```
# Assignment 1 : Data preparation
# Download heart dataset from following link.
# https://www.kaggle.com/zhaoyingzhu/heartcsv

# Perform following operation on given dataset.
#     1. Find Shape of Data
#     2. Find Missing Values
#     3. Find data type of each column
#     4. Finding out Zero's
#     5. Find Mean age of patients
#     6. 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

import numpy as np
import pandas as pd

dataFrame=pd.read_csv('/content/Heart.csv')

dataFrame.shape # shape

dataFrame.info()

dataFrame.head()

dataFrame.dtypes # datatype

dataFrame.isnull() # missing values

dataFrame.isnull().sum() # missing values : count

dataFrame.Age.mean() #mean of age

count = (dataFrame['Age'] == 0).sum() # counting zeros
count

# selected columns
var=dataFrame.loc[:,['Age','Sex','ChestPain','RestBP','Chol']]
var

# Splitting the dataset into train and test sets: 75-25 split
from sklearn.model_selection import train_test_split

X_train, X_test = train_test_split(var, test_size = 0.25, random_state = 42)

X_train.shape, X_test.shape
```

```
# Find accuracy and precision for given example
tp=45
fp=55
fn=05
tn=395
acc=(tp+tn)/(tp+fp+fn+tn)
pre=tp/(tp+fp)
rec=tp/(tp+fn)
print("Accuracy is : {}".format(acc))
print("Precision is : {}".format(pre))
print("Recall is : {}".format(rec))
print("F1-Score is : {}".format((2*pre*rec)/(pre+rec)))
```

```python
# Assignment 2 : Regression Technique
# 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

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

import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

from sklearn.metrics import mean_squared_error,mean_absolute_error,r2_score

temp_dataset = pd.read_csv('temperatures.csv')

temp_dataset.head()
temp_dataset.shape

temp_dataset.isnull().sum()

# model for monthe JAN

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

X=temp_dataset[["YEAR"]] # input column
y=temp_dataset["JAN"]         # target column

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
print(X_train.shape, X_test.shape)

# instantiate the model
lr = LinearRegression()

# fit the model
lr.fit(X_train, y_train)

#predicting the target value from the model for the samples
y_test_lr = lr.predict(X_test)
y_train_lr = lr.predict(X_train)

print("Intercept",lr.intercept_)
print("Slope",lr.coef_)

#computing the accuracy of the model performance
print("Linear Regression: Accuracy on test Data: {:.3f}".format(lr.score(X_test, y_test)))
```

```
#visulaize annaul temperature
plt.plot(X,y)
plt.xlabel("Year")
plt.ylabel("Temperature")
plt.title("Annual Temperature from 1901-2017")
plt.show()

# plot the regression line for the month januwary
#plt.figure(figsize=(8, 6))
plt.scatter(X_test,y_test,color = 'blue');
plt.scatter(X_train,y_train,color = 'red');
plt.plot(X_train,lr.predict(X_train), color = 'black');
plt.legend(['Best fit Regression lIne','Testing Set','Training Set'])
plt.title('Temperature vs Year for month Jan')
plt.xlabel('Year')
plt.ylabel('Temperature')
plt.show();

#Errors for month Jan
print('R-Squared Error :',r2_score(y_test,y_test_lr))
print('Mean Absolute Error :',mean_absolute_error(y_test,y_test_lr))
print('Mean Squared Error :',mean_squared_error(y_test,y_test_lr))
print('Root Mean Squared Error :',np.sqrt(mean_squared_error(y_test,y_test_lr)))
```

# Assignment 3 : Classification technique

```
# 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 (i.e. class column) .

# 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 ifnecessary.
# Perform data-preparation (Train-Test Split)
# Apply Machine Learning Algorithm
# Evaluate Model.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv("Admission_Predict.csv")

# first 5 instances of dataset
data.head()

data.shape

#columns in dataframe
data.columns

# droping Id column as this column is not informative for classification algorithm
data.drop("Serial No.",axis=1,inplace=True) # axix = 1 : means said operation is down the rows

#convert the  "Chance of Admit" (class column to 0 or 1)
data["Chance of Admit "]=data["Chance of Admit "].apply(lambda x: 1 if x>0.5 else 0)

# Find missing values
print("Missing values:\n")
data.isnull().sum()

# Calculating total class Count
data_admit = data[data['Chance of Admit ']==1]
data_non_admit = data[data['Chance of Admit ']==0]
```

```
print("Admitted count       : " ,data_admit.shape[0])
print("Non - Admitted count : " ,data_non_admit.shape[0])

#Vertically Splitting of Dataset into dependent and independent : input and target (class)
X= data.drop("Chance of Admit ",axis =1 )
y= data["Chance of Admit "]

# Splitting the dataset into train and test sets: 80-20 split
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)

# Decision Tree Classifier model
from sklearn.tree import DecisionTreeClassifier

# instantiate the model : create object
tree = DecisionTreeClassifier()

# fit the model : training on data
tree.fit(X_train, y_train)

#predicting the target value from the model for the samples
y_train_tree = tree.predict(X_train)
y_test_tree = tree.predict(X_test)

#computing the accuracy of the model performance
from sklearn.metrics import accuracy_score
acc_test_tree = accuracy_score(y_test,y_test_tree)

print("Decision Tree : Accuracy on test Data: {:.3f}".format(acc_test_tree))

#computing the classification report of the model
from sklearn.metrics import classification_report
print(classification_report(y_test, y_test_tree))

# visualization of tree
import sklearn.tree as tr
fig = plt.figure(figsize=(20,15))
_ = tr.plot_tree(tree,
            feature_names=X.columns,
            class_names=np.array(["Non admit","Admit"]),
            filled=True)
```

# 4. Assignment 4: Bays Classifier

```
# Problem Statement:  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
# Apply Machine Learning Algorithm and Evaluate Model

import pandas as pd
df = pd.read_csv('/content/SMSSpamCollection',delimiter = '\t', header=None)

df.head(n=10)
df.columns = ['Class_label', 'Message']

df['Class_label'] = df['Class_label'].apply(lambda x: 1 if x == 'spam' else 0)

from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(df['Message'], df['Class_label'], test_size = 0.3,
random_state = 0)

from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()

vectorizer.fit(df['Message'])
print(vectorizer.vocabulary_)

new_train_set = vectorizer.transform(x_train)

new_test_set  = vectorizer.transform(x_test)

print(new_train_set.shape)
print(new_test_set.shape)

from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(new_train_set, y_train)

print("classifier accuracy {:.2f}%".format(classifier.score(new_test_set, y_test) * 100))

email = [
   'hey Gauri can you get together for lunch' ,
   'upto 70% discount on exclusive offer'
]

new_email = vectorizer.transform(email)
classifier.predict(new_email)
```

# Assignment 5 : Assignment on Clustering Technique

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

# This dataset givesthe 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.

# 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

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import seaborn as sns

df = pd.read_csv("Mall_Customers.csv")

df.head()

df.shape

df.columns

df.drop("CustomerID",axis=1,inplace=True)

print("Missing values:")
df.isnull().sum()

from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
le = LabelEncoder()

df["Genre"] = le.fit_transform(df["Genre"])

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

wcss = []
for i in range(1,11):
    kmeans = KMeans(n_clusters=i, init='k-means++',random_state=42)
    kmeans.fit(x)
    wcss.append(kmeans.inertia_)
    print('k:',i ,"-> wcss:",kmeans.inertia_)
```

```
# Plotting the results onto a line graph, 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()

#Taking k = 5
km1=KMeans(n_clusters=5)

km1.fit(data)

pred_y =km1.predict(data)

data["label"] = pred_y

#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()

X=data.iloc[:,:4]
y=data.iloc[:,-1]

# Splitting of Data
# 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)

print(X_train.shape,y_train.shape)
print(X_test.shape,y_test.shape)

from sklearn.cluster import KMeans
km=KMeans(n_clusters=5)
km.fit(X_train)

# predicting the target (cluster) from the model for the samples
y_train_km = km.predict(X_train)
y_test_km = km.predict(X_test)

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))
```

```
# Hirarchical Agglomerative clustering
data = data.iloc[:,[2,3]]

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

#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()
```