A logo of university of mumbai

Description automatically generated

**Institute of Distance and Open Learning**

Vidya Nagari, Kalina, Santacruz East – 400098.

***CERTIFICATE***

This is to certify that Mr. **Omkar Balu Auti** of **Master in Computer Application** (MCA) Semester 2 has completed the specified term work in the subject of **Artificial Intelligence and Machine Learning** satisfactorily within this institute as laid down by University of Mumbai during the academic year 2023 to 2024.

\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

Subject In-charge External Examiner Coordinator – M.C.A

**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr No.** | **Title** | **Date** | **Signature** |
| 1 | **Introduction** |  |  |
| 2 | **Relations in PROLOG** |  |  |
| 3 | **Supervised Learning** |  |  |
| 1. **Implement the Linear regression model** |  |  |
|  | **b) Implement Logistic regression model.** |  |  |
|  | **C) K-nearest Neighbours (KNN) Classification Model.** |  |  |
| 4 | **Features and Extraction** |  |  |
| **a) Identify the features in Iris dataset that are strongly correlated** |  |  |
|  | **b) Implementation of principal component analysis (PCA) on the Iris dataset with Python** |  |  |
| 5 | **Unsupervised Learning** |  |  |
| **a) Implement the K-Means clustering method** |  |  |
| 6 | **Classify the data using Support vector machine** |  |  |
| 7 | **Implement the decision tree using python** |  |  |

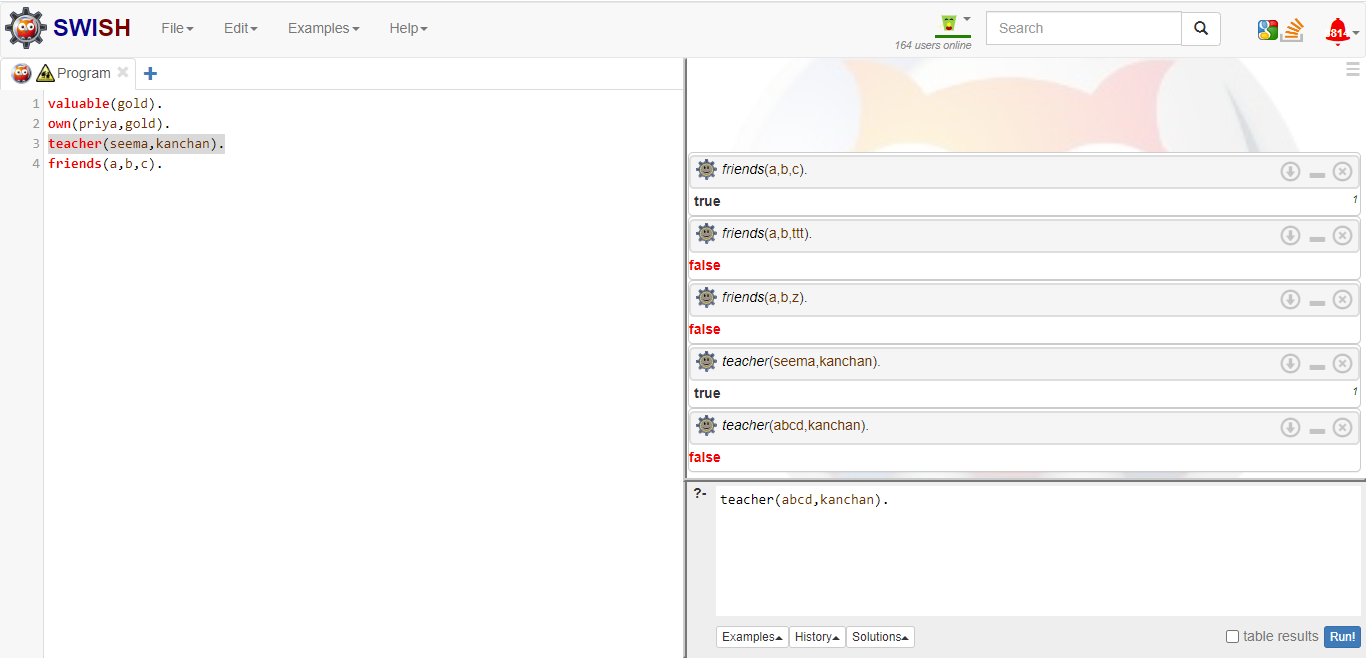
**Practical 01**

valuable(gold).

own(priya,gold).

teacher(seema,kanchan).

friends(a,b,c).



animal(dog).

animal(cat).

animal(elephant).

reptile(snake).

reptile(crocodile).

mammal(Y):- animal(Y).

A screenshot of a computer

Description automatically generated

Q1. Write a program to derive the predicate. For example, Sachin is batsman, batsman is cricketer. Imply that Sachin -> cricketer.

batsman("Sachin").

cricketer(Y):-batsman(Y).

head            body

A screenshot of a computer

Description automatically generated

**Practical 02**

parents(Lata,Raj).

parents(Lata,Rahul).

male(Raj).

male(Rahul).

brothers(Raj,Rahul):-parents(Lata,Raj),parents(Lata,Rahul),male(Raj),male(Rahul).

A screenshot of a computer

Description automatically generated

Q1. Write a program which contains 3 predicates male, female, parent. Make rules for following family relations –

1. Father
2. Mother
3. Grandfather
4. Grandmother
5. Brother
6. Sister
7. Uncle
8. Aunt
9. Nephew
10. Niece
11. Cousin

mother(\_Lata,\_Raj).

mother(Savitri,Lata).

mother(Sumati,Shankar).

father(\_Shankar,\_Rahul).

father(\_Shrinivas, Lata).

father(\_Shriram, Shankar).

husband(Shankar, Lata).

wife(Lata, Shankar).

parents(\_Lata,\_Raj).

parents(\_Lata,\_Rahul).

parents(\_Shankar,\_Raj).

parents(\_Shankar,\_Rahul).

female(\_Savitri).

female(\_Lata).

female(\_Sumati).

male(\_Shriram).

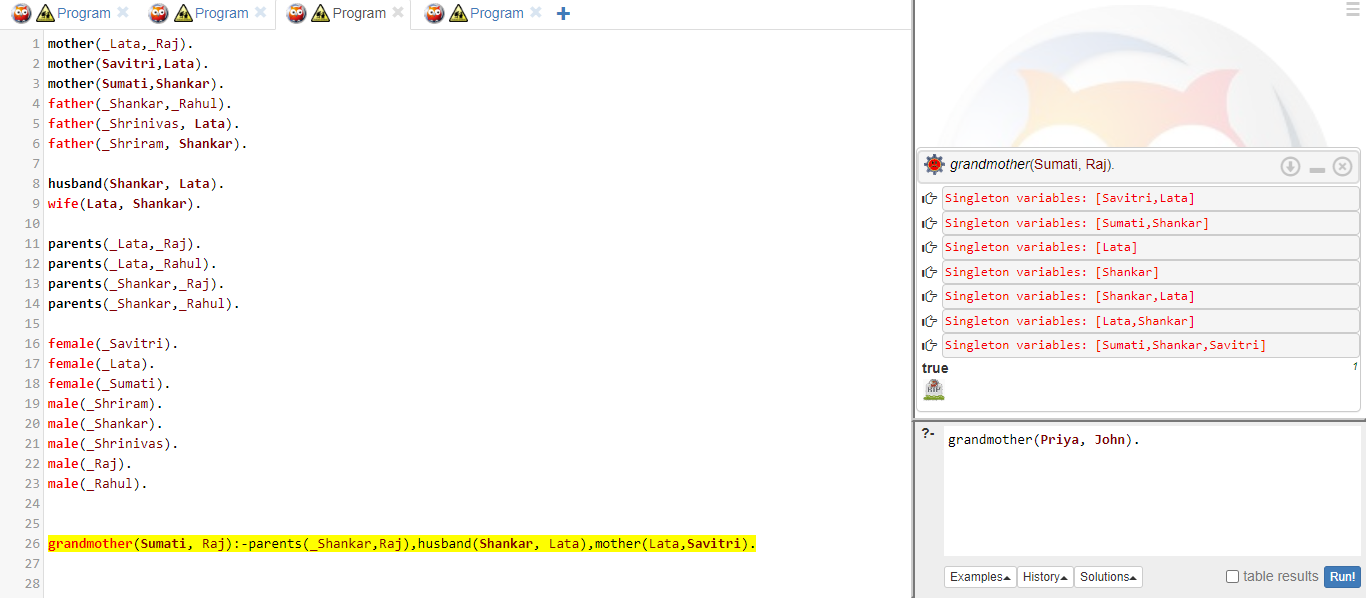
male(\_Shankar).

male(\_Shrinivas).

male(\_Raj).

male(\_Rahul).

grandmother(Sumati, Raj):-parents(\_Shankar,Raj),husband(Shankar, Lata),mother(Lata,Savitri).



Q2. Write simple fact for following –

1. Ram likes mango

likes(Ram, mango).



1. Tanvi is a girl

girl(\_Tanvi).



1. Rose is pink

pink(\_Rose).



1. Ritesh owns land

owns(\_Ritesh, land).



**Practical 03**

**Aim: supervised learning**

1. **Implement the Linear regression model**

**Code:**

import matplotlib.pyplot as plt

from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]

y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std\_err = stats.linregress(x, y)

def myfunc(x):

return slope \* x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)

plt.plot(x, mymodel)

plt.show()

**Output:**

A screen shot of a graph

Description automatically generated

1. **Implement Logistic regression model.**

**Code**:

import numpy as np

import matplotlib.pyplot as plt

from sklearn import datasets

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score, confusion\_matrix

# Load the Iris dataset

iris = datasets.load\_iris()

X = iris.data

y = iris.target

# For binary classification, let's consider only two classes (0 and 1)

X = X[y != 2]

y = y[y != 2]

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Standardize the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Create a logistic regression model

model = LogisticRegression()

# Train the model

model.fit(X\_train, y\_train)

# Make predictions on the test set

predictions = model.predict(X\_test)

# Evaluate the model

accuracy = accuracy\_score(y\_test, predictions)

conf\_matrix = confusion\_matrix(y\_test, predictions)

print(f"Accuracy: {accuracy}")

print(f"Confusion Matrix:\n{conf\_matrix}")

A close-up of a number

Description automatically generated

# Plot decision boundary (works only for 2D datasets)

if X\_train.shape[1] == 2:

h = .02 # Step size in the mesh

x\_min, x\_max = X\_train[:, 0].min() - 1, X\_train[:, 0].max() + 1

y\_min, y\_max = X\_train[:, 1].min() - 1, X\_train[:, 1].max() + 1

xx, yy = np.meshgrid(np.arange(x\_min, x\_max, h), np.arange(y\_min, y\_max, h))

Z = model.predict(np.c\_[xx.ravel(), yy.ravel()])

Z = Z.reshape(xx.shape)

plt.contourf(xx, yy, Z, cmap=plt.cm.Spectral, alpha=0.8)

# Plot the training points

plt.scatter(X\_train[:, 0], X\_train[:, 1], c=y\_train, cmap=plt.cm.Spectral)

plt.title("Logistic Regression Decision Boundary")

plt.xlabel("Feature 1")

plt.ylabel("Feature 2")

plt.show()

**Output:**

A diagram of a logistic regression decision boundary

Description automatically generated

**Aim: Supervised Learning**

1. **K-nearest Neighbours (KNN) Classification Model.**

**Code**:

Implement import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.datasets import make\_blobs

from sklearn.neighbors import KNeighborsClassifier

from sklearn.model\_selection import train\_test\_split

X, y = make\_blobs(n\_samples = 500, n\_features = 2, centers = 4,cluster\_std = 1.5, random\_state = 4)

plt.style.use('seaborn')

plt.figure(figsize = (10,10))

plt.scatter(X[:,0], X[:,1], c=y, marker= '.',s=100,edgecolors='black')

plt.show()

A graph showing a number of dots

Description automatically generated

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, random\_state = 0)

knn5 = KNeighborsClassifier(n\_neighbors = 5)

knn1 = KNeighborsClassifier(n\_neighbors=1)

knn5.fit(X\_train, y\_train)

knn1.fit(X\_train, y\_train)

y\_pred\_5 = knn5.predict(X\_test)

y\_pred\_1 = knn1.predict(X\_test)

from sklearn.metrics import accuracy\_score

print("Accuracy with k=5", accuracy\_score(y\_test, y\_pred\_5)\*100)

print("Accuracy with k=1", accuracy\_score(y\_test, y\_pred\_1)\*100)

plt.figure(figsize = (15,5))

plt.subplot(1,2,1)

plt.scatter(X\_test[:,0], X\_test[:,1], c=y\_pred\_5, marker= '\*', s=100,edgecolors='black')

plt.title("Predicted values with k=5", fontsize=20)

plt.subplot(1,2,2)

plt.scatter(X\_test[:,0], X\_test[:,1], c=y\_pred\_1, marker= '\*', s=100,edgecolors='black')

plt.title("Predicted values with k=1", fontsize=20)

plt.show()

A graph showing a variety of values

Description automatically generated

Accuracy with k=5 93.60000000000001

Accuracy with k=1 90.4

**Practical 04**

**Aim: Features and Extraction**

**a) Identify the features in Iris dataset that are strongly correlated**

import numpy as np

import pandas as pd

from sklearn import datasets

import matplotlib.pyplot as plt

iris=datasets.load\_iris()

iris.data

iris.feature\_names



cov\_data=np.corrcoef(iris.data.T)

cov\_data

A close-up of numbers

Description automatically generated

img=plt.matshow(cov\_data,cmap=plt.cm.rainbow)

plt.colorbar(img,ticks=[-1,0,1],fraction=0.045)

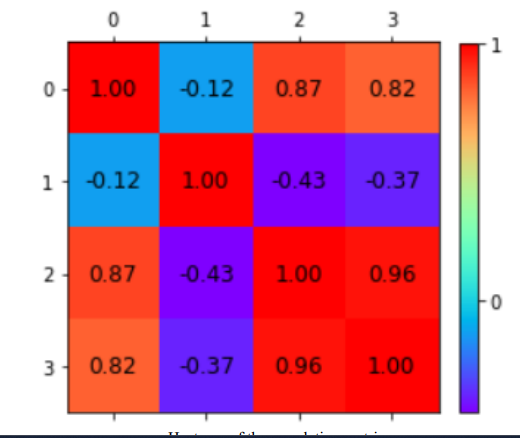
for x in range(cov\_data.shape[0]):

for y in range(cov\_data.shape[1]):

plt.text(x,y,"%0.2f"%cov\_data[x,y],size=12,color='black',ha="center",va="center")

plt.show()

**Output:**



**b) Implementation of principal component analysis (PCA) on the Iris dataset with Python**

**Code:**

import matplotlib.pyplot as plt

import mpl\_toolkits.mplot3d

from sklearn import datasets

iris = datasets.load\_iris()

from sklearn.decomposition import PCA

fig = plt.figure(1, figsize=(8, 6))

ax = fig.add\_subplot(111, projection="3d", elev=-150, azim=110)

X\_reduced = PCA(n\_components=3).fit\_transform(iris.data)

ax.scatter(

X\_reduced[:, 0],

X\_reduced[:, 1],

X\_reduced[:, 2],

c=iris.target,

s=40,

)

ax.set\_title("First three PCA dimensions")

ax.set\_xlabel("1st Eigenvector")

ax.xaxis.set\_ticklabels([])

ax.set\_ylabel("2nd Eigenvector")

ax.yaxis.set\_ticklabels([])

ax.set\_zlabel("3rd Eigenvector")

ax.zaxis.set\_ticklabels([])

plt.show()

**Output**:

A diagram of a graph

Description automatically generated

**Practical 05**

**Aim: Unsupervised Learning**

**a) Implement the K-Means clustering method**

**Code:**

from sklearn.cluster import KMeans

import pandas as pd

from sklearn.preprocessing import MinMaxScaler

from matplotlib import pyplot as plt

%matplotlib inline

df **=** pd**.**read\_csv("income.csv")

df**.**head()

A table of names and numbers

Description automatically generated

plt**.**scatter(df**.**Age,df['Income($)'])

plt**.**xlabel('Age')

plt**.**ylabel('Income($)')

plt.show()

A graph with blue dots

Description automatically generated

km **=** KMeans(n\_clusters**=**3)

y\_predicted **=** km**.**fit\_predict(df[['Age','Income($)']])

y\_predicted



df['cluster']**=**y\_predicted

df**.**head()

A white paper with blue text

Description automatically generated

km**.**cluster\_centers\_



df1 **=** df[df**.**cluster**==**0]

df2 **=** df[df**.**cluster**==**1]

df3 **=** df[df**.**cluster**==**2]

plt**.**scatter(df1**.**Age,df1['Income($)'],color**=**'green')

plt**.**scatter(df2**.**Age,df2['Income($)'],color**=**'red')

plt**.**scatter(df3**.**Age,df3['Income($)'],color**=**'black')

plt**.**scatter(km**.**cluster\_centers\_[:,0],km**.**cluster\_centers\_[:,1],color**=**'purple',marker**=**'\*',label**=**'centroid')

plt**.**xlabel('Age')

plt**.**ylabel('Income ($)')

plt**.**legend()

plt.show()

A graph of income and income

Description automatically generated with medium confidence

scaler **=** MinMaxScaler()

scaler**.**fit(df[['Income($)']])

df['Income($)'] **=** scaler**.**transform(df[['Income($)']])

scaler**.**fit(df[['Age']])

df['Age'] **=** scaler**.**transform(df[['Age']])

A close-up of numbers

Description automatically generated

plt**.**scatter(df**.**Age,df['Income($)'])

A graph with blue dots

Description automatically generated

km **=** KMeans(n\_clusters**=**3)

y\_predicted **=** km**.**fit\_predict(df[['Age','Income($)']])

y\_predicted



df['cluster']**=**y\_predicted

df**.**head()

A table of numbers and a few words

Description automatically generated with medium confidence

km**.**cluster\_centers\_

A number on a white background

Description automatically generated

df1 **=** df[df**.**cluster**==**0]

df2 **=** df[df**.**cluster**==**1]

df3 **=** df[df**.**cluster**==**2]

plt**.**scatter(df1**.**Age,df1['Income($)'],color**=**'green')

plt**.**scatter(df2**.**Age,df2['Income($)'],color**=**'red')

plt**.**scatter(df3**.**Age,df3['Income($)'],color**=**'black')

plt**.**scatter(km**.**cluster\_centers\_[:,0],km**.**cluster\_centers\_[:,1],color**=**'purple',marker**=**'\*',label**=**'centroid')

plt**.**legend()

plt.show()

**Output**:

A graph with red green and blue dots

Description automatically generated

**Practical 06**

**Aim: Classify the data using Support vector machine**

**Code:**

# importing scikit learn with make\_blobs

from sklearn.datasets import make\_blobs

# creating datasets X containing n\_samples

# Y containing two classes

X, Y = make\_blobs(n\_samples=500, centers=2,random\_state=0, cluster\_std=0.40)

import matplotlib.pyplot as plt

import numpy as np

# plotting scatters

plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='spring');

# creating linspace between -1 to 3.5

xfit = np.linspace(-1, 3.5)

# plotting scatter

plt.scatter(X[:, 0], X[:, 1], c=Y, s=50, cmap='spring')

# plot a line between the different sets of data

for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:

yfit = m \* xfit + b

plt.plot(xfit, yfit, '-k')

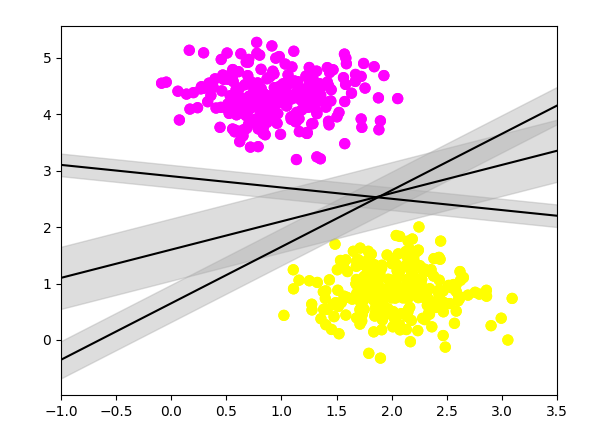
plt.fill\_between(xfit, yfit - d, yfit + d, edgecolor='none',

color='#AAAAAA', alpha=0.4)

plt.xlim(-1, 3.5);

plt.show()

**Output**:



**Practical 07**

**Aim: Implement the decision tree using python**

**Code:**

import pandas as pd

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier

import matplotlib.pyplot as plt

df = pd.read\_csv("data.csv")

d = {'UK': 0, 'USA': 1, 'N': 2}

df['Nationality'] = df['Nationality'].map(d)

d = {'YES': 1, 'NO': 0}

df['Go'] = df['Go'].map(d)

print(df)

A table of numbers with text

Description automatically generated with medium confidence

features = ['Age', 'Experience', 'Rank', 'Nationality']

X = df[features]

y = df['Go']

print(X)

A table of numbers with text

Description automatically generated with medium confidence

print(y)

A number on a white background

Description automatically generated

dtree = DecisionTreeClassifier()

dtree = dtree.fit(X, y)

tree.plot\_tree(dtree, feature\_names=features)

**Output**:

A diagram of a number of samples

Description automatically generated with medium confidence