal.length     150 non-null float64
sepal.width      150 non-null float64
petal.length     150 non-null float64
petal.width      150 non-null float64
variety          150 non-null object
Unnamed: 5       0 non-null float64
dtypes: float64(5), object(1)
memory usage: 6.5+ KB
```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

```
X=data.drop(['Unnamed: 5','variety'],axis=1)
y=data['variety']
# print(X.head())
```

```
print(X.shape)
# print(y.head())
print(y.shape)
```

```
(150, 4)
(150,)
```

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.4,random_state=5) print(X_train.shape) print(y_train.shape) print(X_test.shape) print(y_test.shape)

```
(90, 4)
(90,)
(60, 4)
(60,)
```

clf=RandomForestClassifier(n_estimators=100)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
#Import scikit-learn metrics module for accuracy calculation
fromsklearnimport metrics
 # Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

```
('Accuracy:', 0.95)
```

clf.predict([[3, 5, 4, 2]])

```
array(['Virginica'], dtype=object)
```

Viva voce:

1. What does random refer to in ‘Random Forest’?
2. Explain the structure of a Decision Tree.
3. How is a Random Forest related to Decision Trees?
4. Compare Linear Regression and Decision Trees
5. How would you deal with an Over fitted Decision Tree?
6. How would you define the Stopping Criteria for decision trees?

Experiment: 8 Implement K-means clustering algorithm (cluster some sample data set into disjoint clusters using K-means).

Aim: To Implement k-mean on Clustering Problem and Justify the outcome with relevant Parameters.

Link of Problem: **[K-Mean clustering for Wine Quality Data | Kaggle]()**
Objectives:-

- To Learn about unsupervised learning
- To learn about clustering Techniques and implementation to K-means.
- To Learn about Clustering Model based on K-means algorithm and analysis

Requirement Analysis:

- Goggle CoLab (Online Compiler)
- Jupyter Notebook (Offline)

Hardware Requirement

- Windows 10.
- Power Supply.
- RAM-4GB

Problem statement:

Discussion: Clustering leads to new discovery of knowledge.

Clustering is an branch of Unsupervised Learning. Theory for Clustering Basic Req. when we can say we have clusters. There must be some way to say that 1 observation is closer to A observation than B.
There must be some proximity measure or similarity measure between data points of dataset.
Object should be as homogenous as possible in 1 cluster and object point between 2 cluster should be as homogenous as possible.
Proximity Measure.
Goodness of fit function.
Clustering must be effective i.e it should be complete and correct.
Req. for a Good Clustering Algorithm
Scalable (independent of size of data).
Should be able to deal with different types of data.
Whatever may be the shape of cluster, the algo should be able to handle the clustering.
A good clustering solution will remove Noise and Outliers.
Whatever order the data is feed into algo, the cluster should always be the same.

We will be making use of K-mean clustering techniques.
To make sure the results we will make use of Data manipulation and data analysis so that we

get good clustering results.
We will also make use of various techniques to find homoginity and numbers of clusters we should create to gt best results.

```
X=data.drop(['Unnamed:
5','variety'],axis=1) y=data['variety']
# print(X.head())
print(X.shape)
# print(y.head())
print(y.shape)
```

	sepal.length	sepal.width	petal.length	petal.width	variety
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa

Solutions:

```
importnumpyasnp
importpandasaspd
importseabornassns
importmatplotlib.pyplotasplt

fromsklearn.model_selectionimporttrain_test_split
fromsklearn.ensembleimportRandomForestClassifier
```

```
data=pd.read_csv('Downloads/iris  data.csv')

data.head()
```

	sepal.length	sepal.width	petal.length	petal.width	variety	Unnamed: 5
0	5.1	3.5	1.4	0.2	Setosa	NaN
1	4.9	3.0	1.4	0.2	Setosa	NaN
2	4.7	3.2	1.3	0.2	Setosa	NaN
3	4.6	3.1	1.5	0.2	Setosa	NaN
4	5.0	3.6	1.4	0.2	Setosa	NaN

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
sepal.length    150 non-null float64
sepal.width     150 non-null float64
petal.length    150 non-null float64
petal.width     150 non-null float64
variety         150 non-null object
Unnamed: 5      0 non-null float64
dtypes: float64(5), object(1)
memory usage: 6.5+ KB
```

```
(150, 4)
(150,)
```

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.4,random_state=5) print(X_train.shape) print(y_train.shape) print(X_test.shape) print(y_test.shape)

```
(90, 4)
(90,)
(60, 4)
(60,)
```

km=KMeans(n_clusters=3,random_state=1) km.fit(X_train)

```
KMeans(algorithm='auto', copy_x=True, init='k-means++',
max_iter=300,
n_clusters=3, n_init=10, n_jobs=1, precompute_distances='auto',
random_state=1, tol=0.0001, verbose=0)
```

km.labels_

km.cluster_centers_

x1= data['sepal.length'] x2= data['sepal.width'] x3= data['petal.length'] x4= data['petal.width']
y=km.labels_

colors= ['b', 'g', 'r']
markers= ['o', 'v', 's']

fori, l **in**enumerate(km.labels_):

plt.plot(x1[i],x2[i], color=colors[l],marker=markers[l]) # Kmeans with arbitrary data set

importnumpy as np
importmatplotlib.pyplot as plt fromsklearn.cluster import KMeans

fromsklearn import metrics

x1 = np.array([3, 1, 1, 2, 1, 6, 6, 6, 5, 6, 7, 8, 9, 8, 9, 9, 8])
x2 = np.array([5, 4, 6, 6, 5, 8, 6, 7, 6, 7, 1, 2, 1, 2, 3, 2, 3])

plt.xlim([0, 10])

plt.ylim([0, 10]) plt.title('Dataset') plt.scatter(x1, x2) plt.show()

```
X = np.array(list(zip(x1, x2))).reshape(len(x1), 2) colors = ['b', 'g', 'r']

markers = ['o', 'v', 's']

kmeans_model = KMeans(n_clusters=3).fit(X) kmeans_model.labels_


fori, l in enumerate(kmeans_model.labels_): plt.plot(x1[i], x2[i], color=colors[l],marker=markers[l])


plt.xlim([0, 10])
plt.ylim([0, 10]) plt.title('Kmeans')

fori, l in enumerate(kmeans_model.labels_): plt.plot(x1[i], x2[i], color=colors[l],marker=markers[l])
```

Viva voce:

1. What is Clustering?
2. Explain the steps of k-Means Clustering Algorithm
3. What are some Stopping Criteria for k-Means Clustering?
4. What is the main difference between k-Means and k-Nearest Neighbours?
5. Compare Hierarchical Clustering and k-Means Clustering
6. Explain some cases where k-Means clustering fails to give good results

Experment-9 Implement Principle Component Analysis on any data set.
Aim: To Implement PCA on Clustering Problem and Justify the outcome with relevant Parameters.

Link of Problem: **[Principal Component Analysis | Kaggle]**

Objectives of PCA:

- It is basically a non-dependent procedure in which it reduces attribute space from a large number of variables to a smaller number of factors.
- PCA is basically a dimension reduction process but there is no guarantee that the dimension is interpretable.
- Main task in this PCA is to select a subset of variables from a larger set, based on which original variables have the highest correlation with the principal amount. **Principal Axis Method:** PCA basically search a linear combination of variables so that we can extract maximum variance from the variables. Once this process completes it removes it and search for another linear combination which gives an explanation about the maximum proportion of remaining variance which basically leads to orthogonal factors. In this method, we analyze total variance.

Requirement Analysis:

- Goggle CoLab (Online Compiler)
- Jupyter Notebook (Offline**)**

Hardware Requirement

- Windows 10.
- Power Supply.
- RAM-4GB

Problem statement:

Principal Component Analyis is basically a statistical procedure to convert a set of observation of possibly correlated variables into a set of values of linearly uncorrelated variables.
Each of the principal components is chosen in such a way so that it would describe most of the still available variance and all these principal components are orthogonal to each other. In all principal components first principal component has maximum variance.
Uses of PCA:

- It is used to find inter-relation between variables in the data.
- It is used to interpret and visualize data.
- As number of variables are decreasing it makes further analysis simpler.
- It’s often used to visualize genetic distance and relatedness between populations. These are basically performed on square symmetric matrix. It can be a pure sums of

squares and cross products matrix or Covariance matrix or Correlation matrix. A correlation matrix is used if the individual variance differs much.

Eigenvector: It is a non-zero vector that stays parallel after matrix multiplication. Let's suppose x is eigen vector of dimension r of matrix M with dimension r*r if Mx and x are parallel. Then we need to solve Mx=Ax where both x and A are unknown to get eigen vector and eigen values.
Under Eigen-Vectors we can say that Principal components show both common and unique variance of the variable. Basically, it is variance focused approach seeking to reproduce total variance and correlation with all components. The principal components are basically the linear combinations of the original variables weighted by their contribution to explain the variance in a particular orthogonal dimension.
Eigen Values: It is basically known as characteristic roots. It basically measures the variance in all variables which is accounted for by that factor. The ratio of eigenvalues is the ratio of explanatory importance of the factors with respect to the variables. If the factor is low then it is contributing less in explanation of variables. In simple words, it measures the amount of variance in the total given database accounted by the factor. We can calculate the factor's eigen value as the sum of its squared factor loading for all the variables.

SOLUTION:

Principal Component Analysis with Python.

To get the dataset used in the implementation, click here.

Step 1: Importing the libraries

```
# importing required libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Step 2: Importing the data set

Import the dataset and distributing the dataset into X and y components for data analysis.

```
# importing or loading the dataset
dataset = pd.read_csv('wines.csv')

# distributing the dataset into two components X and Y
X = dataset.iloc[:, 0:13].values
y = dataset.iloc[:, 13].values
```

Step 3: Splitting the dataset into the Training set and Test set

```
# Splitting the X and Y into the
# Training set and Testing set
from sklearn.model_selection import train_test_split
```

x-train, X-test, y-train, y-test = train_test_split(X, y, test-size = 0.2, random-state = 0)

Step 4: Feature Scaling

Doing the pre-processing part on training and testing set such as fitting the Standard scale.

```
# performing preprocessing part
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()

x-train = sc.fit_transform(x-train)
X-test = sc.transform(X-test)
```

Step 5: Applying PCA function

Applying the PCA function into training and testing set for analysis.

```
# Applying PCA function on training
# and testing set of X component
from sklearn.decomposition import PCA

pca = PCA(n_components = 2)

x-train = pca.fit_transform(x-train)
X-test = pca.transform(X-test)

explained_variance = pca.explained_variance_ratio_
```

Step 6: Fitting Logistic Regression To the training set

```
# Fitting Logistic Regression To the training set
from sklearn.linear_model import LogisticRegression

classifier = LogisticRegression(random-state = 0)
classifier.fit(x-train, y-train)
```

Step 7**:** Predicting the test set result

```
# Predicting the test set result using
# predict function under LogisticRegression
Y_prediction = classifier.predict(X-test)
```

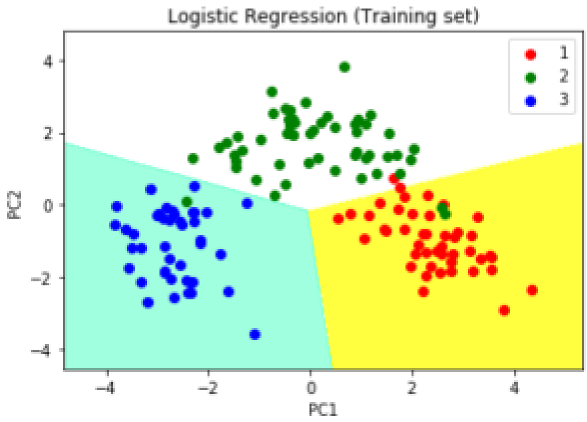
Step 8**:** Making the confusion matrix

```
# making confusion matrix between
# test set of Y and predicted value.
from sklearn.metrics import confusion_matrix

co_mat = confusion_matrix(y-test, Y_prediction)
```

Step 9: Predicting the training set result

```
# Predicting the training set
# result through scatter plot
from matplotlib.colors import ListedColormap
```

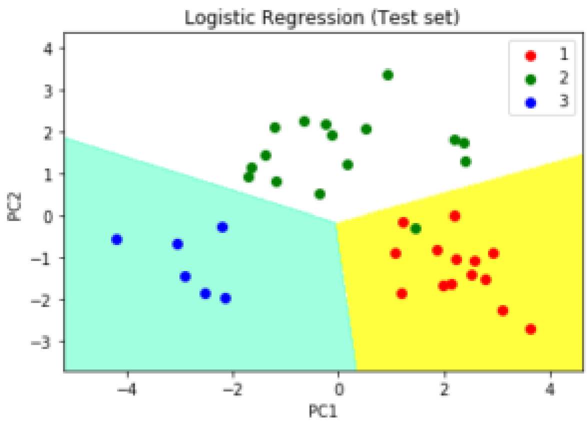
```
X_set, y_set = x-train, y-train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))

plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,
cmap = ListedColormap(('yellow', 'white', 'aquamarine')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
c = ListedColormap(('red', 'green', 'blue'))(i), label = j)

plt.title('Logistic Regression (Training set)')
plt.xlabel('PC1') # for Xlabel
plt.ylabel('PC2') # for Ylabel
plt.legend() # to show legend

# show scatter plot
plt.show()
```



Step 10**:** Visualising the Test set results

```
# Visualising the Test set results through scatter plot
from matplotlib.colors import ListedColormap

X_set, y_set = X-test, y-test

X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1,
stop = X_set[:, 0].max() + 1, step = 0.01),
np.arange(start = X_set[:, 1].min() - 1,
stop = X_set[:, 1].max() + 1, step = 0.01))
```

```
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(),
X2.ravel()]).T).reshape(X1.shape), alpha = 0.75,
cmap = ListedColormap(('yellow', 'white', 'aquamarine')))

plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())

for i, j in enumerate(np.unique(y_set)):
plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
c = ListedColormap(('red', 'green', 'blue'))(i), label = j)

# title for scatter plot
plt.title('Logistic Regression (Test set)')
plt.xlabel('PC1') # for Xlabel
plt.ylabel('PC2') # for Ylabel
plt.legend()

# show scatter plot
plt.show()
```



Viva voce:

1. Is it important to standardize the data before applying PCA?
2. How Principal Component Analysis (PCA) is used for Dimensionality Reduction?
3. Would you use PCA on large datasets or there is a better alternative?
4. What is the relationship between k-Means Clustering and PCA?
5. What are the assumptions taken into consideration while applying PCA?
6. What are the properties of Principal Components in PCA?

Experiment: 10 . Implement Association Rule Mining.

Aim: To ImplementAssociation mining model on Clustering Problem and Justify the outcome with relevant Parameters.

Link of Problem: [**Association Rule Mining | Kaggle**]
Objectives:-

- To Learn about Association Rule Mining
- To learn About Mining and analysis Techniques
- To Learn about Mining Models or algorithms based on Data Mining.

Requirement Analysis:

- Goggle CoLab (Online Compiler)
- Jupyter Notebook (Offline)

Hardware Requirement

- Windows 10.
- Power Supply.
- RAM-4GB

Problem statement:
Market Basket Analysis is one of the key techniques used by large retailers to uncover associations between items. It works by looking for combinations of items that occur together frequently in transactions. To put it another way, it allows retailers to identify relationships between the items that people buy.

Discussion: Association Rules are widely used to analyze retail basket or transaction data and are intended to identify strong rules discovered in transaction data using measures of interestingness, based on the concept of strong rule.

Algorithm Overview
This is the official pseudocode of Apriori

- **Lk:** frequent k-itemset, satisfy minimum support
- **Ck:** candidate k-itemset, possible frequent k-itemsets
-

```
L1={frequent 1-itemsets};
for (k=2; Lk-1 ≠ 0; k++) do begin
    Ck=apriori-gen(Lk-1);
    for each transactions t ∈D do begin //scan DB
     Ct=subset(Ck, t) //get the subsets of t that are candidates
     for each candidate c ∈Ct do
          c.count++;
    end
    Lk={c ∈Ck | c.count ≥ minsup}
end
Answer=∪kLk;
```

Please be aware that the pruning step is already included in the apriori-gen function.

Personally, I found this pseudocode quite confusing. So, I organized it into my own version. It should be way easier to understand.

```
L[1] = {frequent 1-itemsets};
for (k=2; L[k-1] != 0; k ++) do begin
    // perform self-joining
    C[k] = getUnion(L[k-1])
    // remove pruned supersets
    C[k] = pruning(C[k])
    // get itemsets that satisfy minSup
    L[k] = getAboveMinSup(C[k], minSup)
end
Answer = Lk (union)
```

To sum up, the basic components of Apriori can be written as

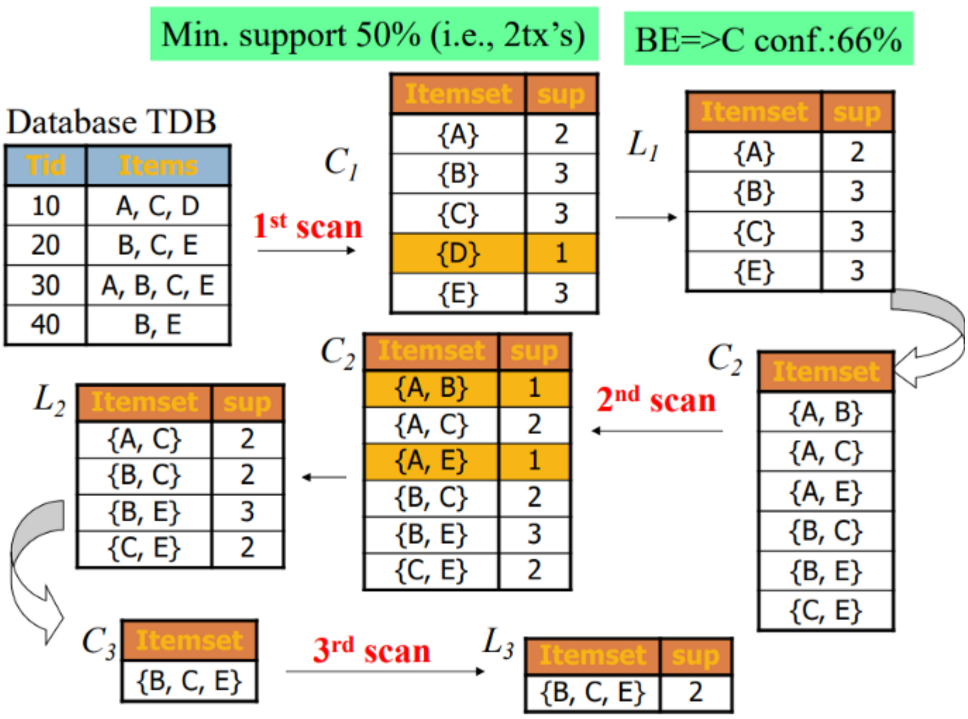
Use k-1 itemsets to generate k itemsets

Getting C[k] by joining L[k-1] and L[k-1]

Prune C[k] with subset testing

Generate L[k] by extracting the itemsets in C[k] that satisfy minSup

Simulate the algorithm in your head and validate it with the example below. The concept should be really clear now.



Python Implementation

Apriori Function

This is the main function of this Apriori Python implementation. The most important part of this function is from **line 16 ~ line 21**. It basically follows my modified pseudocode written above.

1. Generate the candidate set by joining the frequent itemset from the previous stage.
2. Perform subset testing and prune the candidate set if there's an infrequent itemset contained.
3. Calculate the final frequent itemset by getting those satisfy minimum support.

```
def
apriori(itemSetList,
minSup, minConf):
        C1ItemSet = getItemSetFromList(itemSetList)
        # Final result, global frequent itemset
        globalFreqItemSet = dict()
        # Storing global itemset with support count
        globalItemSetWithSup = defaultdict(int)
        L1ItemSet = getAboveMinSup(C1ItemSet, itemSetList, minSup,
    globalItemSetWithSup)
        currentLSet = L1ItemSet
        k = 2
        # Calculating frequent item set
        while(currentLSet):
            # Storing frequent itemset
            globalFreqItemSet[k-1] = currentLSet
            # Self-joining Lk
            candidateSet = getUnion(currentLSet, k)
            # Perform subset testing and remove pruned supersets
            candidateSet = pruning(candidateSet, currentLSet, k-1)
            # Scanning itemSet for counting support
            currentLSet = getAboveMinSup(candidateSet, itemSetList, minSup,
    globalItemSetWithSup)
            k += 1
        rules = associationRule(globalFreqItemSet, globalItemSetWithSup,
    minConf)
        rules.sort(key=lambda x: x[2])
        return globalFreqItemSet, rules
```

Candidate Generation

For self-joining, we simply get all the union through brute-force and only return those are in the specific length.

```
def
getUnion(itemSet,
```

```
length):
                    return set([i.union(j) for i in itemSet for j in itemSet if
                len(i.union(j)) == length])
```

Pruning

To perform subset testing, we loop through all possible subsets in the itemset. If the subset is not in the previous frequent itemset, we prune it.

```
def
pruning(candidateSet,
prevFreqSet, length):
                        tempCandidateSet = candidateSet.copy()
                        for item in candidateSet:
                            subsets = combinations(item, length)
                            for subset in subsets:
                                # if the subset is not in previous K-frequent get, then
                    remove the set
                                if(frozenset(subset) not in prevFreqSet):
                                    tempCandidateSet.remove(item)
                                    break
                        return tempCandidateSet
```

Get Frequent Itemset from Candidate

In the final step, we **turn the candidate sets into frequent itemsets**. Since we are not applying any improvement technique. The only approach we can go for is to brainlessly loop through the item and itemset over and over again to obtain the count. At last, we only retain the itemsets whose support is equal or higher than minimum support.

```
def
getAboveMinSup(itemSet,
itemSetList, minSup,
globalItemSetWithSup):
                        freqItemSet = set()
                        localItemSetWithSup = defaultdict(int)
                        for item in itemSet:
                            for itemSet in itemSetList:
                                if item.issubset(itemSet):
                                    globalItemSetWithSup[item] +=
                    1
                                    localItemSetWithSup[item] +=
                    1
                        for item, supCount in
                    localItemSetWithSup.items():
                            support = float(supCount /
                    len(itemSetList))
```

```
                                        if(support >= minSup):
                                            freqItemSet.add(item)
                                    return freqItemSet
```

Result

```
print(rules)
#  [[{'beer'},  {'rice'},  0.666],  [{'rice'},  {'beer'},  1.000]]
# (rules[0] --> rules[1]), confidence = rules[2]
```

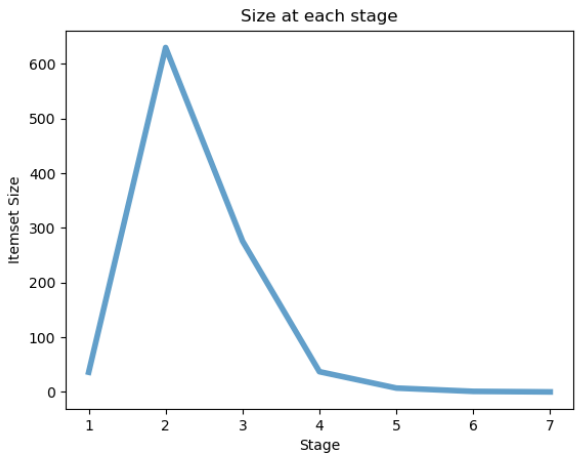
Shortcomings

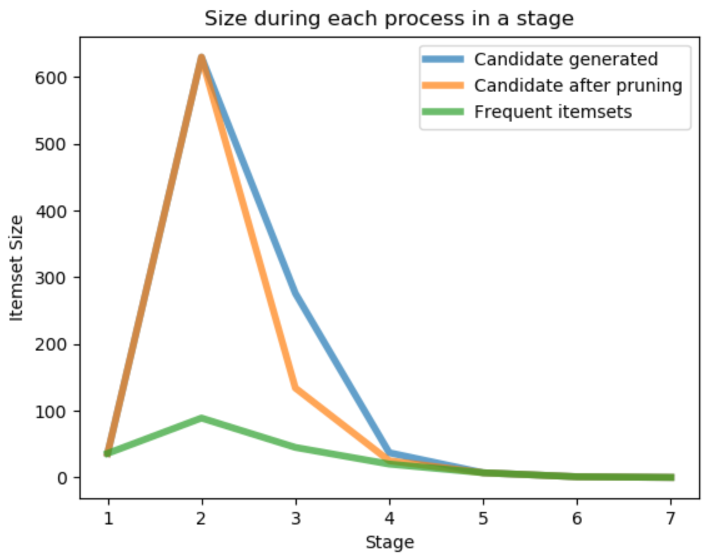
There are two major shortcomings of Apriori Algorithms

- The size of itemset from candidate generation could be extremely large
- Lots of time wasted on counting the support since we have to scan the itemset database over and over again

We will use the data4.csv(generated from IBM generator) in the repo to showcase these shortcomings and see if we can get some interesting observations.

Candidate itemsets size at each stage



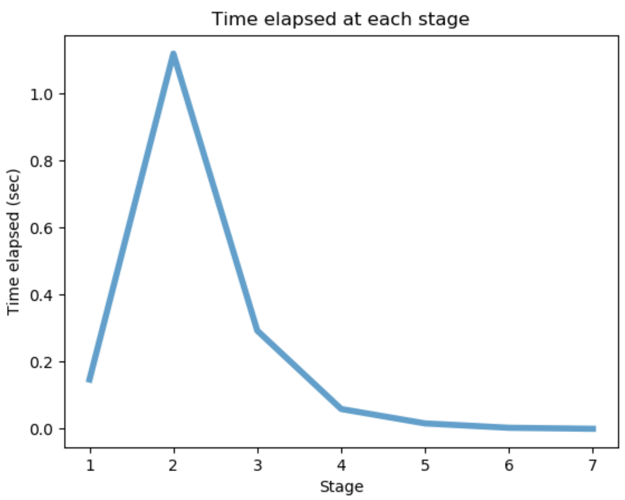


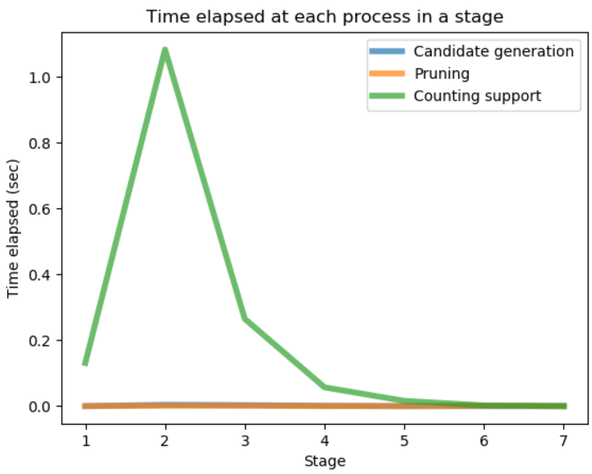
By running Apriori on data4.csv, we can plot the process like the graph above. The shortcomings we mentioned above can be found in the observation of the graphs.

On the right, we can see the itemset size after the three major processes of the algorithm. Two key points can be discovered from the graph

- Size of itemset rapidly increase at the beginning, and gradually decrease as the iteration goes on
- Pruning process may be useless like stage 1 and 2. However, it could help a lot at some cases like stage 3. Half of the itemset is pruned, which means the counting time could be decreased by half!

Time elapsed at each stage





From the plot, we can tell that most of the running is spent on counting the support. The time spent on candidate generation and pruning is nothing comparing to scanning the original itemset database over and over again.

Another observation worth attention is that we get a peak in cost at **stage 2**. Interestingly, This is actually not an accident! Data scientists often meet a bottleneck at stage 2 when using Apriori. Since there are almost no candidates removed at stage 1, the candidates generated at stage 2 are basically all possible combinations of all 1-frequent itemsets. And calculating the support of such a huge itemset leads to extremely high costs.

Viva voce:

1. Define support and confidence in Association rule mining.
2. How are association rules mined from large databases?
3. What is Association rule?
4. What are the Applications of Association rule mining?
5. What are the algorithms to deal with it
6. What type problems lie under it?