DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MACHINE LEARNING

[R20A0590]

LABORATORY MANUAL

B. TECH CSE

(III YEAR–IISEM)

R20 REGULATION

(2023-24)

Name :__________________________________

Roll no:________________________________

Section:________________________________

Year :__________________________________

MALLAREDDYCOLLEGEOFENGINEERING&TECHNOLOGY

(AutonomousInstitution– UGC,Govt.ofIndia) Recognizedunder2(f)and12(B) ofUGCACT1956

(Affiliated to JNTUH, Hyderabad, Approved by AICTE - Accredited by NBA &NAAC–'A' Grade-ISO9001:2015 Certified)

Maisammaguda, Dhulapally (Post Via. Hakimpet), Secunderabad – 500100,TelanganaState,India

INDEX

S.No	Date	Name of the Activity/Experiment	Grade/ Marks	Faculty Signature

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Vision

To acknowledge quality education and instill high patterns of discipline making the students technologically superior and ethically strong which involves the improvement in the quality of life in human race.

Mission

- To achieve and impart holistic technical education using the best of infrastructure, outstanding technical and teaching expertise to establish the students in to competent and confident engineers.
- Evolving the center of excellence through creative and innovative teaching learning practices for promoting academic achievement to produce internationally accepted competitive and world class professionals.

PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

PEO1-ANALYTICALSKILLS

- To facilitate the graduates with the ability to visualize, gather information, articulate, analyze, solve complex problems, and make decisions. These are essential to address the challenges of complex and computation intensive problems increasing their productivity.

PEO2-TECHNICALSKILLS

- Tofacilitatethegraduateswiththetechnicalskillsthatpreparethemforimmediateemployment and pursue certification providing a deeper understanding of the technology in advanced areas of computer science and related fields, thus encouraging to pursue higher education and research based on their interest.

PEO3- SOFTSKILLS

- To facilitate the graduates with the soft skills that include fulfilling the mission, setting goals, showing self confidence by communicating effectively, having a positive attitude ,get involved in team-work, being a leader, managing their career and their life.

PEO4-PROFESSIONALETHICS

- To facilitate the graduates with the knowledge of professional and ethical responsibilities by paying attention to grooming, being conservative with style, following dress codes, safety codes, and adapting them to technological advancements.

PROGRAM SPECIFIC OUTCOMES (PSOs)

After the completion of the course, B.Tech Computer Science and Engineering, the graduates will have the following Program Specific Outcomes:

1.FundamentalsandcriticalknowledgeoftheComputerSystem:-AbletoUnderstand the working principles of the computer System and its components ,Apply the knowledge to build, asses, and analyze the software and hardware aspects of it.

2.The comprehensive and Applicative knowledge of Software Development: Comprehensive skills of Programming Languages, Software process models, methodologies, and able to plan, develop, test, analyze, and manage the software and hardware intensive systems in heterogeneous platforms individually or working in teams.

3.Applications of Computing Domain & Research: Able to use the professional, managerial, interdisciplinary skill set, and domain specific tools in development processes, identify their search gaps, and provide innovative solutions to them.

PROGRAM OUTCOMES (POs)

Engineering Graduates should possess the following:

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

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

3. Design / development of solutions: Design solutions for complex engineering problems anddesignsystemcomponentsorprocessesthatmeetthespecifiedneedswithappropriateconsideration for the public health and safety, and the cultural, societal, and environmental considerations.

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

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

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

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

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

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

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

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

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

MACHINE LEARNING LABORATORY

1. Course Objectives:

1. Learn usage of Libraries for Machine Learning in Python
2. Demonstrate Dimensionality reduction methods
3. Describe appropriate supervised learning algorithms for a given problem.
4. Explore back propagation algorithm and ensemble methods
5. Discuss different unsupervised learning algorithms

2. Course outcomes:

1. Illustrate the applications of Python Machine Learning Libraries.
2. Apply Dimensionality reduction methods for Machine Learning Tasks.
3. Design and analyze various supervised learning mechanisms.
4. Develop back propagation algorithm and Random Forest Ensemble method.
5. Design and analyze various unsupervised learning algorithms

3. Introduction about lab

Minimum System requirements:

- Processors: Intel Atom® processor or Intel® Core™ i3 processor.
- Disk space: 1 GB.
- Operating systems: Windows* 7 or later, mac OS, and Linux.
- Python* versions: 2.7.X, 3.6.X., 3.8.X

About lab:

Python is a general purpose, high-level programming language; other high Level languages you might have heard of C++, PHP, Java and Python. Virtually all modern programming languages make us of an Integrated Development Environment (IDE), which allows the creation, editing, testing, and saving of programs and modules. In Python, the IDE is called IDLE (like many items in the language, this is a reference to the British comedy group Monty Python, and in this case, one of its members, Eric Idle).

Many modern languages use both processes. They are first compiled into a lower level language, called byte code, and then interpreted by a program called a virtual machine. Python uses both processes, but because of the way programmers interact with it, it is usually considered an interpreted language Practical aspects are the key to understanding and conceptual visualization

Of theoretical aspects covered in the laboratory

4. Guidelines to students

A. Standard operating procedure

a) Explanation on today's experiment by the concerned faculty using PPT covering the following aspects:

1) Name of the experiment
2) Aim

b) Writing the python programs by the students

c) Commands for executing programs

Writing of the experiment in the Observation Book

The students will write the today's experiment in the Observation book as per the following format:

a) Name of the experiment
b) Aim
c) Writing the program
d) Viva-Voce Questions and Answers
e) Errors observed (if any) during compilation/execution
f) Signature of the Faculty

B. Guide Lines to Students in Lab

Disciplinary to be maintained by the students in the Lab

- Students are required to carry their lab observation book and record book with completed experiments while entering the lab.
- Students must use the equipment with care. Any damage is caused student is punishable
- Students are not allowed to use their cell phones/pen drives/ CDs in labs.
- Students need to be maintain proper dress code along with ID Card
- Students are supposed to occupy the computers allotted to them and are not supposed to talk or make noise in the lab. Students, after completion of each experiment they need to be updated in observation notes and same to be updated in the record.
- Lab records need to be submitted after completion of experiment and get it corrected with the concerned lab faculty.
- If a student is absent for any lab, they need to be completed the same experiment in the free time before attending next lab.

Instructions to maintain the record

- Before start of the first lab they have to buy the record and bring the record to the lab.
- Regularly (Weekly) update the record after completion of the experiment and get it corrected with concerned lab in-charge for continuous evaluation.
- In case the record is lost inform the same day to the faculty in charge and get the new record within 2 days the record has to be submitted and get it corrected by the faculty.
- If record is not submitted in time or record is not written properly, the evaluation marks (5M) will be deducted.

C. General laboratory instructions

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.
3. Student should enter into the laboratory with:
a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session. b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab. c. Proper Dress code and Identity card.
4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

INDEX

Week1: Write a python program to import and export the data using pandas library

1. Manual Function

Go to Google Page and find www.kaggle.com .Select Datasets and find Titanic dataset , then download train.csv file ans save it to desktop.

1. Read a CSV file import
   pandas as pd
   url='C:/Users/MRCET1/Desk
   top/train.csv'
   dataframe=pd.read_csv(url)
   dataframe.head(5)

2. Write a CSV file import pandas as pd
   marks_data=pd.DataFrame({'ID':{0:23,1:43,2:12,3:13,4:67,5:89},'NAME':{0:'Ram',1:'Deep',2:'Ya
   sh',3:'Arjun',4:'Aditya',5:'Divya'},'Marks':{0:89,1:92,2:45,3:78,4:56,5:76},'Grade':{0:'b',1:'a',2:'f',3:' c',4:'e',5:'c'}})

   filename='C:/Users/MRCET1/Desktop/M
   arksdata.xlsx'
   marks_data.to_excel(filename)

   print('Data frame written to Excel')

3. Read an Excel File import pandas as pd

   url='C:/Users/MRCET1/Desktop/train.csv.xls'

   dataframe=pd.read_excel(url)

   dataframe.head(5)

4. Write an Excel file import pandas as pd

   marks_data=pd.DataFrame({'ID':{0:23,1:43,2:12,3:13,4:67,5:89},'NAME':{0:'Ram',1:'Deep',2:'Y

   ash',3:'Arjun',4:'Aditya',5:'Divya'},'Marks':{0:89,1:92,2:45,3:78,4:56,5:76},'Grade':{0:'b',1:'a',2:'f',

   3:'c',4:'e',5:'c'}})
   filename='C:/Users/MRCET1/Desktop/Marksdata.csv'
   marks_data.to_csv(filename)

   print('Data frame written to CSV');

Viva Questions

1. **What is Machine learning?**
2. **What is the main key difference between supervised and unsupervised machine learning?**
3. **What Are the Different Types of Machine Learning?**
4. **What is numpy in python**

Faculty Signature

WEEK-2: Demonstrate various data pre-processing techniques for a given dataset

1. Detecting and Handling Missing values
1. Type conversion
2. Detecting and Treating Outliers
3. Scaling
4. Dimensionality Reduction

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt import seaborn as sns
df=pd.read_csv('C:/Users/MRCET1/Desktop/train.csv')
df.head(5)

df.shape # to check dimensions of data

df.info()# to list features of data

df.describe() # to display mean, variance ,count and other details of data

df.columns  # to display column names

df.dtypes # to display data types of columns

# To display the total count of null values in each column

column_names = df.columns
for column in column_names:
        print(column + ' - ' + str(df[column].isnull().sum()))

# To display the count of passengers survived

df.Survived.value_counts()
```

DATA VISUALIZATION

```
# To draw a bar graph to visualize
survival data import
matplotlib.pyplot as plt
plt = df.Survived.value_counts().plot.bar(x='Survived or not',y='Passenger Count',rot=0)

# To see probability of survival based on Passenger Class
plt = df.Pclass.value_counts().sort_index().plot.bar(x='PClass',y='Survival Probability',rot=0)


# To display Class wise survival count
df[['Pclass',
'Survived']].groupby('Pclass').count()

df[['Pclass', 'Survived']].groupby('Pclass').mean().Survived.plot.bar(x='PClass',y='Survival
Probability',rot=0)
```

#From the results, we can say that, 1st class has high chance of surviving than the other two classes.

# To display survival count based on Gender
df.Sex.value_counts().sort_index().plot.bar(x='Sex',y='Passenger Count',rot=0) plt.set_xlabel('Sex')
plt.set_ylabel('Passenger count')

IDENTIFYING IRRELEVANT AND REDUNDANT

FEATURES Remove unnecessary columns
We can remove 'Ticket' and 'PassengerId', as they don't contribute to target class.
Remove 'Cabin' as it has a lot of missing values in both train and test data

df= df.drop(columns=['Ticket', 'PassengerId', 'Cabin'])
df.head()

#Map 'Sex' and 'Embarked' to numerical values. df['Sex'] =
df['Sex'].map({'male':0, 'female':1})
df['Embarked'] = df['Embarked'].map({'C':0, 'Q':1, 'S':2})
df.head()

#Preprocess 'Name'
#Extarct title from name of the passenger and categorize them.
#Drop the column 'Name'
df['Title'] = df.Name.str.extract(' ([A-Za-z]+)\.',
expand=False) df = df.drop(columns='Name')
df.Title.value_counts().plot.bar(x='Title',y=0,rot=0)

#Combine some of the classes and group all the rare classes into 'Others'.
df['Title'] = df['Title'].replace(['Dr', 'Rev', 'Col', 'Major', 'Countess', 'Sir', 'Jonkheer', 'Lady', 'Capt', 'Don'],
'Others')
df['Title'] = df['Title'].replace('Ms',
'Miss') df['Title'] =
df['Title'].replace('Mme', 'Mrs') df['Title']
= df['Title'].replace('Mlle', 'Miss')
df.Title.value_counts().sort_index().plot.bar(x='Title',y='Passenger count')

df[['Title', 'Survived']].groupby('Title').mean().Survived.plot.bar(x='Title',y='Survival Probability')

#Map 'Title' to numerical values
df['Title'] = df['Title'].map({'Master':0, 'Miss':1, 'Mr':2, 'Mrs':3, 'Others':4})
df.head()

#Correlation between columns
corr_matrix = df.corr()
import matplotlib.pyplot as plt
plt.figure(figsize=(9, 8))
sns.heatmap(data = corr_matrix,cmap='BrBG', annot=True, linewidths=0.2)

#Handling missing values
df.isnull().sum()

#Impute 'Embarked' with it's majority class.

df['Embarked'].isnull().sum()

#There are two null values in the column 'Embarked'. Let's impute them using majority class.
#The majority class is 'S'. Impute the unkonown values
(NaN) using 'S' df['Embarked'] = df['Embarked'].fillna(2)
df.head()

#Missing values - 'Age'
#Let's find the columns that are useful to predict the
value of Age. corr_matrix = df[['Pclass', 'Sex', 'Age',
'SibSp', 'Parch', 'Fare']].corr()

plt.figure(figsize=(7, 6))
sns.heatmap(data = corr_matrix,cmap='BrBG', annot=True, linewidths=0.2)

#Age is not correlated with 'Sex' and 'Fare'. So, we don't consider these two columns while imputing 'Sex'.#
#'Pclass', 'SibSp' and 'Parch' are negatively correlated with 'Sex'.
#Let's fill Age with the median age of similar rows from 'Pclass', 'SibSp' and 'Parch'.
#If there are no similar rows, fill the age with the median age of
total dataset. NaN_indexes = df['Age'][df['Age'].isnull()].index

```
for i in NaN_indexes:
    pred_age = df['Age'][((df.SibSp == df.iloc[i]["SibSp"]) & (df.Parch == df.iloc[i]["Parch"]) &
    (df.Pclass== df.iloc[i]["Pclass"]))].median()
    if not np.isnan(pred_age):
        df['Age'].iloc[i] = pred_age
        else:
        df['Age'].iloc[i] = df['Age'].median()
```

df.isnull().sum()

#There are no missing values in the data.
df.head(20)

Viva Questions

1. **What is pandas' in python?**
2. **What is meant by data pre-processing?**
3. **What are the ways to handle missing data?**
4. **What is a CSV File?**

Faculty Signature

WEEK-3: Implement Dimensionality reduction using Principle Component Analysis (PCA) method.

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
sns.set_theme(style = "darkgrid")
```

Now, we read the data and check out how the data looks

```
data=pd.read_csv("C:/Users/MRCET1/Desktop/stars.csv")
data.head(5)
```

Now, let us check the shape of the dataset. Before proceeding with the problem statement, understanding the dataset is very important.

```
data.shape
```

The attributes of the data are

```
data.info()
```

Now, we check for missing values.

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

some analysis of the data distribution.

```
a= pd.DataFrame(data['Star color'].value_counts())
plt.figure(figsize=(8,6))
sns.barplot(data= a, x='Star color',y= a.index, palette= 'Spectral')
plt.title("Star Color Analysis")
```

Star Spectral Class Analysis:

```
a= pd.DataFrame(data['Spectral Class'].value_counts())
plt.figure(figsize=(8,6))
sns.barplot(data=a, x='Spectral Class',y= a.index, palette= 'rainbow')
plt.title("Star Spectral Class Analysis")
```

Star Type Analysis

```
a =pd.DataFrame(data['Star type'].value_counts())
plt.figure(figsize=(10,8))
plt.pie(data=a, x='Star type',labels=a.index,autopct='%1.1f%%')
plt.title("Percentage Distribution of Star Type")
```

Correlation Analysis

```
matrix= data.corr()
mask  =  np.zeros_like(matrix,  dtype=float)
mask[np.triu_indices_from(mask)]=     True
plt.figure(figsize=(11,6))
sns.heatmap(matrix,annot=True,cmap='viridis',annot_kws = {'size': 10},mask=mask)
plt.title("Correlation Analysis")
plt.show()

from sklearn import preprocessing
```

 # label_encoder

```
label_encoder = preprocessing.LabelEncoder()
```

Now, we apply the encoder to the dataset.

```
data['Color_Label']=label_encoder.fit_transform(data['Star color'])
data['Spectral_Class_Label']=label_encoder.fit_transform(data['Spec
tral Class']) data.head()

print("Original Colours:") print(data['Star
color'].unique()) print("Labels:")
print(data['Color_Label'].unique())
```

Now, we split the dataset into X and y.

```
y= data["Spectral_Class_Label"].values
X = data.drop(labels=['Spectral_Class_Label','Star color','Spectral Class'], axis=1).values
```

Applying Standard Scaler Normalization

```
from sklearn.preprocessing import
StandardScaler sc = StandardScaler()
X = sc.fit_transform(X)
```

Implementation of PCA

```
from sklearn.decomposition import PCA
pca = PCA()
X = pca.fit_transform(X)
explained_variance = pca.explained_variance_ratio_
explained_variance
```

First, we train the Classifier model by taking the two top features.Inbuilt Random Forest Classifier is used.

```
from sklearn.decomposition import PCA
```

```
pca2 = PCA(n_components=2)
X_2 = pca2.fit_transform(X)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_2, y, test_size=0.2, random_state=5)
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
from sklearn.metrics import
accuracy_score print('Accuracy: ' ,
accuracy_score(y_test, y_pred))
```

number of principal components=3.

```
from sklearn.decomposition import PCA
pca3 = PCA(n_components=3)
X_3 = pca3.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_3, y, test_size=0.2,
random_state=5) classifier = RandomForestClassifier(max_depth=2,
random_state=0) classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
print('Accuracy: ' , accuracy_score(y_test, y_pred))
```

number of principal components=4.

```
from sklearn.decomposition import PCA
pca4 = PCA(n_components=4)
X_4 = pca4.fit_transform(X)
X_train, X_test, y_train, y_test = train_test_split(X_4, y, test_size=0.2, random_state=5)
classifier = RandomForestClassifier(max_depth=2, random_state=0)
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
print('Accuracy: ' , accuracy_score(y_test, y_pred))
```

Viva Questions

1. **What is Dimensionality Reduction?**
2. **What is meant by normalization?**
3. **Write a note on PCA**
4. **What is the use of sklearn in machine learning?**

Faculty Signature

WEEK-4: Write a python program to demonstrate various Data Visualization Techniques.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Downdload dataset and read it
csv_url = &#39;https://archive.ics.uci.edu/ml/machine-learning-
databases/iris/iris.data&#39;
# using the attribute information as the column names
col_names =
[&#39;Sepal_Length&#39;,&#39;Sepal_Width&#39;,&#39;Petal_Length&#39;,&#39;Peta
l_Width
&#39;,&#39;Class&#39;]
iris = pd.read_csv(csv_url, names = col_names)
iris.head()
iris[&quot;Class&quot;].value_counts()
# Line plots
import numpy as np
x = np.linspace(0,20,30)
y= x**2 plt.plot(x, y) plt.show()
# Line plot with grid
x = np.linspace(0,20,30)
y= x**2 plt.plot(x, y)
plt.xlabel(&#39;x-values&#39;) plt.ylabel(&#39;x^2-
values&#39;) plt.title(&#39;line plot&#39;) plt.grid(True)
plt.show()


# Scatter Plot
iris.plot(kind=&quot;scatter&quot;, x=&quot;Sepal_Length&quot;,
y=&quot;Sepal_Width&quot;)
```

```
colours = {&#39;Iris-setosa&#39;:&#39;orange&#39;, &#39;Iris-
versicolor&#39;:&#39;lightgreen&#39;, &#39;Iris-
virginica&#39;:&#39;lightblue&#39;}
for i in range(len(iris[&#39;Sepal_Length&#39;])):
plt.scatter(iris[&#39;Petal_Length&#39;][i],iris[&#39;Petal_Width&#39;][i], color =
colours[iris[&#39;Class&#39;][i]])
plt.title(&#39;Iris&#39;) plt.xlabel(&#39;petal length&#39;)
plt.ylabel(&#39;petal width&#39;) plt.grid(True)
plt.show()


# We can also use the seaborn library to make a similar plot
sns.jointplot(x=&quot;Sepal_Length&quot;, y=&quot;Sepal_Width&quot;,
data=iris, size=5)
# Bar Graph
a= iris[&#39;Class&#39;].value_counts()
species = a.index count = a.values
plt.bar(species,count,color = &#39;lightgreen&#39;)
plt.xlabel(&#39;species&#39;)
plt.ylabel(&#39;count&#39;)
plt.show()


# Box Plot length_width =
iris[[&#39;Petal_Length&#39;,&#39;Petal_Width&#39;,&#39;Sepal_Length&#39;,&#3
9;Sepal_Wi dth&#39;]] #excluding species
column length_width.boxplot() plt.xlabel(&#39;Flower
measurements&#39;) plt.ylabel(&#39;values&#39;)
plt.title(&quot;Iris dataset analysis&quot;)
# We can look at an individual feature in Seaborn through many different kinds of plots.
# Here&#39;s a boxplot
sns.boxplot(x=&quot;Class&quot;, y=&quot;Petal_Length&quot;,
palette=&quot;husl&quot;, data=iris)
```

```
#Histogram
import numpy as np
data_ = np.random.randn(1000) plt.hist(data_,bins =
40,color=&#39;gold&#39;) plt.grid(True)
plt.xlabel(&#39;points&#39;)
plt.title(&quot;Histogram&quot;) plt.show()
#Correlation Matrix correlation = iris.corr() fig ,ax = plt.subplots()
k = ax.imshow(correlation, cmap = &#39;magma_r&#39;)
ax.set_xticks(np.arange(len(correlation.columns)))
ax.set_yticks(np.arange(len(correlation.columns)))
ax.set_xticklabels(correlation.columns)
ax.set_yticklabels(correlation.columns)
cbar = ax.figure.colorbar(k, ax=ax)
cbar.ax.set_ylabel(&#39;color bar&#39;, rotation=-90, va=&quot;bottom&quot;)

plt.setp(ax.get_xticklabels(), rotation=45,
ha=&quot;right&quot;,rotation_mode=&quot;anchor&quot;)
for i in range(len(correlation.columns)):
for j in range(len(correlation.columns)):
text = ax.text(j, i, np.around(correlation.iloc[i, j],
decimals=2),ha=&quot;center&quot;, va=&quot;center&quot;,
color=&quot;lightgreen&quot;)
plt.show()
#Piechart
a= iris[&#39;Class&#39;].value_counts()
species = a.index count = a.values
colors= [&#39;lightblue&#39;,&#39;lightgreen&#39;,&#39;gold&#39;]
explode = (0,0.2,0)
plt.pie(count, labels=species,shadow=True,
colors=colors,explode = explode, autopct=&#39;%1.1f%%&#39;)
plt.xlabel(&#39;species&#39;)
plt.axis(&#39;equal&#39;) plt.show()
sns.set_style(&#39;darkgrid&#39;)
```

```
sns.lineplot(data=iris.drop([&#39;Class&#39;], axis=1))
plt.show()
```

Viva Questions

1. **What is meant by data visualization in machine learning?**
2. **What are the various types of Data Visualization Approaches?**
3. **List some of the Data Visualization Libraries Available in Python**
4. **List some of Feature Selection Techniques in supervised learning.**

Faculty Signature

WEEK-5: Implement Simple and Multiple Linear Regression Models

1. Implementation of Linear Regression

```
import numpy as np
import matplotlib.pyplot as plt
def estimate_coef(x, y):
    # number of observations/points n = np.size(x)

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

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

    # calculating regression coefficients
    b_1 = SS_xy / SS_xx
    b_0 = m_y - b_1*m_x
    return (b_0, b_1)

    def plot_regression_line(x, y, b):
    # plotting the actual points as scatter plot plt.scatter(x, y, color = "m",
                        marker = "o", s = 30)

    # predicted response vector
    y_pred = b[0] + b[1]*x
    # plotting the regression line
    plt.plot(x, y_pred, color = "g")

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

    # function to show plot plt.show()

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

    # estimating coefficients
        b = estimate_coef(x, y)

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

        # plotting regression line
        plot_regression_line(x, y, b)

        if __name__ == "__main__":
        main()
```

2. Multiple linear regression

```
import numpy as np
import matplotlib.pyplot as plt

def estimate_coef(x, y):
                # number of observations/points
                n = np.size(x)
                # mean of x and y vector
                m_x = np.mean(x)
                m_y = np.mean(y)

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

                # calculating regression coefficients
                b_1 = SS_xy / SS_xx
                b_0 = m_y - b_1*m_x
                return (b_0, b_1)
def plot_regression_line(x, y, b):
                # plotting the actual points as scatter plot
                plt.scatter(x, y, color = "m",marker = "o", s = 30)

                # predicted response vector
                y_pred = b[0] + b[1]*x

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

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

                # function to show plot
                plt.show()
def main():

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

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

        # plotting regression line
plot_regression_line(x, y, b)

if __name__ == "__main__":

                main()
```

Viva Questions

1. **What is meant by linear regression?**
2. **List some of the supervised learning algorithms**
3. **List the types of linear regression and define each**

Faculty Signature

WEEK-6: Develop Logistic Regression Model for a given dataset.

```
import matplotlib.pyplot as plt import
numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix

x = np.arange(10).reshape(-1, 1)
y = np.array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
x
y

model = LogisticRegression(solver='liblinear', random_state=0)

model.fit(x, y)

LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
intercept_scaling=1, l1_ratio=None, max_iter=100,
multi_class='warn', n_jobs=None, penalty='l2',
random_state=0, solver='liblinear', tol=0.0001,
verbose=0, warm_start=False)

model = LogisticRegression(solver='liblinear', random_state=0).fit(x, y)
model.classes_ model.intercept_ model.coef_
model.predict_proba(x)
#model.predict(x)
```

Viva Questions

1. **What is meant by logistic regression?**
2. **List the types of logistic regression and explain each**
3. **List the classification model in machine learning**
4. **What are the Steps involved in Logistic Regression**

Faculty Signature

WEEK-7: Develop Decision Tree Classification model for a given dataset and use it to classify a new sample.

1. **Implementation of Decision tree classification**

```
# Importing the required
packages import numpy as np

import pandas as pd

from sklearn.metrics import confusion_matrix

from sklearn.model_selection import
train_test_split from sklearn.tree
import DecisionTreeClassifier from
sklearn.metrics import
accuracy_score

from sklearn.metrics import classification_report


# Function importing Dataset

def importdata():

        balance_data = pd.read_csv(

'https://archive.ics.uci.edu/ml/machine-learning-'+

'databases/balance-scale/balance-scale.data',

        sep= ',', header = None)


        # Printing the dataswet shape

        print ("Dataset Length: ", len(balance_data))

        print ("Dataset Shape: ", balance_data.shape)


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

         return balance_data

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

        # Separating the target
        variable
```

```
        X    =    balance_data.values[:,
        1:5]

        Y = balance_data.values[:, 0]


        # Splitting the dataset into train and test

        X_train, X_test, y_train, y_test = train_test_split(

        X, Y, test_size = 0.3, random_state = 100)

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


# Function to perform training with giniIndex.

def train_using_gini(X_train, X_test, y_train):

        # Creating the classifier object
        clf_gini = DecisionTreeClassifier(criterion = "gini",
                        random_state = 100,max_depth=3, min_samples_leaf=5)
        # Performing training
        clf_gini.fit(X_train, y_train)
        return clf_gini
        # Decision tree with entropy
        clf_entropy = DecisionTreeClassifier(
                        criterion = "entropy", random_state = 100,
                        max_depth = 3, min_samples_leaf = 5)

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


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

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

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

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

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

print("Report : ", classification_report(y_test, y_pred))
# Driver code
def main():

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

clf_entropy = tarin_using_entropy(X_train, X_test, y_train)

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

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

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


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

Viva Questions

1. **What is Decision Trees-ID3 in machine learning?**
2. **What is meant by information gain?**
3. **What are the major issues in decision tree learning?**
4. **How does decision tree help in decision making?**

Faculty Signature

WEEK-8 : Implementation of Naïve Bayes classifier algorithm

```
# Importing the dataset
dataset = pd.read_csv(&#39;titanic.csv&#39;)
X = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, -1].values
# Splitting the dataset into the Training set
and Test set from sklearn.model_selection
import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size = 0.20, random_state = 0)
# Feature Scaling
from sklearn.preprocessing import
StandardScaler sc = StandardScaler()
X_train = sc.fit_transform(X_train) X_test
= sc.transform(X_test)
# Training the Naive Bayes model on the
Training set from sklearn.naive_bayes
import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, y_train)
# Predicting the Test set results
y_pred = classifier.predict(X_test)
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix,
accuracy_score ac = accuracy_score(y_test,y_pred)
cm = confusion_matrix(y_test, y_pred)
```

Viva Questions

1. **What is naive in naive baye’s classifier?**
2. **Write a note on SVM**
3. **Give the formula for Bayes' Theorem**
4. **List some of the advantages and disadvantages of naïve bayes classifier**

Faculty Signature

Week-9: Implementation of K-nearest Neighbour

```
import numpy as np import
pandas as pd
from sklearn.model_selection
import train_test_split
from sklearn.neighbors
import KNeighborsClassifier
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('data.csv')
y = df['diagnosis']
X = df.drop('diagnosis', axis=1)
X = X.drop('Unnamed: 32', axis=1)
X = X.drop('id', axis=1)

# Separating the dependent and independent variable
X_train, X_test, y_train, y_test = train_test_split( X, y,
    test_size=0.3, random_state=0)

# Splitting the data into training and testing data
K = []
training = []
test = []
scores = {}
for k in range(2, 21):
    clf = KNeighborsClassifier(n_neighbors=k)
    clf.fit(X_train, y_train)
   training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test) K.append(k)
    training.append(training_score)

    test.append(test_score)
    scores[k] = [training_score, test_score]
ax = sns.stripplot(training)

ax.set(xlabel='values of k', ylabel='Training Score')
plt.show()
ax = sns.stripplot(test)
ax.set(xlabel='values of k', ylabel='Test Score'
```

```
plt.show()
 plt.scatter(K, training, color='k')
 plt.scatter(K, test, color='g')
 plt.show()
```

Viva Questions

1. **What is KNN in machine learning?**
2. **Why do we need a K-NN Algorithm?**
3. **What is Kernel Method?**
4. **List some of the major Kernel Function in Support Vector Machine**

Faculty Signature

WEEK-10: Build Artificial Neural Network model with back propagation on a given dataset

Let's first understand the term neural networks. In a neural network, where neurons are fed inputs which then neurons consider the weighted sum over them and pass it by an activation function and passes out the output to next neuron.

```
  import numpy as np
 X = np.array(([2, 9], [1, 5], [3, 6]), dtype=float)
  y = np.array(([92], [86], [89]), dtype=float)
 X = X/np.amax(X, axis=0)  # maximum of X array
 longitudinally y = y/100

# Sigmoid Function def sigmoid(x):
    return 1/(1 + np.exp(-x))
# Derivative of Sigmoid Function
def derivatives_sigmoid(x):
    return x * (1 - x)

# Variable initialization
epoch = 5  # Setting training iterations lr = 0.1
# Setting learning rate
inputlayer_neurons = 2
# number of features in data set
hiddenlayer_neurons = 3
# number of hidden layers neurons
output_neurons = 1
# number of neurons at output layer
# weight and bias initialization
wh = np.random.uniform(size=(inputlayer_neurons, hiddenlayer_neurons))
bh = np.random.uniform(size=(1, hiddenlayer_neurons))
wout = np.random.uniform(size=(hiddenlayer_neurons, output_neurons))
bout = np.random.uniform(size=(1, output_neurons))
# draws a random range of numbers
uniformly of dim x*y for i in range(epoch):

    # Forward Propogation hinp1 = np.dot(X, wh)
    hinp = hinp1 + bh
    hlayer_act = sigmoid(hinp)
```

```
    outinp1 = np.dot(hlayer_act, wout)
        outinp = outinp1+bout
        output = sigmoid(outinp)

    # Backpropagation
    EO = y-output
    outgrad = derivatives_sigmoid(output)
    d_output = EO * outgrad
    EH = d_output.dot(wout.T)
    # how much hidden layer wts contributed to error
    hiddengrad = derivatives_sigmoid(hlayer_act)
    d_hiddenlayer = EH * hiddengrad
    # dotproduct of nextlayererror and currentlayerop
    wout += hlayer_act.T.dot(d_output) * lr
    wh += X.T.dot(d_hiddenlayer) * lr
    print("-----------Epoch-", i+1, "Starts----------")
    print("Input: \n" + str(X))
    print("Actual Output: \n" + str(y))
    print("Predicted Output: \n", output)
    print("-----------Epoch-", i+1, "Ends----------\n")
print("Input: \n" + str(X))
print("Actual Output: \n" + str(y))
print("Predicted Output: \n", output)
```

Viva Questions

1. **Define ANN**
2. **What are the types of Artificial Neural Networks?**
3. **List some of the Applications of Artificial Neural Networks**
4. **What is back propagation in neural networks?**

Faculty Signature

WEEK-11: **Implementing Random Forest ensemble method on a given dataset.**

Implementing Random Forestfrom numpy import
mean from numpy import std

```
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.ensemble import RandomForest Classifier

# define dataset
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15,
n_redundant=5, random_state=3)

# define the model
model = RandomForestClassifier()
# evaluate the model
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=3, random_state=1)
n_scores = cross_val_score(
    model, X, y, scoring='accuracy', cv=cv, n_jobs=-1,
                    error_score='raise')
# report performance
print('Accuracy: %.3f (%.3f)' % (mean(n_scores), std(n_scores)))
```

WEEK-11(B) : Implement Boosting ensemble method on a given dataset.

Model Selection, Bagging and Boosting

```
import pandas
from sklearn import model_selection
from sklearn.ensemble import AdaBoostClassifier
url = "pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'test', 'mass', 'pedi', 'age', 'class']
dataframe = pandas.read_csv(url, names=names)
array = dataframe.values
X = array[:, 0:8]
Y = array[:, 8]
seed = 7 num_trees = 30
kfold = model_selection.KFold(n_splits=10, random_state=seed,
shuffle=True)
model = AdaBoostClassifier(n_estimators=num_trees,
random_state=seed)
results = model_selection.cross_val_score(model, X, Y,
cv=kfold)
print(results.mean())
```

Viva Questions

1. **What is random forest in machine learning?**
2. **Differentiate Between Classification And Regression?**
3. **What is meant by over fitting and under fitting in machine learning?**
4. **What is meant by Bagging and boosting?**

Faculty Signature

WEEK-12: Implementing K-means Clustering Algorithm.

```
# ----------installations---------------
# 1.pip install scikit-learn
# 2.pip install matplotlib
# 3.pip install k-means-constrained
# 4.pip install pandas
from sklearn.datasets
import make_blobs
import matplotlib.pyplot as plt
from k_means_constrained import
KMeansConstrained import pandas as pd
df = pd.read_csv('student_clustering.csv')
X = df.iloc[:, :].values
km = KMeansConstrained(n_clusters=4, max_iter=500)
y_means = km.fit_predict(X)
plt.scatter(X[y_means == 0, 0], X[y_means == 0, 1],
color='red')
plt.scatter(X[y_means == 1, 0], X[y_means == 1, 1],
color='blue')
plt.scatter(X[y_means == 2, 0], X[y_means == 2, 1],
color='green')
plt.scatter(X[y_means == 3, 0], X[y_means == 3, 1],
color='yellow') plt.show()
```

Viva Questions

1. **What is meant by unsupervised learning?**
2. **What is meant by reinforcement learning?**
3. **What are the types of unsupervised learning? Define them.**
4. **What is K-Means Algorithm?**

Faculty Signature