1. FIND -S

training\_data = [

['Sunny', 'Warm', 'Normal', 'Strong', 'Warm', 'Same', 'Yes'],

['Sunny', 'Warm', 'High', 'Strong', 'Warm', 'Same', 'Yes'],

['Rainy', 'Cold', 'High', 'Strong', 'Warm', 'Change', 'No'],

['Sunny', 'Warm', 'High', 'Strong', 'Cool', 'Change', 'Yes']

]

def find\_s\_algorithm(training\_data):

hypothesis = None

for instance in training\_data:

attributes, target = instance[:-1], instance[-1]

if target == 'Yes':

if hypothesis is None:

hypothesis = attributes

else:

for i in range(len(hypothesis)):

if hypothesis[i] != attributes[i]:

hypothesis[i] = '?'

return hypothesis

final\_hypothesis = find\_s\_algorithm(training\_data)

print("Training Data:")

for row in training\_data:

print(row)

print("\nFinal Hypothesis:")

print(final\_hypothesis)

2.CANDIDATE ELIMINATION ALGORITHM

data = [

(["Japan", "Honda", "Blue", "1980", "Economy"], "Positive"),

(["Japan", "Toyota", "Green", "1970", "Sports"], "Negative"),

(["Japan", "Toyota", "Blue", "1990", "Economy"], "Positive"),

(["USA", "Chrysler", "Red", "1980", "Economy"], "Negative"),

(["Japan", "Honda", "White", "1980", "Economy"], "Positive")

]

S = ["ϕ", "ϕ", "ϕ", "ϕ", "ϕ"]

G = [["?", "?", "?", "?", "?"]]

def update\_S(S, instance):

for i in range(len(S)):

if S[i] == "ϕ":

S[i] = instance[i]

elif S[i] != instance[i]:

S[i] = "?"

return S

def update\_G(G, instance):

new\_G = []

for g in G:

if any(g[i] != "?" and g[i] != instance[i] for i in range(len(g))):

new\_G.append(g)

else:

for i in range(len(g)):

if g[i] == "?":

new\_g = g[:]

new\_g[i] = instance[i]

new\_G.append(new\_g)

return new\_G

for instance, label in data:

if label == "Positive":

S = update\_S(S, instance)

G = [g for g in G if all(s == "?" or g[i] == "?" or s == g[i] for i, s in enumerate(S))]

elif label == "Negative":

G = update\_G(G, instance)

print("Final Specific Boundary S:", S)

print("Final General Boundary G:", G)

3.LINEAR REGRESSION

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

dataset = pd.read\_csv('Salary\_Data.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

dataset.head()

from sklearn.model\_selection

import train\_test\_split X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 1/3, random\_state = 0)

from sklearn.linear\_model import LinearRegression

regressor = LinearRegression()

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

y\_pred = regressor.predict(X\_test)

. plt.scatter(X\_train, y\_train, color = 'red')

plt.plot(X\_train, regressor.predict(X\_train), color = 'blue')

plt.title('Salary vs Experience (Training set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

4.LOGISTIC REGRESSION

import numpy as np

import pandas as pd

dataset = pd.read\_csv("D:/GEO/BE COURSES/2022 dec/LAB/DATASET/breastcancer.csv")

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = 2)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

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

from sklearn.linear\_model import LogisticRegression

classifier = LogisticRegression(random\_state = 0)

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

from sklearn.metrics import confusion\_matrix,

accuracy\_score y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred) print(cm)

print('Accuracy Score:confusion matrix')

accuracy\_score(y\_test, y\_pred)

5.EXPECTATION AND MAXIMIZATION ALGORITHM

from sklearn.cluster import KMeans

from sklearn.mixture import GaussianMixture

import sklearn.metrics as metrics

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#names = ['Sepal\_Length','Sepal\_Width','Petal\_Length','Petal\_Width', 'Class']

dataset = pd.read\_csv("D:/GEO/BE COURSES/2022 dec/LAB/IRIS.csv")

X = dataset.iloc[:, :-1]

label = {'Iris-setosa': 0,'Iris-versicolor': 1, 'Iris-virginica': 2} y = [label[c] for c in dataset.iloc[:, -1]] plt.figure(figsize=(14,7))

colormap=np.array(['red','lime','black'])

# REAL PLOT

plt.subplot(1,3,1) plt.title('Real')

plt.scatter(X.petal\_length ,X.petal\_width,c=colormap[y])

# GMM PLOT

gmm=GaussianMixture(n\_components=3, random\_state=0).fit(X) y\_cluster\_gmm=gmm.predict(X) plt.subplot(1,3,3)

plt.title('GMM Classification')

plt.scatter(X.petal\_length,X.petal\_width,c=colormap[y\_cluster\_gmm])

print('The accuracy score of EM: ',metrics.accuracy\_score(y, y\_cluster\_gmm))

#print('The Confusion matrix of EM:\n ',metrics.confusion\_matrix(y, y\_cluster\_gmm)

**6.KNN ALGORITHM**

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

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn import datasets

iris=datasets.load\_iris()

x = iris.data y = iris.target print ('sepal-length', 'sepal-width', 'petal-length', 'petal-width') print(x)

print('class: 0-Iris-Setosa, 1- Iris-Versicolour, 2- Iris-Virginica') print(y)

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,test\_size=0.3)

KNeighborsClassifier(n\_neighbors=5)

classifier.fit(x\_train, y\_train)

y\_pred=classifier.predict(x\_test)

print('Confusion Matrix')

print(confusion\_matrix(y\_test,y\_pred))

print('Accuracy Metrics')

print(classification\_report(y\_test,y\_pred))

7.DECISION TREE

car price prediction

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

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

from sklearn.tree import DecisionTreeRegressor

data = pd.read\_csv("CarPrice.csv") data.head() data.shape data.isnull().sum()

data.describe()

data.CarName.unique()

sns.set\_style("whitegrid")

plt.figure(figsize=(15, 10))

sns.distplot(data.price)

plt.show()

print(data.corr())

plt.figure(figsize=(20, 15))

correlations = data.corr()

sns.heatmap(correlations, cmap="coolwarm", annot=True)

plt.show()

#predict = "price"

x=np.array(data[["symboling", "wheelbase", "carlength",

"carwidth", "carheight", "curbweight",

"enginesize", "boreratio", "stroke",

"compressionratio", "horsepower", "peakrpm",

"citympg", "highwaympg", "price"]])

y=np.array(data.price)

#x = np.array(data.drop([predict], 1))

#=data.iloc[:,0:15]

#y = np.array(data[predict])

from sklearn.model\_selection import train\_test\_split xtrain, xtest, ytrain, ytest = train\_test\_split(x, y, test\_size=0.2)

from sklearn.tree import DecisionTreeRegressor

model = DecisionTreeRegressor()

model.fit(xtrain, ytrain)

predictions = model.predict(xtest)

from sklearn.metrics import mean\_absolute\_error

model.score(xtest, predictions)

**8.NAIVEBAYES THEOREM**

import numpy as np

import pandas as pd

dataset = pd.read\_csv("breastcancer.csv")

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

X=dataset.iloc[:,0:10]

y=dataset.Class

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 0)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

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

from sklearn.naive\_bayes import GaussianNB

classifier = GaussianNB()

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

GaussianNB(priors=None, var\_smoothing=1e-09)

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

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

accuracy\_score(y\_test, y\_pred)

9. LINEAR AND POLYNOMIAL REGRESSION

import numpy as np

import pandas as pd

dataset = pd.read\_csv("D:/GEO/BE COURSES/2022 dec/LAB/DATASET/breastcancer.csv")

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

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

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.30, random\_state = 2)

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

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

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

from sklearn.linear\_model import LogisticRegression classifier = LogisticRegression(random\_state = 0)

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

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='warn', tol=0.0001, verbose=0, warm\_start=False)

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

y\_pred = classifier.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

print('Accuracy Score:confusion matrix')

accuracy\_score(y\_test, y\_pred)

10.CREDICT SCORE

11.IRIS FLOWER KNN

12.HOUSE PRICE PREDICTION