CS1702A **MACHINE LEARNING LABORATORY** **L T P C**

0 0 4 2

LIST OF EXPERIMENTS:

1. Implement the concept of decision trees with suitable data set from real world problem and classifythe data set to produce new sample.
2. Detecting Spam mails using Support vector machine
3. Implement facial recognition application with artificial neural network
4. Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points.Select appropriate data set for your experiment and draw graphs.
5. Implement character recognition using Multilayer Perceptron
6. Implement the kmeans algorithm
7. Implement the Dimensionality Reduction techniques
8. Write a program to construct a Bayesian network considering medical data. Use this model to demonstrate the diagnosis of heart patients using standard Heart Disease Data Set. You can use Java/Python ML library classes/API.
9. Using Weka Tool Perform a. Data preprocessing by selecting or filtering attributes b. Data preprocessing for handling missing value
10. Mini-project: students work in team on any socially relevant problem that needs a machine learning based solution, and evaluate the model performance.

TOTAL: 45 PERIODS

COURSE OUTCOMES:

- Understand the implementation procedures for the machine learning algorithms.
- Design Python programs for various Learning algorithms.
- Apply appropriate Machine Learning algorithms to data sets
- Identify and apply Machine Learning algorithms to solve real world problems.

LIST OF EQUIPMENT FOR A BATCH OF 30 STUDENTS:

SOFTWARE: Python/Java with ML Package/R

HARDWARE: 30 terminals.

INDEX

Lab In-charge Sign

EXP NO:1

DATE:

DECISION TREES

AIM:

Implement the concept of decision trees with suitable data set from real world problem and classify the data set to produce new sample.

ALGORITHM:

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 calledthe final node as a leaf node.

PROGRAM:

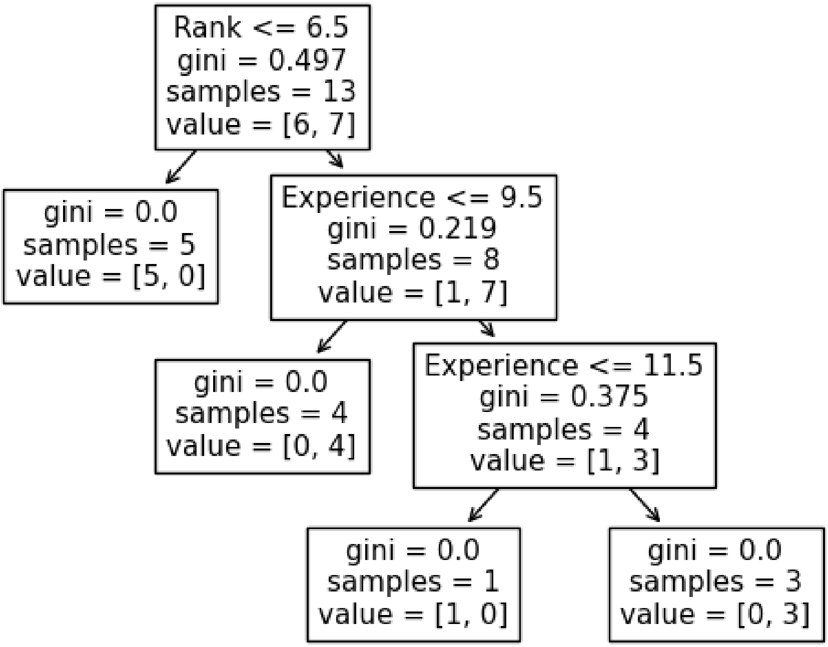
```
from matplotlib import pyplot as plt
import pandas
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
df = pandas.read_csv("data.csv")
d = {'UK': 0, 'USA': 1, 'N': 2}
df['Nationality'] = df['Nationality'].map(d)
d = {'YES': 1, 'NO': 0}
df['Go'] = df['Go'].map(d)
features = ['Age', 'Experience', 'Rank', 'Nationality']
X = df[features]
y = df['Go']
dtree = DecisionTreeClassifier()
```

```
dtree = dtree.fit(X, y)
tree.plot_tree(dtree,feature_names=features)
plt.show()
```

INPUT (data.csv):

```
Age,Experience,Rank,Nationality,Go
36,10,9,UK,NO
42,12,4,USA,NO
23,4,6,N,NO
52,4,4,USA,NO
43,21,8,USA,YES
44,14,5,UK,NO
66,3,7,N,YES
35,14,9,UK,YES
52,13,7,N,YES
35,5,9,N,YES
24,3,5,USA,NO
18,3,7,UK,YES
45,9,9,UK,YES
```

OUTPUT:



RESULT:

Thus the decision tree is implemented successfully.

EXP NO:2

DATE:

DETECTING SPAM MAILS

AIM:

Implement Detecting Spam mails using Support vector machine.

ALGORITHM:

1. Import the Modules
2. read spam mails using pandas
3. Removing the duplicate rows and keeping only the first one. Also, we are using Label Encoder toassign numbers to labels.
    1. not spam
    2. spam
4. Creating the functions to extract important features from the text. We are removing any type ofpunctuation or stop words like the, he, she, etc. Stop words are the words that contribute in the formation of the sentences but these are not useful in detecting whether our SMS is spam or not.
5. Using the train_test_split function to convert our dataset into training and testing.
6. Creating our SVM model using inbuilt function of keras.
7. Finally predicting the result. Also, we are using pickle to save our trained model. Later we will usethis model in Tkinter to create GUI.

PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn import svm
spam = pd.read_csv('data.csv')
z = spam['EmailText']
```

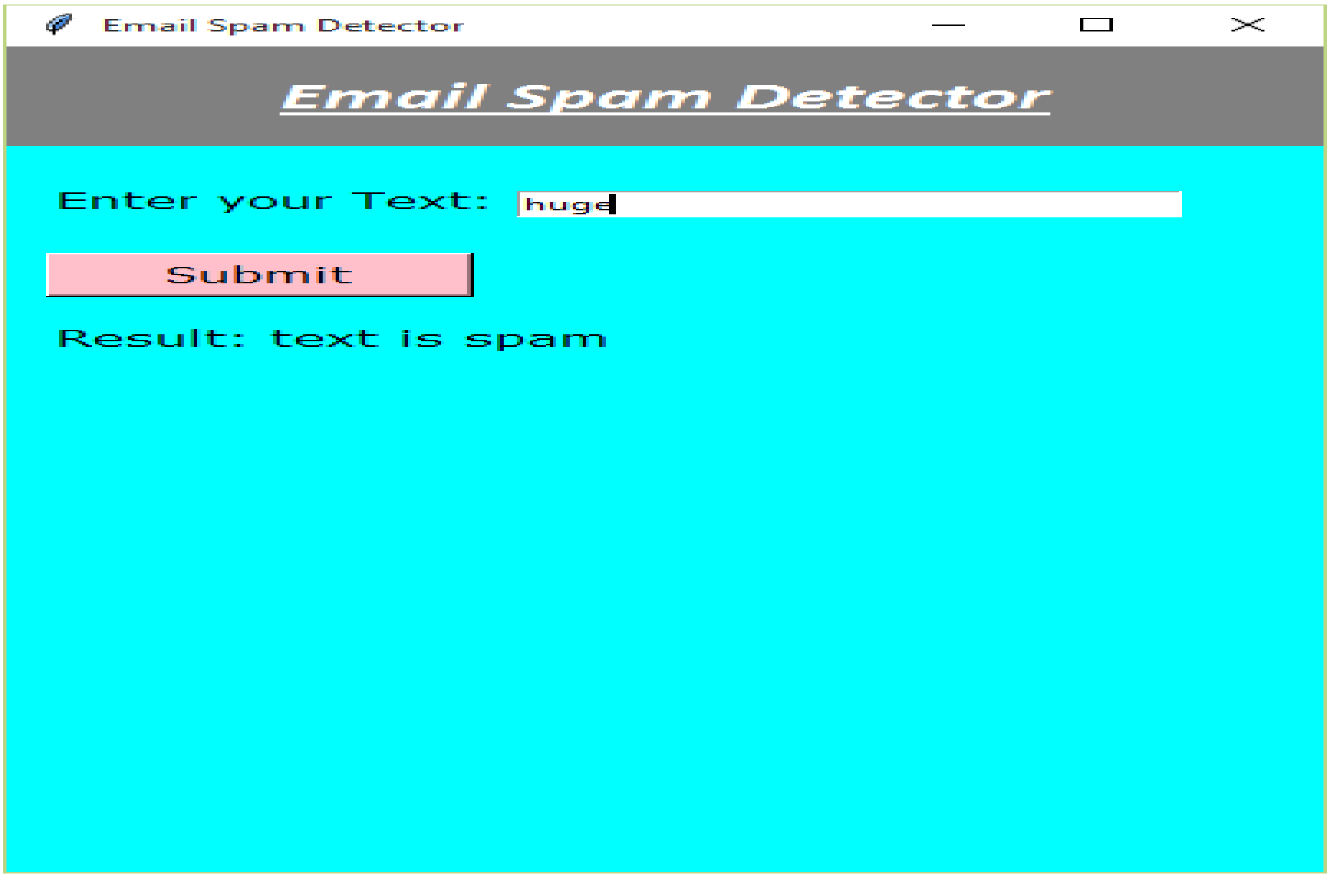
```
y = spam["Label"]
z_train, z_test,y_train, y_test = train_test_split(z,y,test_size = 0.2)
cv = CountVectorizer()
features = cv.fit_transform(z_train)
model = svm.SVC()
model.fit(features,y_train)
features_test = cv.transform(z_test)
print(model.score(features_test,y_test))
features_test = cv.transform(z_test)
print("Accuracy: {}".format(model.score(features_test,y_test)))
import pickle
from tkinter import *
def check_spam():
    text = spam_text_Entry.get()
    with open('data.csv') as file:
        contents = file.read()
        if text in contents:
            print(text,"text is spam")
            my_string_var.set("Result: text is spam")
        else:
             print(text,"text not a spam")
             my_string_var.set("Result: text not a spam")
win = Tk()
win.geometry("400x600")
win.configure(background="cyan")
win.title("Email Spam Detector")
title = Label(win, text="Email Spam Detector",
bg="gray",width="300",height="2",fg="white",font=("Calibri 20 bold italic underline")).pack()
spam_text = Label(win, text="Enter your Text: ",bg="cyan", font=("Verdana
12")).place(x=12,y=100)
spam_text_Entry = Entry(win, textvariable=spam_text,width=33)
spam_text_Entry.place(x=155, y=105)
my_string_var = StringVar()
my_string_var.set("Result: ")
```

```
print_spam = Label(win,textvariable=my_string_var,bg="cyan",font=("Verdana 12")).place(x=12,y=200)
Button = Button(win,
text="Submit",width="12",height="1",activebackground="red",bg="Pink",command=check_spam,
font=("Verdana 12")).place(x=12,y=150)
win.mainloop()
```

Input (data.csv):

```
EmailText,Label
sale,spam
gasssss,ham
huge,spam
tint,spam
ginger,spam
```

OUTPUT:



RESULT:

Spam mails have been successfully detected using SVM.

EXP NO:3

DATE:

FACIAL RECOGNITION APPLICATION

AIM :

Implement facial recognition application with artificial neural network .

ALGORITHM:

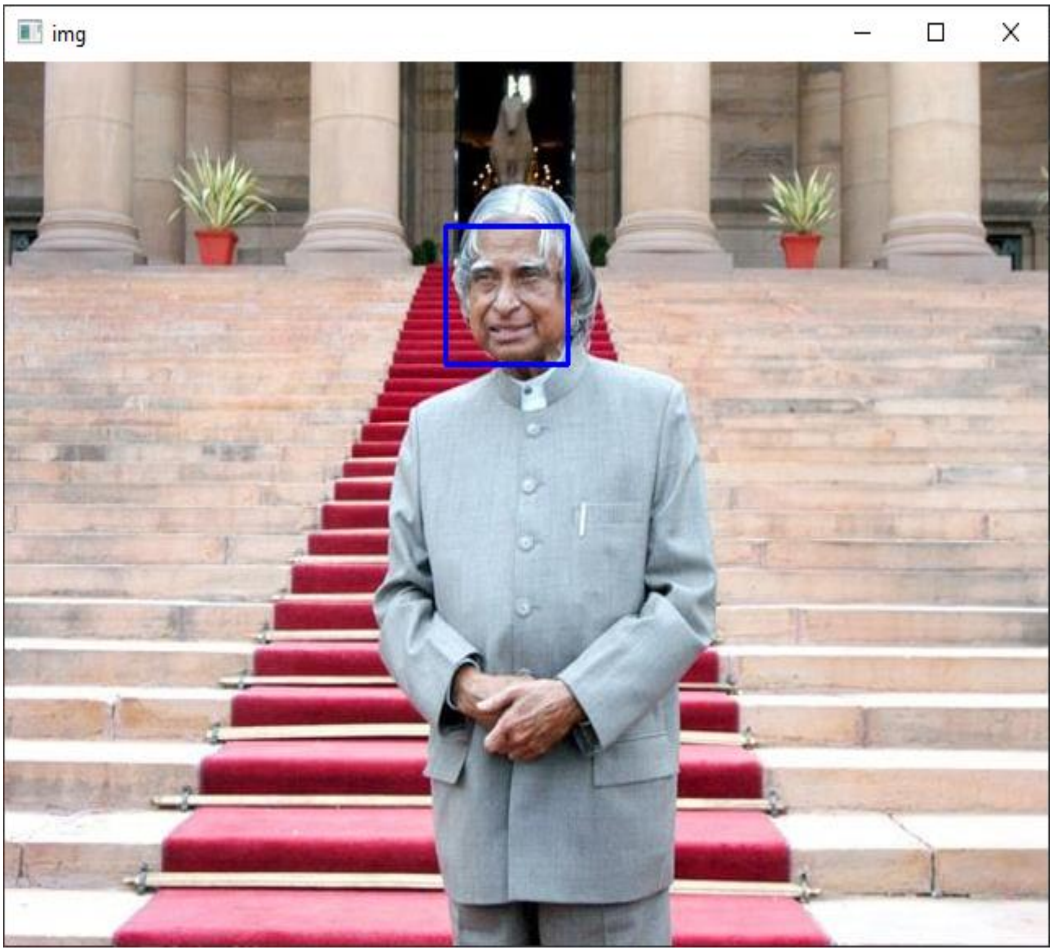
1. Import the python packages opencv,cv-3,cv
2. load all the pre trained packages such as CascadeClassifier, face_cascade.detectMultiScale from the location of python library
3. Read the image from the system
4. Convert the image to Grey Scale
5. Predict the scaling factor and others using the face_cascade.detectMultiScale methods in cv2 library
6. Start with testing the model which we have created

PROGRAM:

```
import cv2
face_cascade = cv2.CascadeClassifier('C:\\Users\\Admin\\AppData\\
Local\\Programs\\Python\\Python310\\Lib\\site-packages\\cv2\\
data\\haarcascade_frontalface_default.xml')
img = cv2.imread('C:\\Users\\Admin\\Desktop\\New notes\\LAB\\kalam.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)
for (x, y, w, h) in faces:
    cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)
cv2.imshow('img', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

INPUT (kalam.png):

OUTPUT:



RESULT:

Thus the Face recognition application is implemented successfully.

EXP NO:4

DATE:

REGRESSION ALGORITHM

AIM:

Implement the non-parametric Locally Weighted Regression algorithm in order to fit data points.Select appropriate data set for your experiment and draw graphs.

ALGORITHM:

1. Read the Given data Sample to X and the curve (linear or non linear) to Y.
2. Set the value for Smoothening parameter or Free parameter say τ
3. Set the bias /Point of interest set x0 which is a subset of X.
4. Determine the weight matrix using :
5. Determine the value of model term parameter β using:
6. Prediction = x0*β

PROGRAM:

```
import numpy as np # linear algebra
import pandas as pd # data processing
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
class LocallyWeightedRegression:
    #maths behind Linear Regression:
    # theta = inv(X.T*W*X)*(X.T*W*Y)this will be our theta whic will
    # be learnt for each point
    # initializer of LocallyWeighted Regression that stores tau as parameters
    def __init__(self, tau = 0.01):
        self.tau = tau
    def kernel(self, query_point, X):
        Weight_matrix = np.mat(np.eye(len(X)))
        for idx in range(len(X)):
```

```
        Weight_matrix[idx,idx] = np.exp(np.dot(X[idx]-query_point, (X[idx]-query_point).T)/
        (-2*self.tau*self.tau))
    return Weight_matrix
# function that makes the predictions of the output of a given query point
def predict(self, X, Y, query_point):
    q = np.mat([query_point, 1])
    X = np.hstack((X, np.ones((len(X), 1))))
    W = self.kernel(q, X)
    theta = np.linalg.pinv(X.T*(W*X))*(X.T*(W*Y))
    pred = np.dot(q, theta)
    return pred
#function that fits and predicts the output of all query points
def fit_and_predict(self, X, Y):
    Y_test, X_test = [], np.linspace(-np.max(X), np.max(X), len(X))
    for x in X_test:
        pred = self.predict(X, Y, x)
        Y_test.append(pred[0][0])
    Y_test = np.array(Y_test)
    return Y_test
# function that computes the score rmse
def score(self, Y, Y_pred):
    return np.sqrt(np.mean((Y-Y_pred)**2))
# function that fits as well as shows the scatter plot of all points
def fit_and_show(self, X, Y):
    Y_test, X_test = [], np.linspace(-np.max(X), np.max(X), len(X))
    for x in X_test:
        pred = self.predict(X, Y, x)
        Y_test.append(pred[0][0])
    Y_test = np.array(Y_test)
    plt.style.use('seaborn')
    plt.title("The scatter plot for the value of tau = %.5f"% self.tau)
    plt.scatter(X, Y, color = 'red')
    plt.scatter(X_test, Y_test, color = 'green')
    plt.show()
```

```
# reading the csv files of the given dataset
dfx = pd.read_csv('weightedX.csv')
dfy = pd.read_csv('weightedY.csv')
# store the values of dataframes in numpy arrays
X = dfx.values
Y = dfy.values
# normalising the data values
u = X.mean()
std = X.std()
X = ((X-u)/std)
tau = 0.2
model = LocallyWeightedRegression(tau)
Y_pred = model.fit_and_predict(X, Y)
model.fit_and_show(X, Y)
```

INPUT:(weightedX.csv)

1.2421

2.3348

0.13264

2.347

6.7389

3.7089

11.853

-1.8708

4.5025

3.2798

1.7573

3.3784

11.47

9.0595

-2.8174

9.3184

8.4211

0.86215

7.5544

-3.9883

INPUT:(weightedY.csv)

1.1718

1.8824

0.34283

2.1057

1.6477

2.3624

2.1212

-0.79712

2.0311

1.9795

1.471

2.4611

1.9819

1.1203

-1.3701

1.0287

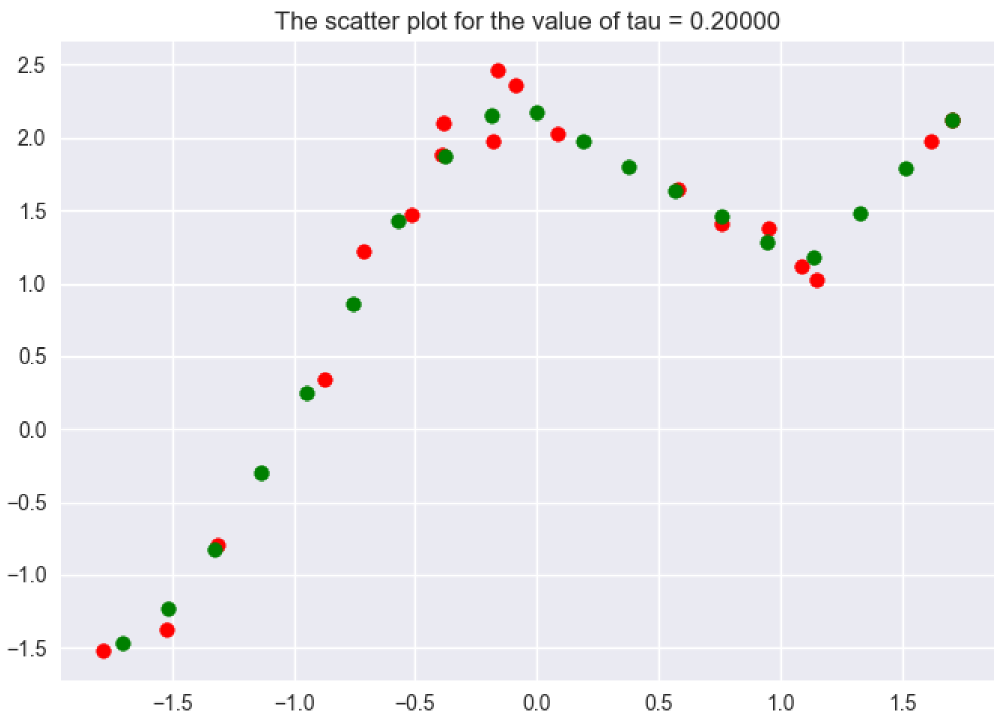
1.3808

1.2178

1.4084

-1.5209

OUTPUT:



RESULT:

Non-parametric Locally Weighted Regression algorithm is implemented successfully.

EXP NO:5

DATE:

K MEANS ALGORITHM

AIM:

To implement the optimal centroid locations using K Means Algorithm

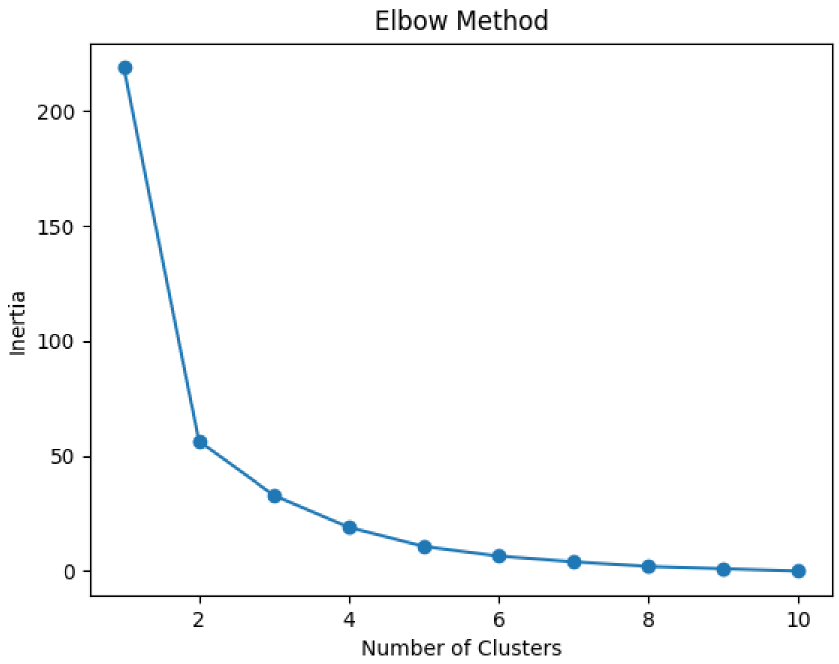
ALGORITHM:

1. Randomly select the first centroid from the data points.
2. For each data point compute its distance from the nearest, previouslychosen centroid.
3. Select the next centroid from the data points such that the probability of choosing a point as centroid is directly proportional to its distance from thenearest, previously chosen centroid.
4. Repeat steps 2 and 3 until k centroids have been sampled

PROGRAM:

```
import sys
import matplotlib
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
x=[4,5,10,4,3,11,14,6,10,12]
y=[21,19,24,17,16,25,24,22,21,21]
data = list(zip(x, y))
inertias = []
for i in range(1,11):
    kmean=KMeans(n_clusters=i)
    kmean.fit(data)
    inertias.append(kmean.inertia_)
plt.plot(range(1,11),inertias,marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```

OUTPUT:



RESULT:

Thus the Program for K Means Algorithms has been executed successfully

.

EXP NO:6

DATE:

CHARACTER RECOGNITION

AIM :

Implement character recognition using Multilayer Perceptron

ALGORITHM

1. Import the python library and label encoder
2. Load the data using the read_csv function
3. Display the data .
4. Convert all the string labels into numbers.
5. Split the dataset into training and testing dataset.
6. Visualizing the learnt weights of the input layer
7. Calculate the accuracy score.

PROGRAM:

```
import numpy as np
import pandas as pd
# Load data
data=pd.read_csv('HR_comma_sep.csv')
data.head()
# Import LabelEncoder
from sklearn import preprocessing

# Creating labelEncoder
le = preprocessing.LabelEncoder()

# Converting string labels into numbers.
data['salary']=le.fit_transform(data['salary'])
data['Departments']=le.fit_transform(data['Departments'])
```

```
# Spliting data into Feature and
X=data[['satisfaction_level','last_evaluation','number_project','average_montly_hours','time_spend_company','Work_accident','promotion_last_5years','Departments','salary']]
y=data['left']

# Import train_test_split function
from sklearn.model_selection import train_test_split

# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=45)  # 70% training and 30% test
# Import MLPClassifer
from sklearn.neural_network import MLPClassifier

# Create model object
clf = MLPClassifier(hidden_layer_sizes=(6,5),
                    random_state=5,
                    verbose=True,
                    learning_rate_init=0.01)

# Fit data onto the model
clf.fit(X_train,y_train)
# Make prediction on test dataset
ypred=clf.predict(X_test)

# Import accuracy score
from sklearn.metrics import accuracy_score

# Calcuate accuracy
accuracy_score(y_test,ypred)
```

INPUT :(HR_comma_sep.csv)

satisfaction_level,last_evaluation,number_project,average_montly_hours,

time_spend_company,Work_accident,left,promotion_last_5years,Departments, salary

0.38,0.53,2,157,3,0,1,0,sales,low

0.80,0.86,5,262,6,0,1,0,sales,medium

0.11,0.88,7,272,4,0,1,0, sales,medium

0.72,0.87,5,223,5,0,1,0, sales,low

0.37,0.52,2,159,3,0,1,0,sales,low

OUTPUT:

```
C:\Users\Administrator\PycharmProjects\veena\venv\Scripts\python.exe
C:\Users\Administrator\PycharmProjects\veena\percept.py
Iteration 1, loss = 0.00456078
Iteration 2, loss = 0.00055361
Iteration 3, loss = 0.00029325
Iteration 4, loss = 0.00025340
Iteration 5, loss = 0.00024135
Iteration 6, loss = 0.00023436
Iteration 7, loss = 0.00022864
Iteration 8, loss = 0.00022336
Iteration 9, loss = 0.00021831
Iteration 10, loss = 0.00021343
Iteration 11, loss = 0.00020870
Iteration 12, loss = 0.00020411
Iteration 13, loss = 0.00019966
Iteration 14, loss = 0.00019535
Training loss did not improve more than tol=0.000100 for 10 consecutive epochs. Stopping.

Process finished with exit code 0
```

RESULT:

Thus the Character recognition is performed successfully using MLP.

EXP NO:7

DATE:

DIMENSIONALITY REDUCTION TECHNIQUES

AIM:

Construct a Bayesian network considering medical data. Use this model todemonstrate the diagnosis of heart patients using a standard Heart Disease Data Set.

ALGORITHM:

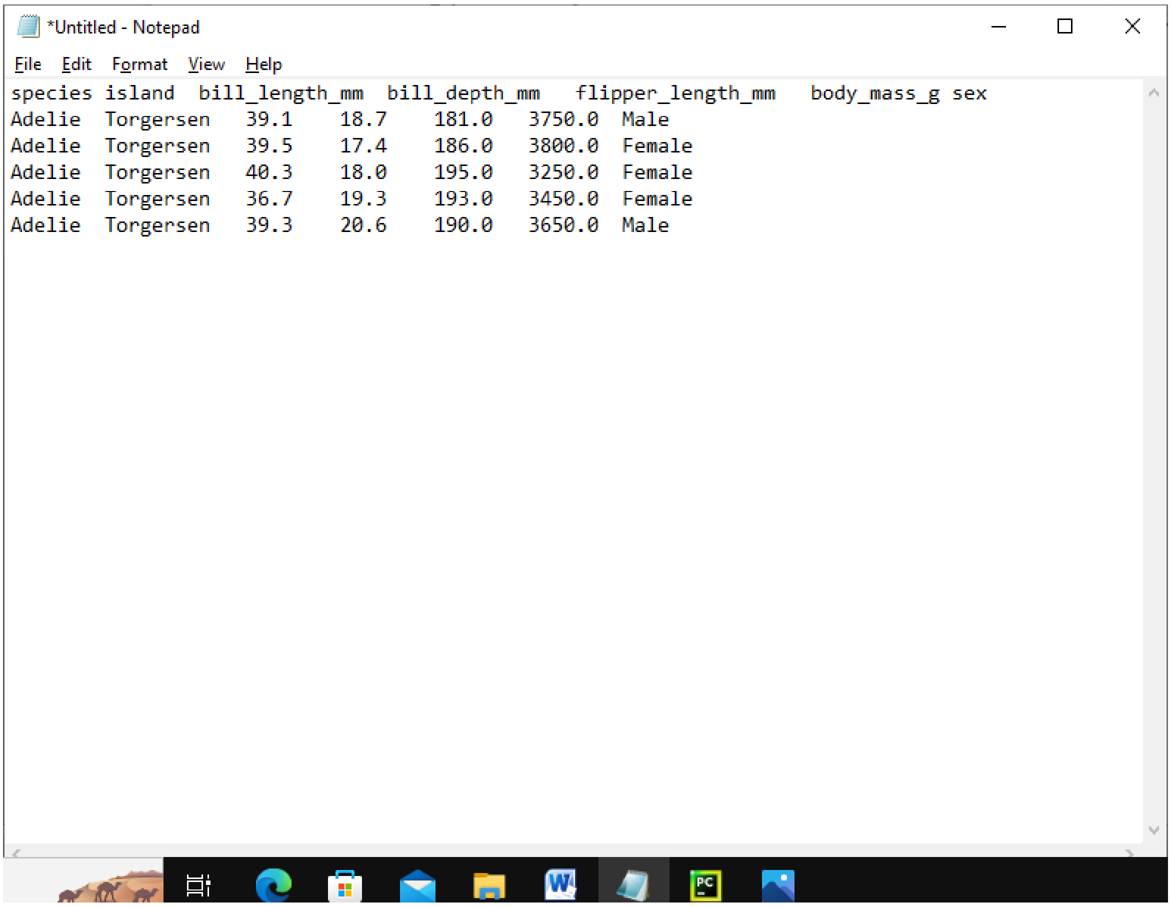
1. Import the python library and label encoder
2. Load the data using the read_csv function through input data (penguins.csv)
3. Display the data .
    - species: species of penguin
    - island: places in island
    - bill length: in mm
    - bill depth: in mm
    - flipp length : in mm
    - body mass : in mm
    - sex: male/female
4. Convert all the string labels into numbers.
5. Split the dataset into training and testing dataset.
6. Reduce the dimensions of data set and create the diagram on principle component analysis.
7. Calculate the accuracy score.

PROGRAM:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler, MinMaxScaler
from sklearn.decomposition import PCA, KernelPCA, FastICA, NMF, FactorAnalysis
```

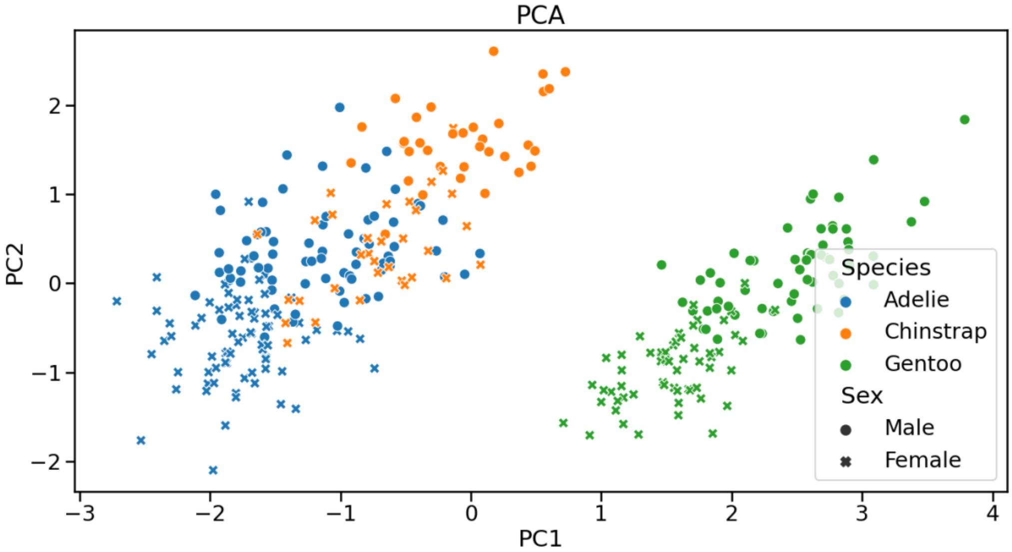
```
penguins = sns.load_dataset("penguins")
penguins = (penguins.dropna())
penguins.head()
data = (penguins.select_dtypes(np.number))
data.head()
random_state = 0
pca_pl = make_pipeline(StandardScaler(),PCA(n_components=2,random_state=random_state))
pcs = pca_pl.fit_transform(data)
pcs[0:5,:]
pcs_df = pd.DataFrame(data = pcs ,columns = ['PC1', 'PC2'])
pcs_df['Species'] = penguins.species.values
pcs_df['Sex'] = penguins.sex.values
pcs_df.head()
plt.figure(figsize=(12,10))
with sns.plotting_context("talk",font_scale=1.25):
    sns.scatterplot(x="PC1", y="PC2",data=pcs_df,hue="Species",style="Sex",s=100)
    plt.xlabel("PC1")
    plt.ylabel("PC2")
    plt.title("PCA", size=24)
plt.savefig("PCA_Example_in_Python.png",format='png',dpi=75)
plt.show()
```

INPUT:




```
species island  bill_length_mm  bill_depth_mm   flipper_length_mm   body_mass_g sex
Adelie  Torgersen   39.1    18.7    181.0   3750.0  Male
Adelie  Torgersen   39.5    17.4    186.0   3800.0  Female
Adelie  Torgersen   40.3    18.0    195.0   3250.0  Female
Adelie  Torgersen   36.7    19.3    193.0   3450.0  Female
Adelie  Torgersen   39.3    20.6    190.0   3650.0  Male
```

OUTPUT:



RESULT:

Thus the Dimensionality reduction for image is successfully performed.

EXP NO:8

DATE:

BAYESIAN NETWORK

AIM:

Construct a Bayesian network considering medical data. Use this model todemonstrate the diagnosis of heart patients using a standard Heart Disease Data Set.

ALGORITHM:

1. age: age in years
2. sex: sex (1 = male; 0 = female)
3. cp: chest pain type
    1. Value 1: typical angina
    2. Value 2: atypical angina
    3. Value 3: non-anginal pain
    4. Value 4: asymptomatic
4. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
5. chol: serum cholestoral in mg/dl
6. fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg: resting electrocardiographic results
    5. Value 0: normal
    6. Value 1: having ST-T wave abnormality (T wave inversions and/or STelevation or depression of > 0.05 mV)
    7. Value 2: showing probable or definite left ventricular hypertrophy by Estes'criteria
8. thalach: maximum heart rate achieved
9. exang: exercise induced angina (1 = yes; 0 = no)
10. oldpeak = ST depression induced by exercise relative to rest
11. slope: the slope of the peak exercise ST segment
    8. Value 1: upsloping
    9. Value 2: flat
    10. Value 3: downsloping
12. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
13. Heartdisease: It is integer valued from 0 (no presence) to 4.

PROGRAM:

```
import numpy as np

import csv

import pandas as pd

from pgmpy.models import BayesianModel

from pgmpy.estimators import MaximumLikelihoodEstimator
from pgmpy.inference import VariableElimination

#read Cleveland Heart Disease data

heartDisease = pd.read_csv('heart.csv')

heartDisease = heartDisease.replace('?',np.nan)

#display the data

print('Few examples from the dataset are given below')

print(heartDisease.head())

#Model Bayesian Network

Model=BayesianModel([('age','trestbps'),('age','fbs'),

('sex','trestbps'),('exang','trestbps'),('trestbps','heartdise

ase'),('fbs','heartdisease'),('heartdisease','restecg'),

('heartdisease','thalach'),('heartdisease','chol')])

#Learning CPDs using Maximum Likelihood Estimators

print('\n Learning CPD using Maximum likelihood estimators')

model.fit(heartDisease,estimator=MaximumLikelihoodEstimator)

# Inferencing with Bayesian Network

print('\n Inferencing with Bayesian Network:')

HeartDisease_infer = VariableElimination(model)

#computing the Probability of HeartDisease given Age

print('\n 1. Probability of HeartDisease given Age=30')

q=HeartDisease_infer.query(variables=['heartdisease'],evidence

={'age':28})
```

```
print(q['heartdisease'])
```

#computing the Probability of HeartDisease given cholesterol

```
print('\n 2. Probability of HeartDisease given cholesterol=100')

q=HeartDisease_infer.query(variables=['heartdisease'],evidence={'chol':100})

print(q['heartdisease'])
```

OUTPUT:

Few examples from the dataset are given below

age sex cp trestbps ...slope ca thal heartdisease

0 63 1 1 145 ... 3 0 6 0

1 67 1 4 160 ... 2 3 3 2

2 67 1 4 120 ... 2 2 7 1

3 37 1 3 130 ... 3 0 3 0

4 41 0 2 130 ... 1 0 3 0

[5 rows x 14 columns]

Learning CPD using Maximum likelihood estimators

Inferencing with Bayesian Network:

1. Probability of HeartDisease given Age=28

heartdisease	phi(heartdisease)
heartdisease_0	0.6791
heartdisease_1	0.1212

heartdisease_2	0.0810
heartdisease_3	0.0939
heartdisease_4	0.0247

2. Probability of HeartDisease given cholesterol=100

heartdisease	phi(heartdisease)
heartdisease_0	0.5400
heartdisease_1	0.1533
heartdisease_2	0.1303
heartdisease_3	0.1259
heartdisease_4	0.0506

RESULT:

Thus Bayesian Network on Medical Data(Heart Disease Dataset) is successfully Implemented.

EXP NO:9

DATE:

DATA PREPROCESSING

AIM :

Using Weka Tool Perform a. Data preprocessing by selecting or filtering attributes b. Data preprocessing for handling missing value

ALGORITHM

1. To demonstrate the preprocessing, we will use the **Weather** database that is provided in theinstallation.

- **Open file** ... option under the **Preprocess** tag select the **weather-nominal.arff** file.
- The **weather** database contains five fields - outlook, temperature, humidity, windy and play.
- We select an attribute from this list by clicking on it, further details on the attribute itself aredisplayed on the right hand side.

2. In the Selected Attribute subwindow, you can observe the following −

- The name and the type of the attribute are displayed.
- The type for the temperature attribute is Nominal.
- The number of Missing values is zero.
- There are three distinct values with no unique value.
- The table underneath this information shows the nominal values for this field ashot, mild and cold.
- It also shows the count and weight in terms of a percentage for each nominalvalue.

3. Removing Attributes

- Many a time, the data that you want to use for model building comes with many irrelevantfields. For example, the customer database may contain his mobile number which is relevant in analysing his credit rating.

4. Applying Filters

- Some of the machine learning techniques such as association rule mining requires categoricaldata. To illustrate the use of filters, we will use weather-numeric.arff database that contains two numeric attributes - temperature and humidity.
- We will convert these to nominal by applying a filter on our raw data. Click on the Choosebutton in the Filter subwindow and select the following filter −
- weka→filters→supervised→attribute→Discretize
- select the best attributes for deciding the play. Select and apply the following filter
- weka→filters→supervised→attribute→AttributeSelection

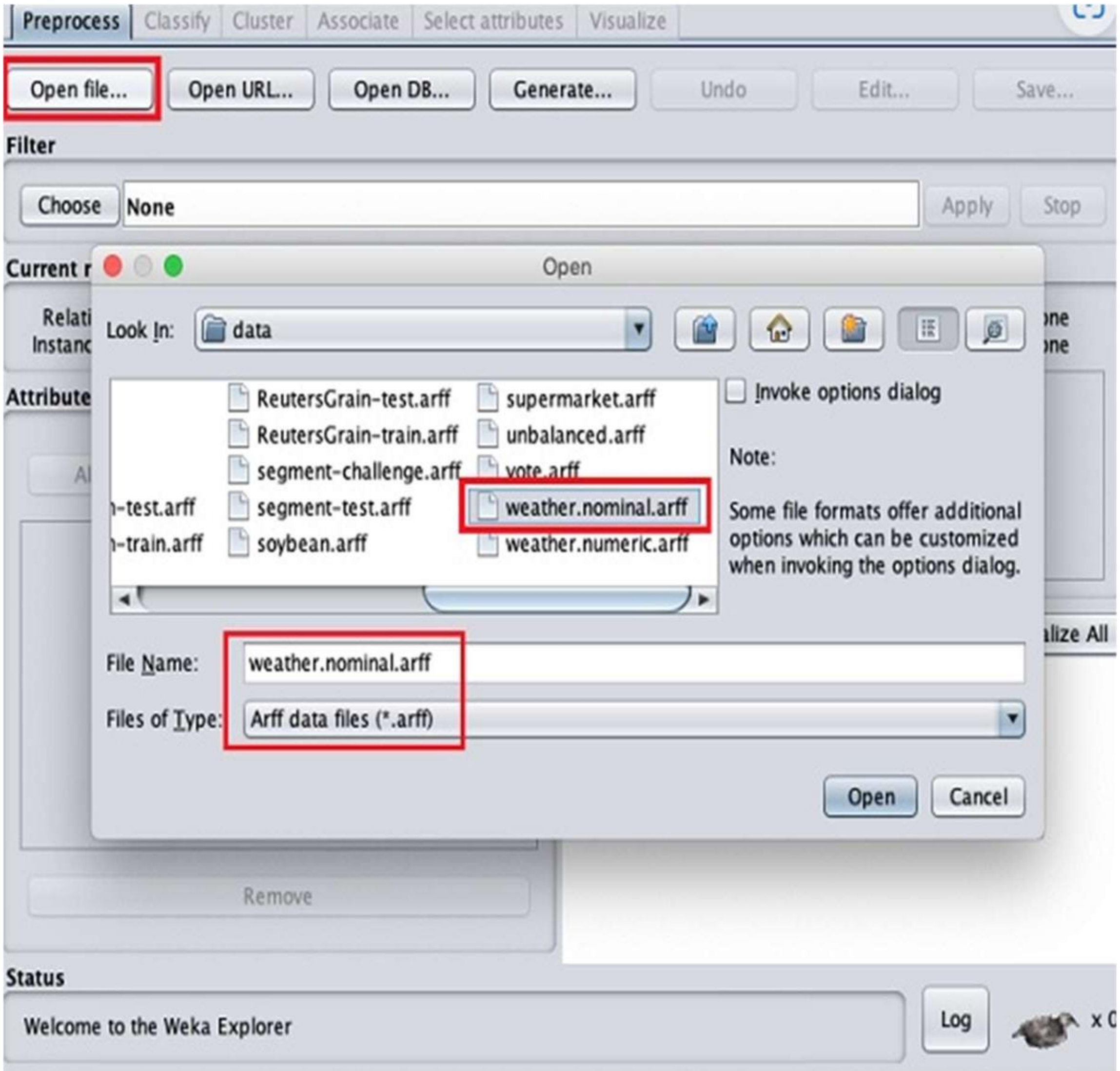
5. Data preprocessing for handling missing value

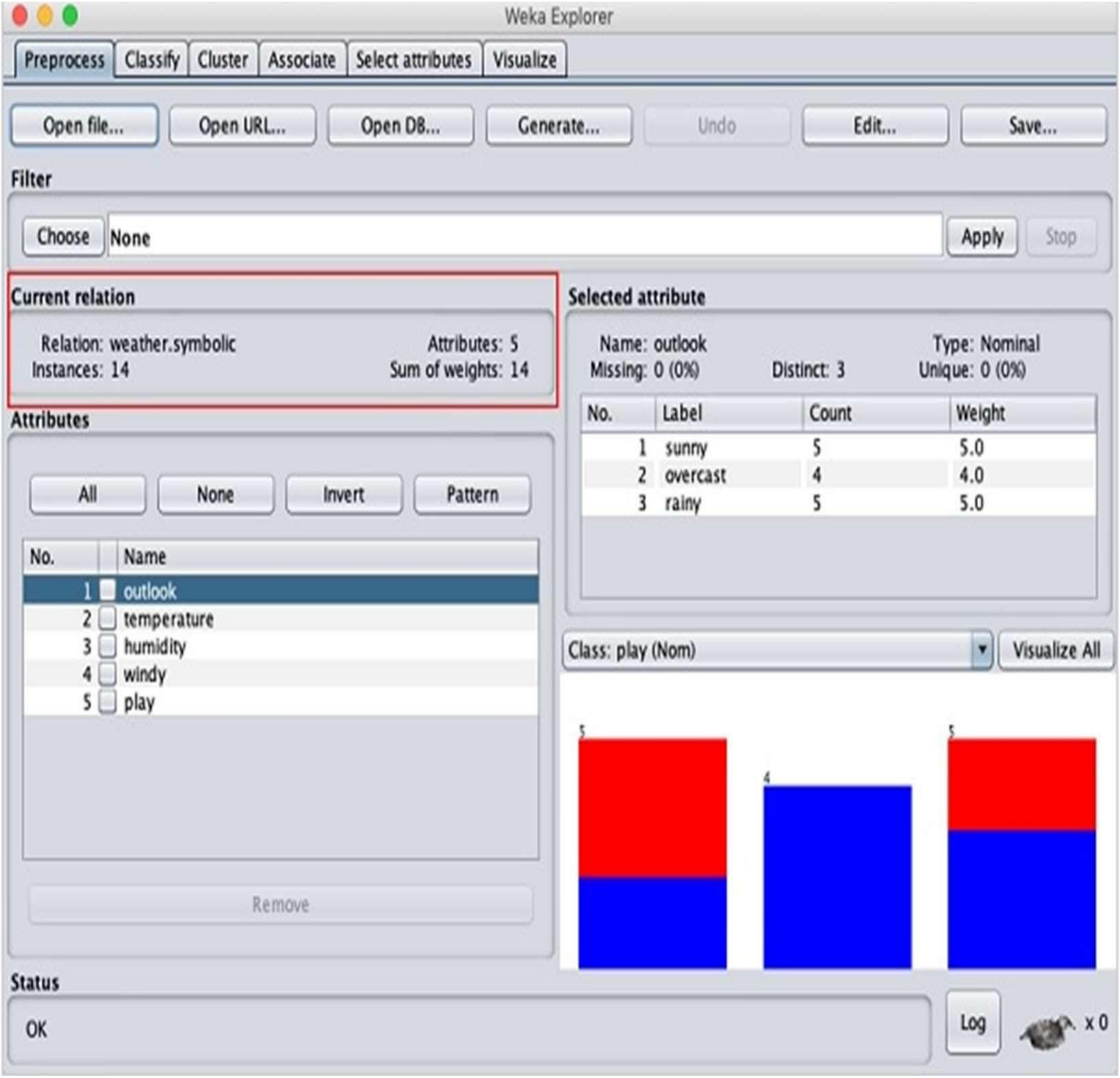
6. Predict the Onset of Diabetes

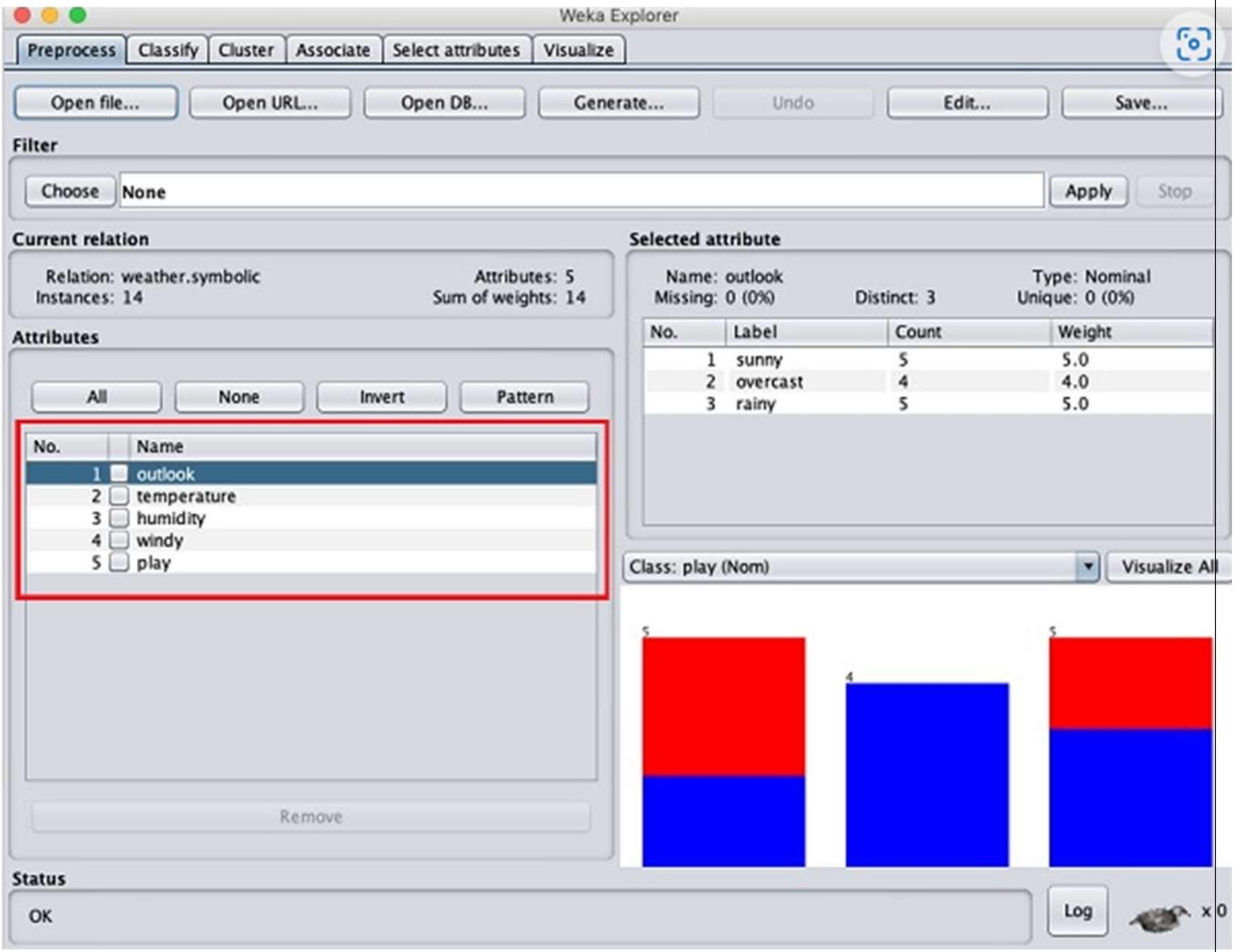
The problem used for this example is the Pima Indians onset of diabetes dataset.

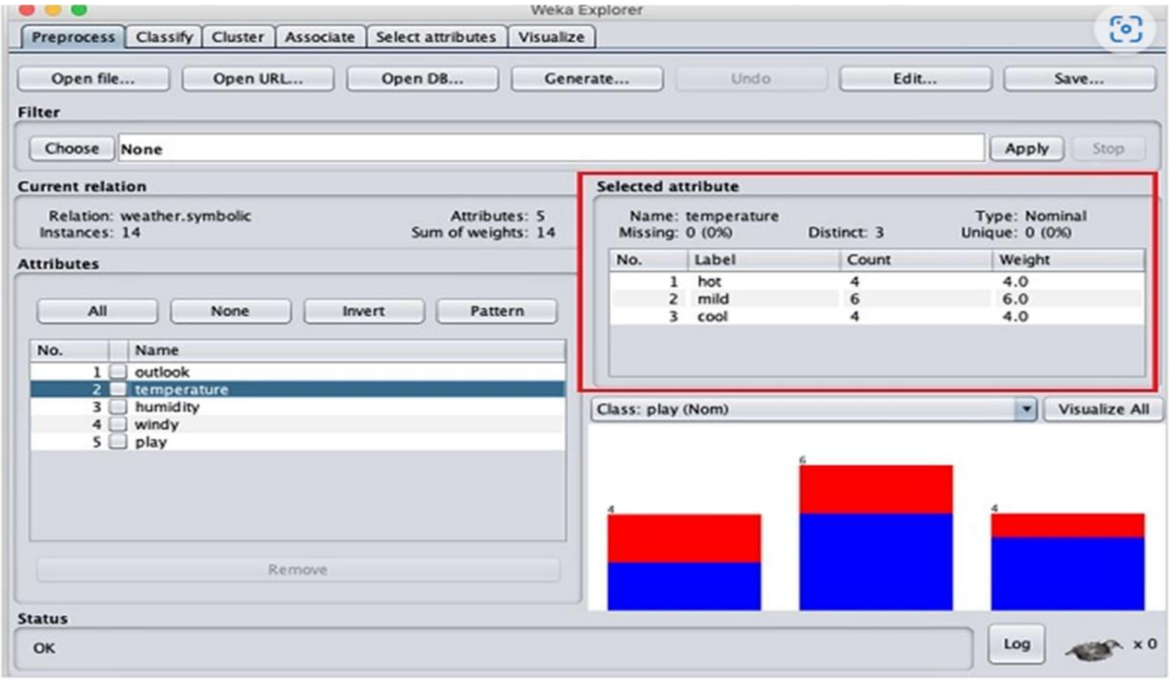
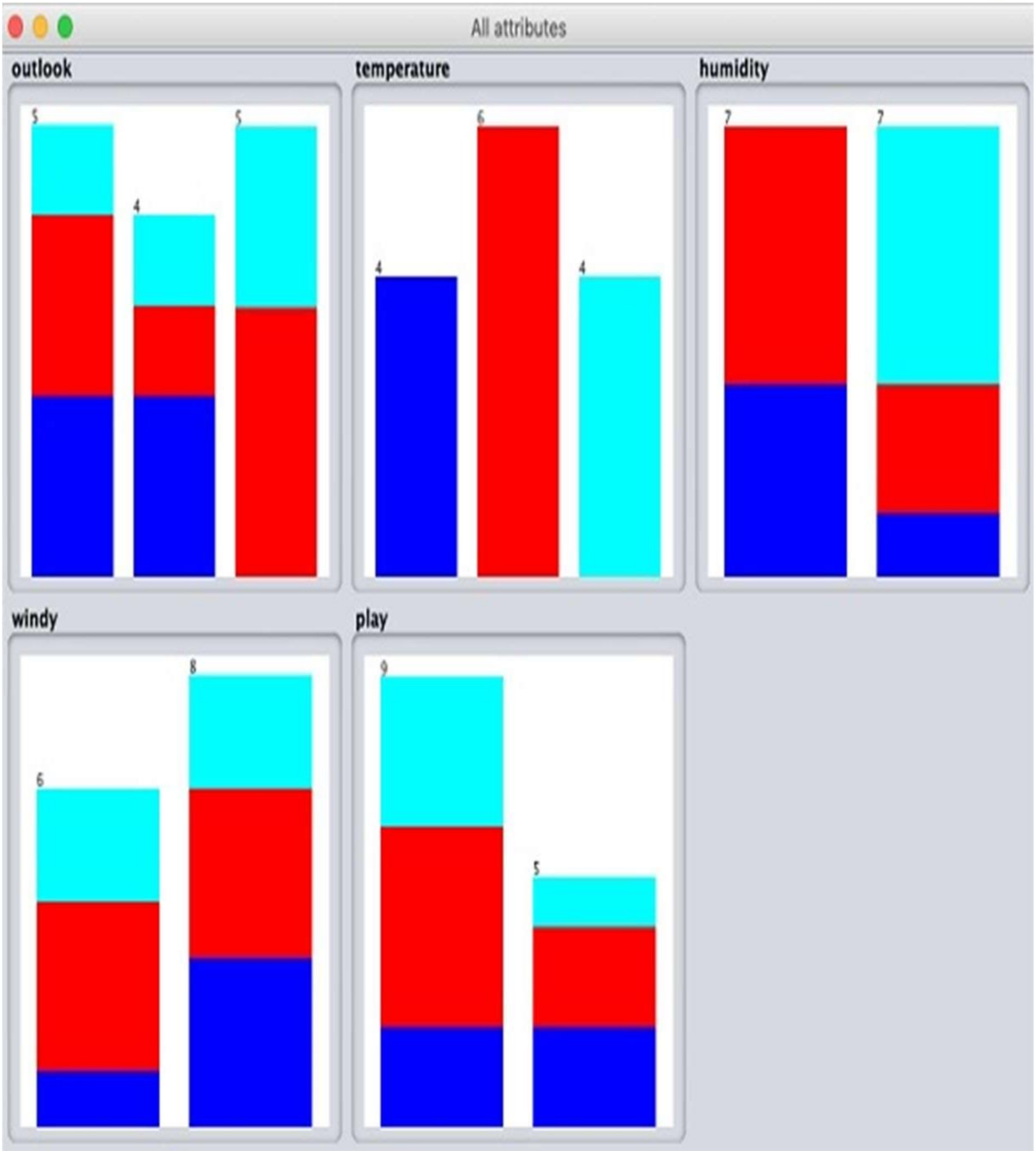
It is a classification problem where each instance represents medical details for one patient and thetask is to predict whether the patient will have an onset of diabetes within the next five years.

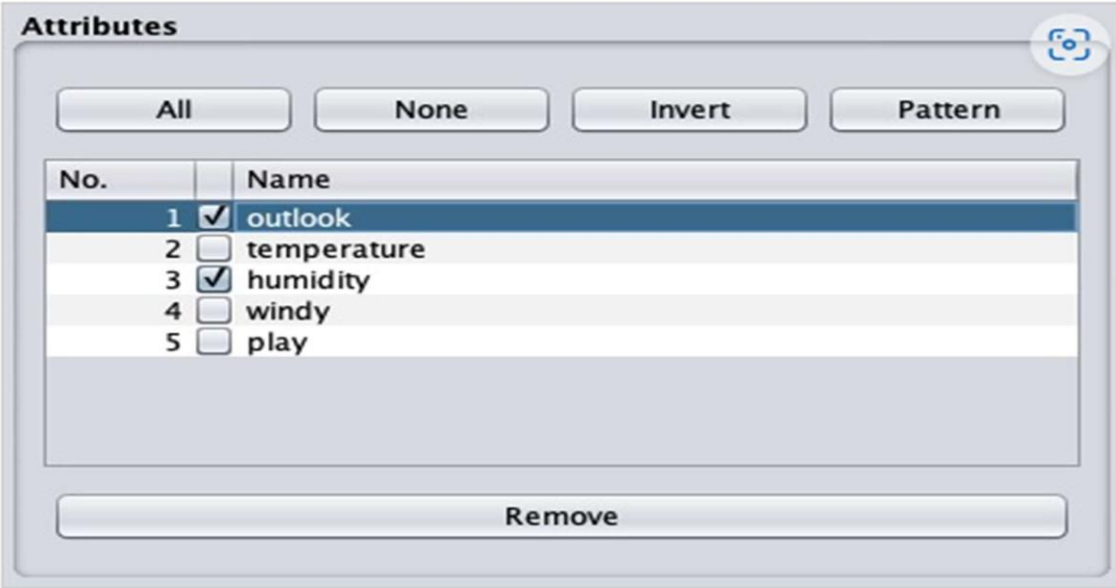
OUTPUT:


Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Open
Look In:
data
ReutersGrain-test.arff
ReutersGrain-train.arff
segment-challenge.arff
segment-test.arff
soybean.arff
supermarket.arff
unbalanced.arff
vote.arff
weather.nominal.arff
weather.numeric.arff
Invoke options dialog
Note:
Some file formats offer additional options which can be customized when invoking the options dialog.
File Name:
weather.nominal.arff
Files of Type:
Arff data files (*.arff)
Open
Cancel
Remove
Status
Welcome to the Weka Explorer
Log


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: weather.symbolic
Instances: 14
Attributes: 5
Sum of weights: 14
Attributes
All
None
Invert
Pattern
No. Name
1 outlook
2 temperature
3 humidity
4 windy
5 play
Remove
Selected attribute
Name: outlook
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 sunny 5 5.0
2 overcast 4 4.0
3 rainy 5 5.0
Class: play (Nom)
Visualize All
5
4
5
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: weather.symbolic
Instances: 14
Attributes: 5
Sum of weights: 14
Attributes
All
None
Invert
Pattern
No.
Name
1 outlook
2 temperature
3 humidity
4 windy
5 play
Remove
Selected attribute
Name: outlook
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 sunny 5 5.0
2 overcast 4 4.0
3 rainy 5 5.0
Class: play (Nom)
Visualize All
Status
OK
Log


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
None
Apply
Stop
Current relation
Relation: weather.symbolic
Instances: 14
Attributes: 5
Sum of weights: 14
Attributes
All
None
Invert
Pattern
No. Name
1 outlook
2 temperature
3 humidity
4 windy
5 play
Remove
Selected attribute
Name: temperature
Missing: 0 (0%)
Distinct: 3
Type: Nominal
Unique: 0 (0%)
No. Label Count Weight
1 hot 4 4.0
2 mild 6 6.0
3 cool 4 4.0
Class: play (Nom)
Visualize All
Status
OK
Log
x 0

All attributes
outlook
temperature
humidity
windy
play
5
4
5
4
6
4
7
7
6
8
9
5



RESULT:

Data preprocessing by selecting or filtering attributes, Data preprocessing for handling missing valueoperations were performed using Weka tool.

EXP NO:10

DATE:

MINI PROJECT

EXP NO:11

DATE:

IMAGE RECOGNITION USING DEEP LEARNING

AIM:

To implement an image recognition system using a Convolutional Neural Network (CNN).

ALGORITHMS:

Step 1. Load and Pre-process Data:
- Load the dataset containing labeled images of cats and dogs.
- Rescale the pixel values of images to a range between 0 and 1 for efficient training.

Step 2. Resize Images:
- Resize all images to a uniform size (e.g., 128x128 or 224x224 pixels) to ensure consistency across the dataset.

Step 3. Data Augmentation:
- Apply data augmentation techniques such as rotation, flipping, and zooming to increase dataset diversity and prevent overfitting.

Step 4. Build the CNN Model:
- Design a CNN architecture with convolutional layers, pooling layers, and fully connected layers to extract and classify features from the images.

Step 5. Convolutional Layers:
- Use multiple convolutional layers to detect various features in the images, such as edges and textures, by applying filters.

Step 6. Pooling Layers:
- Apply max pooling to downsample the feature maps, reducing dimensionality while retaining important information from the images.

Step 7. Activation Functions:
- Use the ReLU activation function in the hidden layers to introduce non-linearity and speed up convergence.
- Use the Softmax activation function in the output layer to generate class probabilities for cats and dogs.

Step 8. Compile the Model:
- Choose the **Adam Optimizer** for efficient weight updates and **binary cross-entropy loss** as the loss function for the binary classification problem.

Step 9. Train the Model:
- Train the CNN model on the training set while applying dropout to reduce overfitting by randomly dropping neurons during each iteration.

Step 10. Evaluate the Model:

- Use accuracy to measure the overall performance of the model.
- Analyze the confusion matrix to gain insights into correct and incorrect predictions for the two classes (cat or dog).

Importing Libraries

```
import zipfile
from PIL import Image
import matplotlib.pyplot as plt
import io
import numpy as np
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import seaborn as sns
```

Load Dataset

```
zip_train_path = '/kaggle/input/dogs-vs-cats/train.zip'
zip_test_path = '/kaggle/input/dogs-vs-cats/test1.zip'

# Data Visualization
def visualize(input_zip, num_images=10):
    with zipfile.ZipFile(input_zip, 'r') as archive_zip:
        archives = archive_zip.namelist()

        images = [archive for archive in archives if archive.endswith(('.png', '.jpg', '.jpeg'))]

        for i, img_path in enumerate(images[:num_images]):
            with archive_zip.open(img_path) as image_zip:
                img = Image.open(io.BytesIO(image_zip.read()))
                plt.subplot(1, num_images, i + 1)
                plt.imshow(img)
                plt.axis('off')

        plt.show()
# train data
visualize(zip_train_path, num_images=10)
```

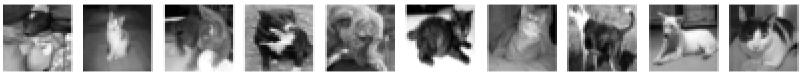
```
# test data
visualize(zip_test_path, num_images=10)
```

Rescaling

```
new_size = (128, 128)
def resize(input_zip, num_images=10, size=new_size):
    with zipfile.ZipFile(input_zip, 'r') as archive_zip:
        archives = archive_zip.namelist()
        images = [archive for archive in archives if archive.endswith(('.png', '.jpg', '.jpeg'))]

        for i, img_path in enumerate(images[:num_images]):
            with archive_zip.open(img_path) as image_zip:
                img = Image.open(io.BytesIO(image_zip.read()))
```

```
                img = img.convert('L')
                img = img.resize(size)

                plt.subplot(1, num_images, i + 1)
                plt.imshow(img, cmap='gray')
                plt.axis('off')

        plt.show()
resize(zip_train_path, num_images=10, size=(128, 128))
```

```
resize(zip_test_path, num_images=10, size=(128, 128))
```



```
# Count dataset size
def count(input_zip):
    with zipfile.ZipFile(input_zip, 'r') as archive_zip:
        archives = archive_zip.namelist()

        images = [archive for archive in archives if archive.endswith(('.png', '.jpg', '.jpeg'))]

        num_images = len(images)
        print(f"Images in {input_zip}: {num_images}")
# Preprocess images by converting to grayscale and resizing


new_size = (128, 128)

def preprocess(input_zip, size=new_size):
    images_processed = []

    with zipfile.ZipFile(input_zip, 'r') as archive_zip:
        archives = archive_zip.namelist()

        images = [archive for archive in archives if archive.endswith(('.png', '.jpg', '.jpeg'))]

        for img_path in images:
            with archive_zip.open(img_path) as image_zip:
                img = Image.open(io.BytesIO(image_zip.read()))

                img = img.convert('L')

                img = img.resize(size)

                img_array = np.array(img) / 255.0

                images_processed.append(img_array)

    dataset = np.array(images_processed)

    return dataset
```

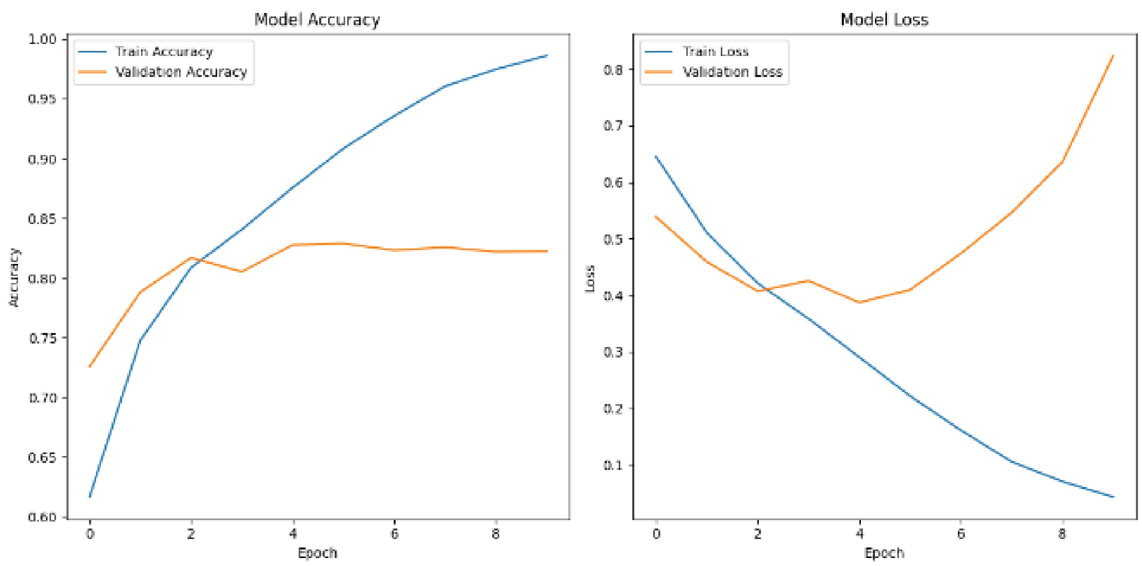
```python
train_dataset = preprocess(zip_train_path, new_size)
test_dataset = preprocess(zip_test_path, new_size)
train_dataset
# extract class tags
def extract_tags(input_zip):
    results = []
    with zipfile.ZipFile(input_zip, 'r') as archive_zip:
        archives = archive_zip.namelist()

        images = [archive for archive in archives if archive.endswith(('.png', '.jpg', '.jpeg'))]

        for img_path in images:
            if 'dog' in img_path.lower():
                results.append(('dog', img_path))
            elif 'cat' in img_path.lower():
                results.append(('cat', img_path))
            else:
                results.append(('unknown', img_path))  # Si no es identificable

    return results
train_labels = extract_tags(zip_train_path)
for label, img_path in train_labels[:10]:
    print(f"Image: {img_path}, Label: {label}")
```

```
Image: train/cat.0.jpg, Label: cat
Image: train/cat.1.jpg, Label: cat
Image: train/cat.10.jpg, Label: cat
Image: train/cat.100.jpg, Label: cat
Image: train/cat.1000.jpg, Label: cat
Image: train/cat.10000.jpg, Label: cat
Image: train/cat.10001.jpg, Label: cat
Image: train/cat.10002.jpg, Label: cat
Image: train/cat.10003.jpg, Label: cat
Image: train/cat.10004.jpg, Label: cat
```

Building Model

```python
train_binary = np.array([1 if label == 'dog' else 0 for label, _ in train_labels])

train_dataset = np.expand_dims(train_dataset, axis=-1)

# Train split
X_train, X_temp, y_train, y_temp = train_test_split(train_dataset, train_binary, test_size=0.2, random_state=42)

# Validation-Test Split
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5, random_state=42)

# Create Model
model = Sequential()

# Layer 1: Convolution+Max Pooling
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Layer 2: Convolution+Max Pooling
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
# Layer 3: Convolution+Max Pooling
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

# Ftatten Layer
model.add(Flatten())

# Dense Layer
model.add(Dense(128, activation='relu'))

# Output Layer for binary classification
model.add(Dense(1, activation='sigmoid'))
# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# Train the model
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))
# Plot training & validation accuracy values
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(loc='upper left')

# Plot training & validation loss values
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(loc='upper left')

plt.tight_layout()
plt.show()
```

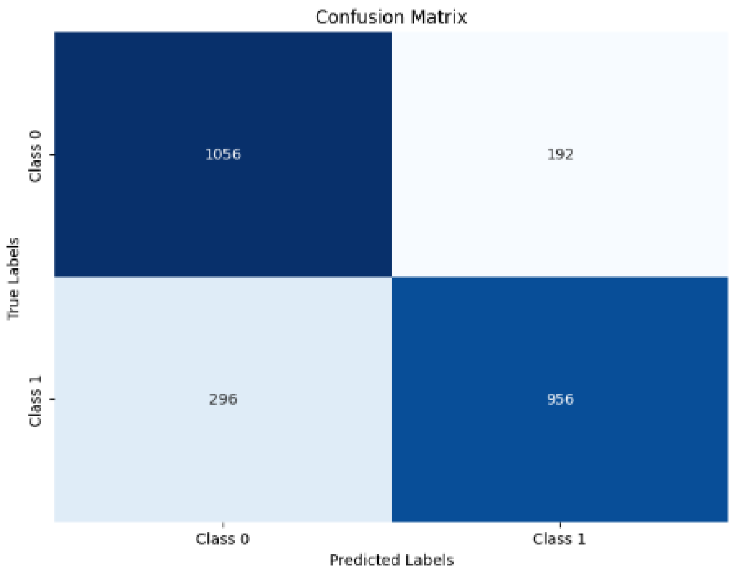


Model Evaluation

```
loss, accuracy = model.evaluate(X_val, y_val)
print(f'Accuracy: {accuracy:.4f}')
print(f'Loss: {loss:.4f}')

Accuracy: 0.8224
Loss: 0.8236
```

Evaluate on Test Set

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f'Accuracy: {accuracy:.4f}')
print(f'Loss: {loss:.4f}')
Accuracy: 0.8048
Loss: 0.8984

# Confusion Matrix
y_pred = model.predict(X_test)
y_pred = (y_pred > 0.5).astype(int)
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,
        xticklabels=['Class 0', 'Class 1'], yticklabels=['Class 0', 'Class 1'])
plt.xlabel('Predicted Labels')
plt.ylabel('True Labels')
plt.title('Confusion Matrix')
plt.show()
```




```
# few random predictions
num_samples = 5  # Number of samples to display
random_indices = np.random.choice(len(X_test), num_samples, replace=False)
plt.figure(figsize=(15, 5))
for i, idx in enumerate(random_indices):
    plt.subplot(1, num_samples, i + 1)
    plt.imshow(X_test[idx].reshape(128, 128), cmap='gray')  # Reshape for display and use grayscale
    true_label = 'dog' if y_test[idx] == 1 else 'cat'  # True label
    pred_label = 'dog' if y_pred[idx][0] == 1 else 'cat'  # Predicted label
```

```
    plt.title(f'True: {true_label}\nPred: {pred_label}')
    plt.axis('off')
plt.tight_layout()
plt.show()
```

OUTPUT:

RESULT:

Thus the Image recognition using Deep learning (CNN) algorithm is successfully implemented.

Virtual Lab

Back Propagation

INTRODUCTION

The Backpropagation neural network is a multilayered, feedforward neural network and is by far the most extensively used. It is also considered one of the simplest and most general methods used for supervised training of multilayered neural networks. Backpropagation works by approximating the non-linear relationship between the input and the output by adjusting the weight values internally. It can further be generalized for the input that is not included in the training patterns (predictive abilities).

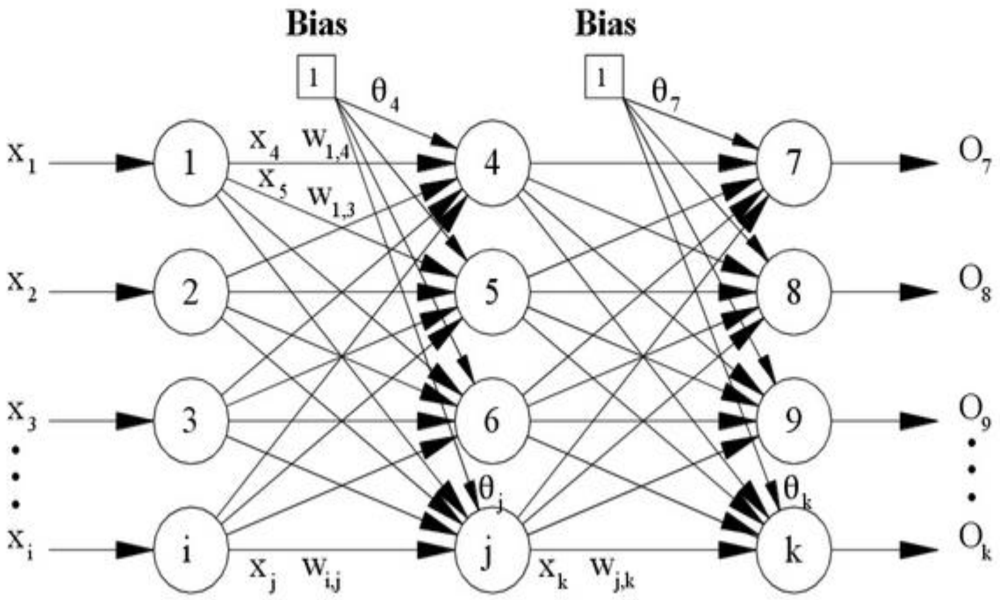
Generally, the Backpropagation network has two stages, training and testing. During the training phase, the network is "shown" sample inputs and the correct classifications.

For example, the input might be an encoded picture of a face, and the output could be represented by a code that corresponds to the name of the person.

A further note on encoding information - a neural network, as most learning algorithms, needs to have the inputs and outputs encoded according to an arbitrary user defined scheme.

The scheme will define the network architecture so that once a network is trained, the scheme cannot be changed without creating a totally new net. Similarly there are many forms of encoding the network response.

The following figure shows the topology of the Backpropagation neural network that includes and input layer, one hidden layer and an output layer. It should be noted that Backpropagation neural networks can have more than one hidden layer.



PROCEDURE

Step 0. Set w and alpha (learning to random rate) values.

Step 1. Perform steps 2 to 9 when stopping condition is false.

Step 2. Perform steps 3 to 8 for each training pair.

Step 3. Receive ip signal from xi and transfer to zj

Step 4. In Hidden Unit calculate the net ip and op:

$$net_{h1} = w_1 * i_1 + w_2 * i_2 + b_1 * 1$$

$$out_{h1} = \frac{1}{1+e^{-net_{h1}}}$$

The function used above is the bipolar sigmoid ie:

$$g(net) = \frac{1-e^{-\lambda \cdot net}}{1+e^{-\lambda \cdot net}}$$

Its binary counterpart is: $y = \dfrac{1}{1+e^{-x}}$

Step 5. Similarly, calculate the net ip and op for each op unit.

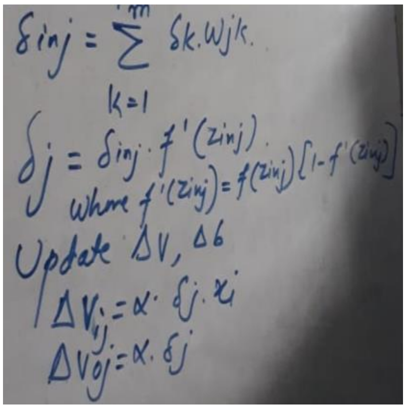
Step 6. Calculate error correction factor for o/p unit.

$$E_{total} = \sum \frac{1}{2}(target - output)^2$$

$$E_{total} = E_{o1} + E_{o2}$$

and send this to hidden layer

Step 7. Each hidden unit "j", sums its delta inputs from op units.



Step 8. Update the weight by weight(new) = weight(old) + offset

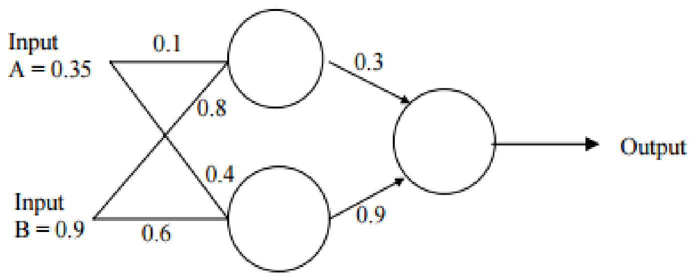
Step 9. Check for stopping condition

*train certain no. of epochs

*where actual op equals to target op

ILLUSTRATION

Consider the simple network below:



Assume that the neurons have a Sigmoid activation function and

Answer:

(i) Perform a forward pass on the network.

(ii) Perform a reverse pass (training) once (target = 0.5).

(iii) Perform a further forward pass

and comment on the result.

Input to top neuron = (0.35x0.1)+(0.9x0.8)=0.755. Out = 0.68.

Input to bottom neuron (0.9x0.6)+(0.35x0.4)=0.68. Out = 0.6637. Input to final neuron = (0.3x0.68)+(0.9x0.6637)=0.80133. Out = 0.69.

(ii)

Output error 8=(t-o) (1-0)o = (0.5-0.69) (1-0.69)0.69=-0.0406.

New weights for output layer

w1 = w1+(8 x input) = 0.3+ (-0.0406x0.68)=0.272392. w2*=w2+(8 x input) = 0.9+(-0.0406x0.6637) = 0.87305.

Errors for hidden layers:

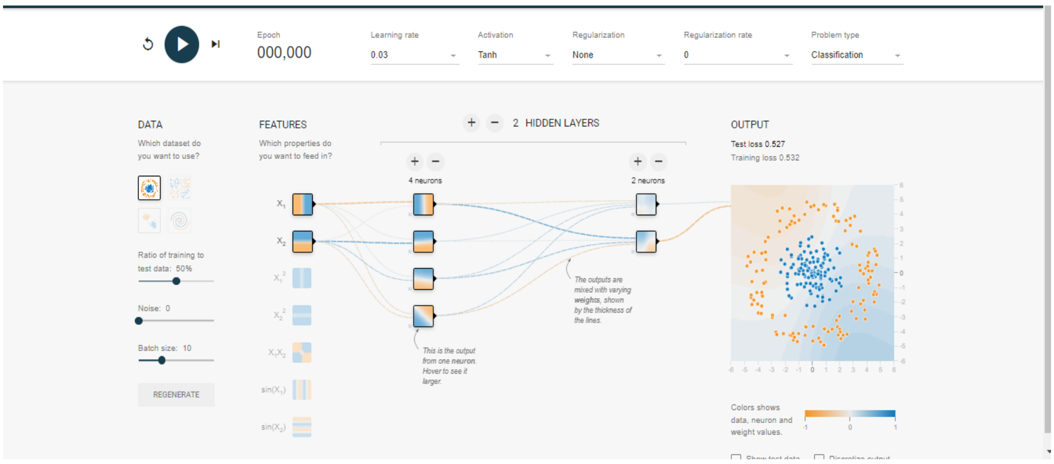
81 = 8 xw1 = -0.0406 x 0.272392 x (1-0)o = -2.406x10-3 82=8 x w2 = -0.0406 x 0.87305 x (1-0)o=-7.916x103

New hidden layer weights:

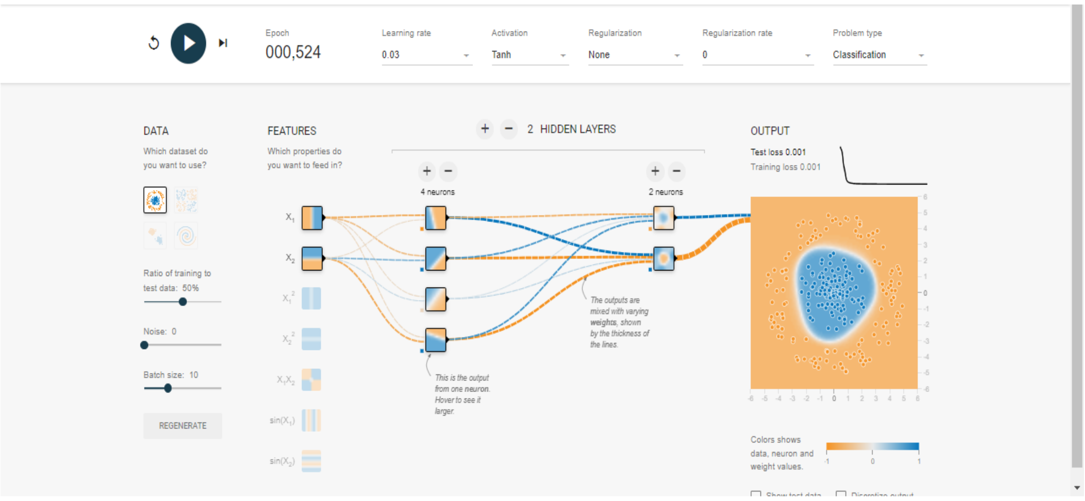
w3=0.1+(-2.406 x 103 x 0.35)=0.09916. w4=0.8+(-2.406 x 103 x 0.9)=0.7978. w5 0.4+(-7.916 x 103 x 0.35) = 0.3972. w6=0.6+(-7.916 x 103 x 0.9)=0.5928.

(iii)

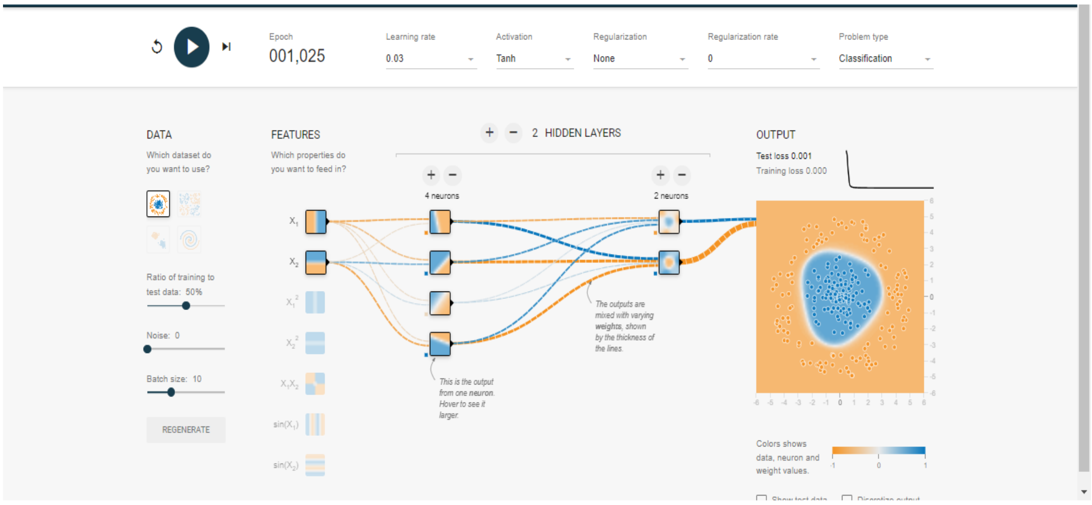
Old error was -0.19. New error is -0.18205. Therefore error has reduced.



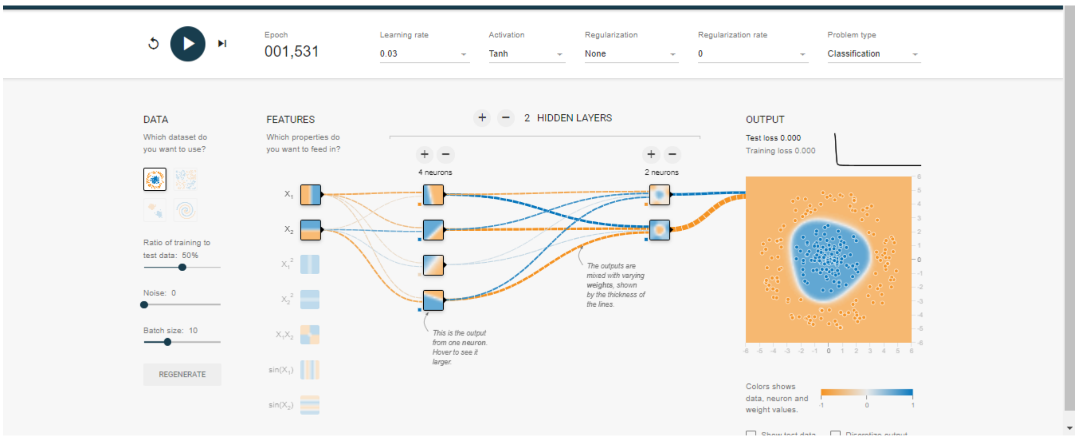
Simulation 1



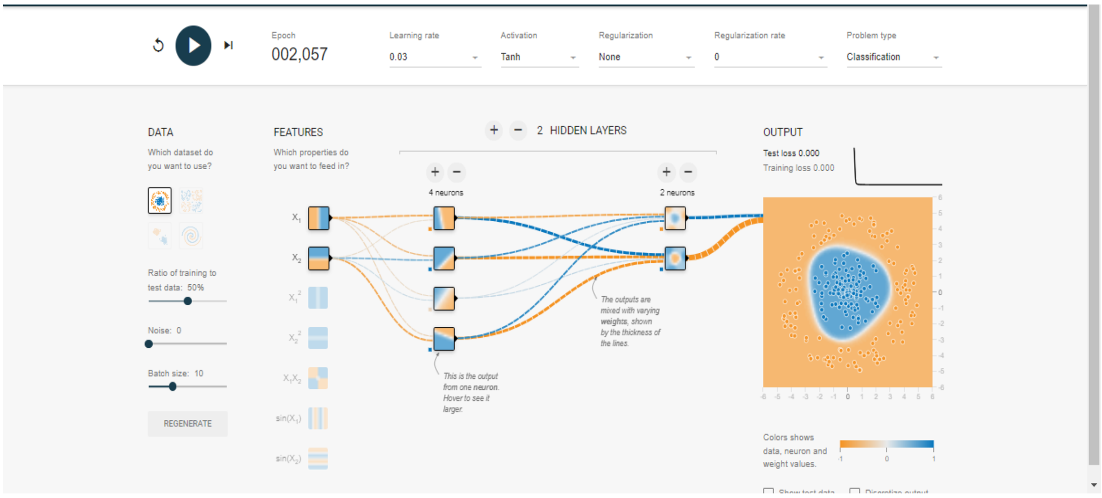
Simulation 2



Simulation 3



Simulation 4



Simulation 5

RESULT:

Thus the Virtual lab for Back Propagation is successfully implemented.

GitHub Commands

cd <file path> command

First create a file where your code will be stored on your pc. For that, you have to create a file on your desktop. After that open Git Bash and type cd <File directory> to go to file and branch.

git clone command

If you want to open-source contribution. First, you have to copy an existing repository (the repository, where you want to contribute) on your local repository (Your repository). For that, you have to click the fork button on the repo of the existing repository on GitHub.

- **What is forking:** Forking any repository means make a copy of a real repository in your GitHub account and make changes in your copy. Thus, a real repository won't get affected by your code changes. (After that you have to make a pull request to the real repository for merging your code change, we will come to that part later)
- **How to do fork*:*** Just go to the real repo and tap on the fork button
- **Copy URL:** Then a copy of real repository will be created in your local repository. After that, you have to copy the URL from your local repo. For doing that click to code and copy the URL. After that, you have to create a file on your desktop. Then open Git Bash and go to the file using cd command and click enter and type git clone <copied url> to copy the code in your desktop file. With that, you are able to get the code on your desktop.

git status command

After making code changes, add the files for that you have to check which files are not added. For that use git status. The command git status can show you the status of your current file whether it is added or committed or pushed.

git add <File name> command

When you get to know which files are not added by typing git status(red-colored files are not added). Then type git add <file name> to add files.

git commit -m <message> or git commit -am<message> command

After that, commit those added files (type git status to check status, and green colored files are not yet committed). Type git commit -m <message> (message is nothing but a text that tells about what is changed in files) (there are many types of commit command you can check out git documentation in git official website).

git push command

At last, push your code changes in your local repo by typing git push and then make a pull request.

Getting & Creating Projects

Command	Description
git init	Initialize a local Git repository
git clone ssh://git@github.com/[username]/[repository-name].git	Create a local copy of a remote repository

Basic Snapshotting

Command	Description
git status	Check status
git add [file-name.txt]	Add a file to the staging area
git add –A	Add all new and changed files to the staging area
git commit -m "[commit message]"	Commit changes
git rm -r [file-name.txt]	Remove a file (or folder)
git remote -v	View the remote repository of the currently working file or directory

Branching & Merging

Command	Description
git branch	List branches (the asterisk denotes the current branch)
git branch -a	List all branches (local and remote)
git branch [branch name]	Create a new branch
git branch -d [branch name]	Delete a branch
git push origin --delete [branch name]	Delete a remote branch
git checkout -b [branch name]	Create a new branch and switch to it
git checkout -b [branch name] origin/[branch name]	Clone a remote branch and switch to it
git branch -m [old branch name] [new	Rename a local branch

Command	Description
branch name]	
git checkout [branch name]	Switch to a branch
git checkout -	Switch to the branch last checked out
git checkout -- [file-name.txt]	Discard changes to a file
git merge [branch name]	Merge a branch into the active branch
git merge [source branch] [target branch]	Merge a branch into a target branch
git stash	Stash changes in a dirty working directory
git stash clear	Remove all stashed entries
git stash pop	Apply latest stash to working directory

Sharing & Updating Projects

Command	Description
git push origin [branch name]	Push a branch to your remote repository
git push -u origin [branch name]	Push changes to remote repository (and remember the branch)
git push	Push changes to remote repository (remembered branch)
git push origin --delete [branch name]	Delete a remote branch
git pull	Update local repository to the newest commit
git pull origin [branch name]	Pull changes from remote repository
git remote add origin ssh://git@github.com/[username]/[repository-name].git	Add a remote repository
git remote set-url origin ssh://git@github.com/[username]/[repository-name].git	Set a repository's origin branch to SSH

Inspection & Comparison

Command	Description
git log	View changes
git log --summary	View changes (detailed)
git log --oneline	View changes (briefly)
git diff [source branch] [target branch]	Preview changes before merging