SINHGAD TECHNICAL EDUCATION SOCIETY'S

SINHGAD INSTITUTE OF TECHNOLOGY

Kusgaon (Bk), Lonavala 410401

DEPARTMENT OF INFORMATION TECHNOLOGY

LABORATORY MANUAL V-1.0

Laboratory Practices-I

Part 1: Machine Learning [314448]

T.E. IT (SEM –I) (2019 Course)

AY 2022-23

Developed By

Prof. Vandana P. Tonde

TEACHING SCHEME	EXAMINATION SCHEME
Practical: 4 Hrs/Week	Term Work: 25Marks
	Practical: 25Marks

Vision and Mission of Institute

VISION

उत्तमपुरुषान् उत्तमाभियंतॄन् निर्मातुं कटिबद्धाःवयम्

“We are committed to produce not only good engineers but good human beings, also.”

MISSION

- We believe in and work for the holistic development of students and teachers.
- We strive to achieve this by imbibing a unique value system, transparent work culture, excellent academic and physical environment conducive to learning, creativity and technology transfer.

Vision and Mission of the Department

VISION

To provide excellent Information Technology education by building teaching and research environment.

MISSION

1) To transform the students into innovative, competent and high quality IT professionals to meet the growing global challenges.
2) To achieve and impart quality education with an emphasis on practical skills and social relevance.
3) To endeavour for continuous up-gradation of technical expertise of students to cater to the needs of the society.
4) To achieve an effective interaction with industry for mutual benefits.

Program Educational Objectives (PEO’s)

PEO1	Possess strong fundamental concepts in mathematics, science, engineering and Technology to address technological challenges.
PEO2	Possess knowledge and skills in the field of Computer Science and Information Technology for analyzing, designing and implementing complex engineering problems of any domain with innovative approaches.
PEO3	Possess an attitude and aptitude for research, entrepreneurship and higher studies in the field of Computer Science and Information Technology.
PEO4	Have commitment ethical practices, societal contributions through communities and life-long learning.
PEO5	Possess better communication, presentation, time management and team work skills leading to responsible & competent professional sand will be able to address challenges in the field of IT at global level.

Program Outcomes: POs

PO1	**Engineering knowledge**	An ability to apply knowledge of mathematics, computing, science, engineering and technology.
PO2	**Problem analysis**	An ability to define a problem and provide a systematic solution with the help of conducting experiments, analyzing the problem and interpreting the data.
PO3	**Design / Development of Solutions**	An ability to design, implement, and evaluate software or a software /hardware system, component, or process to meet desired need switch in realistic constraints.
PO4	**Conduct Investigation of Complex Problems**	An ability to identify, formulates, and provide essay schematic solutions to complex engineering /Technology problems.
PO5	**Modern Tool Usage**	An ability to use the techniques, skills, and modern engineering technology tools, standard processes necessary for practice as a IT professional.
PO6	**The Engineer and Society**	An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer- based systems with necessary constraints and assumptions.
PO7	**Environment and Sustainability**	An ability to analyze and provide solution for the local and global impact of information technology on individuals, organizations and society.
PO8	**Ethics**	An ability to understand professional, ethical, legal, security and social issues and responsibilities.
PO9	**Individual and Team Work**	An ability to function effectively as an individual or as a team member to accomplish a desired goal(s).
PO10	**Communication Skills**	An ability to engage in life-long learning and continuing professional development to cope up with fast changes in the technologies /tools with the help of electives, profession along animations and extra- curricular activities.
PO11	**Project Management and Finance**	An ability to communicate effectively in engineering community at large by means of effective presentations, report writing, paper publications, demonstrations.
PO12	**Life-long Learning**	An ability to understand engineering, management, financial aspects, performance, optimizations and time complexity necessary for professional practice.

Program Specific Outcomes: PSOs

PSO1	An ability to apply the theoretical concepts and practical knowledge of Information Technology in analysis, design, development and management of information processing systems and applications in the interdisciplinary domain.
PSO2	An ability to analyze a problem, and identify and define the computing infrastructure and operations requirements appropriate to its solution. IT graduates should be able to work on large-scale computing systems.
PSO3	An understanding of professional, business and business processes, ethical, legal, security and social issues and responsibilities.
PSO4	Practice communication and decision-making skills through the use of appropriate technology and be ready for professional responsibilities.

Prerequisites:

1. Python programming language

Course Description :

Machine Learning is concerned with computer programs that automatically improve their performance through experience. This course covers the theory and practical algorithms for machine learning from a variety of perspectives. We cover topics such as introduction to Python programming, Classification ,Linear Regression , Decision tree (ID3 Algorithm), Naïve Bayesian classifier, Bayesian Network, k-Means Algorithm, k-Nearest Neighbor Algorithm.

Course Objectives:

1. The objective of this course is to provide students with the fundamental elements of machine learning for classification, regression, clustering.
2. Design and evaluate the performance of a different machine learning models.

Course Outcomes:

On completion of the course, students will be able to–
CO1: Implement different supervised and unsupervised learning algorithms.
CO2: Evaluate performance of machine learning algorithms for real-world applications. .

CERTIFICATE

This is to certify that Mr. /Ms ____ of class TEIT Div _____ Roll No._____ Examination Seat No./PRN No.___________________________ has completed all the practical work in the **Laboratory Practices-I(Machine Learning)** satisfactorily, as prescribed by Savitribai Phule Pune University , Pune in academic year 2022 - 23 (Semester I).

Course In-charge Head of Department Principal

Date:

INDEX [LP-I Lab ML]

SN	Title of experiment	Date of Submission	Marks Obtained (10)	Sign of Faculty
1	**Data preparation:** Download heart dataset from following link. https://www.kaggle.com/zhaoyingzhu/heartcsv Perform following operation on given dataset. a) Find Shape of Data b) Find Missing Values c) Find data type of each column d) Finding out Zero's e) Find Mean age of patients f) Now extract only Age, Sex, ChestPain, RestBP, Chol. Randomly divide dataset in training (75%) and testing (25%).			
2	**Assignment on Regression technique** Apply Linear Regression using suitable library function and predict the Month-wise Download temperature data from below link. https://www.kaggle.com/venky73/temperatures- of-india? select=temperatures.csv			
3	**Assignment on Classification technique** Data Set : https://www.kaggle.com/mohansacharya/graduate-admissions A. Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary. B. Perform data-preparation (Train-Test Split) C. Apply Machine Learning Algorithm D. Evaluate Model.			
4	**Assignment on Improving Performance of Classifier Models** A SMS unsolicited mail (every now and then known as cell smartphone junk mail) is any junk message brought to a cellular phone as textual content messaging via the Short Message Service (SMS). Use probabilistic approach (Naive Bayes Classifier / Bayesian Network) to implement SMS Spam Filtering system. SMS messages are categorized as SPAM or HAM using features like length of message, word depend, unique keywords etc. Download Data -Set from : http://archive.ics.uci.edu/ml/datasets/sms+spam+collection This dataset is composed by just one text file, where			

	each line has the correct class followed by the raw message. **A .**Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary B. Perform data-preparation (Train-Test Split) C. Apply at least two Machine Learning Algorithms and Evaluate Models D.Apply Cross-Validation and Evaluate Models and compare performance. E. Apply Hyper parameter tuning and evaluate models and compare performance.			
5	**Assignment on Clustering Techniques** Download the following customer dataset from below link: Data Set: https://www.kaggle.com/shwetabh123/mall-customers This dataset gives the data of Income and money spent by the customers visiting a Shopping Mall. The data set contains Customer ID, Gender, Age, Annual Income, and Spending Score. Therefore, as a mall owner you need to find the group of people who are the profitable customers for the mall owner. Apply at least two clustering algorithms (based on Spending Score) to find the group of customers. Apply Data pre-processing (Label Encoding , Data Transformation….) techniques if necessary. Perform data-preparation(Train-Test Split) Apply Machine Learning Algorithm Evaluate Model. Apply Cross-Validation and Evaluate Model			
6	**Assignment on Association Rule Learning** Download Market Basket Optimization dataset from below link. DataSet: https://www.kaggle.com/hemanthkumar05/market-basket-optimization This dataset comprises the list of transactions of a retail company over the period of one week. It contains a total of 7501 transaction records where each record consists of the list of items sold in one transaction. Using this record of transactions and items in each transaction, find the association rules between items. There is no header in the dataset and the first row			

	contains the first transaction, so mentioned header = None here while loading dataset. **A.** Follow following steps : **B.** Data Pre-processing **C.** Generate the list of transactions from the dataset **D.** Train Apriori algorithm on the dataset **E.** Visualize the list of rules **F.** Generated rules depend on the values of hyper parameters. By increasing the minimum confidence value and find the rules accordingly			
7	**Assignment on Multilayer Neural Network Model** Download the dataset of National Institute of Diabetes and Digestive and Kidney Diseases from below link : Data Set: https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians- diabetes.data.csv The dataset is has total 9 attributes where the last attribute is "Class attribute" having values 0 and 1. (1="Positive for Diabetes", 0="Negative") Load the dataset in the program. Define the ANN Model with Keras. Define at least two hidden layers. Specify the ReLU function as activation function for the hidden layer and Sigmoid for the output layer. Compile the model with necessary parameters. Set the number of epochs and batch size and fit the model. Evaluate the performance of the model for different values of epochs and batch sizes. Evaluate model performance using different activation functions Visualize the model using ANN Visualizer.			

Name & Signature of Course In-charge

INTRODUCTION TO LAB:

Machine Learning is used anywhere from automating mundane tasks to offering intelligent insights, industries in every sector try to benefit from it. You may already be using a device that utilizes it. For example, a wearable fitness tracker likes Fitbit, or an intelligent home assistant like Google Home. But there are much more examples of ML in use.

- **Prediction:** Machine learning can also be used in the prediction systems. Considering the loan example, to compute the probability of a fault, the system will need to classify the available data in groups.
- **Image recognition**: Machine learning can be used for face detection in an image as well. There is a separate category for each person in a database of several people.
- **Speech Recognition:** It is the translation of spoken words into the text. It is used in voice searches and more. Voice user interfaces include voice dialing, call routing, and appliance control. It can also be used a simple data entry and the preparation of structured documents.
- **Medical diagnoses:** ML is trained to recognize cancerous tissues.
- **Financial industry:** and trading companies use ML in fraud investigations and credit checks.

Types of Machine Learning

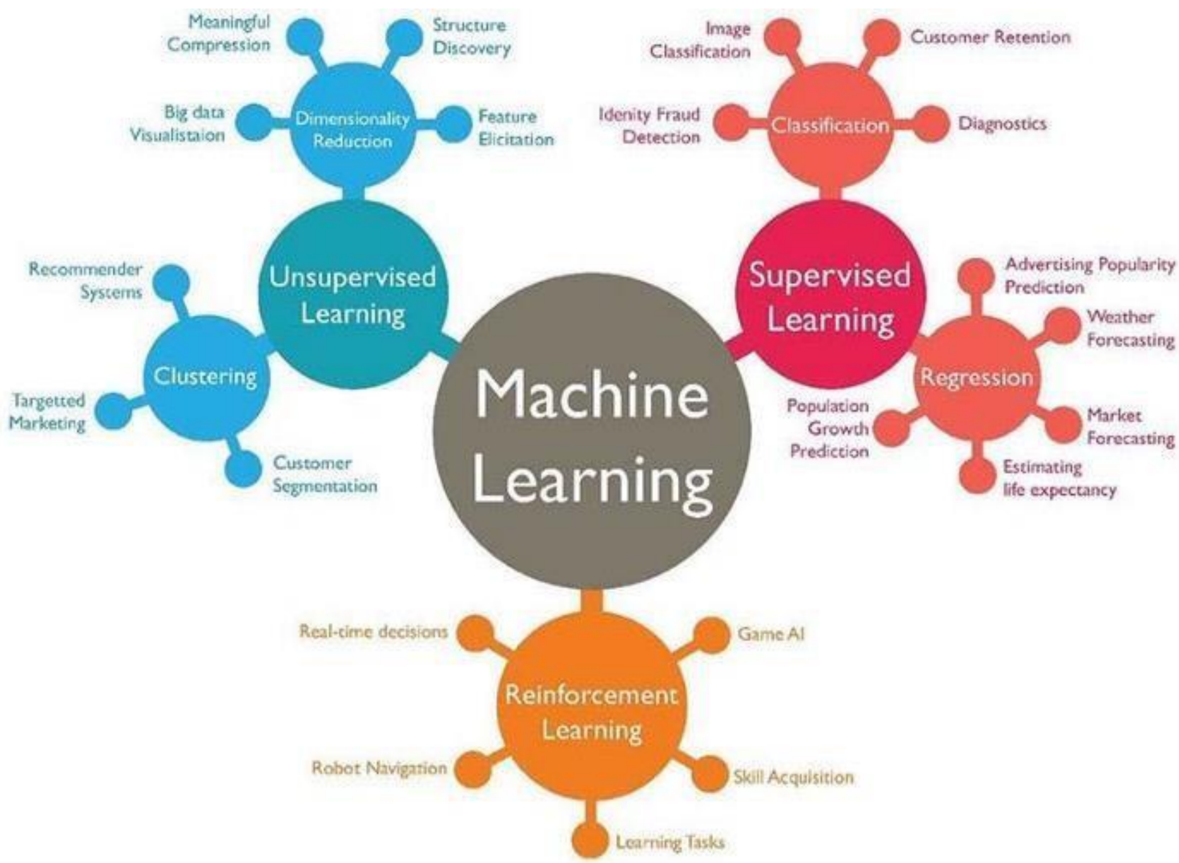
Machine learning can be classified into 3 types of algorithms

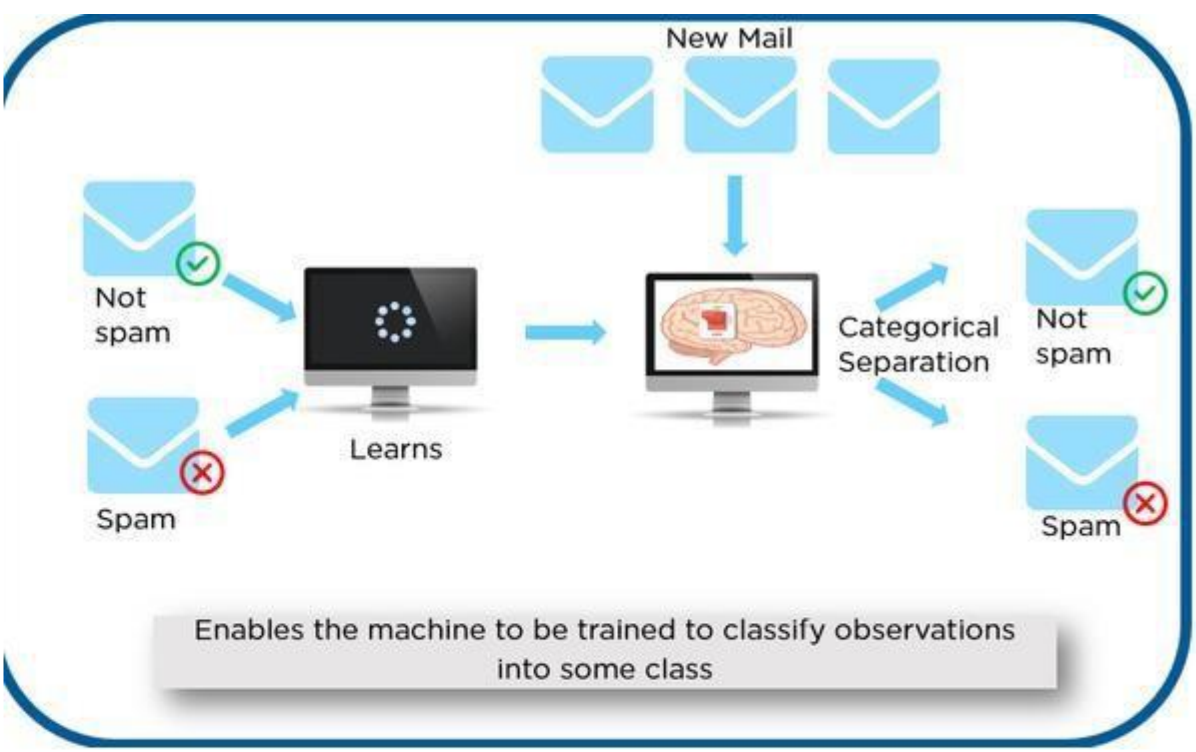
1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

Overview of Supervised Learning Algorithm

In Supervised learning, an AI system is presented with data which is labeled, which means that each data tagged with the correct label

The goal is to approximate the mapping function so well that when you have new input data (x) that you can predict the output variables (Y) for that data


Machine Learning
Unsupervised Learning
Dimensionality Reduction
Meaningful Compression
Structure Discovery
Big data Visualistaion
Feature Elicitation
Clustering
Recommender Systems
Targetted Marketing
Customer Segmentation
Supervised Learning
Classification
Image Classification
Customer Retention
Idenity Fraud Detection
Diagnostics
Regression
Advertising Popularity Prediction
Weather Forecasting
Population Growth Prediction
Market Forecasting
Estimating life expectancy
Reinforcement Learning
Real-time decisions
Game AI
Robot Navigation
Skill Acquisition
Learning Tasks


New Mail
Not spam
Spam
Learns
Categorical Separation
Not spam
Spam
Enables the machine to be trained to classify observations into some class

As shown in the above example, we have initially taken some data and marked them as ‘Spam’ or ‘Not Spam’. This labeled data is used by the training supervised model, this data is used to train the model.

Once it is trained we can test our model by testing it with some test new mails and checking of the model is able to predict the right output.

Types of Supervised learning

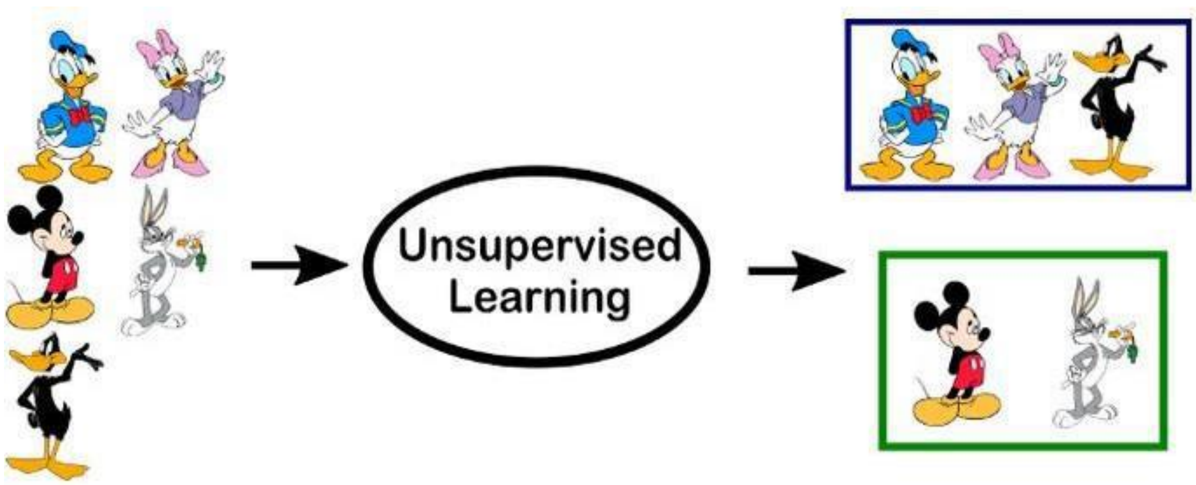
- **Classification**: A classification problem is when the output variable is a category, such as “red” or “blue” or “disease” and “no disease”.
- **Regression**: A regression problem is when the output variable is a real value, such as “dollars” or “weight”.

Overview of Unsupervised Learning Algorithm

In unsupervised learning, an AI system is presented with unlabeled, uncategorized data and the system’s algorithms act on the data without prior training. The output is dependent upon the coded algorithms. Subjecting a system to unsupervised learning is one way of testing AI.

Types of Unsupervised learning:

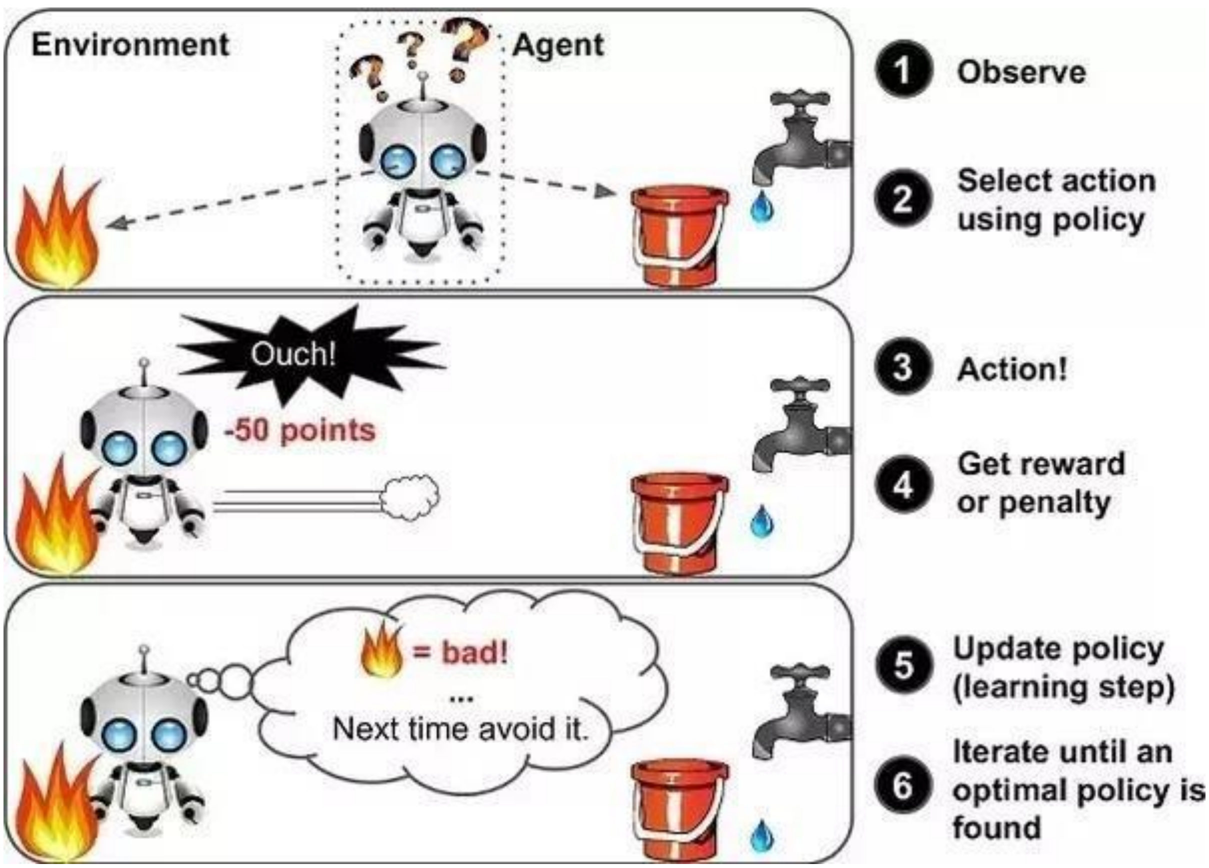
- **Clustering**: A clustering problem is where you want to discover the inherent groupings in the data, such as grouping customers by purchasing behavior.
- **Association**: An association rule learning problem is where you want to discover rules that describe large portions of your data, such as people that buy X also tend to buy Y.



Example of Unsupervised Learning

Overview of Reinforcement Learning

A reinforcement learning algorithm, or agent, learns by interacting with its environment. The agent receives rewards by performing correctly and penalties for performing incorrectly. The agent learns without intervention from a human by maximizing its reward and minimizing its penalty. It is a type of dynamic programming that trains algorithms using a system of reward and punishment.



in the above example, we can see that the agent is given 2 options i.e. a path with water or a path with fire. A reinforcement algorithm works on reward a system i.e. if the agent uses the fire path then the rewards are subtracted and agent tries to learn that it should avoid the fire path. If it had chosen the water path or the safe path then some points would have been added to the reward points, the agent then would try to learn what path is safe and what path isn't.

It is basically leveraging the rewards obtained; the agent improves its environment knowledge to select the next action.

Assignment No.01: Data Preparation

Sinhgad Institutes

Name of the Student: ______________________________________ Roll no: ___

CLASS: - T.E. IT **Subject Name: - LP-I Lab (Part I- ML)**

Experiment No. 01

** Data Preparation: **

Marks: /10

Date of Performance: / /2022 **Sign with Date:**

Problem Statement:

Download heart dataset from following link.

https://www.kaggle.com/zhaoyingzhu/heart.csv

Perform following operation on given dataset.
a) Find Shape of Data
b) Find Missing Values
c) Find data type of each column
d) Finding out Zero's
e) Find Mean age of patients
f) Now extract only Age, Sex, ChestPain, RestBP, Chol. Randomly divide dataset in training (75%) and testing (25%).

Through the diagnosis test I predicted 100 report as COVID positive, but only 45 of those were actually positive. Total 50 people in my sample were actually COVID positive. I have total 500 samples.
Create confusion matrix based on above data and find
I Accuracy
II Precision
III Recall
IV F-1 score

Theory:

Data Preparation: It is the process of transforming raw data into a particular form so that data scientists and analysts can run it through machine learning algorithms to uncover insights or make predictions. All projects have the same general steps; they are:
Step 1: Define Problem.
Step 2: Prepare Data.
Step 3: Evaluate Models.

Step 4: Finalize Model.
We are concerned with the data preparation step (step 2), and there are common or standard tasks that you may use or explore during the data preparation step in a machine learning project.

Data Preparation Tasks

1. **Data Cleaning:** There are many reasons data may have incorrect values, such as being mistyped, corrupted, duplicated, and so on. Domain expertise may allow obviously erroneous observations to be identified as they are different from what is expected.

2. **Feature Selection:** Feature selection refers to techniques for selecting a subset of input features that are most relevant to the target variable that is being predicted. Feature selection techniques are generally grouped into those that use the target variable (**supervised**) and those that do not (**unsupervised**). Additionally, the supervised techniques can be further divided into models that automatically select features as part of fitting the model (**intrinsic**), those that explicitly choose features that result in the best performing model (**wrapper**) and those that score each input feature and allow a subset to be selected (**filter**).

3. **Data Transforms**: Data transforms are used to change the type or distribution of data variables.

   - **Numeric Data Type:** Number values.
   - **Integer:** Integers with no fractional part.
   - **Real:** Floating point values.
   - **Categorical Data Type:** Label values.
   - **Ordinal:** Labels with a rank ordering.
   - **Nominal:** Labels with no rank ordering.
   - **Boolean:** Values True and False.

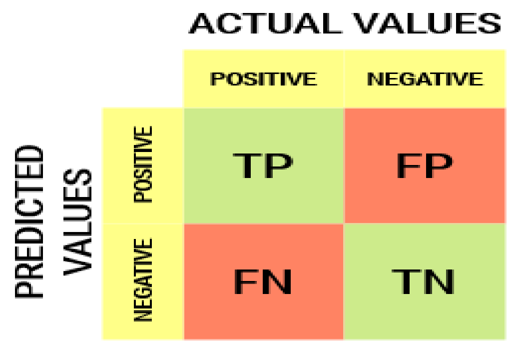
4. **Feature Engineering:** Feature engineering refers to the process of creating new input variables from the available data. Engineering new features is highly specific to your data and data types. As such, it often requires the collaboration of a subject matter expert to help identify new features that could be constructed from the data.

5. **Dimensionality Reduction:** The number of input features for a dataset may be considered the dimensionality of the data. This motivates feature selection, although an alternative to feature selection is to create a projection of the data into a lower-dimensional space that still preserves the most important properties of the original data. The most common approach to dimensionality reduction is to use a matrix factorization technique:

   - Principal Component Analysis (PCA)
   - Linear Discriminant Analysis (LDA)

Confusion Matrix

A Confusion matrix is an N x N matrix used for evaluating the performance of a classification model, where N is the number of target classes. The matrix compares the actual target values with those predicted by the machine learning model.



The explanation of the terms associated with confusion matrix is as follows −

- **True Positives (TP)** − It is the case when both actual class & predicted class of data point is 1.
- **True Negatives (TN) −** It is the case when both actual class & predicted class of data point is 0.
- **False Positives (FP) −** It is the case when actual class of data point is 0 & predicted class of datapoint is 1.
- **False Negatives (FN) −** It is the case when actual class of data point is 1 & predicted class ofdata point is 0.

Algorithm:-

Step # 1 *Importing the required libraries.*

Step # 2 *Loading the dataset.*

Step # 3 *Let's get some useful information about dataset. Applying pandas "info ()" function*

Step #4 for *Applying pandas "shape () and size ()" function.*

Step # 5 *Let's check for useful descriptive statistical values. Applying pandas "describe ()" function*

Step # 6 *Let's check for not NULL values in the data set*

Step # 7 *Printing values.*

Step # 8 *Finding mean of age of patients.*

Step # 9 *Finding zeros.*

Step # 10 *Now extracting only Age, Sex, ChestPain, RestBP, Chol. And dividing dataset in training (75%) and testing (25%).*

Python Code:-

Step # 1 Importing the required libraries

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import hvplot.pandas
from scipy import stats

%matplotlib inline
sns.set_style("whitegrid")
plt.style.use("fivethirtyeight")
```

Step #2 Loading the Dataset

```
data = pd.read_csv("E:/2021_22 Sem I/ML TE IT/Datasets/heart.csv")
data.head()
```

Output 1:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Step#3 Let's get the some useful information about dataset. Applying pandas "info()" function

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

Step#4 For Shape of dataset applying pandas "shape() and size ()" function

```
data=pd.read_csv('../input/heartcsv/Heart.csv')
```

```
data.shape
```

```
(303, 14)
```

data.size

```
4545
```

Step#5 Let’s check for useful descriptive statistical values Applying pandas “describe()” function

data.describe()

Out[5]:

	Unnamed: 0	Age	Sex	RestBP	Chol	Fbs	RestECG	MaxHR	ExA
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303
mean	152.000000	54.438944	0.679868	131.689769	246.693069	0.148515	0.990099	149.607261	0.32
std	87.612784	9.038662	0.467299	17.599748	51.776918	0.356198	0.994971	22.875003	0.46
min	1.000000	29.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.00
25%	76.500000	48.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.00
50%	152.000000	56.000000	1.000000	130.000000	241.000000	0.000000	1.000000	153.000000	0.00
75%	227.500000	61.000000	1.000000	140.000000	275.000000	0.000000	2.000000	166.000000	1.00
max	303.000000	77.000000	1.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.00

Step#6 Let’s check for not NULL values in the dataset.

```
data=pd.read_csv('../input/heartcsv/Heart.csv')
data.notnull()
```

ut[4]:

	Unnamed: 0	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
0	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
1	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
2	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
3	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
4	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
299	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
300	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
301	True	True	True	True	True	True	True	True	True	True	True	True	True	True	True
302	True	True	True	True	True	True	True	True	True	True	True	True	False	True	True

303 rows × 15 columns

Step#7 Prining values.

```
data=pd.read_csv('../input/heartcsv/Heart.csv')
data.values
```

```
df.values
```

```
array([[1, 63, 1, ..., 0.0, 'fixed', 'No'],
       [2, 67, 1, ..., 3.0, 'normal', 'Yes'],
       [3, 67, 1, ..., 2.0, 'reversable', 'Yes'],
       ...,
       [301, 57, 1, ..., 1.0, 'reversable', 'Yes'],
       [302, 57, 0, ..., 1.0, 'normal', 'Yes'],
       [303, 38, 1, ..., nan, 'normal', 'No']], dtype=object)
```

Step#8 Finding mean of age of patients.

```
data["Age"].mean()
```

```
ut[6]:
54.43894389438944
```

Step#9 Finding zeros

```
data=pd.read_csv('../input/heartcsv/Heart.csv')
data.isin([0]).any()
(data==0).sum()
```

```
:[7]:
True
```

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

Step# 10 Now extracting only Age, Sex, ChestPain, RestBP, Chol. And dividing dataset in training (75%) and testing (25%)

```
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn.model_selection import train_test_split

data = pd.read_csv('../input/heartcsv/Heart.csv')

X = data[['ChestPain', 'Age', 'Sex', 'RestBP', 'Chol']]
y = data[['RestBP', 'Chol']]
X_train, X_test, y_train, y_text = train_test_split(X, y, test_size=0.25,
random_state=10)
X_train
```

	ChestPain	Age	Sex	RestBP	Chol	Fbs	RestECG
280	asymptomatic	57	1	110	335	0	0
235	asymptomatic	54	1	122	286	0	2
260	nonanginal	44	0	118	242	0	0
76	asymptomatic	60	1	125	258	0	2
275	typical	64	1	170	227	0	2
...	...	...	...	...	...	...	...
156	asymptomatic	51	1	140	299	0	0
123	asymptomatic	55	1	140	217	0	0
15	nonanginal	57	1	150	168	0	0
125	nontypical	45	0	130	234	0	2
265	asymptomatic	42	1	136	315	0	0

227 rows × 7 columns

Conclusion: Thus we have studied different data preparation techniques.

Assignment No.02: Regression

Sinhgad Institutes

Name of the Student: ______________________________ **Roll no:** ___

CLASS: - T.E. IT **Subject Name: - LP-I Lab (Part I- ML)**

Experiment No. 02

** Regression Techniques: **

Marks: /10

Date of Performance: / /2022 **Sign with Date:**

Problem Statement:-

Download temperature data from below link.
https://www.kaggle.com/venky73/temperatures- of-india?select=temperatures.csv
This data consists of temperatures of INDIA averaging the temperatures of all places month wise. Temperatures values are recorded in CELSIUS

a. Apply Linear Regression using suitable library function and predict the Month-wise temperature.
b. Assess the performance of regression models using MSE, MAE and R-Square metrics
c. Visualize simple regression model.

Theory:

Regression:

Regression is a supervised learning technique which helps in finding the correlation between variables and enables us to predict the continuous output variable based on the one or more predictor variables. It is mainly used for prediction, forecasting, time series modeling and determining the causal-effect relationship between variables. In Regression, we plot a graph between the variables which best fits the given datapoints, using this plot, the machine learning model can make predictions about the data.

Terminologies Related to the Regression Analysis:

Dependent Variable: The main factor in Regression analysis which we want to predict or understand is called the dependent variable. It is also called target variable.

Independent Variable: The factors which affect the dependent variables or which are used to predict the values of the dependent variables are called independent variable, also called as a predictor.

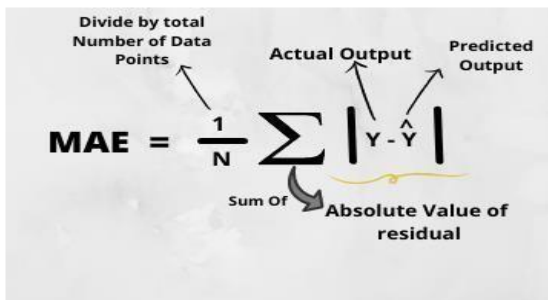
Outliers: Outlier is an observation which contains either very low value or very high value in comparisonto other observed values. An outlier may hamper the result, so it should be avoided.

Multicollinearity: If the independent variables are highly correlated with each other than other variables, then such condition is called Multicollinearity. It should not be present in the dataset, because it creates problem while ranking the most affecting variable.

Underfitting and Overfitting: If our algorithm works well with the training dataset but not well withtest dataset, then such problem is called Overfitting. And if our algorithm does not perform well even with training dataset, then such problem is called underfitting.

Cost Functions:

1. **Mean Absolute Error (MAE**): MAE is a very simple metric which calculates the absolute difference between actual and predicted values.



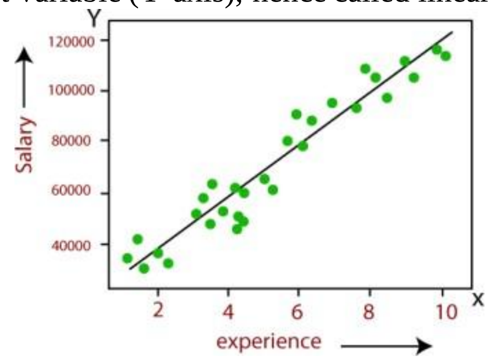
2. **Mean Squared Error(MSE):** Mean squared error states that finding the squared difference between actual and predicted value. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

$$MSE = \frac{1}{n} \Sigma \left(\underbrace{y - \widehat{y}}_{\text{The square of the difference between actual and predicted}} \right)^2$$

3. **Root Mean Squared Error(RMSE):** As RMSE is clear by the name itself, that it is a simplesquare root of mean squared error.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)}$$

Linear Regression: Linear regression is a statistical regression method which is used for predictive analysis. It is one of the very simple and easy algorithms which works on regression and shows the relationship between the continuous variables. It shows the linear relationship between the independent variable (X-axis) and the dependent variable (Y-axis), hence called linear regression.



Below is the mathematical equation for Linear regression:Y= aX+b

Here, Y= Independent Variable (Target Variable), X= Dependent Variable (Predictor Variable)

Steps in Linear Regression:

1. Loading the Data
2. Exploring the Data
3. Slicing The Data
4. Train and Split Data
5. Generate The Model
6. Evaluate The accuracy

Algorithm:-

Step # 1 Importing the required libraries

Step # 2 Loading the dataset

Step # 3 Let’s check for useful descriptive statistical values. Applying pandas “describe()” function

Step # 4 Let’s check for any missing or NA values in the training and testing data set

Step # 5 Let’s drop the record with missing value in the training dataset. As it is only one record, removing it will not be much of concern.

Step # 6 Let’s define our dependent and independent variable for training and testing data

Step # 7 Let’s split the dataset into two sub datasets “Training” and “Testing” Dataset.

Step # 8 Let’s define the model and fit it.

Step # 9 Let’s look at different parameters of the model summary and interpret it:

Step # 10: Now let’s visualise the regression equation fitment on the data

Step # 11: *Now let's check how our model is doing on the testing data, which we kept aside for testing our model performance*

Step # 12: *Assess the performance of regression models using MSE,* MAE and R-Square metrics

Python Code:-

Step # 1 Importing the required libraries

```
import numpy as np                  // for Numeric Operations
import pandas as pd                  //For Dataframe Operations
import matplotlib.pyplot as plt   // For Plotting and Visualization
From sklearn_linear.model import LinearRegression //sklearn implementation of LinearRegress
```

Step#2 Loading the dataset

```
trainData = pd.read_csv("E:/2021_22 Sem I/ML TE IT/Datasets/temperatures.csv")
```

```
trainData.head(n=10)
```

Output1:

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP	OCT	NOV	DEC	ANNUAL	JAN-FEB	MAR-MAY	JUN-SEP
0	1901	22.40	24.14	29.07	31.91	33.41	33.18	31.21	30.39	30.47	29.97	27.31	24.49	28.96	23.27	31.46	31.27
1	1902	24.93	26.58	29.77	31.78	33.73	32.91	30.92	30.73	29.80	29.12	26.31	24.04	29.22	25.75	31.76	31.09
2	1903	23.44	25.03	27.83	31.39	32.91	33.00	31.34	29.98	29.85	29.04	26.08	23.65	28.47	24.24	30.71	30.92
3	1904	22.50	24.73	28.21	32.02	32.64	32.07	30.36	30.09	30.04	29.20	26.36	23.63	28.49	23.62	30.95	30.66
4	1905	22.00	22.83	26.68	30.01	33.32	33.25	31.44	30.68	30.12	30.67	27.52	23.82	28.30	22.25	30.00	31.33
5	1906	22.28	23.69	27.31	31.93	34.11	32.19	31.01	30.30	29.92	29.55	27.60	24.72	28.73	23.03	31.11	30.86
6	1907	24.46	24.01	27.04	31.79	32.68	31.92	31.05	29.58	30.67	29.87	27.78	24.44	28.65	24.23	29.92	30.80
7	1908	23.57	25.26	28.86	32.42	33.02	33.12	30.61	29.55	29.59	29.35	26.88	23.73	28.83	24.42	31.43	30.72
8	1909	22.67	24.36	29.22	30.79	33.06	31.70	29.81	29.81	30.06	29.25	27.69	23.69	28.38	23.52	31.02	30.33
9	1910	23.24	25.16	28.48	31.42	33.51	31.84	30.42	29.86	29.82	28.91	26.32	23.37	28.53	24.20	31.14	30.48

```
trainData.dtypes
trainData.columns
```

Output2

```
Index(['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP',
       'OCT', 'NOV', 'DEC', 'ANNUAL', 'JAN-FEB', 'MAR-MAY', 'JUN-SEP',
       'OCT-DEC'],
      dtype='object')
```

Step#3 Let’s check for useful descriptive statistical values by applying pandas “describe()” function

```
trainData.describe()
```

output 3:

```
trainData.describe()
```

	YEAR	JAN	FEB	MAR	APR	MAY	JUN	JUL	AUG	SEP
count	117.000000	117.000000	117.000000	117.000000	117.000000	117.000000	117.000000	117.000000	117.000000	117.00
mean	1959.000000	23.687436	25.597863	29.085983	31.975812	33.565299	32.774274	31.035897	30.507692	30.486
std	33.919021	0.834588	1.150757	1.068451	0.889478	0.724905	0.633132	0.468818	0.476312	0.5442
min	1901.000000	22.000000	22.830000	26.680000	30.010000	31.930000	31.100000	29.760000	29.310000	29.070
25%	1930.000000	23.100000	24.780000	28.370000	31.460000	33.110000	32.340000	30.740000	30.180000	30.120
50%	1959.000000	23.680000	25.480000	29.040000	31.950000	33.510000	32.730000	31.000000	30.540000	30.520
75%	1988.000000	24.180000	26.310000	29.610000	32.420000	34.030000	33.180000	31.330000	30.760000	30.810
max	2017.000000	26.940000	29.720000	32.620000	35.380000	35.840000	34.480000	32.760000	31.840000	32.220

Step#4 Let’s check for any missing or NA values in the training and testing data set

```
trainData.isnull().sum()
```

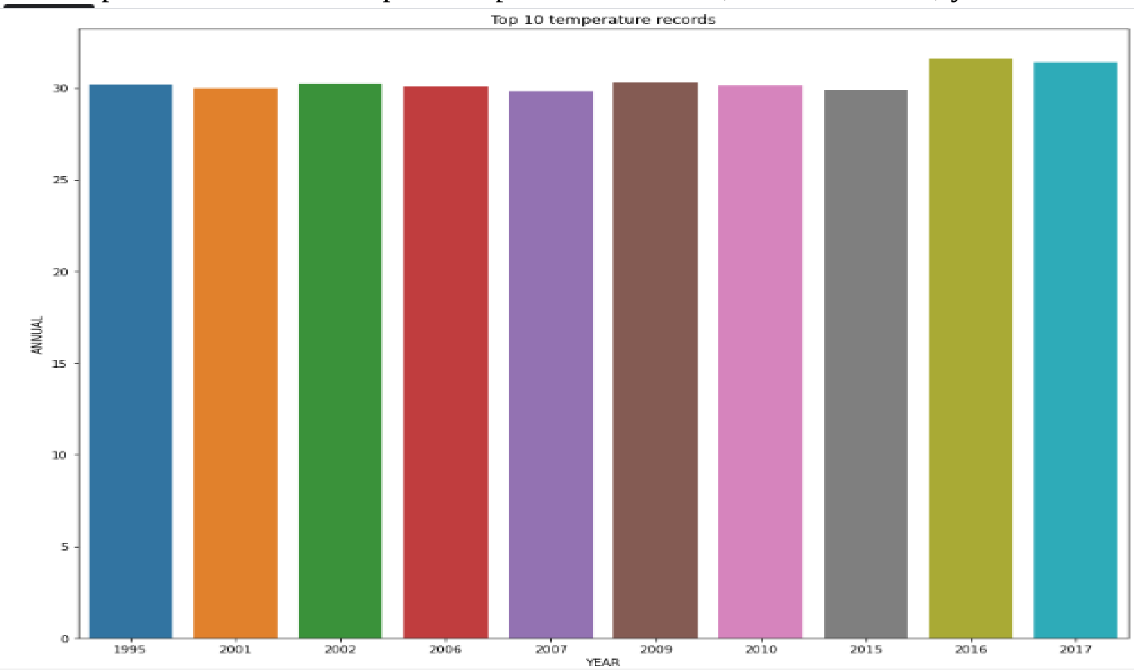
Output 4:

```
YEAR       0
JAN        0
FEB        0
MAR        0
APR        0
MAY        0
JUN        0
JUL        0
AUG        0
SEP        0
OCT        0
NOV        0
DEC        0
ANNUAL     0
JAN-FEB    0
MAR-MAY    0
JUN-SEP    0
OCT-DEC    0
dtype: int64
```

```
top_10_data = trainData.nlargest(10, "ANNUAL")
plt.figure(figsize=(14,12))
plt.title("Top 10 temperature records")
sns.barplot(x=top_10_data.YEAR, y=top_10_data.ANNUAL)
```

Output 5

<AxesSubplot:title={'center':'Top 10 temperature records'}, xlabel='YEAR', ylabel='ANNUAL'>



***Step#6** Let's define our dependent and independent variable for training and testing data*

```
from sklearn import linear_model, metrics
```

```
trainData.columns
```

Output 6:

```
Index(['YEAR', 'JAN', 'FEB', 'MAR', 'APR', 'MAY', 'JUN', 'JUL', 'AUG', 'SEP',
       'OCT', 'NOV', 'DEC', 'ANNUAL', 'JAN-FEB', 'MAR-MAY', 'JUN-SEP',
       'OCT-DEC'],      dtype='object')
```

```
X=trainData[["YEAR"]]
Y=trainData[["JAN"]]
```

Step#7 Let's split the dataset into two sub datasets "Training" and "Testing" Dataset

```
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=1)
```

```
len(X_train)
```

Output 7:

93

```
len(X_test)
```

Output 8:

24

```
trainData.shape
```

Output 9:

(117, 18)

```
reg = linear_model.LinearRegression()
```

```
print(X_train)
```

output 10 :

```
     YEAR
56   1957
94   1995
35   1936
38   1939
93   1994
..    ...
9    1910
72   1973
12   1913
107  2008
37   1938

[93 rows x 1 columns]
```

Step#8 Let's define the model and fit it.

```
model = reg.fit(X_train, Y_train)
```

Step#9 Let's look at different parameters of the model summary and interpret it:

```
r_sq = reg.score(X_train, Y_train)
print ("Determination coefficient:", r_sq)
print('Intercept:', model.intercept_)
print('Slope:', model.coef_)
```

Output 11:

Determination coefficient: 0.3548045849122119

Intercept: [-5.35338281]

Slope: [[0.01486008]]

```
Y_pred = model.predict(X_test)
print('predicted response:', Y_pred, sep='\n')
```

Output 12:

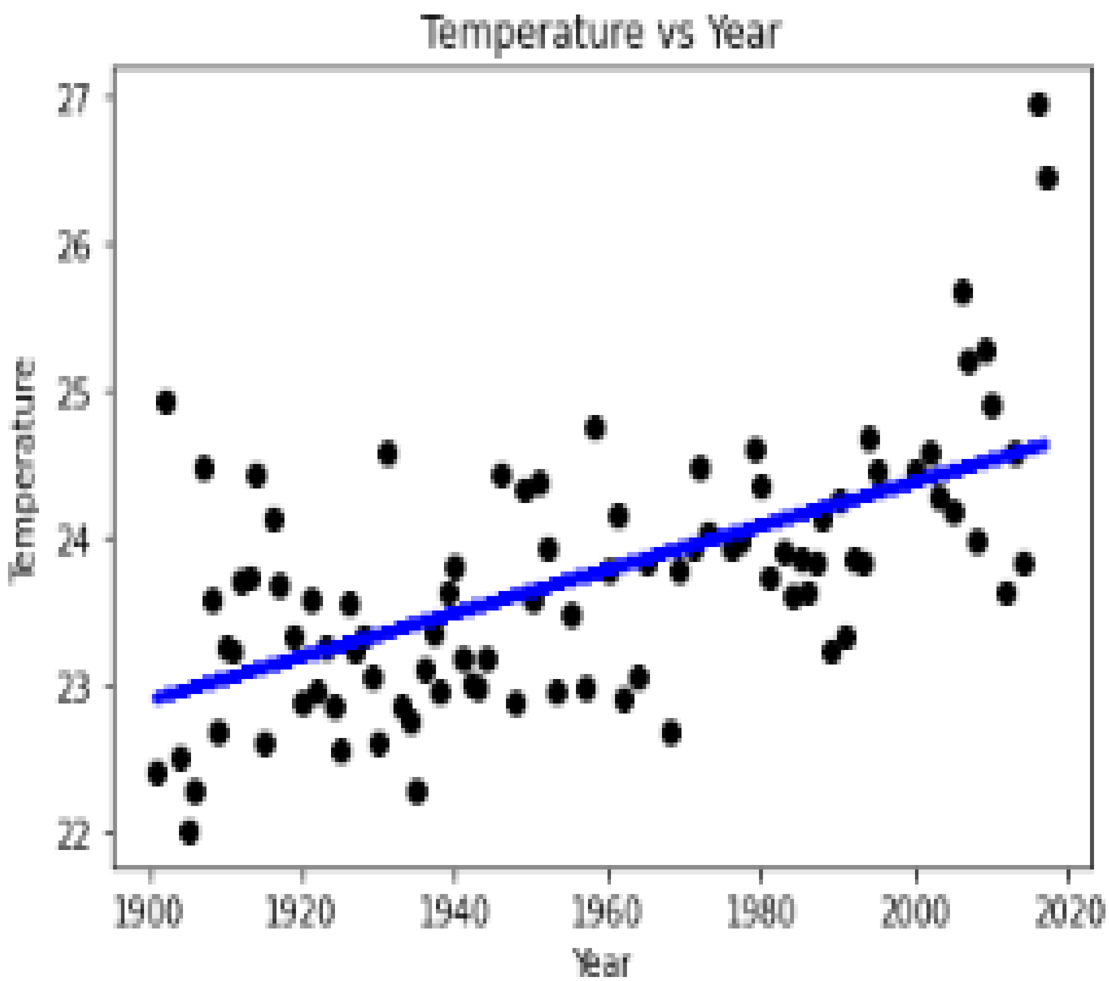
```
predicted response:
[[23.92097555]
 [23.5791937 ]
 [23.75751466]
 [24.58967916]
 [23.98041587]
 [24.35191788]
 [23.35629249]
 [23.68321426]
 [23.86153523]
 [24.32219772]
 [24.30733764]
 [24.3370578 ]
 [22.92535016]
 [23.81695498]
 [24.53023884]
 [23.71293442]
 [24.42621828]
 [24.38163804]
 [23.87639531]
 [23.54947354]
 [24.03985619]
 [23.14825137]
 [24.09929651]
 [23.99527595]]
```

Step#10: Now let’s visualise the regression equation fitment on the data

Visualization on Training Data

```
plt.scatter(X_train, Y_train, color='black')
plt.plot(X_train, reg.predict(X_train), color='blue', linewidth=3)
plt.title("Temperature vs Year")
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.show()
```

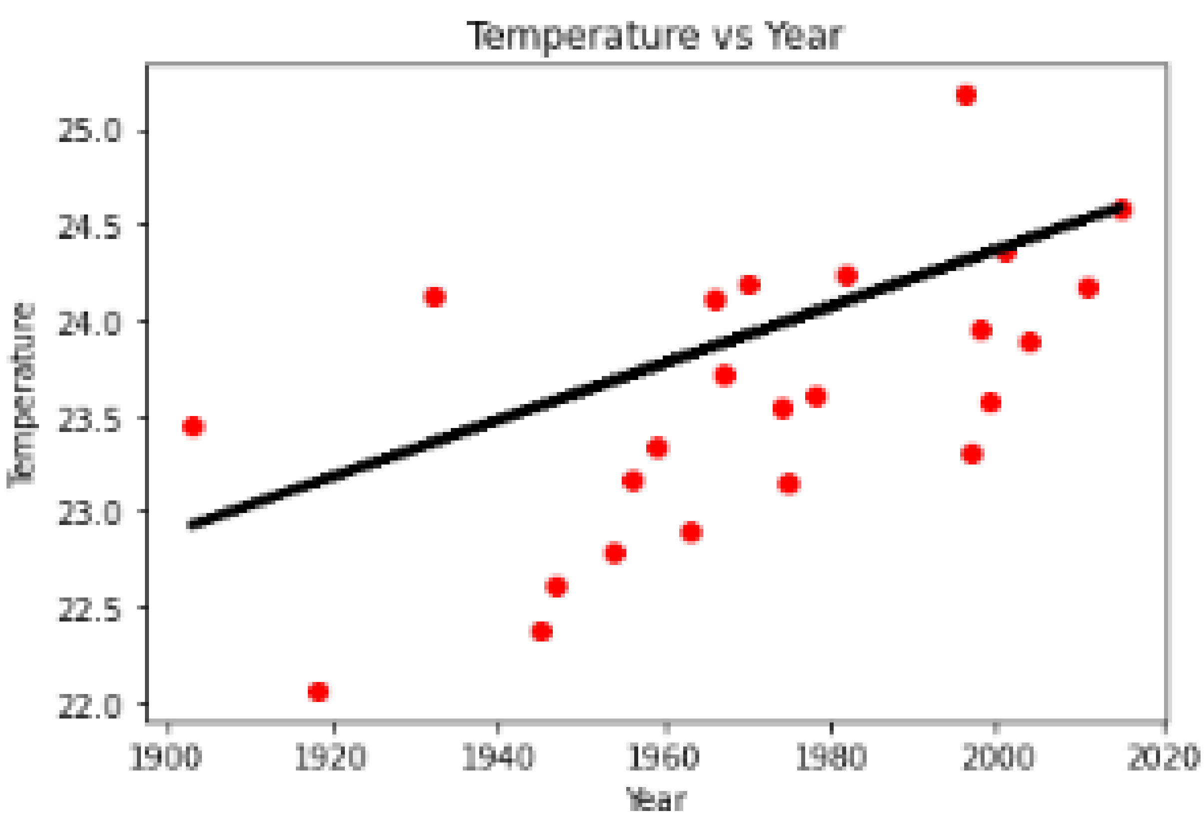
Output 13:



Step#11: Now let's check how our model is doing on the testing data, which we kept aside for testing our model performance Visualization on Testing Data

```
plt.scatter(X_test, Y_test, color='red')
plt.plot(X_test, reg.predict(X_test), color='black', linewidth=3)
plt.title("Temperature vs Year")
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.show()
```

Output 14:



Step # 12: Assess the performance of regression models using MSE, MAE and R-Square metrics

```
mse = np.sum((Y_pred - Y_test)**2)    # mean squared error (MSE)

rmse = np.sqrt(mse/24)                    # root mean squared error (RMSE)

print("Mean Squared Error(MSE):", mse)
print("Root Mean Squared Error(RMSE):", rmse)
```

Output 15:

```
Mean Squared Error (MSE) JAN    10.737906
dtype: float64

Root Mean Squared Error (RMSE) JAN    0.668889
dtype: float64
```

```
SSR = np.sum((Y_pred - Y_test)**2)   #Sum of square of Residuals/Errors SSR/SSE

SST = np.sum((Y_test - np.mean(Y_test))**2)    #  total sum of squares


r2_score = 1 - (SSR/SST)    # R2 score

print('SST:', SST)
print('SSR', SSR)
print('R2 square:', r2_score)
```

Output 16:

```
SST: JAN    12.452996
dtype: float64
SSR JAN    10.737906
dtype: float64
R2 square: JAN    0.137725
dtype: float64
```

Conclusion: Thus we have studied Regression techniques and implemented simple linear regression for given problem statement.

Assignment No.03: Classification

Name of the Student: ______________________________ Roll no: ___

CLASS: - T.E. IT **Subject Name: - LP-I Lab (Part I- ML)**

Experiment No. 03

** Decision Tree Classification Technique: **

Marks: /10

Date of Performance: / /2022 **Sign with Date:**

Problem Statement:-

Every year many students give the GRE exam to get admission in foreign Universities. The data set contains GRE Scores (out of 340), TOEFL Scores (out of 120), University Rating (out of 5), Statement of Purpose strength (out of 5), Letter of Recommendation strength (out of 5), Undergraduate GPA (out of 10), Research Experience (0=no, 1=yes), Admitted (0=no, 1=yes). Admitted is the target variable.

Data Set Available on kaggle (The last column of the dataset needs to be changed to 0 or 1)

Data Set: https://www.kaggle.com/mohansacharya/graduate-admissions

The counselor of the firm is supposed check whether the student will get an admission or not based on his/her GRE score and Academic Score. So to help the counselor to take appropriate decisions build a machine learning model classifier using Decision tree to predict whether a student will get admission or not.

A. Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary.
B. Perform data-preparation (Train-Test Split)
C. Apply Machine Learning Algorithm
D. Evaluate Model.

Theory:

Classification: Classification may be defined as the process of predicting class or category from observed values or given data points. The categorized output can have the form such as "Black" or "White" or "spam" or "no spam". Mathematically, classification is the task of approximating a mapping function (f) from input variables (X) to output variables (Y).

Building a Classifier in Python:
Step1: Importing necessary python package
Step2: Importing dataset
Step3: Organizing data into training & testing sets
Step4: Model evaluation
Step5: Finding accuracy

Decision Tree Algorithm:

Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions tress is the most powerful algorithms that falls under the category of supervised algorithms.

Decision Tree Algorithm Steps:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
Step-3: Divide the S into subsets that contains possible values for the best attributes.
Step-4: Generate the decision tree node, which contains the best attribute.
Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Solve decision tree such problems there is a technique which is called as **Attribute selection measure or ASM**. There are two popular techniques for ASM, which are:

1. **Information Gain:** Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree.

   Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

2. **Entropy:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

   Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

Where,

S= Total number of samples,

P(yes)= probability of yes,

P(no)= probability of no

3. **Gini Index:** Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index.

$$\text{Gini Index} = 1 - \sum_j P_j^2$$

Applications of Classifications Algorithms:

1. Sentiment Analysis
2. Email Spam Classification
3. Document Classification
4. Image Classification

Algorithm:-

Step # 1: Importing the required libraries

Step#2: loading the dataset

Step#3: Let's check for useful information by applying pandas "info ()" function

Step#4: Let's check for useful descriptive statistical values by applying pandas "describe ()" function

Step#5: Let's check for any missing or NA values in the training and testing data set

Step#6: Let's print number of columns and rows i. e shape of dataset

Step#7: let's use drop () function and perform some basic data frame functions

Step#8: Let's Visualize SOP (attribute) data from dataset

Step#9: Let's split the dataset into two sub datasets "Training" and "Testing" Dataset

Step#10: Let's build the model using training data samples and fit it.

Step#11: Now let's check how our model is doing on the testing data, which we kept aside for testing our model performance

Step#12: Now visualize the decision tree model

Python Code:-

Step # 1 Importing the required libraries

```
In[1]:
import pandas as pd
import numpy as np
import seaborn as sns
# Import Decision Tree Classifier
from sklearn.tree import DecisionTreeClassifier
# Import train_test_split function
from sklearn.model_selection import train_test_split
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
```

Step#2 loading the dataset

```
In[2]:
data=pd.read_csv("C:/Users/Administrator/LPI/Datasets/Admission_Predict_Ver1.
csv")
data.head()
```

Out[2]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Classlabel
0	1	337	118	4	4.5	4.5	9.65	1	1
1	2	324	107	4	4.0	4.5	8.87	1	1
2	3	316	104	3	3.0	3.5	8.00	1	1
3	4	322	110	3	3.5	2.5	8.67	1	1
4	5	314	103	2	2.0	3.0	8.21	0	1

Step#3 Let's check for useful information by applying pandas "info()" function

```
In[3]
data.info()
```

Out[3]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Serial No.         500 non-null    int64
 1   GRE Score          500 non-null    int64
 2   TOEFL Score        500 non-null    int64
 3   University Rating  500 non-null    int64
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64
 8   Classlabel         500 non-null    int64
dtypes: float64(3), int64(6)
memory usage: 35.3 KB
```

Step#4 Let's check for useful descriptive statistical values by applying pandas "describe()" function

In[4];
data.describe()

Out[4]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Classlabel
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.000000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.926000
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.262033
min	1.000000	290.00000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.000000

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Classlabel
		0							
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	1.000000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	1.000000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	1.000000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	1.000000

Step#5 Let's check for any missing or NA values in the training and testing data set

In[5]:

```
data.isnull().sum()
```

Out[5]:

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Classlabel           0
dtype: int64
```

Step#6 Let's print number of columns and rows i. e shape of dataset

```
In[6]:
print("There are {} rows and {}columns.".format(data.shape[0],data.shape[1]))
```

Out[6]:

```
There are 500 rows and 9 columns.
```

Step#7 let's use drop () function and perform some basic data frame functions

```
In[7]:
data= data.drop(['Serial No.'], axis=1)
data
```

Out[7]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Classlabel
0	337	118	4	4.5	4.5	9.65	1	1
1	324	107	4	4.0	4.5	8.87	1	1
2	316	104	3	3.0	3.5	8.00	1	1
3	322	110	3	3.5	2.5	8.67	1	1
4	314	103	2	2.0	3.0	8.21	0	1
...	...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1	1
496	337	117	5	5.0	5.0	9.87	1	1
497	330	120	5	4.5	5.0	9.56	1	1
498	312	103	4	4.0	5.0	8.43	0	1
499	327	113	4	4.5	4.5	9.04	0	1

500 rows × 8 columns

In [8]:

```
data.info()
```

Out [8]:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   GRE Score          500 non-null    int64
 1   TOEFL Score        500 non-null    int64
 2   University Rating  500 non-null    int64
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64
 7   Classlabel         500 non-null    int64
```

```
dtypes: float64(3), int64(5)
memory usage: 31.4 KB
```

In [9]:
```
data['Classlabel'].value_counts()
```

Out[9]:
```
1    463
0     37
Name: Classlabel, dtype: int64
```

In [10]:
```
data['SOP'].value_counts()
```

Out[10]:
```
4.0    89
3.5    88
3.0    80
2.5    64
4.5    63
2.0    43
5.0    42
1.5    25
1.0     6
Name: SOP, dtype: int64
```

In [11]:
```
print(data.Classlabel== 1)
```

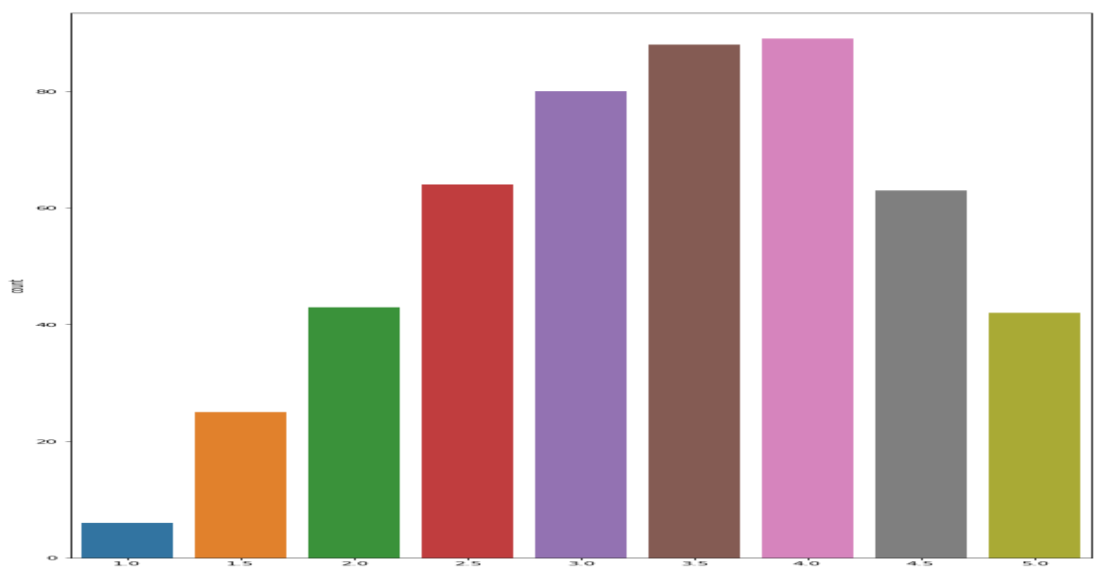
```
Out[11]:
0      True
1      True
2      True
3      True
4      True
       ...
495    True
496    True
497    True
498    True
499    True
Name: Classlabel, Length: 500, dtype: bool
```

Step#8 Let's Visualize SOP(attribute) data from dataset

In [12]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,20));
sns.countplot(data['SOP'].values);
```

Out[12]:



Step#9 Let's split the dataset into two sub datasets "Training" and "Testing" Dataset

In [13]:

```
#split dataset in features and target variable
feature_cols = ['GRE Score','TOEFL Score','University
Rating','SOP','CGPA','Research']
X = data[feature_cols] # Features
y = data.Classlabel # Target variable
```

In [14]:

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1) # 70% training and 30% test
```

In [15]:
```
len(X_train)
```

Out[15]:
```
350
```

In [16]:
```
len(X_test)
```

Out[16]:
```
150
```

Step#10 Let's build the model using training data samples and fit it.

In [17]:
```
# Create Decision Tree classifer object
dtclf = DecisionTreeClassifier()

# Fit Decision Tree Classifer
dtclf = dtclf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = dtclf.predict(X_test)
print(y_pred)
```

Out [17]:
```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 1 1 0
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
 0 1]
```

Step#11: Now let's check how our model is doing on the testing data, which we kept aside for testing our model performance

In [18]:
```
# Model Accuracy, how often is the classifier correct?
accuracy=metrics.accuracy_score(y_pred,y_test)*100
print("Accuracy of the model is {:.2f}".format(accuracy))
```

```
Out[18]:
Accuracy of the model is 91.33
```

In [19]:
```
metrics.confusion_matrix(y_pred,y_test)
```

Out[19]:
```
array([[  7,   6],
       [  7, 130]], dtype=int64)
```

In [20]:
```
pip install graphviz
ut[20]:
Requirement already satisfied: graphviz in c:\programdata\anaconda3\lib\site-packages (0.20.1)Note: you may need to restart the kernel to use updated packages.
```

In [21]:
```
pip install pydotplus
Out[21]:
Requirement already satisfied: pydotplus in c:\programdata\anaconda3\lib\site-packages (2.0.2)Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: pyparsing>=2.0.1 in c:\programdata\anaconda3\lib\site-packages (from pydotplus) (2.4.7)
```

Step#12: *Now Visualize the decision tree model*

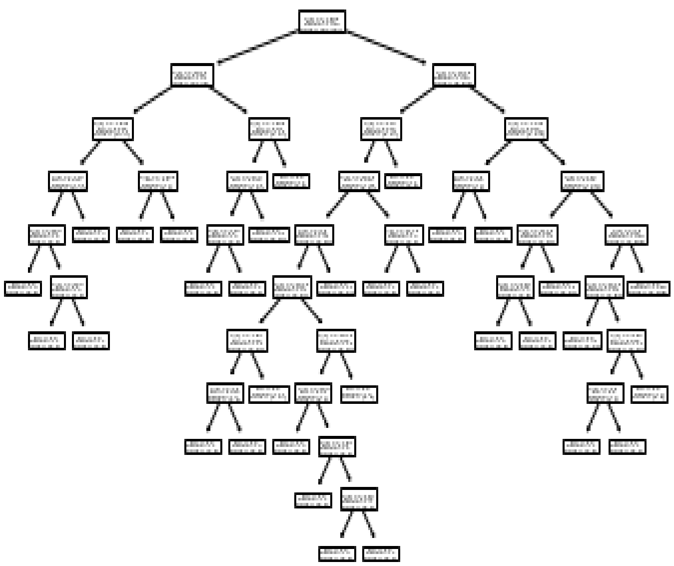
In [22]:
```
from sklearn import tree
tree.plot_tree(dtclf)
```

Out[22]:
```
[Text(160.425, 207.55636363636364, 'X[4] <= 7.665\ngini = 0.123\nsamples = 350\nvalue = [23, 327]'),
 Text(94.86, 187.7890909090909, 'X[0] <= 300.0\ngini = 0.499\nsamples = 27\nvalue = [14, 13]'),
 Text(55.8, 168.0218181818182, 'X[4] <= 7.645\ngini = 0.245\nsamples = 14\nvalue = [12, 2]'),
 Text(33.480000000000004, 148.25454545454545, 'X[3] <= 1.75\ngini = 0.153\nsamples = 12\nvalue = [11, 1]'),
 Text(22.32, 128.48727272727274, 'X[1] <= 98.5\ngini = 0.32\nsamples = 5\nvalue = [4, 1]'),
```

```
 Text(11.16, 108.72, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(33.480000000000004, 108.72, 'X[1] <= 99.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(22.32, 88.95272727272729, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(44.64, 88.95272727272729, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(44.64, 128.48727272727274, 'gini = 0.0\nsamples = 7\nvalue = [7, 0]'),
 Text(78.12, 148.25454545454545, 'X[4] <= 7.655\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(66.96000000000001, 128.48727272727274, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(89.28, 128.48727272727274, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(133.92000000000002, 168.0218181818182, 'X[4] <= 7.655\ngini = 0.26\nsamples = 13\nvalue = [2, 11]'),
 Text(122.76, 148.25454545454545, 'X[1] <= 97.5\ngini = 0.153\nsamples = 12\nvalue = [1, 11]'),
 Text(111.6, 128.48727272727274, 'X[3] <= 2.25\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(100.44, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(122.76, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(133.92000000000002, 128.48727272727274, 'gini = 0.0\nsamples = 10\nvalue = [0, 10]'),
 Text(145.08, 148.25454545454545, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(225.99, 187.7890909090909, 'X[1] <= 101.5\ngini = 0.054\nsamples = 323\nvalue = [9, 314]'),
 Text(189.72, 168.0218181818182, 'X[3] <= 3.75\ngini = 0.223\nsamples = 47\nvalue = [6, 41]'),
 Text(178.56, 148.25454545454545, 'X[0] <= 316.5\ngini = 0.194\nsamples = 46\nvalue = [5, 41]'),
 Text(156.24, 128.48727272727274, 'X[5] <= 0.5\ngini = 0.165\nsamples = 44\nvalue = [4, 40]'),
 Text(145.08, 108.72, 'X[0] <= 301.5\ngini = 0.252\nsamples = 27\nvalue = [4, 23]'),
 Text(122.76, 88.95272727272729, 'X[4] <= 7.83\ngini = 0.133\nsamples = 14\nvalue = [1, 13]'),
 Text(111.6, 69.18545454545455, 'X[3] <= 2.0\ngini = 0.444\nsamples = 3\nvalue = [1, 2]'),
 Text(100.44, 49.418181818181836, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(122.76, 49.418181818181836, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(133.92000000000002, 69.18545454545455, 'gini = 0.0\nsamples = 11\nvalue = [0, 11]'),
```

```
 Text(167.4, 88.95272727272729, 'X[0] <= 310.0\ngini = 0.355\nsamples =
13\nvalue = [3, 10]'),
 Text(156.24, 69.18545454545455, 'X[1] <= 99.5\ngini = 0.49\nsamples =
7\nvalue = [3, 4]'),
 Text(145.08, 49.418181818181836, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(167.4, 49.418181818181836, 'X[4] <= 7.93\ngini = 0.48\nsamples =
5\nvalue = [3, 2]'),
 Text(156.24, 29.650909090909096, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(178.56, 29.650909090909096, 'X[3] <= 3.25\ngini = 0.375\nsamples =
4\nvalue = [3, 1]'),
 Text(167.4, 9.883636363636384, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(189.72, 9.883636363636384, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(178.56, 69.18545454545455, 'gini = 0.0\nsamples = 6\nvalue = [0, 6]'),
 Text(167.4, 108.72, 'gini = 0.0\nsamples = 17\nvalue = [0, 17]'),
 Text(200.88, 128.48727272727274, 'X[0] <= 317.5\ngini = 0.5\nsamples =
2\nvalue = [1, 1]'),
 Text(189.72, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(212.04, 108.72, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(200.88, 148.25454545454545, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(262.26, 168.0218181818182, 'X[4] <= 7.745\ngini = 0.022\nsamples =
276\nvalue = [3, 273]'),
 Text(234.36, 148.25454545454545, 'X[3] <= 3.5\ngini = 0.444\nsamples =
3\nvalue = [1, 2]'),
 Text(223.2, 128.48727272727274, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(245.52, 128.48727272727274, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(290.16, 148.25454545454545, 'X[3] <= 2.25\ngini = 0.015\nsamples =
273\nvalue = [2, 271]'),
 Text(267.84000000000003, 128.48727272727274, 'X[0] <= 311.5\ngini =
0.095\nsamples = 20\nvalue = [1, 19]'),
 Text(256.68, 108.72, 'X[4] <= 8.28\ngini = 0.245\nsamples = 7\nvalue = [1,
6]'),
 Text(245.52, 88.95272727272729, 'gini = 0.0\nsamples = 6\nvalue = [0, 6]'),
 Text(267.84000000000003, 88.95272727272729, 'gini = 0.0\nsamples = 1\nvalue
= [1, 0]'),
 Text(279.0, 108.72, 'gini = 0.0\nsamples = 13\nvalue = [0, 13]'),
 Text(312.48, 128.48727272727274, 'X[5] <= 0.5\ngini = 0.008\nsamples =
253\nvalue = [1, 252]'),
 Text(301.32, 108.72, 'X[0] <= 322.5\ngini = 0.023\nsamples = 85\nvalue = [1,
84]'),
 Text(290.16, 88.95272727272729, 'gini = 0.0\nsamples = 74\nvalue = [0,
74]'),
```

```
 Text(312.48, 88.95272727272729, 'X[4] <= 8.65\ngini = 0.165\nsamples = 11\nvalue = [1, 10]'),
 Text(301.32, 69.18545454545455, 'X[2] <= 3.5\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(290.16, 49.418181818181836, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(312.48, 49.418181818181836, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(323.64, 69.18545454545455, 'gini = 0.0\nsamples = 9\nvalue = [0, 9]'),
 Text(323.64, 108.72, 'gini = 0.0\nsamples = 168\nvalue = [0, 168]')]
```



Conclusion: Thus we have studied Decision tree classification technique and implemented it for given problem statement.

Assignment No.04: Classification techniques

Sinhgad Institutes

Name of the Student: ____ **Roll no:** ___

CLASS: - T.E. IT **Subject Name: - LP-I Lab (Part I- ML)**

Experiment No. 04

** Classification Techniques: **

Marks: /10

Date of Performance: / /2022 **Sign with Date:**

Problem statement:

Assignment on Classification technique

Every year many students give the GRE exam to get admission in foreign Universities. The data set contains GRE Scores (out of 340), TOEFL Scores (out of 120), University Rating (out of 5), Statement of Purpose strength (out of 5), Letter of Recommendation strength (out of 5), Undergraduate GPA (out of 10), Research Experience (0=no, 1=yes), Admitted (0=no, 1=yes). Admitted is the target variable. Data Set Available on kaggle (The last column of the dataset needs to be changed to 0 or 1)Data Set : https://www.kaggle.com/mohansacharya/graduate-admissions The counselor of the firm is supposed check whether the student will get an admission or not based on his/her GRE score and Academic Score. So to help the counselor to take appropriate decisions build a machine learning model classifier using Decision tree to predict whether a student will get admission or not. Apply Data pre-processing (Label Encoding, Data Transformation….) techniques if necessary. Perform data-preparation (Train-Test Split) C. Apply Machine Learning Algorithm D. Evaluate Model.

Theory:

Classification: Classification may be defined as the process of predicting class or category from observed values or given data points. The categorized output can have the form such as "Black" or "White" or "spam" or "no spam".Mathematically, classification is the task of approximating a mapping function (f) from input variables (X) to output variables (Y).

Building a Classifier in Python:

Step1: Importing necessary python packageStep2: Importing dataset

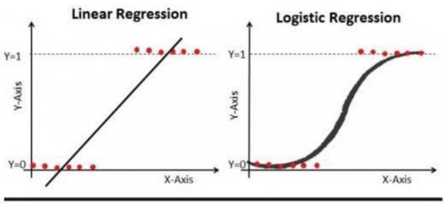
Step3: Organizing data into training & testing setsStep4: Model evaluation

Step5: Finding accuracy

Classification Algorithms Include:

Naive Bayes, Logistic regression, K-nearest neighbours, (Kernel) SVM, Decision tree

1. **Logistic Regression Algorithm:** It is a Machine Learning classification algorithm that is used to predict the probability of a categorical dependent variable. In logistic regression, the dependent variable is a binary variable that contains data coded as 1 (yes, success, etc.) or 0 (no, failure, etc.). Logistic regression model predicts P(Y=1) as a function of X.



1. **Logistic Regression Algorithm Equation:**

The Logistic regression equation can be obtained from the Linear Regression equation. The mathematical steps to get Logistic Regression equations are given below:

- We know the equation of the straight line can be written as:

$$y = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n$$

- In Logistic Regression y can be between 0 and 1 only, so for this let's divide the above equation by (1-y):

$$\frac{y}{1-y}\text{; 0 for y= 0, and infinity for y=1}$$

- But we need range between -[infinity] to +[infinity], then take logarithm of the equation it will become:

$$\boldsymbol{log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n}$$

The above equation is the final equation for Logistic Regression.

Steps in Logistic Regression: To implement the Logistic Regression using Python, we will use the same steps as we have done in previous topics of Regression. Below are the steps:

1. Data Pre-processing step
2. Fitting Logistic Regression to the Training set
3. Predicting the test result
4. Test accuracy of the result(Creation of Confusion matrix)
5. Visualizing the test set result.

2. **Decision Tree Algorithm:** Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions tress is the most powerful algorithms that falls under the category of supervised algorithms.

Decision Tree Algorithm Steps:

Step-1: Begin the tree with the root node, says S, which contains the complete dataset.

Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).

Step-3: Divide the S into subsets that contains possible values for the best attributes.

Step-4: Generate the decision tree node, which contains the best attribute.

Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

Solve decision tree such problems there is a technique which is called as **Attribute selection measure or ASM**. There are two popular techniques for ASM, which are:

1. **Information Gain:** Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute. It calculates how much information a feature provides us about a class. According to the value of information gain, we split the node and build the decision tree.

   Information Gain= Entropy(S)- [(Weighted Avg) *Entropy(each feature)

2. **Entropy:** Entropy is a metric to measure the impurity in a given attribute. It specifies randomness in data. Entropy can be calculated as:

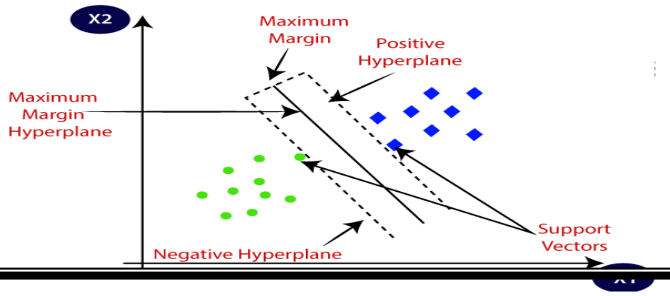
   Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

   Where,S= Total number of samples, P(yes)= probability of yes, P(no)= probability of no

3. **Gini Index:** Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm. An attribute with the low Gini index should be preferred as compared to the high Gini index.

   Gini Index= 1- ∑jPj2

3. **SVM Algorithm:** Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems.The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine. Consider the below diagram in which there are two different categories that are classified using a decision boundary or hyperplane.



SVM Algorithm Steps:

1. Importing the dataset
2. Splitting the dataset into training and test samples
3. Classifying the predictors and target
4. Initializing Support Vector Machine and fitting the training data
5. Predicting the classes for test set
6. Attaching the predictions to test set for comparing
7. Comparing the actual classes and predictions
8. Calculating the accuracy of the predictions

Applications of Classifications Algorithms:

1. Sentiment Analysis
2. Email Spam Classification
3. Document Classification
4. Image Classification

Python Code:

```
# To load the dataset

import pandas as pd

import matplotlib.pyplot as plt

#seaborn: for data visualization and exploratory data analysis

Import seaborn as sns

import warnings warnings.filterwarnings("ignore")

#Read data in csv file store into dataframe

df = pd.read_csv('Admission_Predict.csv')

print(df.head(5))

##############################################################################

#To drop the irrelevant column and check if there are any null values in the dataset

df = df.drop(['Serial No.'], axis=1)

print(df.isnull().sum())

#To see the distribution of the variables of graduate applicants. #distplot() plot distributed data as observations

#KDE: Kerner Density Estimate, probability density function of a continuous random variable Show GRE Score
```

```
fig = sns.distplot(df['GRE Score'], kde=False)

plt.title("Distribution of GRE Scores")

 plt.show()
#Show TOEFL Score
fig = sns.distplot(df['TOEFL Score'], kde=False)

plt.title("Distribution of TOEFL Scores")

plt.show()
#Show University Ratings
fig = sns.distplot(df['University Rating'], kde=False)

plt.title("Distribution of University Rating")

 plt.show()
#Show SOP Ratings
fig = sns.distplot(df['SOP'], kde=False)

plt.title("Distribution of SOP Ratings")

plt.show()
#Show CGPA
fig = sns.distplot(df['CGPA'], kde=False)

plt.title("Distribution of CGPA")

plt.show()
#It is clear from the distributions, students with varied merit apply  for the university.
#Understanding the relation between different factors responsible for graduate admissions GRE Score vs TOEFL Score

#regplot() :Plot data and a linear regression model fit.
fig = sns.regplot(x="GRE Score", y="TOEFL Score", data=df)
```

plt.title("GRE Score vs TOEFL Score")

plt.show()

#People with higher GRE Scores also have higher TOEFL Scores which is justified because both TOEFL and GRE have a verbal section which although not similar are relatable

#GRE Score vs CGPA

fig = sns.regplot(x="GRE Score", y="CGPA", data=df)

plt.title("GRE Score vs CGPA")

plt.show()

#although there are exceptions, people with higher CGPA usually has higher GRE scores maybe because they are smart or hard working

#LOR vs CGPA show wheather Research 0 or 1

#lmplot():a 2D scatterplot with an optional overlaid regression line.

#hue: Variables that define subsets of the data, which will be drawn on separate facets in the grid.

fig = sns.lmplot(x="CGPA", y="LOR ", data=df, hue="Research")

plt.title("LOR vs CGPA")

plt.show()

#LORs (Letter of Recommendation strength) are not that related with CGPA so it is clear that a person's LOR is not dependent on that persons academic excellence.

#Having research experience is usually related with a good LOR which might be justified by the fact that supervisors have personal interaction with the students performing research which usually results in good LORs

#GRE Score vs LOR SHOW WHEATHER Research 0 or 1

fig = sns.lmplot(x="GRE Score", y="LOR ", data=df, hue="Research")
plt.title("GRE Score vs LOR")

plt.show()

#GRE scores and LORs are also not that related. People with different kinds of LORs have all kinds of GRE scores

#SOP vs CGPA

fig = sns.regplot(x="CGPA", y="SOP", data=df)

plt.title("SOP vs CGPA")

plt.show()

#CGPA and SOP are not that related because Statement of Purpose is related to academic performance, but since people with good CGPA tend to be more hard working so they have good things to say in their SOP which might explain the slight move towards higher CGPA as along with good SOPs

#GRE Score vs SOP

fig = sns.regplot(x="GRE Score", y="SOP", data=df)

plt.title("GRE Score vs SOP")

plt.show()

#Similary, GRE Score and CGPA is only slightly related#SOP vs TOEFL

fig = sns.regplot(x="TOEFL Score", y="SOP", data=df)

plt.title("SOP vs TOEFL")

plt.show()

.#Correlation among variables

Import numpy as np

#corr():Find the pairwise correlation of all columns in the dataframe

corr = df.corr()

print(corr)

#plt.subplot:Crate a figure & set sub plots

fig, ax = plt.subplots(figsize=(8, 8))

#Make a diverging palette between two HUSL colors.#cmap: colour map set

colormap = sns.diverging_palette(220, 10, as_cmap=True)

#zeros_like():Returns an array of given shape and type as given array, with zeros.

dropSelf = np.zeros_like(corr)

#np.triu_indices_from(dropSelf): Return indices of array

dropSelf[np.triu_indices_from(dropSelf)] = True

colormap = sns.diverging_palette(220, 10, as_cmap=True)

sns.heatmap(corr, cmap=colormap, linewidths=.5, annot=True, fmt=".2f", mask=dropSelf)

plt.show()

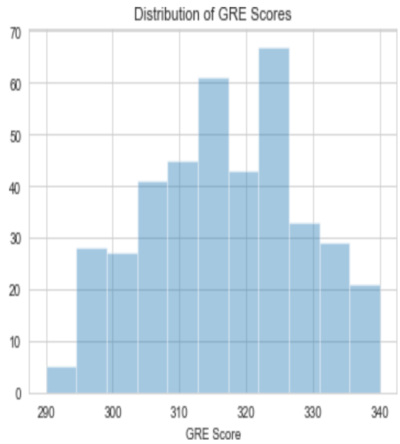
fromsklearn.model_selection import train_test_split

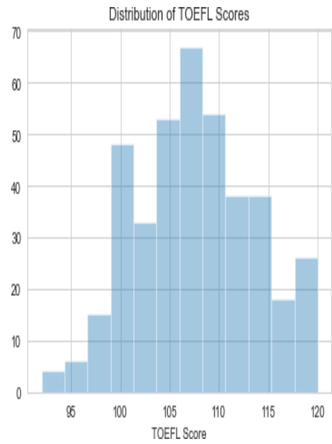
```
#drop col chances of admission

X = df.drop(['Chance of Admit '], axis=1)y = df['Chance of Admit ']

#split data for training & tasting

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.20, shuffle=False)

#DecisionTree, Random Forest, K Neighbor, SVR, Linear Regression

from sklearn.tree import DecisionTreeRegressor

from sklearn.ensemble import RandomForestRegressor

from sklearn.svm import SVR

from sklearn.linear_model import LinearRegression

from sklearn.metrics import mean_squared_error

#These methods predict the future applicant's chances of admission.

models = [['DecisionTree :',DecisionTreeRegressor()],

['Linear Regression :', LinearRegression()],['SVM :', SVR()]]

print("Results...")

#For loop for generating model results

For name,model in models:model = model

#Fit training data of x & y axis

model.fit(X_train, y_train)

#Pass predicted or test result

predictions = model.predict(X_test)

#Difference between actual value & predicted value

print(name, (np.sqrt(mean_squared_error(y_test, predictions))))classifier = RandomForestRegressor()

classifier.fit(X,y)

#X.columns features in dataset

feature_names = X.columns print(feature_names)
```

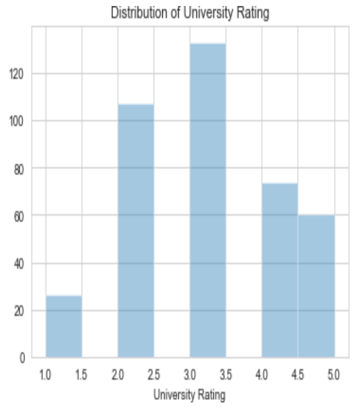
```
#Initialize importance_frame[] in 2 dim array.

importance_frame = pd.DataFrame()

#Two Dimensional Array Format column names

importance_frame['Features'] = X.columns

#classifier.feature_importance is decision tree based on correlation value As per importance of admission

importance_frame['Importance'] = classifier.feature_importances_

#Sort the features by high to low bar graph

importance_frame = importance_frame.sort_values(by=['Importance'], ascending=True)

#Visualize 7 Feature Importances

#bar: plots horizontal rectangles with constant heights.

plt.barh([1,2,3,4,5,6,7],

importance_frame['Importance'],

 align='center', alpha=0.5)

#yticks: set feature lable on y axis

plt.yticks([1,2,3,4,5,6,7],

importance_frame['Features'])

plt.xlabel('Importance')

#Clearly, CGPA is the most factor for graduate admissions followed by GRE Score.

plt.title('Feature Importances')

plt.show()
```

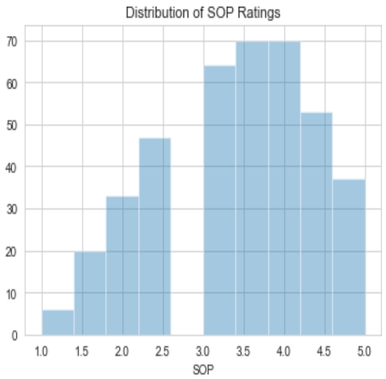
Output:

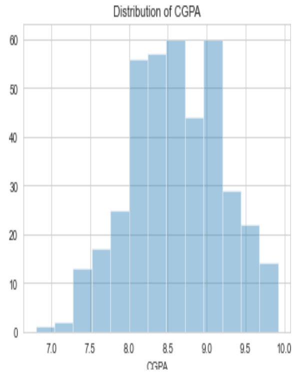
```
In [2]: runfile('C:/Users/Vrushali/Test_Project/Practicle_2.py', wdir='C:/Users/Vrushali/Test_Project')
   Serial No.  GRE Score  TOEFL Score  ...  CGPA  Research  Chance of Admit
0           1        337          118  ...  9.65         1             0.92
1           2        324          107  ...  8.87         1             0.76
2           3        316          104  ...  8.00         1             0.72
3           4        322          110  ...  8.67         1             0.80
4           5        314          103  ...  8.21         0             0.65

[5 rows x 9 columns]
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
                   GRE Score  TOEFL Score  ...  Research  Chance of Admit
GRE Score           1.000000     0.835977  ...  0.580391         0.802610
TOEFL Score         0.835977     1.000000  ...  0.489858         0.791594
University Rating   0.668976     0.695590  ...  0.447783         0.711250
SOP                 0.612831     0.657981  ...  0.444029         0.675732
LOR                 0.557555     0.567721  ...  0.396859         0.669889
CGPA                0.833060     0.828417  ...  0.521654         0.873289
Research            0.580391     0.489858  ...  1.000000         0.553202
Chance of Admit     0.802610     0.791594  ...  0.553202         1.000000

[8 rows x 8 columns]
```

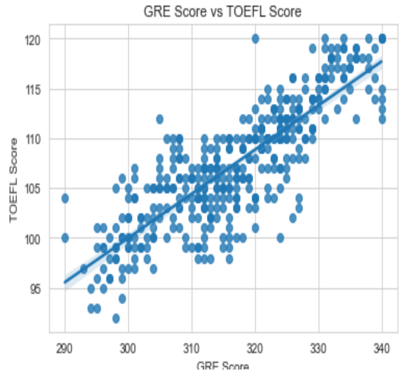

Distribution of GRE Scores
GRE Score


Distribution of TOEFL Scores
TOEFL Score

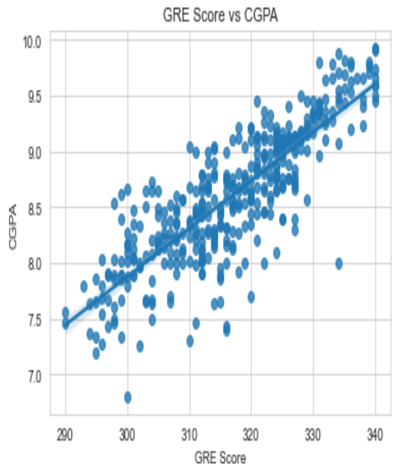

Distribution of University Rating
University Rating

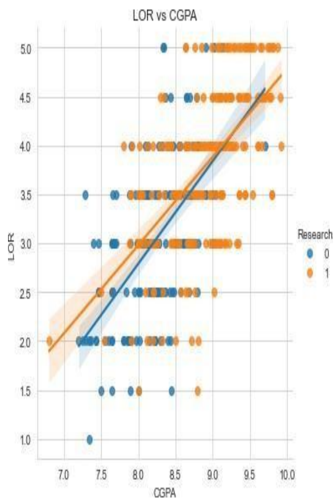

Distribution of SOP Ratings
SOP

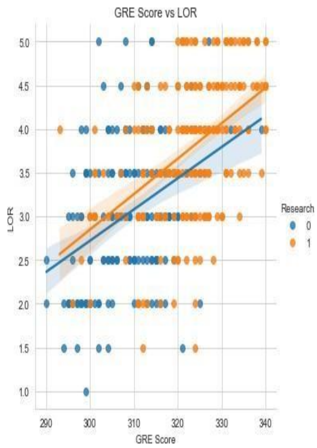

Distribution of CGPA
CGPA

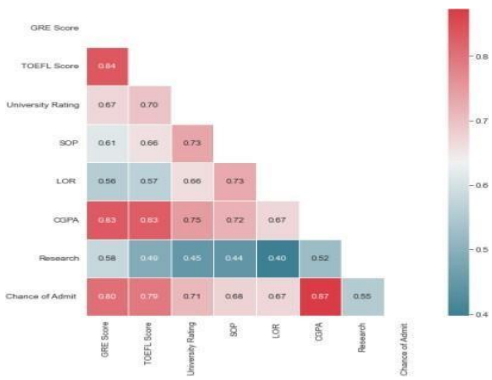

GRE Score vs TOEFL Score
TOEFL Score
GRE Score

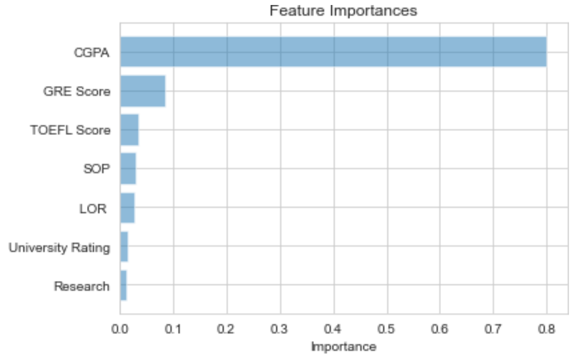
–


GRE Score vs CGPA
CGPA
GRE Score


LOR vs CGPA
LOR
CGPA
Research
0
1


GRE Score vs LOR
LOR
GRE Score
Research
0
1






```
[8 rows x 8 columns]
Results...
DecisionTree : 0.09370832406995656
Linear Regression : 0.0647331169578209
SVM : 0.08180727044650482
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research'],
      dtype='object')
```

<u>Conclusion:</u> Thus we have studied different classification techniques.

Assignment No.05: Clustering

Name of the Student: ________________________________ Roll no: ___

CLASS: - T.E. IT Subject Name: - LP-I Lab (Part I- ML)

Experiment No. 05

**Clustering Techniques: **

Marks: /10

Date of Performance: / /2022 Sign with Date:

Problem statement:

Assignment on Clustering Techniques

Download the following customer dataset from below link: Data Set: https://www.kaggle.com/shwetabh123/mall-customers

This dataset gives the data of Income and money spent by the customers visiting a Shopping Mall. The data set contains Customer ID, Gender, Age, Annual Income, and Spending Score. Therefore, as a mall owner you need to find the group of people who are the profitable customers for the mall owner. Apply at least two clustering algorithms (based on Spending Score) to find the group of customers.

a. Apply Data pre-processing (Label Encoding , Data Transformation….) techniques if necessary.
b. Perform data-preparation(Train-Test Split)
c. Apply Machine Learning Algorithm
d. Evaluate Model.
e. Apply Cross-Validation and Evaluate Model

Theory:

Approach of Clustering : Clustering or cluster analysis is a machine learning technique, which groups the unlabelled dataset. It can be defined as "A way of grouping the data points into different clusters, consisting of similar data points. The objects with the possible similarities remain in a group that has less or no similarities with another group

Applications of Clustering: Market Segmentation, Statistical data analysis, Social network analysis, Image segmentation**,** Anomaly detection, etc.

K-Means Clustering:

K-Means clustering is the most popular unsupervised learning algorithm. It is used when we have unlabelled data which is data without defined categories or groups. The algorithm follows an easy or simple way to classify a given data set through a certain number of clusters, fixed apriori.

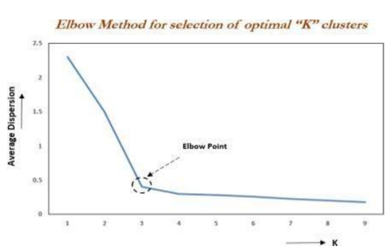
K-Means Algorithm:

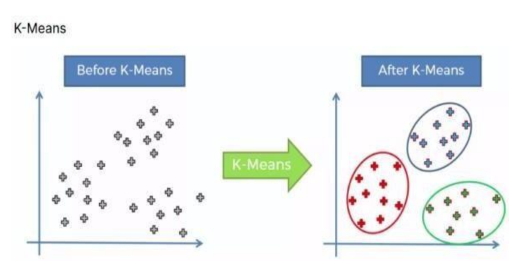
Step-1: Select the number K to decide the number of clusters.

Step-2: Select random K points or centroids. (It can be other from the input dataset).
Step-3: Assign each data point to their closest centroid, which will form the predefined Kclusters.
Step-4: Calculate the variance and place a new centroid of each cluster.
Step-5: Repeat the third steps, which means reassign each datapoint to the new closest centroid ofeach cluster.
Step-6: If any reassignment occurs, then go to step-4 else go to FINISH.
Step-7: The model is ready.

K-Means Clustering Intuition:

1. **Centroid:** A centroid is a data point at the centre of a cluster. In centroid-based clustering, clusters are represented by a centroid. The algorithm requires number of clusters K and the data set as input. The data set is a collection of features for each data point. The algorithm starts with initial estimates for the K centroids.
2. **Data Assignment Step:** Each centroid defines one of the clusters. In this step, each data point is assigned to its nearest centroid, which is based on the squared Euclidean distance. So, if ci is the collection of centroids in set C, then each data point is assigned to a cluster based on minimum Euclidean distance.
3. **Centroid update Step**: In this step, the centroids are recomputed and updated. This is done bytaking the mean of all data points assigned to that centroid's cluster.
4. **Choosing the value of K:** The K-Means algorithm depends upon finding the number of clustersand data labels for a pre-defined value of K. We should choose the optimal value of K that gives us best performance. There are different techniques available to find the optimal value of K. Themost common technique is the elbow method.
5. The elbow method: The elbow method is used to determine the optimal number of clusters in K-means clustering.





WCSS List: Elbow method uses the concept of WCSS value. **WCSS** stands for **Within Cluster Sum of Squares**, which defines the total variations within a cluster. To find the optimal value of clusters, the elbow method follows the below steps:

Python Implementation of K-means Clustering Algorithm

In [1]:

```
#importing required libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')

# Reading Data
```

In [2]:

```
#Loading data into dataframe
df = pd.read_csv("Mall_Customers.csv")
```

In [3]:

```
# first 5 instances of dataset
df.head()
```

Out[3]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

In [4]:

```
# last 5 instances of dataset
df.tail()
```

Out[4]:

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

In [5]:
```
#Shape of dataframe
df.shape
```
Out[5]:
(200, 5)

In [6]:
```
#columns in dataframe
df.columns
```
Out[6]:
```
Index(['CustomerID', 'Genre', 'Age', 'Annual Income (k$)',
       'Spending Score (1-100)'],
      dtype='object')
```

In [7]:
```
# droping Id column
df.drop("CustomerID",axis=1,inplace=True)
```
In [8]:
```
# Checking Dataset
df
```
Out[8]:

	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	Male	19	15	39
1	Male	21	15	81
2	Female	20	16	6
3	Female	23	16	77
4	Female	31	17	40
...	...	...	...	...
195	Female	35	120	79
196	Female	45	126	28
197	Male	32	126	74
198	Male	32	137	18
199	Male	30	137	83

200 rows × 4 columns

#Step 2. Pre-Processing of Data
In [9]:
#Find missing values
print("Missing values:")
df.isnull().sum()
Missing values:

Out[9]:
Genre 0
Age 0
Annual Income (k$) 0
Spending Score (1-100) 0
dtype: int64
From this we can come to know that there is no missing value in dataset.

In [10]:
 # describtion of dataset
df.describe()
Out[10]:

	Age	Annual Income (k$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000
mean	38.850000	60.560000	50.200000
std	13.969007	26.264721	25.823522
min	18.000000	15.000000	1.000000
25%	28.750000	41.500000	34.750000
50%	36.000000	61.500000	50.000000
75%	49.000000	78.000000	73.000000
max	70.000000	137.000000	99.000000

In [11]:
 # info about dataset
df.info()
Out[11]:
RangeIndex: 200 entries, 0 to 199

```
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Genre                   200 non-null    object
 1   Age                     200 non-null    int64
 2   Annual Income (k$)      200 non-null    int64
 3   Spending Score (1-100)  200 non-null    int64
dtypes: int64(3), object(1)
memory usage: 6.4+ KB
```

In [12]:

```
#no. of classes in Dataset
df.nunique()
```

Out[12]:

```
Genre                      2
Age                       51
Annual Income (k$)        64
Spending Score (1-100)    84
dtype: int64
```

In [13]:

```
#Correlation among dataset
df.corr()
```

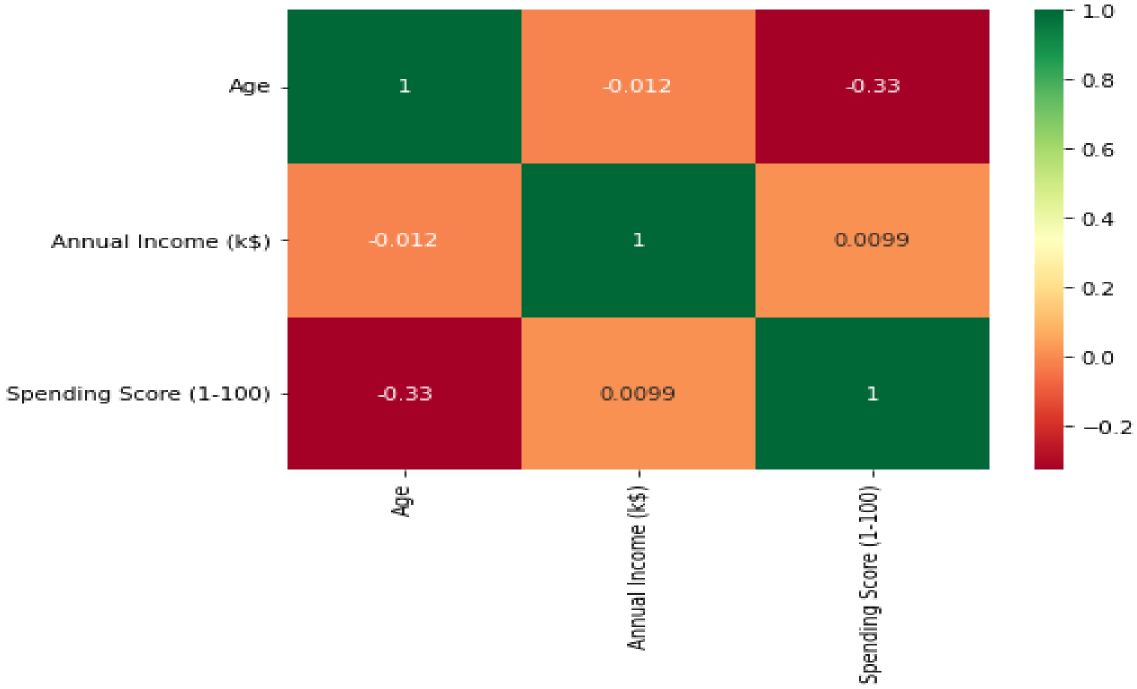
Out[13]:

	Age	Annual Income (k$)	Spending Score (1-100)
Age	1.000000	-0.012398	-0.327227
Annual Income (k$)	-0.012398	1.000000	0.009903
Spending Score (1-100)	-0.327227	0.009903	1.000000

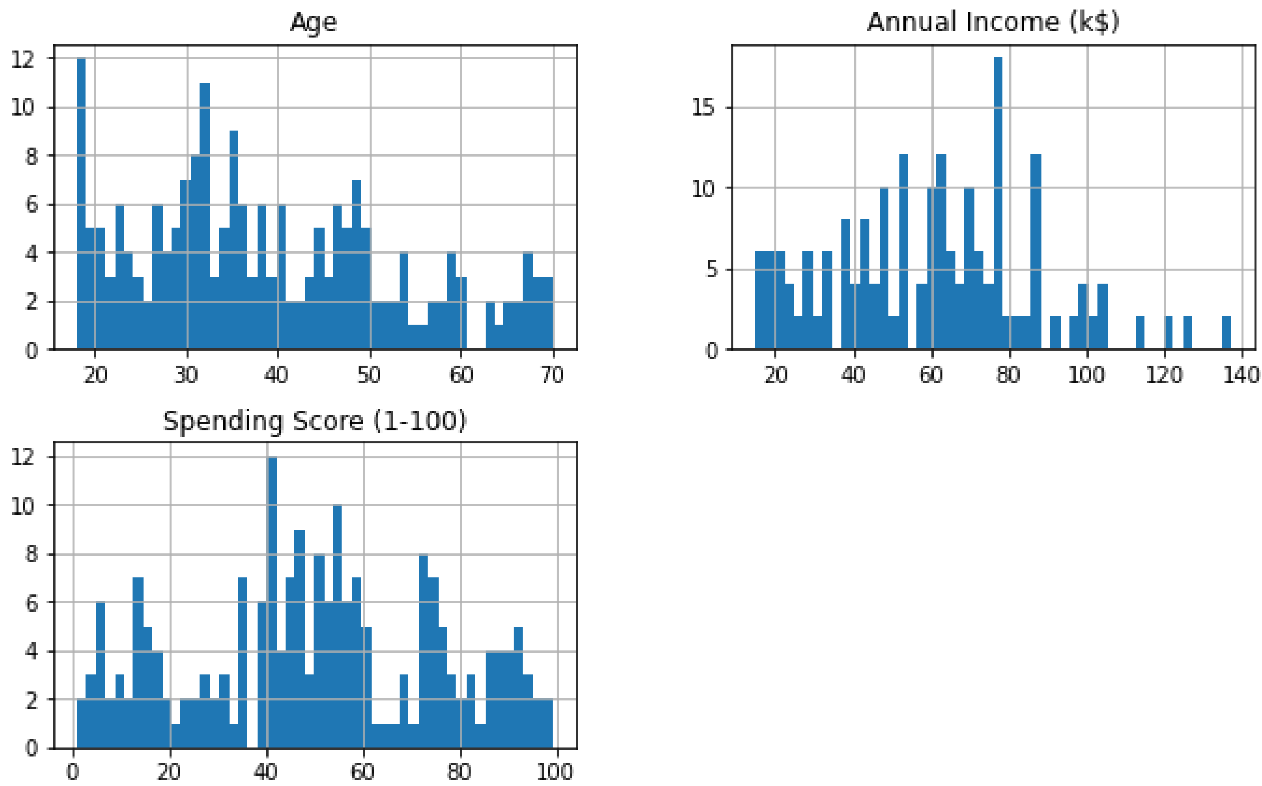
#Step 3.Visualizing Data

In [14]:

```
#Correlation heatmap
plt.figure(figsize=(7,5))
sns.heatmap(df.corr(), annot=True, cmap='RdYlGn')
plt.show()
```

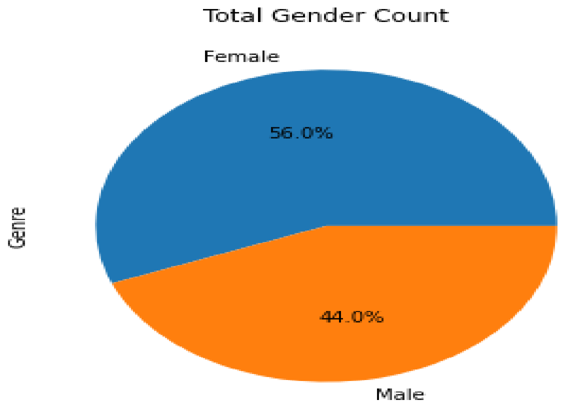


In [15]:
df.hist(bins = 50,figsize = (10,6))



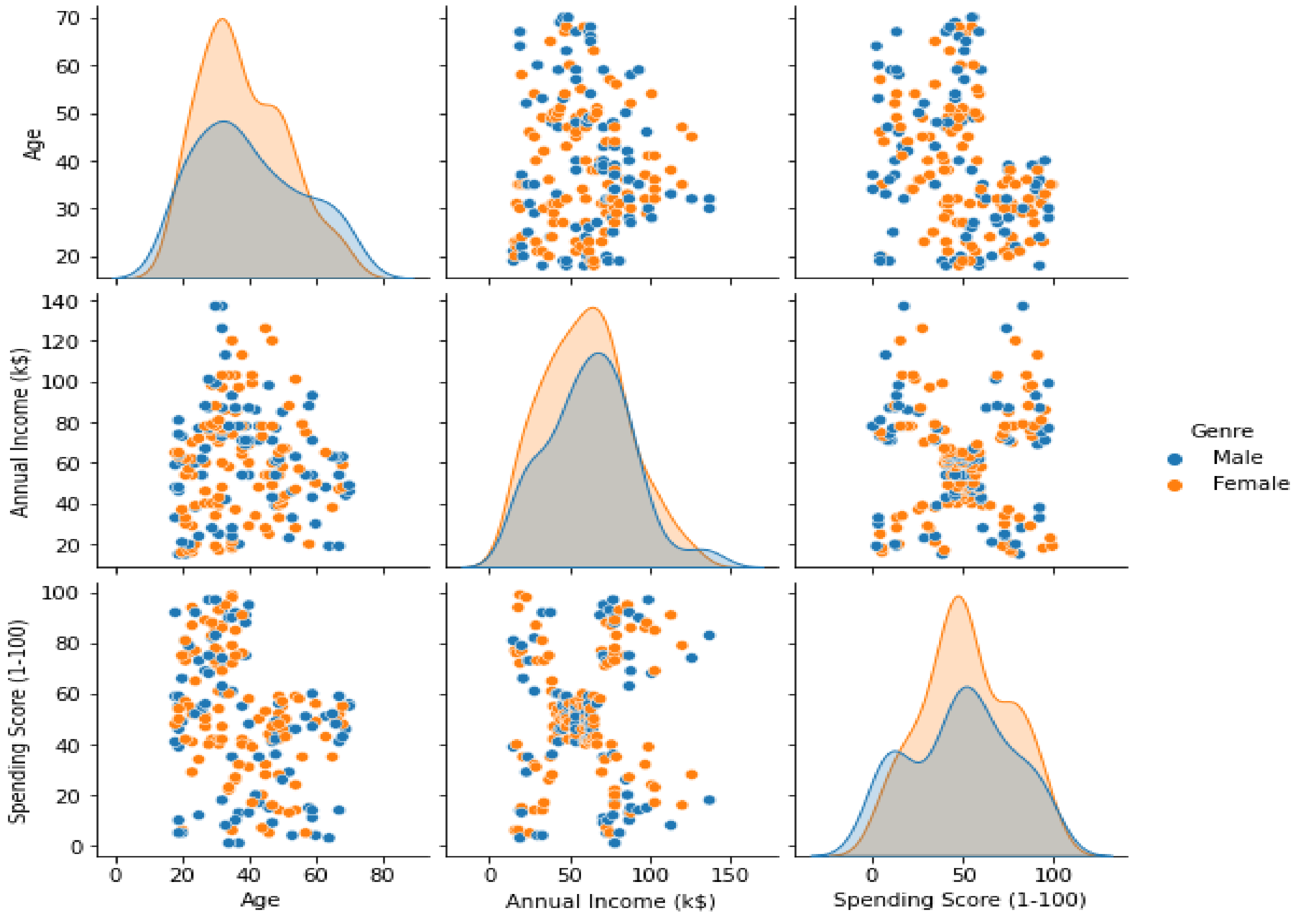
In [16]:

```
# pie chart for "Total Gender Count"
df['Genre'].value_counts().plot(kind='pie',figsize=(5,5),autopct='%1.1f%%')
plt.title("Total Gender Count")
plt.show()
```



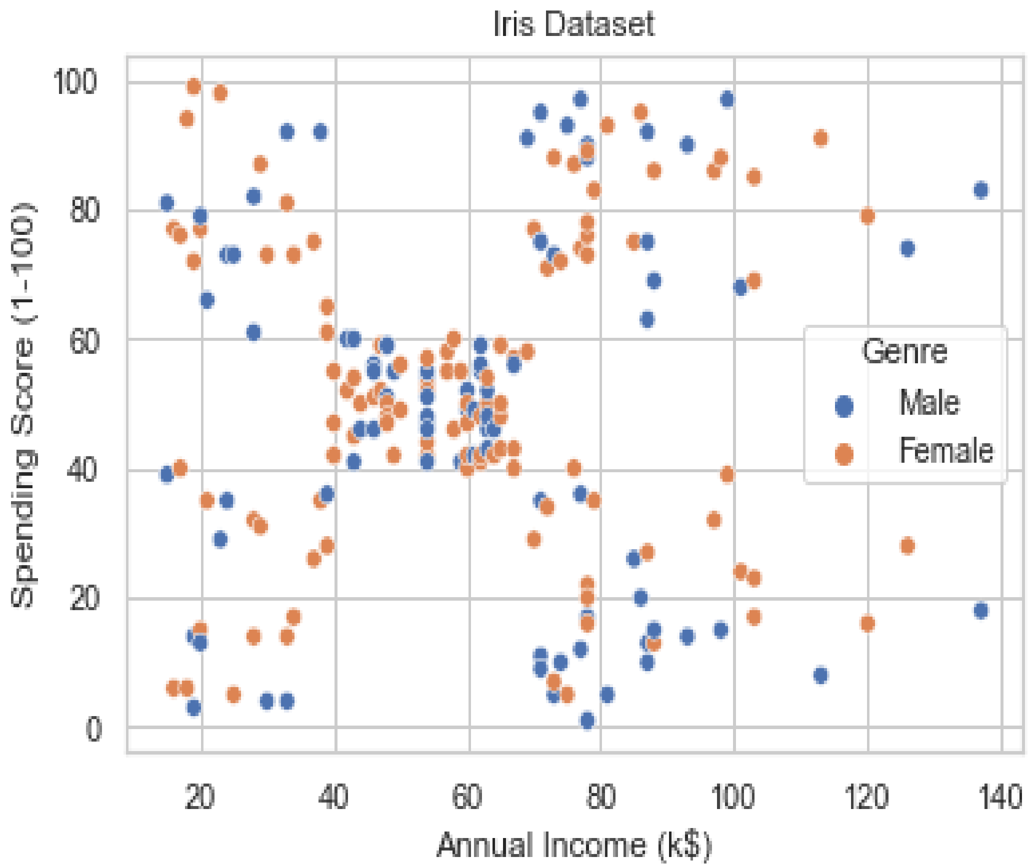
In [17]:

```
sns.pairplot(df,hue="Genre");
```



In [18]:

```
sns.set(style = 'whitegrid')
sns.scatterplot(y = 'Spending Score (1-100)',x ='Annual Income (k$)',data = df,hue= "Genre");
plt.title('Iris Dataset')
plt.show()
```



In [19]:

```
# LabelEncoder for encoding binary categories in a column
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
le = LabelEncoder()
# One single vector so it is ovbious what we want to encode
df["Genre"] = le.fit_transform(df["Genre"])
```

In [20]:
df
Out[20]:

	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	19	15	39
1	1	21	15	81
2	0	20	16	6
3	0	23	16	77
4	0	31	17	40
...	...	...	...	...
195	0	35	120	79
196	0	45	126	28
197	1	32	126	74
198	1	32	137	18
199	1	30	137	83

200 rows × 4 columns

#Step 4:Use of First clustering algorithm i.e.. K - Mean Clustering
In [21]:

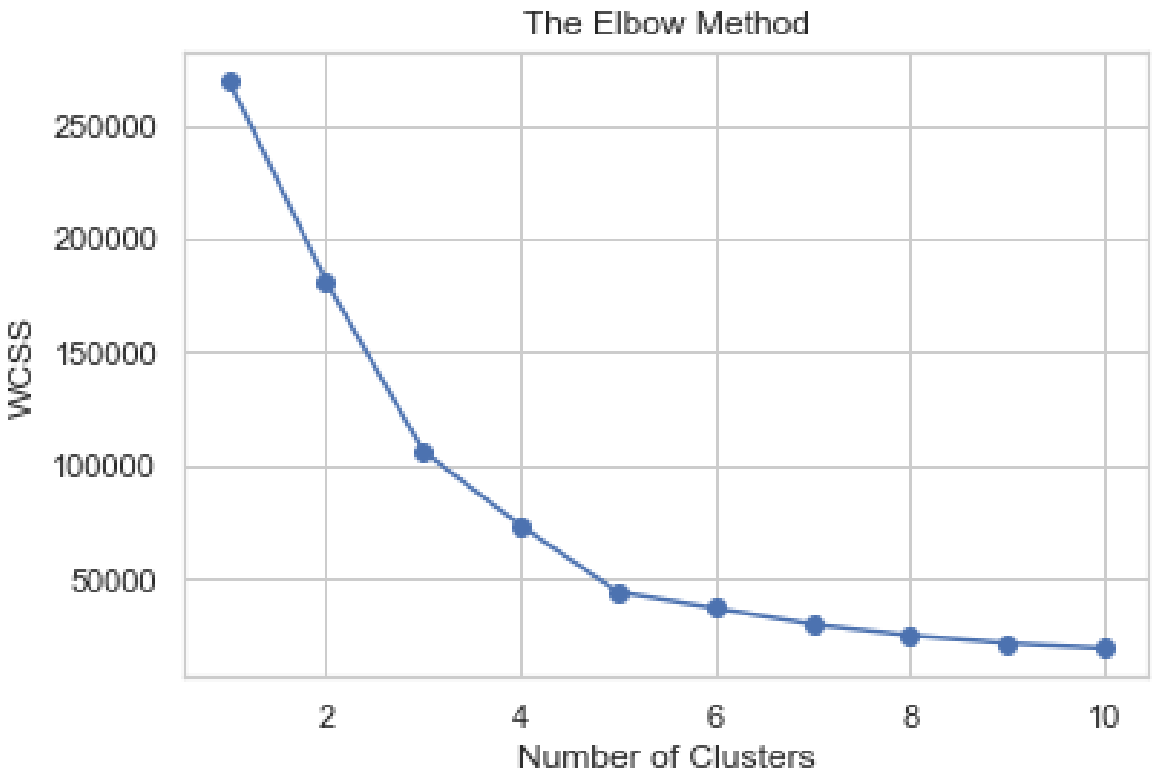
```
# Finding the optimum number of clusters using k-means
data = df.copy()
x = data.iloc[:,[2,3]]

#importing Kmean model
from sklearn.cluster import KMeans
wcss = []
for i in range(1,11):
   kmeans = KMeans(n_clusters=i, init='k-means++', random_state=42)
   kmeans.fit(x)
   # appending the WCSS to the list
   #(kmeans.inertia_ returns the WCSS value for an initialized cluster)
   wcss.append(kmeans.inertia_)
   print('k:',i ,"-> wcss:",kmeans.inertia_)
```

Out [21]:

```
k: 1 -> wcss: 269981.28000000014
k: 2 -> wcss: 181363.59595959607
k: 3 -> wcss: 106348.37306211119
k: 4 -> wcss: 73679.78903948837
k: 5 -> wcss: 44448.45544793369
k: 6 -> wcss: 37265.86520484345
k: 7 -> wcss: 30241.34361793659
k: 8 -> wcss: 25336.94686147186
k: 9 -> wcss: 21850.16528258562
k: 10 -> wcss: 19634.554629349972
```

In [22]:

```
# Plotting the results onto a line graph, allowing us to observe 'The elbow'

plt.plot(range(1,11),wcss,marker='o')
plt.title('The Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('WCSS')
plt.show()
```



In [23]:

```
#Taking 5 clusters
km1=KMeans(n_clusters=5)
#Fitting the input data
km1.fit(data)
#predicting the labels of the input data
y=km1.predict(data)
#adding the labels to a column named label
data["label"] = y
#The new dataframe with the clustering done
data.head()
```

Out[23]:

	Genre	Age	Annual Income (k$)	Spending Score (1-100)	label
0	1	19	15	39	4
1	1	21	15	81	2
2	0	20	16	6	4
3	0	23	16	77	2
4	0	31	17	40	4

In [24]:

```
#Scatterplot of the clusters
plt.figure(figsize=(6,4))
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="label",
                palette=['green','brown','orange','red','dodgerblue'],data = data )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
```

In [25]:
X=data.iloc[:,:4]
y=data.iloc[:,-1]
Splitting of Data
In [26]:
 # Splitting of dataset into train and test

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=42)
 # Shape of train Test Split
print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)
(160, 4) (160,)
(40, 4) (40,)

In [27]:
from sklearn.cluster **import** KMeans
km=KMeans(n_clusters=5)
km.fit(X_train)
#predicting the target value from the model for the samples
y_train_km = km.predict(X_train)
y_test_km = km.predict(X_test)

In [28]:

```
from sklearn.metrics.cluster import adjusted_rand_score

acc_train_gmm = adjusted_rand_score(y_train,y_train_km)
acc_test_gmm = adjusted_rand_score(y_test,y_test_km)

print("K mean : Accuracy on training Data: {:.3f}".format(acc_train_gmm))
print("K mean : Accuracy on test Data: {:.3f}".format(acc_test_gmm))
```

K mean : Accuracy on training Data: 0.982
K mean : Accuracy on test Data: 0.912

#Step 4:2. Hierarchical clustering

In [29]:

```
data = df.copy()
data = data.iloc[:,[2,3]]
data
```

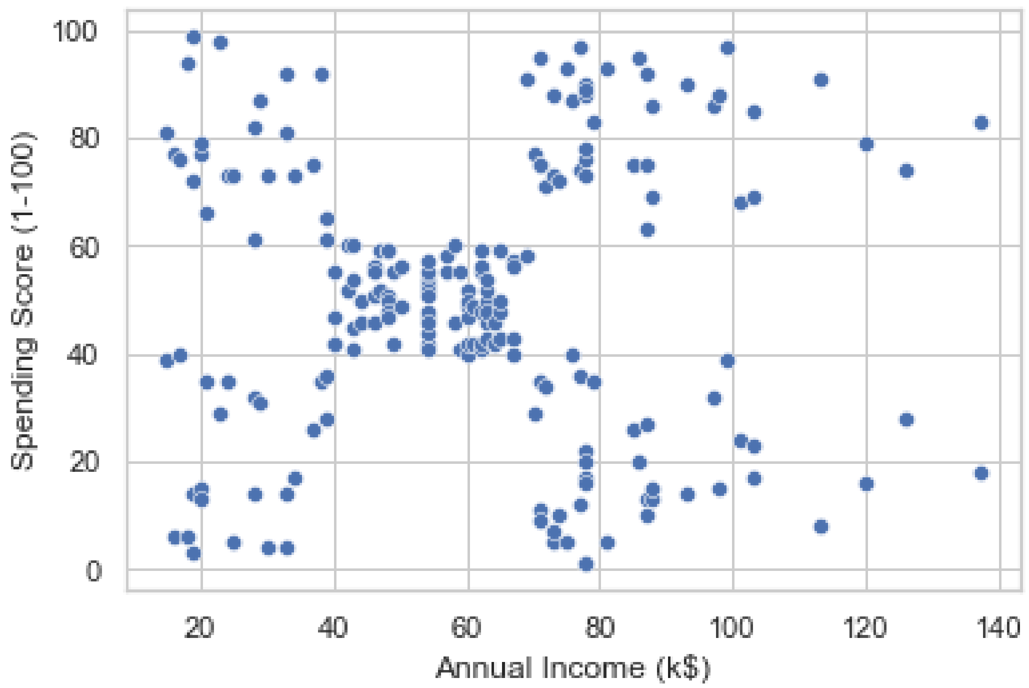
Out[29]:

	Annual Income (k$)	Spending Score (1-100)
0	15	39
1	15	81
2	16	6
3	16	77
4	17	40
...	...	...
195	120	79
196	126	28
197	126	74
198	137	18
199	137	83

200 rows × 2 columns

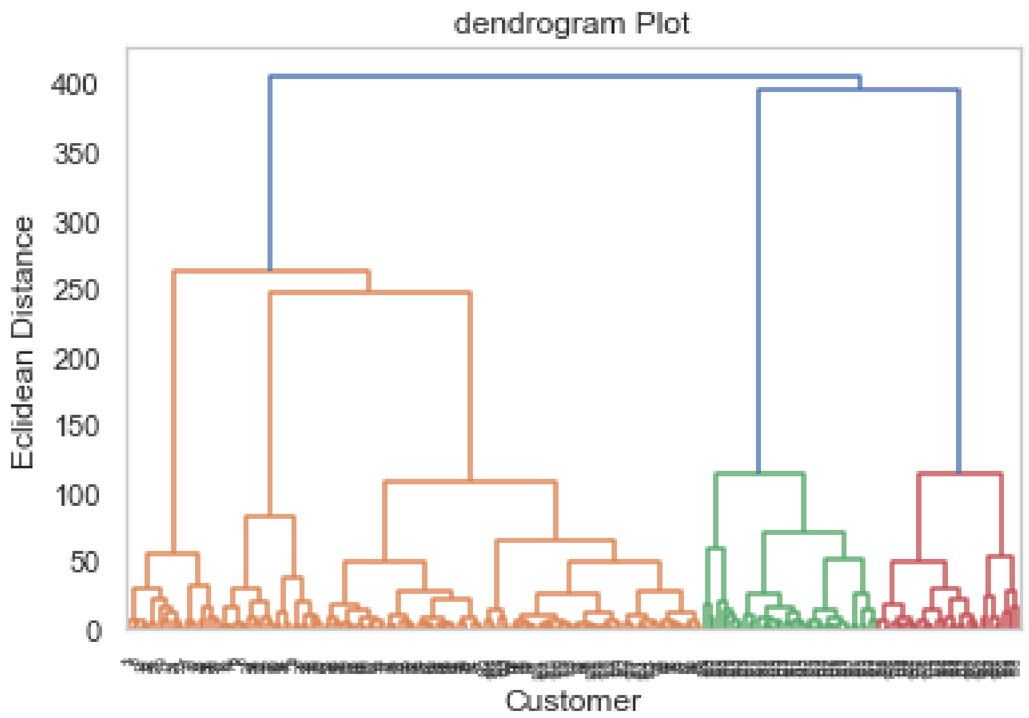
In [30]:

```
sns.scatterplot(x="Annual Income (k$)",y="Spending Score (1-100)",data = data );
```



In [31]:

```
import scipy.cluster.hierarchy as shc
dendrogram = shc.dendrogram(shc.linkage(data,method="ward"))
plt.title("dendrogram Plot")
plt.xlabel("Customer")
plt.ylabel("Eclidean Distance")
plt.grid(False)
```



In [32]:

```
from sklearn.cluster import AgglomerativeClustering
agc = AgglomerativeClustering(n_clusters=5)
data["label"] = agc.fit_predict(data)
data
```

Out[32]:

	Annual Income (k$)	Spending Score (1-100)	label
0	15	39	4
1	15	81	3
2	16	6	4
3	16	77	3
4	17	40	4
...	...	...	...
195	120	79	2
196	126	28	0
197	126	74	2
198	137	18	0
199	137	83	2

200 rows × 3 columns

In [34]:

```
#Scatterplot of the clusters
sns.scatterplot(x = 'Annual Income (k$)',y = 'Spending Score (1-100)',hue="label",
                palette=['green','brown','orange','red','dodgerblue'],data = data )
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.title('Spending Score (1-100) vs Annual Income (k$)')
plt.show()
```

Conclusion

1. There are 5 clusters in dataset.
2. Accuracy of model comes tobe 98% for training dataset and 91% on testing dataset.
3. Accuracy of testing dataset various with random_state value.
4. We have successfully find the group of customers.