DEPARTMENT OF INFORMATION TECHNOLOGY

LAB MANUAL

CLASS: TE-IT

SEMESTER: I

SUBJECT: LABORATORY PRACTICE-I (MACHINE LEARNING)

COURSE: 2019 PATTERN

ACDEMIC YEAR: 2021-22

INDEX

Vision

To equip students with core and state of the art Information Technologies.

Mission

Imparting knowledge of Information Technology and teaching its application through innovative practices and to instil high morale, ethics, lifelong learning skills, concern for the society and environment.

Program Educational Objectives

I. To prepare students to identify, formulate, and solve multifaceted and complex IT problems.
II. To teach core professional skills with latest information technologies that prepares students for immediate employment in Information Technology Industry.
III. To teach students soft skills that prepares them for leadership roles along diverse career paths.
IV. To make students aware of their social responsibilities in building the nation/society.

Program Specific Outcomes

1. Graduates will be able to demonstrate database, networking and programming technologies.
2. Graduates will be able to apply core professional state of the art Information Technology

Program Outcomes

Graduates will be able to

1. Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.**[Engineering knowledge]**
2. Identify, formulate, research literature, and analyse complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.**[Problem analysis]**

3. Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.**[Design/development of solutions]**
4. Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.**[Conduct investigations of complex problems]**
5. Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations. **[Modern tool usage]**
6. Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.**[The engineer and society]**
7. Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.**[Environment and sustainability]**
8. Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.**[Ethics]**
9. Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.**[Individual and team work]**
10. Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.**[Communication]**
11. Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.[Project management and finance]
12. Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.[Life-long learning]

Savitribai Phule Pune University, Pune
Third Year Information Technology (2019 Course)
314448 : Laboratory Practice-I (Machine Learning)

Teaching Scheme:	Credit Scheme:	Examination Scheme:
Practical (PR) : 4 hrs/week	02 Credits	PR : 25 Marks TW: 25 Marks

Prerequisites:

1. Python programming language

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.

Guidelines for Instructor's Manual

The faculty member should prepare the laboratory manual for all the experiments and it should bemade available to students and laboratory instructor/Assistant.

Guidelines for Student's Lab Journal

1. Students should submit term work in the form of a handwritten journal based on specified listof assignments.
2. Practical Examination will be based on the term work.
3. Students are expected to know the theory involved in the experiment.
4. The practical examination should be conducted if and only if the journal of the candidate is complete in all respects.

Guidelines for Lab /TW Assessment

HOME

1. Examiners will assess the term work based on performance of students considering the parameters such as timely conduction of practical assignment, methodology adopted for implementation of practical assignment, timely submission of assignment in the form of handwritten write-up along with results of implemented assignment, attendance etc.
2. Examiners will judge the understanding of the practical performed in the examination by asking some questions related to theory & implementation of experiments he/she has carried out.
3. Appropriate knowledge of usage of software and hardware related to respective laboratories shouldbe as a conscious effort and little contribution towards Green IT and environment awareness, attaching printed papers of the program in a journal may be avoided. There must be hand-written write-ups for every assignment in the journal. The DVD/CD containing student programs should be attached to the journal by every student and the same to be maintained by the department/lab In-charge is highly encouraged. For reference one or two journals may be maintained with program prints at Laboratory.

Guidelines for Laboratory Conduction

1. All the assignments should be implemented using python programming language
2. **Implement any 4 assignments out of 6**
3. **Assignment clustering with K-Means is compulsory**
4. The instructor is expected to frame the assignments by understanding the prerequisites, technological aspects, utility and recent trends related to the topic.
5. The instructor may frame multiple sets of assignments and distribute them among batches of students.
6. All the assignments should be conducted on multicore hardware and 64-bit open-sources software

Guidelines for Practical Examination

1. Both internal and external examiners should jointly set problem statements for practical examination. During practical assessment, the expert evaluator should give the maximum weightage to the satisfactory implementation of the problem statement.
2. The supplementary and relevant questions may be asked at the time of evaluation to judge the student ‘s understanding of the fundamentals, effective and efficient implementation.
3. The evaluation should be done by both external and internal examiners.

List of Laboratory Assignments

Sr.No.	Practical List
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%). 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 I. Accuracy II II. Precision III III. Recall IV. F-1 score
2	**Assignment on Regression technique** a. 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 This data consists of temperatures of INDIA averaging the temperatures of all places month wise. Temperatures values are recorded in CELSIUS temperature. b. Assess the performance of regression models using MSE, MAE and R-Square metrics c. Visualize simple regression model
3	**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.
4	**Assignment on Improving Performance of Classifier Models** 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. 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
5	**Assignment on Clustering Techniques** Download the following customer dataset from below link: Data Set: https://www.kaggle.com/shwetabh123/mall-customer 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, 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.
6	**Assignment on Association Rule Learning** Download Market Basket Optimization dataset from below link. Data Set: 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 Preprocessing 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** a. 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. b. Compile the model with necessary parameters. Set the number of epochs and batch size and fit the model. c. Evaluate the performance of the model for different values of epochs and batch sizes. d. Evaluate model performance using different activation functions Visualize the model using ANN Visualizer. 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”)

Assignment 1

Aim:

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%).

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.

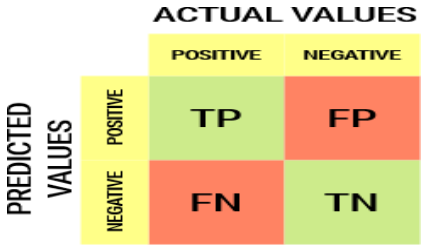
4. Feature Engineering: Feature engineering refers to the process of creating new input variables from the available data. Engineering new features is highly specific to your data and data types. As such, it often requires the collaboration of a subject matter expert to help identify new features that 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 data point is 1.
- **False Negatives (FN)** − It is the case when actual class of data point is 1 & predicted class of data point is 0.

Code:

```
#Imporiting Required Libraries

#converting an entire data table into a NumPy matrix array.

#data manipulation and analysis

import pandas as pd

importnumpy as np#array manipulation

importmatplotlib.pyplot as plt#graph plotting libraries

importseaborn as sns#data visualization and exploratory data analysis.

#making statistical graphics.

#%matplotlib inline

#Loading the data

#Data frame is a two-dimensional data structure,

#i.e., data is aligned in a tabular fashion in rows and columns.
```

```
df = pd.read_csv('heart.csv') # read csv file store into dataframedf
print(df.head(3)) # print first 3 row, if df print complete data
print() #print for spacing
#Features of the data set
print('Below are the features of dataset:')
df.info()
 #Details of Rows & Columns (Count, Datatypes, Null Values & Memory Usage)
#Dimensions of the dataset
print()
print('Below are the diamensions of dataset:')
#Shape method denotes count of rows &colums
print('Number of rows in the dataset: ',df.shape[0])
print('Number of columns in the dataset: ',df.shape[1])
#Checking for null values in the dataset
print()
print('Checking for null values in the dataset:')
print(df.isnull().sum()) #Field has no value present
#There are no null values in the dataset
print(df.describe())
#The features described in the above data set are:
#1. Count tells us the number of NoN-empty rows in a feature.
#2. Mean tells us the mean value of that feature.
#3. Std tells us the Standard Deviation Value of that feature.
```

```
#4. Min tells us the minimum value of that feature.
#5. 25%, 50%, and 75% are the percentile/quartile of each features.
#6. Max tells us the maximum value of that feature.
#Checking features of various attributes
#1. Sex -->
male =len(df[df['sex'] == 1]) #df=complete df #df = column in df
female = len(df[df['sex']== 0])
plt.figure(figsize=(8,6)) #8 by 6 inch
# Data to plot specifications
labels = 'Male','Female'
sizes = [male,female]
colors = ['skyblue', 'yellowgreen']
explode = (0, 0) # explode 1st slice don't separate
# Plot actual figure
#autopct: according len calculate percentage
#pie: show piechart according parameter
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)
plt.axis('equal') #x & y equal axis
plt.show()
#2. Chest Pain Type -->
plt.figure(figsize=(8,6))
# Data to plot
```

```
labels = 'Chest Pain Type:0','Chest Pain Type:1','Chest Pain Type:2','Chest Pain Type:3'
sizes = [len(df[df['cp'] == 0]),len(df[df['cp'] == 1]),
len(df[df['cp'] == 2]),
len(df[df['cp'] == 3])]
colors = ['skyblue', 'yellowgreen','orange','gold']
explode = (0, 0,0,0)  # explode 1st slice
# Plot specifications
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=180)
plt.axis('equal')
plt.show()
#3. fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
plt.figure(figsize=(8,6))
# Data to plot
labels = 'fasting blood sugar < 120 mg/dl','fasting blood sugar > 120 mg/dl'
sizes = [len(df[df['fbs'] == 0]),len(df[df['cp'] == 1])] #bp value
colors = ['skyblue', 'yellowgreen','orange','gold']
explode = (0.1, 0)  # explode 1st slice
# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=180)
plt.axis('equal')
plt.show()
```

```
#4.exang: exercise induced angina (1 = yes; 0 = no)
plt.figure(figsize=(8,6))
# Data to plot
labels = 'No','Yes'
sizes = [len(df[df['exang'] == 0]),len(df[df['exang'] == 1])]
colors = ['skyblue', 'yellowgreen']
explode = (0.1, 0) # explode 1st slice
# Plot
plt.pie(sizes, explode=explode, labels=labels, colors=colors,
autopct='%1.1f%%', shadow=True, startangle=90)
plt.axis('equal')
plt.show()
#Exploratory Data Analysis
sns.set_style('whitegrid') #set background white
#1. Heatmap
plt.figure(figsize=(14,8)) #14/8
#heatmap:Graphical representation of data that uses a system of
#color-coding to represent different values
#corr(): pairwise correlation of all columns in the dataframe
#annot: Value in each field bool or rectangular dataset
#cmap: Colourmap
sns.heatmap(df.corr(), annot = True, cmap='coolwarm',linewidths=.1)
plt.show()
```

```
#Plotting the distribution of various attributes
#1. thalach: maximum heart rate achieved
sns.distplot(df['thalach'],kde=False,bins=30,color='violet')
#2.chol: serum cholestoral in mg/dl
sns.distplot(df['chol'],kde=False,bins=30,color='red')
plt.show()
#3. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
sns.distplot(df['trestbps'],kde=False,bins=30,color='blue')
plt.show()
#4. Number of people who have heart disease according to age
plt.figure(figsize=(15,6))
sns.countplot(x='age',data = df, hue = 'target',palette='GnBu')
plt.show()
#5.Scatterplot for thalach vs. chol
plt.figure(figsize=(8,6))
sns.scatterplot(x='chol',y='thalach',data=df,hue='target')
plt.show()
#6.Scatterplot for thalach vs. trestbps
plt.figure(figsize=(8,6))
sns.scatterplot(x='trestbps',y='thalach',data=df,hue='target')
plt.show()
#Making Predictions
#Splitting the dataset into training and test set
```

```
X= df.drop('target',axis=1)
y=df['target']
fromsklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=.3,random_state=42)
#Preprocessing - Scaling the features
fromsklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_train = pd.DataFrame(X_train_scaled)
X_test_scaled = scaler.transform(X_test)
X_test = pd.DataFrame(X_test_scaled)
#1. k-NearestNeighor Algorithm
#ImplementingGridSearchCv to select best parameters and applying k-NN Algorithm
fromsklearn.neighbors import KNeighborsClassifier
fromsklearn.model_selection import GridSearchCV
knn =KNeighborsClassifier()
params = {'n_neighbors':list(range(1,20)),
    'p':[1, 2, 3, 4,5,6,7,8,9,10],
    'leaf_size':list(range(1,20)),
    'weights':['uniform', 'distance'] }
model = GridSearchCV(knn,params,cv=3, n_jobs=-1)
model.fit(X_train,y_train)
print(model.best_params_)          #print's parameters best values
```

```
#Making predictions
predict = model.predict(X_test)
#Checking accuracy
fromsklearn.metrics import accuracy_score,confusion_matrix
print()
print('Accuracy Score: ',accuracy_score(y_test,predict))
print('Using k-NN we get an accuracy score of: ',
round(accuracy_score(y_test,predict),5)*100,'%')
print()
#Confusion Matrix
class_names = [0,1]
fig,ax = plt.subplots()
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks,class_names)
plt.yticks(tick_marks,class_names)
cnf_matrix = confusion_matrix(y_test,predict)
print('Below is the confusion matrix')
print(cnf_matrix)
#create a heat map
sns.heatmap(pd.DataFrame(cnf_matrix), annot = True, cmap = 'YlGnBu',fmt = 'g')
ax.xaxis.set_label_position('top')
plt.tight_layout()
plt.title('Confusion matrix for k-Nearest Neighbors Model', y = 1.1)
```

```
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
#Classification report
fromsklearn.metrics import classification_report
print(classification_report(y_test,predict))
#Receiver Operating Characterstic(ROC) Curve
fromsklearn.metrics import roc_auc_score,roc_curve
#Get predicted probabilites from the model
y_probabilities = model.predict_proba(X_test)[:,1]
#Create true and false positive rates
false_positive_rate_knn,true_positive_rate_knn,threshold_knn = roc_curve(y_test,y_probabilities)
#Plot ROC Curve
plt.figure(figsize=(10,6))
plt.title('Revceiver Operating Characterstic')
plt.plot(false_positive_rate_knn,true_positive_rate_knn)
plt.plot([0,1],ls='--')
plt.plot([0,0],[1,0],c='.5')
plt.plot([1,1],c='.5')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.show()
#Calculate area under the curve
```

print(roc_auc_score(y_test,y_probabilities))

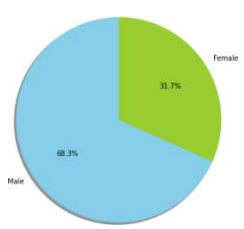
Output:

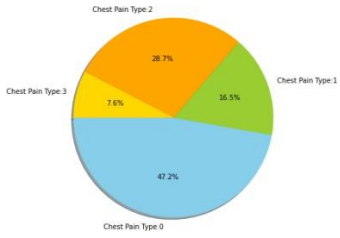
```
In [1]: runfile('C:/Users/Vrushali/Test_Project/Practicle_1.py', wdir='C:/Users/Vrushali/Test_Project'
   age  sex  cp  trestbps  chol  fbs  ...  exang  oldpeak  slope  ca  thal  target
0   63    1   3       145   233    1  ...      0      2.3      0   0     1       1
1   37    1   2       130   250    0  ...      0      3.5      0   0     2       1
2   41    0   1       130   204    0  ...      0      1.4      2   0     2       1

[3 rows x 14 columns]
```

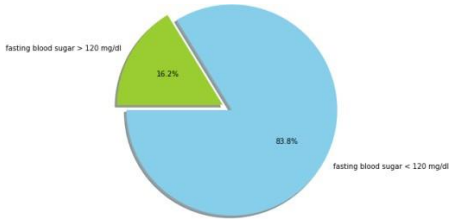
```
Below are the features of dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        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
```

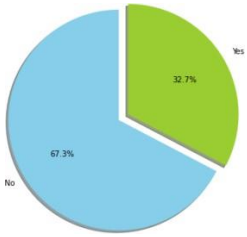
```
Below are the diamensions of dataset:
Number of rows in the dataset:  303
Number of columns in the dataset:  14

Checking for null values in the dataset:
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

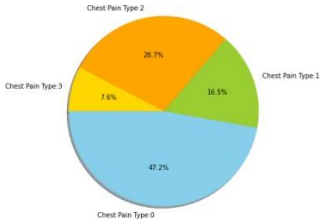
```
              age         sex          cp  ...          ca        thal      target
count  303.000000  303.000000  303.000000  ...  303.000000  303.000000  303.000000
mean    54.366337    0.683168    0.966997  ...    0.729373    2.313531    0.544554
std      9.082101    0.466011    1.032052  ...    1.022606    0.612277    0.498835
min     29.000000    0.000000    0.000000  ...    0.000000    0.000000    0.000000
25%     47.500000    0.000000    0.000000  ...    0.000000    2.000000    0.000000
50%     55.000000    1.000000    1.000000  ...    0.000000    2.000000    1.000000
75%     61.000000    1.000000    2.000000  ...    1.000000    3.000000    1.000000
max     77.000000    1.000000    3.000000  ...    4.000000    3.000000    1.000000

[8 rows x 14 columns]
```

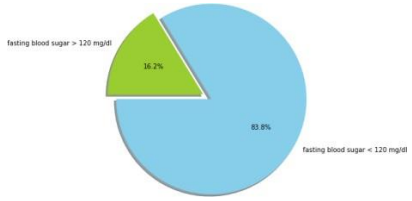

Female
31.7%
68.3%
Male

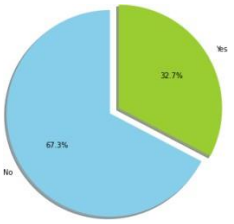

Chest Pain Type:2
28.7%
Chest Pain Type:3
7.6%
Chest Pain Type:1
16.5%
47.2%
Chest Pain Type:0


fasting blood sugar > 120 mg/dl
16.2%
83.8%
fasting blood sugar < 120 mg/dl

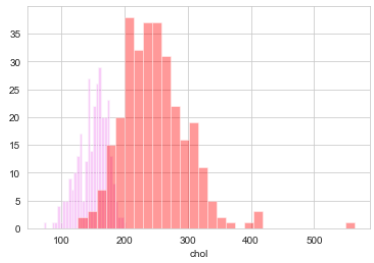

Yes
32.7%
67.3%
No

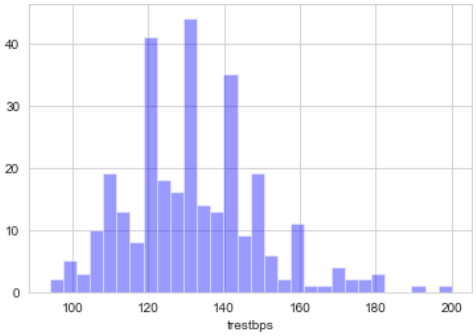

Chest Pain Type:2
28.7%
Chest Pain Type:3
7.6%
Chest Pain Type:1
16.5%
47.2%
Chest Pain Type:0

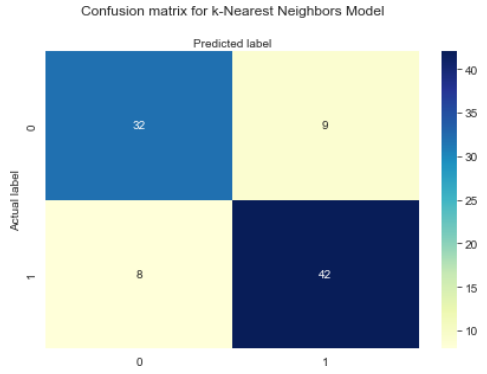

fasting blood sugar > 120 mg/dl
16.2%
83.8%
fasting blood sugar < 120 mg/dl


Yes
32.7%
67.3%
No


age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target


35
30
25
20
15
10
5
0
100
200
300
400
500
chol


40
30
20
10
0
100
120
140
160
180
200
trestbps


Confusion matrix for k-Nearest Neighbors Model
Predicted label
Actual label
32
9
8
42
0
1
0
1
40
35
30
25
20
15
10

```
Accuracy Score:  0.8131868131868132
Using k-NN we get an accuracy score of:  81.319 %

Below is the confusion matrix
[[32  9]
 [ 8 42]]
              precision    recall  f1-score   support

           0       0.80      0.78      0.79        41
           1       0.82      0.84      0.83        50

    accuracy                           0.81        91
   macro avg       0.81      0.81      0.81        91
weighted avg       0.81      0.81      0.81        91

0.8651219512195122
```

Conclusion: Thus we have studied different data preparation techniques.

ASSIGNMENT 2

Aim:

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 package

Step2: Importing dataset

Step3: Organizing data into training & testing sets

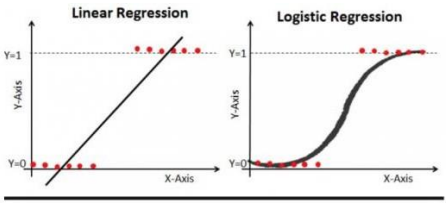
Step4: Model evaluation

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.



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:

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

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:

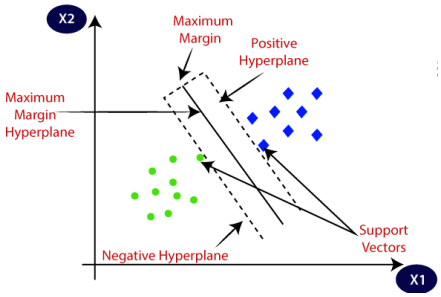
Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)

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

Code:

```
# To load the dataset

import pandas as pd

importmatplotlib.pyplot as plt

#seaborn: for data visualization and exploratory data analysis

importseaborn 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 have 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 persons 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

importnumpy 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
fromsklearn.tree import DecisionTreeRegressor
fromsklearn.ensemble import RandomForestRegressor
fromsklearn.svm import SVR
```

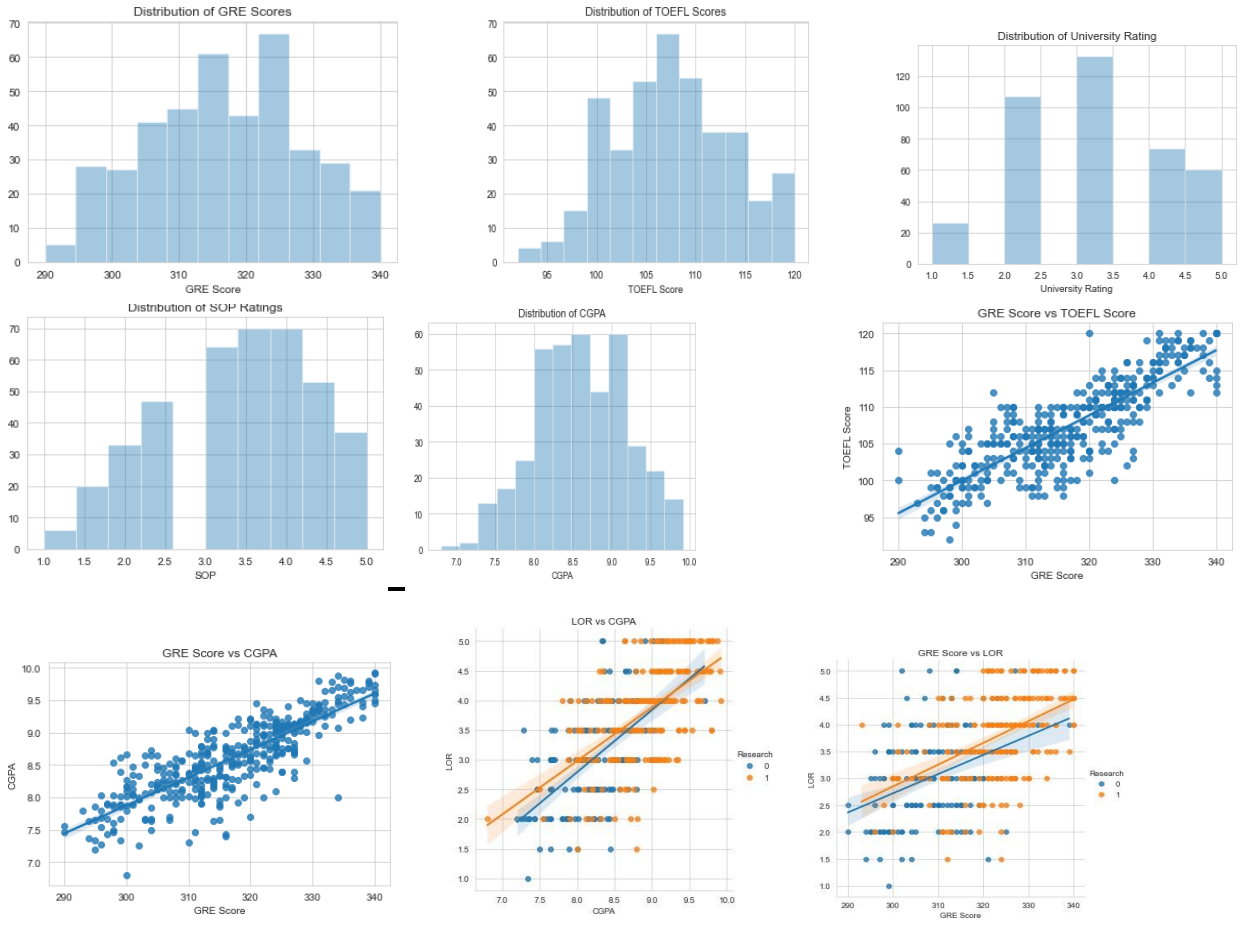
```
fromsklearn.linear_model import LinearRegression
fromsklearn.metrics import mean_squared_error
#These methods predict the future applicant's chances of admission.
models = [['DecisionTree :',DecisionTreeRegressor()],
        ['Linear Regression :', LinearRegression()],
        ['SVM :', SVR()]]
print("Results...")
#For loop for generating model results
forname,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()
```

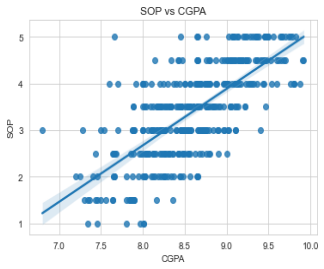
```
#Two Dimensional Array Format column names
importance_frame['Features'] = X.columns
#classifier.feature_importance is decision tree based on correlation value As per importance of admission
importance_frame['Importance'] = classifier.feature_importances_
#Sort the features by high to low bar graph
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()
```

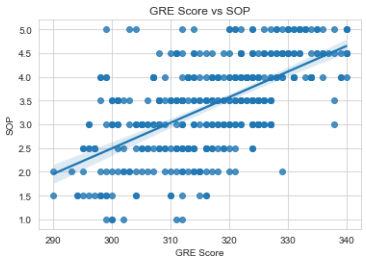
Output:

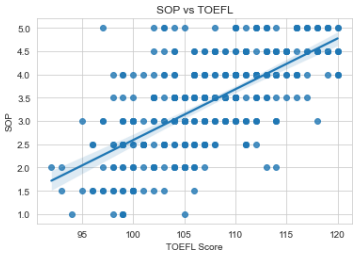
```
In [2]: runfile('C:/Users/Vrushali/Test_Project/Practicle_2.py', wdir='C:/Users/Vrushali/Test_Project')
   Serial No.  GRE Score  TOEFL Score  ...  CGPA  Research  Chance of Admit
0           1        337          118  ...  9.65         1              0.92
1           2        324          107  ...  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
```

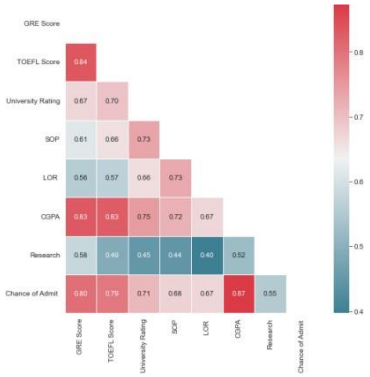
```
                   GRE Score  TOEFL Score  ...  Research  Chance of Admit
GRE Score           1.000000     0.835977  ...  0.580391         0.802610
TOEFL Score         0.835977     1.000000  ...  0.489858         0.791594
University Rating   0.668976     0.695590  ...  0.447783         0.711250
SOP                 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]
```

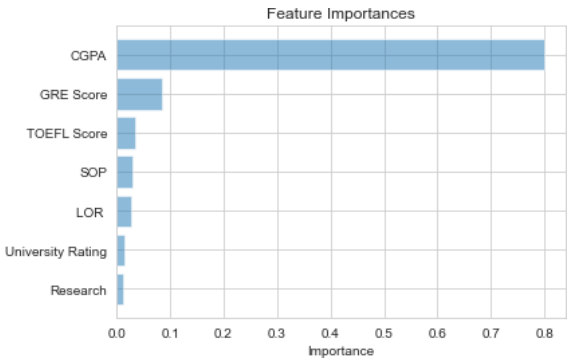













```
[8 rows x 8 columns]
Results...
DecisionTree : 0.09370832406995656
Linear Regression : 0.0647331169578209
SVM : 0.08180727044650482
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research'],
      dtype='object')
```

Conclusion: Thus we have studied different classification techniques.

ASSIGNMENT 3

Aim:

Assignment on Regression technique

a. 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
This data consists of temperatures of INDIA averaging the temperatures of all places month wise. Temperatures values are recorded in CELSIUS 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.

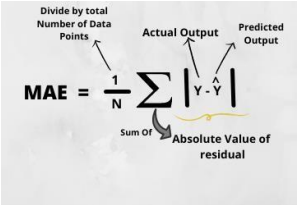
Outliers: Outlier is an observation which contains either very low value or very high value in comparison to other observed values. An outlier may hamper the result, so it should be avoided.

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 with test 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.



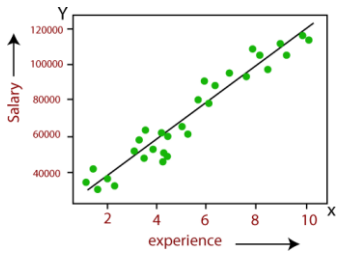
2. **Mean Squared Error(MSE):** Mean squared error states that finding the squared difference between actual and predicted value. we perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

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

3. **Root Mean Squared Error(RMSE):** As RMSE is clear by the name itself, that it is a simple square 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

Code:

```
#Importing required libraries
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np
#Reading the input dataset
trainData = pd.read_csv("temperatures.csv")
#Print first 10 records
print(trainData.head(n=10))
#Printing datatypes and columns in the dataset
#datatypes columwise
print("Below are the datatypes of columns:")
print(trainData.dtypes)
print()
#column names
```

```
print("Below are the columns in the dataset:")
print(trainData.columns)
print()
#describe min, max temp, count, std dev values
print("Descritive information about data set:")
print(trainData.describe())
#To check if dataset has null values or not
print(trainData.isnull().sum())
#To find top 10 temperature
#As per 'Annual col' find 'top 10' temp data
top_10_data = trainData.nlargest(10, "ANNUAL")
#Mentioned figure size
plt.figure(figsize=(14,12))
plt.title("Top 10 temperature records")
#In barplot x & y axis year & temp resp
sns.barplot(x=top_10_data.YEAR, y=top_10_data.ANNUAL)
#It is found that highest record of temperature is in 2016 roughly
#about 32 degree Celsius
#Analyse 2016 data
data_2016 = trainData[trainData["YEAR"]==2016]
 #x axis temp data in array format
xticks = np.array(data_2016[["JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP",
"OCT", "NOV", "DEC"]].values)
 #y axis months labels
yticks = ["JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV",
"DEC"]
#To plot the graph
#Mentioned figsize
plt.figure(figsize=(10,8))
#barh: xticks & yticks get and set the current tick locations and labels of the x & y-axis.
plt.barh(yticks,xticks[0])
plt.title("Month wise temperature data of 2016")
plt.xlabel("Temperature in degree celsius")
plt.ylabel("Month")
plt.show()
```

```
#From the above graph it is clear that May month recorded highest temperature around 35 degree celsius
#Genearate Regresion Model of Training & Testing Data
from sklearn import linear_model, metrics
#train data according columns
print(trainData.columns)
#x axis = year
X=trainData[["YEAR"]]
# y axis= month wise temp
Y=trainData[["JAN"]]
#import training & testing features
from sklearn.model_selection import train_test_split
#split data in training & testing part
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=1)
 print(len(X_train)) #lenth of x-train data

 print(len(X_test))  #length of y-testing data

print(trainData.shape) #Show total row & column (117,18)

 #Used Linear Regression Model Features to show data
reg = linear_model.LinearRegression()
print(X_train)
 #fit decision line in Regression Model
model = reg.fit(X_train, Y_train)
#Predict test data
Y_pred = model.predict(X_test)
#Year wise predtion data
print('predicted response:', Y_pred,  sep='\n')
#training regression model Scatter black color plots
plt.scatter(X_train, Y_train, color='black')
#Blue line indicate predicted training data
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()
```

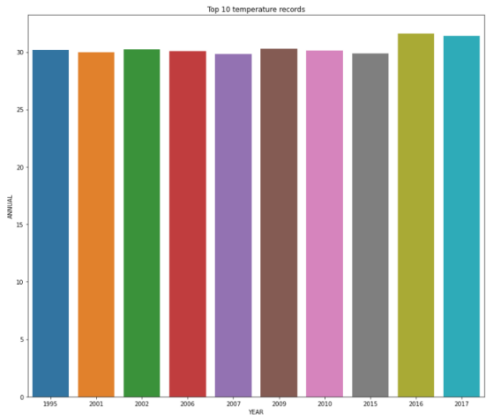
```
#testing regression model Scatter red color plots
plt.scatter(X_test, Y_test, color='red')
#Acc year machine predict temp
plt.plot(X_test, reg.predict(X_test), color='black', linewidth=3)
plt.title("Temperature vs Year")
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.show()
```

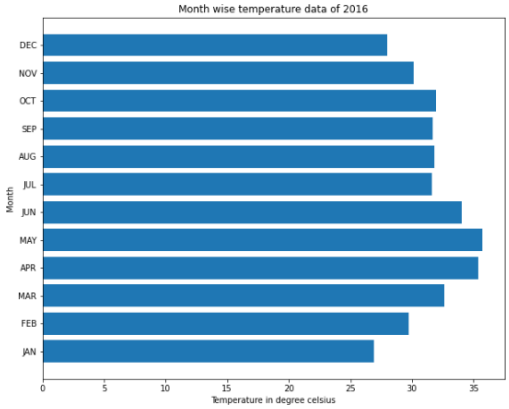
Output:

```
   YEAR    JAN    FEB    MAR    APR  ...  ANNUAL  JAN-FEB  MAR-MAY  JUN-SEP  OCT-DEC
0  1901  22.40  24.14  29.07  31.91  ...   28.96    23.27    31.46    31.27    27.25
1  1902  24.93  26.58  29.77  31.78  ...   29.22    25.75    31.76    31.09    26.49
2  1903  23.44  25.03  27.83  31.39  ...   28.47    24.24    30.71    30.92    26.26
3  1904  22.50  24.73  28.21  32.02  ...   28.49    23.62    30.95    30.66    26.40
4  1905  22.00  22.83  26.68  30.01  ...   28.30    22.25    30.00    31.33    26.57
5  1906  22.28  23.69  27.31  31.93  ...   28.73    23.03    31.11    30.86    27.29
6  1907  24.46  24.01  27.04  31.79  ...   28.65    24.23    29.92    30.80    27.36
7  1908  23.57  25.26  28.86  32.42  ...   28.83    24.42    31.43    30.72    26.64
8  1909  22.67  24.36  29.22  30.79  ...   28.38    23.52    31.02    30.33    26.88
9  1910  23.24  25.16  28.48  31.42  ...   28.53    24.20    31.14    30.48    26.20

[10 rows x 18 columns]
```

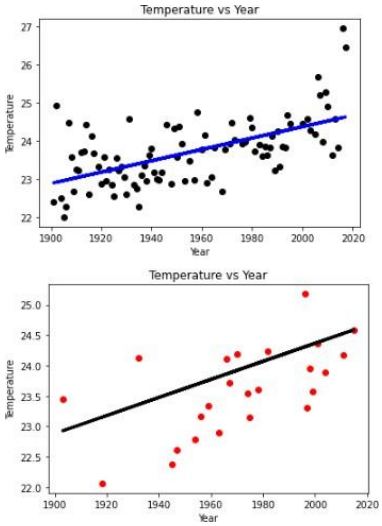
```
JAN        float64
FEB        float64
MAR        float64
APR        float64
MAY        float64
JUN        float64
JUL        float64
AUG        float64
SEP        float64
OCT        float64
NOV        float64
DEC        float64
ANNUAL     float64
JAN-FEB    float64
MAR-MAY    float64
JUN-SEP    float64
OCT-DEC    float64
dtype: object

Below are the columns in the dataset:
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')

Descritive information about data set:
              YEAR         JAN         FEB  ...     MAR-MAY     JUN-SEP     OCT-DEC
count   117.000000  117.000000  117.000000  ...  117.000000  117.000000  117.000000
mean   1959.000000   23.687436   25.597863  ...   31.517607   31.198205   27.208120
std      33.919021    0.834588    1.150757  ...    0.740585    0.420508    0.672003
min    1901.000000   22.000000   22.830000  ...   29.920000   30.240000   25.740000
25%    1930.000000   23.100000   24.780000  ...   31.040000   30.920000   26.700000
50%    1959.000000   23.680000   25.480000  ...   31.470000   31.190000   27.210000
75%    1988.000000   24.180000   26.310000  ...   31.890000   31.400000   27.610000
max    2017.000000   26.940000   29.720000  ...   34.570000   32.410000   30.030000

[8 rows x 18 columns]
```

```
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
```






```
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')
93
24
(117, 18)
```



Conclusion: Thus we have studied Regression techniques.

ASSIGNMENT 4

Aim:

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, 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.

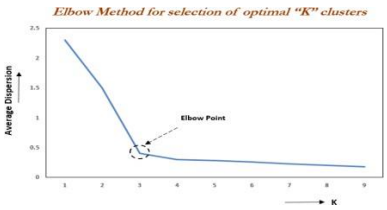
K-Means Algorithm:

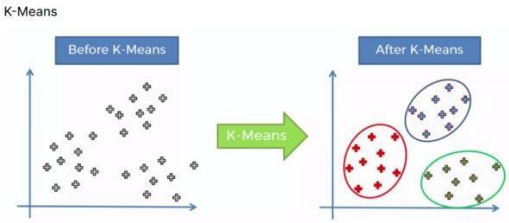
- **Step-1:** Select the number K to decide the number of clusters.

- **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 K clusters.
- **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 of each 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 by taking 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 clusters and 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. The most 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.





6. **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

1. Data Pre-processing
2. Finding the optimal number of clusters using the elbow method
3. Training the K-means algorithm on the training dataset
4. Visualizing the clusters

Code:

```
#Import required libraries

import pandas as pd

#  Visualization Library

import matplotlib.pyplot as plt

from matplotlib.lines import Line2D

# Scaling

from sklearn.preprocessing import StandardScaler

# Dimensional

from sklearn.decomposition import PCA

# Clustering

from sklearn.cluster import KMeans

#import numpy as np

#import seaborn as sns
```

```
#Load Data csv file
data = pd.read_csv('Mall_Customers.csv')
#Data Preprocessing Steps
print('For printing sample data:')
print(data.head())
print() #for creating blank space
print('To get total rows and columns:')
print(data.shape)
print()
#Column names, Count, Data Types, Null Values
print('To get info about columns:')
print(data.info())
print()
#Rename column name
data.rename(columns = {'Genre':'Gender'} , inplace = True)
#Describe Datasets
print(data.describe())
print()
#Drop useless columns
data.drop(labels = 'CustomerID' , axis = 1 , inplace = True)
#Missing values
print(data.isnull().sum())
print()
```

```
#Encoding finding data types of data present in csv
print(data.dtypes)
print()
#Find Gender Counts
print(data['Gender'].value_counts())
#Consider Male=1 & Female=0
data['Gender'].replace({'Male':1 , 'Female':0} , inplace = True)
print(data.info())
#Scaling
#Clustering algorithms such as K-means do need feature scaling before they are fed to the algorithm.
# Since, clustering techniques use Euclidean Distance to form the cohorts,
#it will be wise to scale the variables.
#Data coverted into normalization distribution
sc = StandardScaler()
data_scaled = sc.fit_transform(data)
#Dimensionality reduction
pca = PCA(n_components = 2)
data_pca = pca.fit_transform(data_scaled)
print("data shape after PCA :",data_pca.shape)
print("data_pca is:",data_pca)
# KMeans Clustering
''' Elbow plot Details : Finding optimal value of clusters
K is a hyperparameter in KMeans algorithm.
```

```
WCSS : Within Cluster Sum of Squares, in other word it's sum of squared
distance between each point and the centroid in a cluster
Lower WCSS shows a better clustering(because points in a cluster are more similar to each other,
 this is what we want)
Increasing the k value always results in a lower WCSS.
if we put k to be equal to the number of samples(so each point is a special cluster)
then WCSS = 0 , but this is not a wise way.
Here we will use elbow plot to find the best k.
Elbow point will show the best k.
How to find this point ?
After this point the speed of WCSS decreasing will be lowered. '"
#font size
plt_font = {'family':'serif' , 'size':16}
'"WCSS: Within Cluster Sum of Squares, in other word it's sum of squared
distance between each point and the centroid in a cluster
Lower WCSS shows a better clustering(because points in a cluster are more similar to each other,
 this is what we want)
Increasing the k value always results in a lower WCSS."'
#Create blank list
#Minimum no. of clusters & squared distance
wcss_list = []
for i in range(1, 15):
    kmeans = KMeans(n_clusters = i , init = 'k-means++'  , random_state = 1)
```

```
    kmeans.fit(data_pca)
    wcss_list.append(kmeans.inertia_)
#Draw Elbow plot
#X & Y axis range
plt.plot(range(1,15) , wcss_list)
plt.plot([4,4] , [0 , 500] , linestyle = '--' , alpha = 0.7)
#Elbow line
plt.text(4.2 , 300 , 'Elbow = 4')
#X & Y axis labels
plt.xlabel('K' , fontdict = plt_font)
plt.ylabel('WCSS' , fontdict = plt_font)
plt.show()
#KMeans Algorithm
kmeans = KMeans(n_clusters = 4 , init = 'k-means++' , random_state = 1)
kmeans.fit(data_pca)
cluster_id = kmeans.predict(data_pca)
result_data = pd.DataFrame()
result_data['PC1'] = data_pca[:,0]
result_data['PC2'] = data_pca[:,1]
result_data['ClusterID'] = cluster_id
#KMeans clustered ploting features
#cluster colors & tab details
cluster_colors = {0:'tab:red' , 1:'tab:green' , 2:'tab:blue' , 3:'tab:pink'}
```

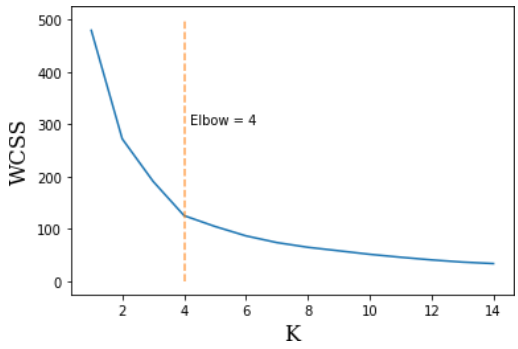
```
cluster_dict = {'Centroid':'tab:orange','Cluster0':'tab:red' , 'Cluster1':'tab:green'
        , 'Cluster2':'tab:blue' , 'Cluster3':'tab:pink'}
#Scatter data
#X & Y Value, result & cluster colors
plt.scatter(x = result_data['PC1'] , y = result_data['PC2']
        , c = result_data['ClusterID'].map(cluster_colors))
handles = [Line2D([0], [0], marker='o', color='w', markerfacecolor=v, label=k, markersize=8)
      for k, v in cluster_dict.items()]
plt.legend(title='color', handles=handles, bbox_to_anchor=(1.05, 1), loc='upper left')
plt.scatter(x = kmeans.cluster_centers_[:,0] , y = kmeans.cluster_centers_[:,1] ,
       marker = 'o' , c = 'tab:orange', s = 150 , alpha = 1)
#Heading details
plt.title("Clustered by KMeans" , fontdict = plt_font)
plt.xlabel("PC1" , fontdict = plt_font)
plt.ylabel("PC2" , fontdict = plt_font)
#Show all data
plt.show()
```

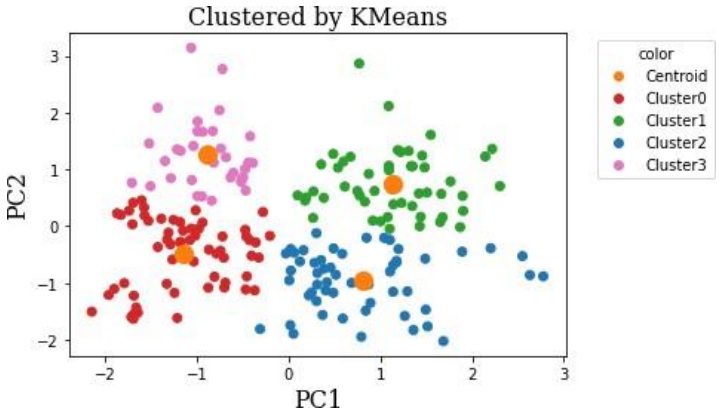
Output:

```
For printing sample data:
   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

To get total rows and columns:
(200, 5)

To get info about columns:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              200 non-null    int64
 1   Genre                   200 non-null    object
 2   Age                     200 non-null    int64
 3   Annual Income (k$)      200 non-null    int64
 4   Spending Score (1-100)  200 non-null    int64
dtypes: int64(4), object(1)
memory usage: 7.9+ KB
None
```

```
       CustomerID         Age  Annual Income (k$)  Spending Score (1-100)
count  200.000000  200.000000          200.000000              200.000000
mean   100.500000   38.850000           60.560000               50.200000
std     57.879185   13.969007           26.264721               25.823522
min      1.000000   18.000000           15.000000                1.000000
25%     50.750000   28.750000           41.500000               34.750000
50%    100.500000   36.000000           61.500000               50.000000
75%    150.250000   49.000000           78.000000               73.000000
max    200.000000   70.000000          137.000000               99.000000

Gender                    0
Age                       0
Annual Income (k$)        0
Spending Score (1-100)    0
dtype: int64

Gender                    object
Age                        int64
Annual Income (k$)         int64
Spending Score (1-100)     int64
dtype: object

Female    112
Male       88
Name: Gender, dtype: int64
```

```
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Gender                  200 non-null    int64
 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(4)
memory usage: 6.4 KB
None
data shape after PCA : (200, 2)
data_pca is: [[-4.06382715e-01 -5.20713635e-01]
```





Conclusion: Thus we have studied Clustering techniques.

ASSIGNMENT 5

Aim:

Assignment of exploring Machine Learning libraries.

Demonstrate multiple methods of Machine Learning libraries like NumPy & Pandas. Perform multiple operations on given dataset. Below is the link of dataset

https://olympus.greatlearning.in/courses/10899/files/1753546?module_item_id=903335

Theory:

- **NumPy:**

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Operations using NumPy:

Using NumPy, a developer can perform the following operations −

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Methods of Numpy:

1. **np.array:** This method useful for creating one & multidimensional array.

```
import numpy as np
n1=np.array([10,20,30,40])
n1
```

```
import numpy as np
n2=np.array([[10,20,30,40],[40,30,20,10]])
n2
```

2. **np.zeros:** Returns a new array of specified size, filled with zeros.

```
import numpy as np
n1=np.zeros((1,2))
n1
```

```
import numpy as np
n1=np.zeros((5,5))
n1
```

3. **np.arrange:** Return records in given order.

```
import numpy as np
n1=np.arange(10,20) n1
```

4. **np.full:** Return full array according given dimensions.

```
 import numpy as np
n1=np.full((2,2),10)
n1
```

5. **np.random.randint:** Return random values in given range.

```
import numpy as np
n1=np.random.randint(1,100,5)
n1
```

6. **n1.shape:** Return number of rows & columns in given array.

```
   n1.shape
```

7. **np.sum:** Return sum of the array.

```
import numpy as np
n1=np.array([10,20])
n2=np.array([30,40])
np.sum([n1,n2])
```

8. **np.equal:** Check similarities of values in given array.

```
import numpy as np
n1=np.array([10,20,30])
n2=np.array([10,30,20])
   np.equal(n1,n2)
```

9. **np.vstack/ np.hstack/np.column_stack**: Return multiple arrays in vertical, horizontal & column format.

```
import numpy as np
n1=np.array([10,20,30])
n2=np.array([40,50,60])

np.vstack((n1,n2))
```

```
import numpy as np
n1=np.array([10,20,30])
n2=np.array([40,50,60])

np.hstack((n1,n2))
```

```
import numpy as np
n1=np.array([10,20,30])
n2=np.array([40,50,60])

np.column_stack((n1,n2))
```

10. NumPy Manipulations:

```
#Division:
import numpy as np
n1=np.array([10,20,30])
n1=n1/2
n1
```

```
#Intersection:
import numpy as np
n1=np.array([10,20,30,40,50,60])
n2=np.array([50,60,70,80,90])

np.intersect1d(n1,n2)
```

```
#Difference:
import numpy as np
n1=np.array([10,20,30,40,50,60])
n2=np.array([50,60,70,80,90])
np.setdiff1d(n1,n2)
```

- **PANDAS:**

Pandas is a Python library used for working with data sets. It has functions for analyzing, cleaning, exploring, and manipulating data. The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis" and was created by Wes McKinney in 2008.

Why Use Pandas? A panda allows us to analyze big data and make conclusions based on statistical theories. Pandas can clean messy data sets, and make them readable and relevant.

Methods of Pandas:

1. **pd.Series:** A Pandas Series is like a column in a table. It is a one-dimensional array holding data of any type. With the index argument, you can name your own labels.

```
import pandas as pd
s1=pd.Series([1,2,3,4,5])
s1
```

```
import pandas as pd
s1=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
s1
```

```
import pandas as pd
pd.Series({'a':10,'b':20,'c':30})
```

2. **pd.DataFrame:** A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

```
import pandas as pd
pd.DataFrame({"Name":['Bob','Sam','Anne'],"Marks":[76,25,92]})
```

3. **pd.read_csv:** A simple way to store big data sets is to use CSV files (comma separated files). CSV files contain plain text and are a well know format that can be read by everyone including Pandas.

```
import pandas as pd
iris=pd.read_csv('iris.csv')
```

4. **head():** Returns the headers and a specified number of rows, starting from the top.

```
iris.head()
```

5. **tail():** Returns the headers and a specified number of rows, starting from the bottom.

```
iris.tail()
```

6. **shape():** Return number of rows & colums count.

```
iris.shape
```

7. **describe():** Return more information about the data set

```
iris.describe()
```

8. **min():** Return minimum value in each columns.

```
iris.min()
```

9. **max():** Return maximum value in each columns.

```
iris.max()
```

10.iloc[]: Return one or more specified row(s)

```
iris.iloc[0:3,0:2]
```

Output:

```
[8] import numpy as np

    n1=np.array([10,20,30,40])
    n1

    array([10, 20, 30, 40])
```

```
[9] import numpy as np

    n2=np.array([[10,20,30,40],[40,30,20,10]])
    n2

    array([[10, 20, 30, 40],
           [40, 30, 20, 10]])
```

```
[10] type(n2)

     numpy.ndarray
```

```
[11] n1[0]

     10
```

```
[13] import numpy as np
     n1=np.zeros((1,2))
     n1

     array([[0., 0.]])
```

```
[14] import numpy as np
     n1=np.zeros((5,5))
     n1

     array([[0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0.]])
```

```
[15] import numpy as np
     n1=np.arange(10,20)
     n1

     array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

```
[16] import numpy as np
     n1=np.full((2,2),10)
     n1

     array([[10, 10],
            [10, 10]])
```

```
[18] import numpy as np
     n1=np.random.randint(1,100,5)
     n1

     array([36, 87,  1, 33, 11])
```

```
[19] n1.shape

     (5,)
```

```
[20] import numpy as np
     n1=np.array([10,20,30])
     n2=np.array([40,50,60])

     np.vstack((n1,n2))

     array([[10, 20, 30],
            [40, 50, 60]])
```

```
[21] import numpy as np
     n1=np.array([10,20,30])
     n2=np.array([40,50,60])

     np.hstack((n1,n2))

     array([10, 20, 30, 40, 50, 60])
```

```
[22] import numpy as np
     n1=np.array([10,20,30])
     n2=np.array([40,50,60])

     np.column_stack((n1,n2))

     array([[10, 40],
            [20, 50],
            [30, 60]])
```

```
[23] import numpy as np
     n1=np.array([10,20,30])
     n1=n1/2
     n1

     array([ 5., 10., 15.])
```

```
     import numpy as np
     n1=np.array([10,20,30,40,50,60])
     n2=np.array([50,60,70,80,90])
     np.intersect1d(n1,n2)

     array([50, 60])
```

```
[25] np.setdiff1d(n1,n2)

     array([10, 20, 30, 40])
```

```
[ ] n1.shape

     (5,)
```

```
[2] import numpy as np
    n1=np.array([10,20])
    n2=np.array([30,40])

    np.sum([n1,n2])

    100
```

```
    import numpy as np
    n1=np.array([10,20,30])
    n2=np.array([10,30,20])

    np.equal(n1,n2)

    array([ True, False, False])
```

```
[1] import pandas as pd
    s1=pd.Series([1,2,3,4,5])
    s1

    0    1
    1    2
    2    3
    3    4
    4    5
    dtype: int64
```

```
[2] import pandas as pd
    s1=pd.Series([1,2,3,4,5],index=['a','b','c','d','e'])
    s1

    a    1
    b    2
    c    3
    d    4
    e    5
    dtype: int64
```

```
[3] import pandas as pd
    pd.Series({'a':10,'b':20,'c':30})

    a    10
    b    20
    c    30
    dtype: int64
```

```
[4] import pandas as pd

    pd.DataFrame({"Name":['Bob','Sam','Anne'],"Marks":[76,25,92]})
```

	Name	Marks
0	Bob	76
1	Sam	25
2	Anne	92

```
[5] from google.colab import files
    uploaded = files.upload()
```

Choose Files iris.csv

- **iris.csv**(application/vnd.ms-excel) - 3867 bytes, last modified: 11/24/2021 - 100% done

Saving iris.csv to iris.csv

```
[8] import pandas as pd

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

```
[9] iris.head()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
[10] iris.tail()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

```
[11] iris.shape

(150, 5)
```

```
[12] iris.describe()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333
std	0.828066	0.435866	1.765298	0.762238
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

```
[13] iris.min()

Sepal.Length       4.3
Sepal.Width          2
Petal.Length         1
Petal.Width        0.1
Species         setosa
dtype: object
```

```
[14] iris.max()

Sepal.Length          7.9
Sepal.Width           4.4
Petal.Length          6.9
Petal.Width           2.5
Species         virginica
dtype: object
```

```
[16] iris.head()
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
iris.iloc[0:3,0:2]
```

	Sepal.Length	Sepal.Width
0	5.1	3.5
1	4.9	3.0
2	4.7	3.2

Conclusion: Thus we have studied differe

nt methods in machine learning libraries.