SCTR's Pune Institute of Computer Technology
(PICT) Pune

B.E. Machine Learning and Data Science (Honors)

SUBMITTED BY
Name: Tanaya Prasad Peshave
ABC Id: 931-253-282-048
PRN no: 72017327C
Class: BE-08
Roll no: 42465

Under the guidance of
Prof. H. B. Mali

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING
ACADEMIC YEAR 2022-23

DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

SCTR's Pune Institute of Computer Technology (PICT), Pune
Maharashtra 411043

CERTIFICATE

This is certified that ML&DS laboratory experiments submitted by Ms. Tanaya Prasad Peshave has satisfactorily completed the curriculum-based B.E. Machine learning and Data Science honors experiments under the guidance of Prof. H. B. Mali towards the partial fulfillment of final year Electronics and Telecommunication Engineering Semester VII (Honors Degree), Academic Year 2022-23 of Savitribai Phule Pune University.

Prof. H. B. Mali

Principal

LAB MANUAL

Academic Year: 2022-23

Class: B.E. **Date: 09/11/2022**

Subject: Machine Learning and Data Science (Honours)

Semester: I

Lab Expt. No	Problem Statement.
1.	Creating & Visualizing Neural Network for the given data. (Use python) Note: download dataset using Kaggal. Keras, ANN visualizer, graph viz libraries are equired.
2.	Recognize optical character using ANN
3.	Implement basic logic gates using Hebbnet neural networks
4.	Exploratory analysis on Twitter text data Perform text pre-processing, Apply Zips and heaps law, Identify topics
5.	Text classification for Sentimental analysis using KNN Note: Use twitter data
6.	Write a program to recognize a document is positive or negative based on polarity words using suitable classification method.

Machine Learning Data Science Laboratory (410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.1

Title: Creating and visualizing neural networks for the given data.

Objectives:

1. To handle given data for creating and visualizing neural network.
2. To analyze data using a python programming language.

Software Requirement:

Windows /Linux

Theory:

Neural network was inspired by the design and functioning of human brain and components

Definition:

Information processing model that is inspired by the way biological nervous system (i.e the brain) process information, is called Neural Network. Neural Network has the ability to learn by examples. It is not designed to perform fix /specific task, rather task which need thinking (e.g. Predictions). ANN is composed of large number of highly interconnected processing elements(neurons)working in unison to solve problems. It mimic human brain. With advanced in deep learning, you can now visualize the entire deep learning process or just the Convolutional Neural Network you've built.We are going to build simple neural network using Keras and then use ANN visualizer to visualize our neural network

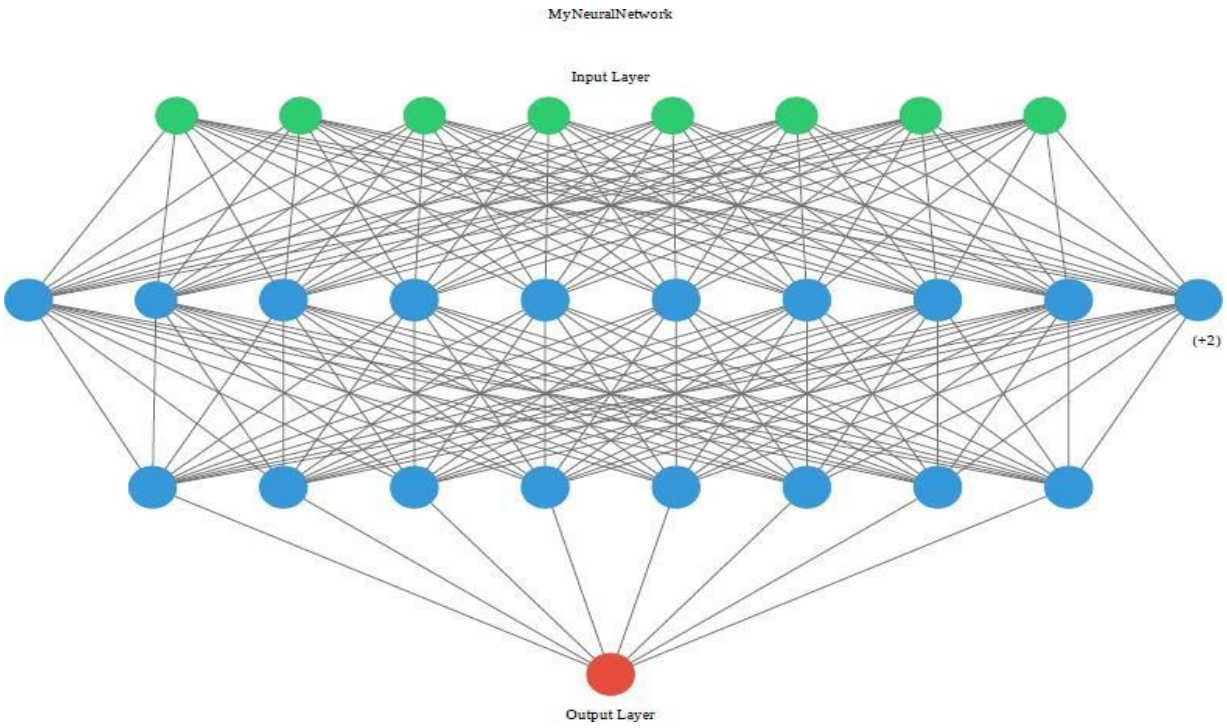
ANN Visualizer:

It is a python library that enables us to visualize an Artificial Neural Network using just a single line of code. It is used to work with Keras and makes use of python's graphviz library to create a neat and presentable graph of the neural network you're building

Visualize a Neural Network in Python using Graphviz

- Import module.
- Create a new object of Diagraph.
- Add **node ()** and **edge ()** into graph object.
- Save the source code with render () object.

Output :



Conclusion:

Here, we studied creating and visualizing neural network for the given data using
python

Machine Learning And Data Science Laboratory(410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.1

NAME : TANAYA PESHAVE

ROLL NO : 42465

PRN NO : 72017327C

ABC ID : 931-253-282-048

BRANCH : E&TC

COLLEGE : PICT

```
from keras.models import Sequential
from keras.layers import Dense
import numpy


# fix random seed for reproducibility
numpy.random.seed(7)
# load pima indians dataset
dataset = numpy.loadtxt(r"pima-indians-diabetes.csv", delimiter=",")


X = dataset[:,0:8]
Y = dataset[:,8]
# create model
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# Compile model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Fit the model
model.fit(X, Y, epochs=150, batch_size=10)
# evaluate the model
scores = model.evaluate(X, Y)
```

```
77/77 [==============================] - 0s 2ms/step - loss: 0.5229 - accuracy: 0.747
Epoch 123/150
```

```
77/77 [==============================] - 0s 2ms/step - loss: 0.5211 - accuracy: 0.751
Epoch 124/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5306 - accuracy: 0.747
Epoch 125/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4938 - accuracy: 0.759
Epoch 126/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5137 - accuracy: 0.724
Epoch 127/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4904 - accuracy: 0.768
Epoch 128/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5135 - accuracy: 0.764
Epoch 129/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4982 - accuracy: 0.761
Epoch 130/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5358 - accuracy: 0.740
Epoch 131/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5011 - accuracy: 0.765
Epoch 132/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5722 - accuracy: 0.701
Epoch 133/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5046 - accuracy: 0.764
Epoch 134/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5222 - accuracy: 0.742
Epoch 135/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5015 - accuracy: 0.750
Epoch 136/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5048 - accuracy: 0.756
Epoch 137/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5031 - accuracy: 0.748
Epoch 138/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4906 - accuracy: 0.764
Epoch 139/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5226 - accuracy: 0.738
Epoch 140/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4904 - accuracy: 0.770
Epoch 141/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4987 - accuracy: 0.743
Epoch 142/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4829 - accuracy: 0.769
Epoch 143/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5321 - accuracy: 0.739
Epoch 144/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4859 - accuracy: 0.761
Epoch 145/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5027 - accuracy: 0.752
Epoch 146/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4887 - accuracy: 0.768
Epoch 147/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4885 - accuracy: 0.781
Epoch 148/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4928 - accuracy: 0.759
Epoch 149/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5027 - accuracy: 0.744
Epoch 150/150
77/77 [==============================] - 0s 2ms/step - loss: 0.4836 - accuracy: 0.755
24/24 [==============================] - 0s 2ms/step - loss: 0.4558 - accuracy: 0.786
```

```
print("\n%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

```
   accuracy: 78.65%
```

```
!pip install -q ann_visualizer
```

```
from ann_visualizer.visualize import ann_viz;
ann_viz(model, title="My graph")
```

Machine Learning Data Science Laboratory (410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.2

Title: Recognize Optical Character using ANN

Objective:

To recognize the optical characters using ANN

Theory:

What is Optical Character Recognition?

Optical character recognition or **optical character reader** (**OCR**) is the electronic or mechanical conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example: from a television broadcast).[1]

Widely used as a form of data entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs.[2] Some systems are capable of

reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

What is keras_ocr?

keras-ocr provides out-of-the-box OCR models and an end-to-end training pipeline to build new OCR models. Using this we get pre trained data and weights so our time and effort is saved.

Conclusion:

Here, we studied how to recognize optical characters using ANN.

Machine Learning And Data Science Laboratory(410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.2

NAME : TANAYA PESHAVE

ROLL NO : 42465

PRN NO : 72017327C

ABC ID : 931-253-282-048

BRANCH : E&TC

COLLEGE : PICT

```
from google.colab import drive
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from keras.utils import np_utils
```

```
df = pd.read_csv("/content/A_Z Handwritten Data.csv")
df.head()
```

	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.639	0.640	0.641	0.642	0.64
0	0	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0	0.
1	0	0	0	0	0	0	0	0	0	0		0.0	0.0	0.0	0.0	0

```
df.rename(columns={'0':'target'}, inplace=True)
df.head()
```

	target	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	...	0.639	0.640	0.641	0.642
0	0	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0
1	0	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0
2	0	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0
3	0	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0
4	0	0	0	0	0	0	0	0	0	0	...	0.0	0.0	0.0	0.0

5 rows × 785 columns

```
df['target'].value_counts()
```

```
0    3352
Name: target, dtype: int64
```

```
X = df.drop('target',axis = 1)
y = df['target']

print(X.shape)
print(y.shape)
```

```
(3352, 784)
(3352,)
```

Split into Training and Testing

```
x_train,x_test,y_train,y_test = train_test_split(X,y, test_size=0.2)

standard_scaler = MinMaxScaler()
standard_scaler.fit(x_train)

# scaling data
x_train = standard_scaler.transform(x_train)
x_test = standard_scaler.transform(x_test)

# One hot encoding targets
```

```
y_train = np_utils.to_categorical(y_train)

print(x_train.shape)
print(x_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(2681, 784)
(671, 784)
(2681, 1)
(671, 1)
```

```
#no. of features per data sample
x_train.shape[1]
```

```
784
```

```
#total targets
len(y.unique())
```

```
1
```

Create ANN Model

```
# creating ANN model

model = keras.Sequential()
model.add(layers.Dense(500, activation="relu" , input_dim = x_train.shape[1]))
model.add(layers.Dense(400, activation="relu"))
model.add(layers.Dense(300, activation="relu"))
model.add(layers.Dense(200, activation="relu"))
model.add(layers.Dense(100, activation="relu"))
model.add(layers.Dense(len(y.unique()), activation="softmax"))

model.compile(loss='categorical_crossentropy', optimizer="adam" , metrics = ['acc'])

model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 500)               392500

 dense_1 (Dense)             (None, 400)               200400

 dense_2 (Dense)             (None, 300)               120300
```

```
 dense_3 (Dense)             (None, 200)               60200

 dense_4 (Dense)             (None, 100)               20100

 dense_5 (Dense)             (None, 1)                 101

=================================================================
Total params: 793,601
Trainable params: 793,601
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(x_train, y_train, epochs=10, validation_data = (x_test, y_test))
```

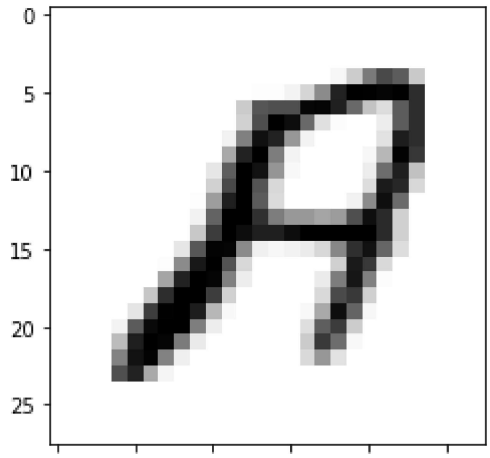
```
Epoch 1/10
84/84 [==============================] - 2s 16ms/step - loss: nan - acc: 0.6203 - val_lo
Epoch 2/10
84/84 [==============================] - 1s 12ms/step - loss: nan - acc: 0.0000e+00 - va
Epoch 3/10
84/84 [==============================] - 1s 12ms/step - loss: nan - acc: 0.0000e+00 - va
Epoch 4/10
84/84 [==============================] - 1s 12ms/step - loss: nan - acc: 0.0000e+00 - va
Epoch 5/10
84/84 [==============================] - 1s 12ms/step - loss: nan - acc: 0.0000e+00 - va
Epoch 6/10
84/84 [==============================] - 1s 13ms/step - loss: nan - acc: 0.0000e+00 - va
Epoch 7/10
84/84 [==============================] - 1s 13ms/step - loss: nan - acc: 0.0000e+00 - va
Epoch 8/10
84/84 [==============================] - 1s 13ms/step - loss: nan - acc: 0.0000e+00 - va
Epoch 9/10
84/84 [==============================] - 1s 12ms/step - loss: nan - acc: 0.0000e+00 - va
Epoch 10/10
84/84 [==============================] - 1s 12ms/step - loss: nan - acc: 0.0000e+00 - va
<keras.callbacks.History at 0x7f981b59bd50>
```

Make predictions

```
model.evaluate(x_test,y_test)
```

```
21/21 [==============================] - 0s 5ms/step - loss: nan - acc: 0.0000e+00
[nan, 0.0]
```

```
# original target
plt.imshow(x_test[150].reshape(28,28), cmap='Greys')
plt.show()
```




```
# predicted target
test_pred = model.predict(x_test[150].reshape(1,784))
chr(np.argmax(test_pred)+65)
```

```
1/1 [==============================] - 0s 111ms/step
'A'
```

Machine Learning Data Science Laboratory (410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.3

Aim: Implement basic logic gates using Mc-Culoch-Pitts or Hebbnet neural networks.

Objectives:

1.The student will be able to obtain the fundamentals and different architecture of neural networks.

2. The student will have a broad knowledge in developing the different algorithms for neural networks.

Software Requirements:

Ubuntu 18.04 /windows

Hardware Requirements:

Pentium IV system with latest configuration

Outcomes: The students will be able to,

- Describe the relation between real brains and simple artificial neural network models.
- Understand the role of neural networks in engineering.
- Apply the knowledge of computing and engineering concept to this discipline.

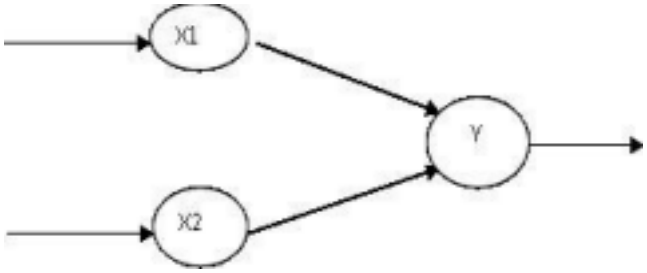
Theory:

Neural network was inspired by the design and functioning of human brain and components. Definition: Information processing model that is inspired by the way biological nervous system (i.e the brain) process information, is called Neural Network.

Neural Network has the ability to learn by examples. It is not designed to perform fix /specific task, rather task which need thinking (e.g. Predictions).

ANN is composed of large number of highly interconnected processing elements(neurons) working in unison to solve problems. It mimic human brain. It is configured for special application such as pattern recognition and data classification through a learning process. ANN is 85-90% accurate.

Basic Operation of a Neural Network:



X1 and X2 – input neurons.

Y- output neuron

Weighted interconnection links- W1 and W2.

Net input calculation is :

Yin= x1w1+x2w2

Output is :

y=f(Yin)

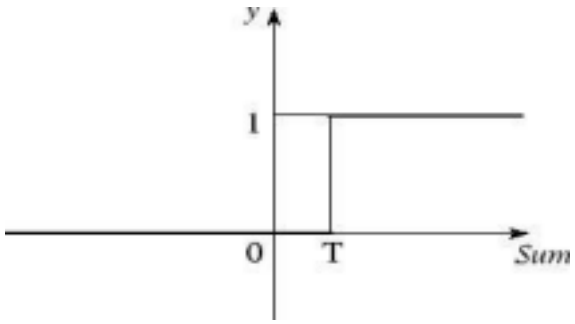
Output= function

The McCulloch-Pitts Model of Neuron:

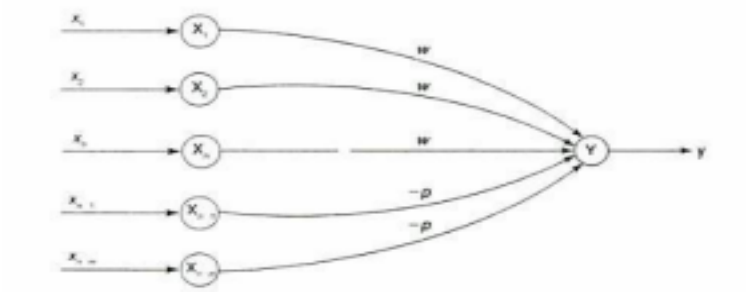
The early model of an artificial neuron is introduced by Warren McCulloch and Walter Pitts in 1943. The McCulloch-Pitts neural model is also known as linear threshold gate. It is a neuron of a set of inputs I1,I2,I3…Im and one output y . The linear threshold gate simply classifies the set of

inputs into two different classes. Thus the output y is binary. Such a function can be described mathematically using these equations:

$$y = f(Sum). \quad Sum = \sum_{i=1}^{N} I_i W_i,$$



W1,W2…Wm are weight values normalized in the range of either (0,1) or (-1,1) and associated with each input line, Sum is the weighted sum, and T is a threshold constant. The function f is a linear step function at threshold T as shown in figure



A simple M-P neuron is shown in the figure.

It is excitatory with weight (w>0) / inhibitory with weight –p (p<0).

In the Fig., inputs from x1 to xn possess excitatory weighted connection and Xn+1 to xn+m has inhibitory weighted interconnections.

Since the firing of neuron is based on threshold, activation function is defined as

$$f(y_{in}) = \begin{cases} 1 & if\ y_{in} \geq \theta \\ 0 & if\ y_{in} < \theta \end{cases}$$

For inhibition to be absolute, the threshold with the activation function should satisfy the following condition:θ >nw –p

Output will fire if it receives ―k‖ or more excitatory inputs but no inhibitory inputs where kw≥θ>(k-1) w

- The M-P neuron has no particular training algorithm.

- An analysis is performed to determine the weights and the threshold. - It is used as a building block where any function or phenomenon is modelled based on a logic function.

Activation function Yin is as follows: $Yin=x_1w_1+x_2w_2$

Input:

Input1	Input2	Output
0	0	0
0	1	0
1	0	0
1	1	1

Conclusion: Mc-Culloch pits Model is implemented for XOR function by using the thresholding activation function. Activation of M-P neurons is binary (i.e) at any time step the neuron may fire or may not fire. Threshold plays major role here.

Machine Learning And Data Science Laboratory(410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.3

NAME : TANAYA PESHAVE

ROLL NO : 42465

PRN NO : 72017327C

ABC ID : 931-253-282-048

BRANCH : E&TC

COLLEGE : PICT

```
import numpy as np
from random import randint
#inputs

X=np.array([[1,1,1],[1,-1,1],[-1,1,1],[-1,-1,1]])
#output
Y_ad=np.array([[1],[-1],[-1],[-1]])
Y_o=np.array([[1],[1],[1],[-1]])



print('input is:')
print(X)
print('output for And Gate is: ')
print(Y_ad)
print('output for Or Gate is: ')
print(Y_ad)
weights_ad=np.zeros((3))
weights_o=np.zeros((3))
print(weights_ad)


# update weight for and gate /logic
def update_weight_ad(X,Y,weights):
    for i in range(4):
```

```
        weights=weights+X[i]*Y[i]

        #print weights
        slope =-(weights[0]/weights[1])
        c=-(weights[2]/weights[0])
        if slope<0 and weights[0]>0:
            weights_main=weights

    return weights_main

def update_weight_o(X,Y,weights):
    for i in range(4):

        weights=weights+X[i]*Y[i]

        #print weights
        slope =-(weights[0]/weights[1])
        c=-(weights[2]/weights[0])
        if slope<0 and weights[0]>0:
            weights_main=weights

    return weights_main



weights_ad=update_weight_ad(X,Y_ad,weights_ad)
weights_o=update_weight_o(X,Y_o,weights_o)


print('Checking after learning selectg a input')
rand_int=int(input('Enter the test case no you want to try'))
print('Select a logic you also want  to check')
logicgate=input()
print(weights_ad)
print('selected input is %d '%rand_int)
print(X[rand_int])
def check_learning(X,weights,rand_int):
    Yin=0
    for i in range(3):
        Yin+=X[rand_int,i]*weights[i]
    if Yin<0:
        Yin=-1
    else:
        Yin=1
    return Yin


if logicgate=='a' or logicgate=='A':
    weights_in=weights_ad
else:
    weights_in=weights_o
```

```
Yin=check_learning(X,weights_in,rand_int)
print(Yin)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:31: RuntimeWarning: invalic
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:43: RuntimeWarning: divide
input is:
[[ 1  1  1]
 [ 1 -1  1]
 [-1  1  1]
 [-1 -1  1]]
output for And Gate is:
[[ 1]
 [-1]
 [-1]
 [-1]]
output for Or Gate is:
[[ 1]
 [-1]
 [-1]
 [-1]]
[0. 0. 0.]
Checking after learning selectg a input
Enter the test case no you want to try2
Select a logic you also want  to check
A
[ 2.  2. -2.]
selected input is 2
[-1  1  1]
-1
```

Colab paid products - Cancel contracts here

Machine Learning Data Science Laboratory (410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.4

Aim:

Exploratory analysis on Twitter Text Data. Perform text preprocessing. Apply Zipf's and Heaps law , Identify topics.

Objectives:

1. The student will be able to perform text preprocessing
2. The student will learn the concept of Zipf's law.

Software Requirements: Windows with python and Jupyter notebook.

Outcomes:

The students will be able to perform text preprocessing.And will learn the Zipf's law concept.

Theory:

Data Preprocessing :

Data preprocessing is a technique which is used to transform the raw data in a useful and efficient format.

Steps Involved in Data Preprocessing:

1. Data Cleaning:

The data can have many irrelevant and missing parts. To handle this part, data

cleaning is done. It involves handling of missing data, noisy data etc.

- **(a). Missing Data:**
  This situation arises when some data is missing in the data. It can be handled in various ways.
  Some of them are:
    1. **Ignore the tuples:**
       This approach is suitable only when the dataset we have is quite large and multiple values are missing within a tuple.

    2. **Fill the Missing values:**
       There are various ways to do this task. You can choose to fill the missing values manually, by attribute mean or the most probable value.

- **(b). Noisy Data:**
  Noisy data is meaningless data that can't be interpreted by machines.It can be generated due to faulty data collection, data entry errors etc. It can be handled in following ways :
    1. **Binning Method:**
       This method works on sorted data in order to smooth it. The whole data is divided into segments of equal size and then various methods are performed to complete the task. Each segment is handled separately. One can replace all data in a segment by its mean or boundary values can be used to complete the task.

    2. **Regression:**
       Here data can be made smooth by fitting it to a regression function.The regression used may be linear (having one independent variable) or multiple (having multiple independent variables).

    3. **Clustering:**

This approach groups the similar data in a cluster. The outliers may be undetected or it will fall outside the clusters.

2. Data Transformation:

This step is taken in order to transform the data in appropriate forms suitable for the mining process. This involves following ways:

1. **Normalization:**
   It is done in order to scale the data values in a specified range (-1.0 to 1.0 or 0.0 to 1.0)

2. **Attribute Selection:**
   In this strategy, new attributes are constructed from the given set of attributes to help the mining process.

3. **Discretization:**
   This is done to replace the raw values of numeric attributes by interval levels or conceptual levels.

4. **Concept Hierarchy Generation:**
   Here attributes are converted from lower level to higher level in hierarchy. For Example-The attribute “city” can be converted to “country”.

3. Data Reduction:

Since data mining is a technique that is used to handle huge amounts of data. While working with a huge volume of data, analysis became harder in such cases. In order to get rid of this, we use data reduction techniques. It aims to increase the storage efficiency and reduce data storage and analysis costs.

The various steps to data reduction are:

1. **Data Cube Aggregation:**
   Aggregation operation is applied to data for the construction of the data cube.

2. **Attribute Subset Selection:**
   The highly relevant attributes should be used, rest all can be discarded. For performing attribute selection, one can use the level of significance and p- value of the attribute.The attribute having p-value greater than significance level can be discarded.

3. **Numerosity Reduction:**
   This enables us to store the model of data instead of whole data, for example: Regression Models.

4. **Dimensionality Reduction:**
   This reduces the size of data by encoding mechanisms.It can be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction is called lossless reduction, else it is called lossy reduction. The two effective methods of dimensionality reduction are:Wavelet transforms and PCA (Principal Component Analysis).

Zipf's Law:-

Zipf's law is a law about the frequency distribution of words in a language (or in a collection that is large enough so that it is representative of the language). To illustrate Zipf's law let us suppose we have a collection and let there be V unique words in the collection (the vocabulary). For each word in the collection we need to compute the freq(word) = how many times a word occurs in the collection. Then we

rank the words descending by their frequency (most frequent words have rank 1, next frequent word has rank 2, ...).**Zipf's law**, in probability, assertion that the frequencies *f* of certain events are inversely proportional to their rank *r*. The law was originally proposed by American linguist George Kingsley Zipf (1902–50) for the frequency of usage of different words in the English language; this frequency is given approximately by $f(r) \cong 0.1/r$. Thus, the most common word (rank 1) in English, which is *the*, occurs about one-tenth of the time in a typical text; the next most common word (rank 2), which is *of*, occurs about one-twentieth of the time; and so forth. Another way of looking at this is that a rank *r* word occurs $1/r$ times as often as the most frequent word, so the rank 2 word occurs half as often as the rank 1 word, the rank 3 word one-third as often, the rank 4 word one-fourth as often, and so forth. Beyond about rank 1,000, the law completely breaks down.

Zipf's law purportedly has been observed for many other statistics that follow an exponential distribution. For example, in 1949 Zipf claimed that the largest city in a country is about twice the size of the next largest, three times the size of the third largest, and so forth. While the fit is not perfect for languages, populations, or any other data, the basic idea of Zipf's law is useful in schemes for data compression and in allocation of resources by urban planners.

Heaps' law:-

Heaps' law (also called **Herdan's law**) is an empirical law which describes the number of distinct words in a document (or set of documents) as a function of the document length

Conclusion: Learnt text preprocessing technique and Zipf's law.

Machine Learning And Data Science Laboratory(410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.4

NAME : TANAYA PESHAVE

ROLL NO : 42465

PRN NO : 72017327C

ABC ID : 931-253-282-048

BRANCH : E&TC

COLLEGE : PICT

```
import csv
import nltk
import numpy as np
import re
import random
from nltk.corpus import wordnet
from nltk.tokenize import TweetTokenizer as tt
import math
tokens=[]
d=dict()
with open(r'/content/tweets-dataset.csv','r',encoding="utf8") as csvfile:
    reader1=csv.reader(csvfile)
    c=0
    for i in reader1:
        token=tt().tokenize(i[0])
        actual=[]
        for i in token:
            k=re.findall(r"[A-Za-z']+",i)
            if len(k)==1 and len(i)==len(k[0]):
                if 'http' in k[0]:
                    k=[k[0].replace('http','')]
                if 'https' in k[0]:
                    k=[k[0].replace('https','')]
                k=[k[0].lower()]
                if not ((len(k[0])==1 and (k[0]=="'" )) or len(k[0])==0):
```

```
                    actual+=k
                if k[0] in d:
                    d[k[0]]+=1
                else:
                    d[k[0]]=1
        tokens+=actual
types=set(tokens)
n_tokens=len(tokens)
n_types=len(types)
print('Number of tokens : '+str(n_tokens))
print('Number of types : '+str(n_types))
print('TTR : '+str(n_types/n_tokens))
```

```
Number of tokens : 283330
Number of types : 30260
TTR : 0.10680125648537042
```

Zipf's Law of Length

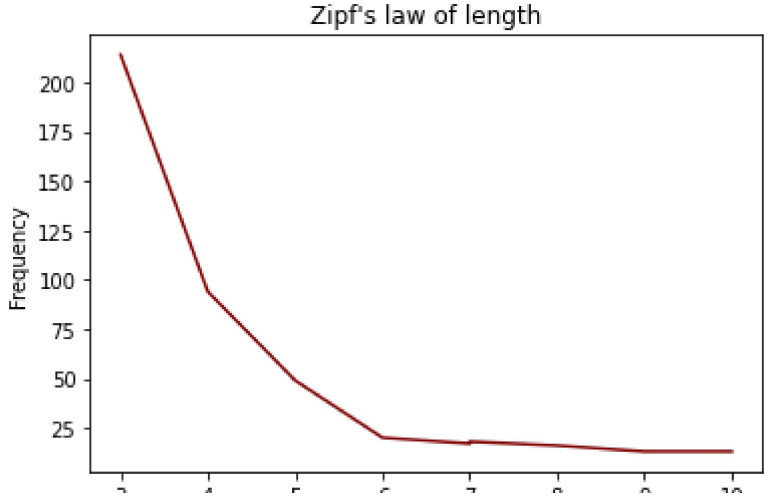
```
import matplotlib.pyplot as plt

%matplotlib inline
frequency_types=dict()
for i in tokens:
    if i in frequency_types:
        frequency_types[i]+=1
    else:
        frequency_types[i]=1
#wordlength_frequency=dict()
words_chosen=['show','complete','welcome','famous','holiday','champions','one','girls','par

axes=[]
for i in words_chosen:
    axes.append([len(i),frequency_types[i],i])
axes.sort()
x=[]
y=[]
for i in axes:
    x.append(i[0])
    y.append(i[1])

plt.plot(x,y,color='maroon')
plt.title("Zipf's law of length")
plt.xlabel('length')
plt.ylabel('Frequency')
plt.show()
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
```



According to this law $frequency * wordlength = constant$, In the plot below the graph traces hyperbola $xy = c$, from which we can say that Zipf's law of length holds good.

Zip's Law of Meanings

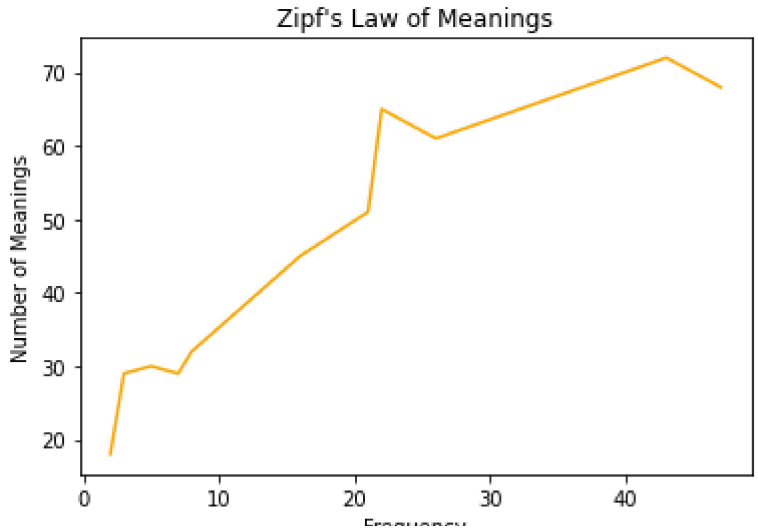
```
import nltk

nltk.download('all')

nltk.download('wordnet')
l1=[]
freq=[]
num_meanings=[]
words=['clubs','crackers','tip','matches','cases','starts','cover','deal','marks','getting',
meanings=[]
tokennumber=0
for i in words:
    c=0
    mean_i=[]
    for syn in wordnet.synsets(i):
        for j in syn.lemmas():
            mean_i.append(j.name())
    meanings.append(mean_i)
    num_meanings.append(len(set(mean_i)))
    freq.append(frequency_types[i])

plt.plot(freq,num_meanings,color='orange')
plt.title("Zipf's Law of Meanings")
plt.xlabel('Frequency')
plt.ylabel('Number of Meanings')
plt.show()
```

```
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/twitter_samples.zip.
[nltk_data]     | Downloading package udhr to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/udhr.zip.
[nltk_data]     | Downloading package udhr2 to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/udhr2.zip.
[nltk_data]     | Downloading package unicode_samples to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping corpora/unicode_samples.zip.
[nltk_data]     | Downloading package universal_tagset to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping taggers/universal_tagset.zip.
[nltk_data]     | Downloading package universal_treebanks_v20 to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     | Downloading package vader_lexicon to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     | Downloading package verbnet to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/verbnet.zip.
[nltk_data]     | Downloading package verbnet3 to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/verbnet3.zip.
[nltk_data]     | Downloading package webtext to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/webtext.zip.
[nltk_data]     | Downloading package wmt15_eval to /root/nltk_data...
[nltk_data]     |   Unzipping models/wmt15_eval.zip.
[nltk_data]     | Downloading package word2vec_sample to
[nltk_data]     |     /root/nltk_data...
[nltk_data]     |   Unzipping models/word2vec_sample.zip.
[nltk_data]     | Downloading package wordnet to /root/nltk_data...
[nltk_data]     |   Package wordnet is already up-to-date!
[nltk_data]     | Downloading package wordnet2021 to /root/nltk_data...
[nltk_data]     | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]     | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/wordnet_ic.zip.
[nltk_data]     | Downloading package words to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/words.zip.
[nltk_data]     | Downloading package ycoe to /root/nltk_data...
[nltk_data]     |   Unzipping corpora/ycoe.zip.
[nltk_data]     |
[nltk_data]   Done downloading collection all
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```



Frequency

Machine Learning Data Science Laboratory (410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.5

Title: Text classification for Sentiment analysis using KNN

Objectives:

1. To handle Twitter Data for performing computing.
2. To analyze data using R programming tools.

Theory:

Sentiment analysis refers to the use of natural language processing, text analysis, and computational linguistics to systematically identify, extract, quantify, and study effective states and subjective information. Sentiment analysis is widely applied to customer materials such as reviews and survey responses. The most common type of sentiment analysis is 'polarity detection' and involves classifying customer materials/reviews as positive, negative or neutral.

Text Processing

With the increasing importance of computational text analysis in research, many researchers face the challenge of learning how to use advanced software that enables this text analysis. Text processing has a direct application to Natural Language Processing, also known as NLP. NLP is aimed at processing the languages spoken or written by humans when they communicate with one another. This is different from the communication

between a computer and a human where the communication is either a computer program written by a human or some gesture by a human like clicking the mouse at some position. NLP tries to understand the natural language spoken by humans and classify it, analyze it as well if required to respond to it. Python has a rich set of libraries which cater to the needs of NLP. The Natural Language ToolKit (NLTK) is a suite of such libraries which provides the functionalities required for NLP..

Twitter Data

Twitter is an online microblogging tool that disseminates more than 400 million messages per day, including vast amounts of information about almost all industries from entertainment to sports, health to business etc. One of the best things about Twitter—indeed, perhaps its greatest appeal-is in its accessibility. It's easy to use both for sharing information and for collecting it.Twitter provides unprecedented access to our lawmakers and to our celebrities, as well as to news as it's happening. Twitter represents an important data source for the business models of huge companies as well. All the above characteristics make twitter a best place to collect real time and latest data to analyse and do any sought of research for real life situations.

DATASET DESCRIPTION

We are given a Twitter US Airline Sentiment dataset that contains around 14,601 tweets about each major U.S. airline. The tweets are labelled as positive, negative, or neutral based on the nature of the respective Twitter user's feedback regarding the airline. The dataset is further segregated into training and test sets in a stratified fashion. Train set contains 11,680 tweets whereas the test set contains 2,921 tweets.Our task is to develop and train a k-nearest neighbors classifier on the training set and use it to predict sentiment classes of the tweets present in the test set.

Pre-Processing

Raw tweets scraped from twitter generally result in a noisy dataset. This is due to the casual nature of people's usage of social media. Tweets have certain special characteristics such as retweets, emoticons, user mentions, etc. which

have to be suitably extracted. Therefore, raw twitter data has to be normalized to create a dataset which can be easily learned by various classifiers. We have applied an extensive number of pre-processing steps to standardize the dataset and reduce its size. We first do some general pre-processing on tweets which is as follows.

- Convert the tweet to lower case.
- Replace 2 or more dots (.) with space.
- Strip spaces and quotes (" and ') from the ends of tweet.
- Replace 2 or more spaces with a single space.

Special twitter features as follows.

URL:

Users often share hyperlinks to other webpages in their tweets. Any particular URL is not important for text classification as it would lead to very sparse features. Therefore, we replace all the URLs in tweets with the word URL. The regular expression used to match URLs is ((www\.[\S]+)|(https?://[\S]+)).

User Mention

Every twitter user has a handle associated with them. Users often mention other users in their tweets by @handle. It replaces all user mentions with the word USER_MENTION. The regular expression used to match user mention is @[\S]+.

K-Nearest Neighbours

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection.It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data).

KNN algorithm is used to classify by finding the K nearest matches in training data and then using the label of closest matches to predict. Traditionally,

distance such as euclidean is used to find the closest match.KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

Feature Extraction

In the feature extraction step, we will need to represent each tweet as a bag-of-words (BoW), i.e. an unordered set of words with their positions ignored and all of the emphasis placed on the respective frequencies of each word.

For example, consider these two tweets:

T1 = Welcome to machine learning, machine!

T2 = kNN is a powerful machine learning algorithm.

The bag-of-words representation (ignoring case and punctuation) for the above two tweets are:

Vocabulary	welcome	to	machine	learning	knn	is	a	powerful	algorithm
T1	1	1	2	1	0	0	0	0	0
T2	0	0	1	1	1	1	1	1	1

In order to create this bag-of-words representation, we would first need to extract out the unique words from all of our tweets in the training dataset.

Conclusion:

Hence, we studi ed On Twitter Data performs computing using Business Intelligence analytical tools electively.

Machine Learning And Data Science Laboratory(410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.5

NAME : TANAYA PESHAVE

ROLL NO : 42465

PRN NO : 72017327C

ABC ID : 931-253-282-048

BRANCH : E&TC

COLLEGE : PICT

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


train = pd.read_csv(r"/content/train.csv")
test = pd.read_csv(r"/content/test.csv")
```

```
train
```

	id	label	tweet
0	1	0	@user when a father is dysfunctional and is s...
1	2	0	@user @user thanks for #lyft credit i can't us...
2	3	0	bihday your majesty
3	4	0	#model i love u take with u all the time in ...

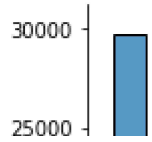
```
test
```

	id	tweet
0	31963	#studiolife #aislife #requires #passion #dedic...
1	31964	@user #white #supremacists want everyone to s...
2	31965	safe ways to heal your #acne!! #altwaystohe...
3	31966	is the hp and the cursed child book up for res...
4	31967	3rd #bihday to my amazing, hilarious #nephew...
...	...	...
17192	49155	thought factory: left-right polarisation! #tru...
17193	49156	feeling like a mermaid ð #hairflip #neverre...
17194	49157	#hillary #campaigned today in #ohio((omg)) &am...
17195	49158	happy, at work conference: right mindset leads...
17196	49159	my song "so glad" free download! #shoegaze ...

17197 rows × 2 columns

```
sns.displot(train['label'])
```

```
<seaborn.axisgrid.FacetGrid at 0x7fecc3ebf590>
```




```
label_cnt = train['label'].value_counts()
label_cnt
```

```
0    29720
1     2242
Name: label, dtype: int64
```

```
label_pct = train['label'].value_counts() / len(train)
label_pct
```

```
0    0.929854
1    0.070146
Name: label, dtype: float64
```

```
label = train['label']

train.drop(['label'], axis=1, inplace=True)
train
```

	id	tweet
0	1	@user when a father is dysfunctional and is s...
1	2	@user @user thanks for #lyft credit i can't us...
2	3	bihday your majesty
3	4	#model i love u take with u all the time in ...
4	5	factsguide: society now #motivation
...	...	...
31957	31958	ate @user isz that youuu?ð ð ð ð ð ð...
31958	31959	to see nina turner on the airwaves trying to...
31959	31960	listening to sad songs on a monday morning otw...
31960	31961	@user #sikh #temple vandalised in in #calgary,...
31961	31962	thank you @user for you follow

31962 rows × 2 columns

```
combi = train.append(test)
combi
```

	id	tweet
0	1	@user when a father is dysfunctional and is s...
1	2	@user @user thanks for #lyft credit i can't us...
2	3	bihday your majesty
3	4	#model i love u take with u all the time in ...
4	5	factsguide: society now #motivation
...	...	...
17192	49155	thought factory: left-right polarisation! #tru...
17193	49156	feeling like a mermaid ð #hairflip #neverre...
17194	49157	#hillary #campaigned today in #ohio((omg)) &am...
17195	49158	happy, at work conference: right mindset leads...
17196	49159	my song "so glad" free download! #shoegaze ...

49159 rows × 2 columns

```
tweets = combi['tweet']

count_words = tweets.str.findall(r'(\w+)').str.len()
print(count_words.sum())
```

```
681137
```

```
import re
import nltk
nltk.download('all')
from nltk.corpus import stopwords

tweets = tweets.str.lower()


tweets = tweets.apply(lambda x : re.sub("[^a-z\s]","",x) )


tweets = tweets.str.replace("#", " ")


tweets = tweets.apply(lambda x: ' '.join([w for w in x.split() if len(w)>2]))


stopwords = set(stopwords.words("english"))
tweets = tweets.apply(lambda x : " ".join(word for word in x.split() if word not in stopwords
```

```
count_words = tweets.str.findall(r'(\w+)').str.len()
print(count_words.sum())
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/alpino.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/basque_grammars.zip.
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/biocreative_ppi.zip.
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown_tei.zip.
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_cat.zip.
[nltk_data]    | Downloading package cess_esp to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_esp.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/comparative_sentences.zip.
[nltk_data]    | Downloading package comtrans to /root/nltk_data...
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package conll2007 to /root/nltk_data...
[nltk_data]    | Downloading package crubadan to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/crubadan.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
```

```
[nltk_data]    |   Unzipping corpora/dependency_treebank.zip.
[nltk_data]    | Downloading package dolch to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dolch.zip.
[nltk_data]    | Downloading package europarl_raw to
[nltk_data]    |     /root/nltk_data...
```

```
most_freq_words = pd.Series(' '.join(tweets).lower().split()).value_counts()[:25]
tweets = tweets.apply(lambda x : " ".join(word for word in x.split() if word not in most_freq
print(most_freq_words)

count_words = tweets.str.findall(r'(\w+)').str.len()
print(count_words.sum())
```

```
user        27008
love         4217
day          3471
happy        2630
amp          2433
time         1745
life         1719
today        1555
new          1546
like         1527
positive     1423
get          1406
thankful     1403
people       1331
bihday       1327
good         1313
cant         1239
one          1219
see          1136
fathers      1134
dont         1133
smile        1077
want          986
healthy       962
take          945
dtype: int64
328789
```

```
from collections import Counter
from itertools import chain

v = tweets.str.split().tolist()

c = Counter(chain.from_iterable(v))

tweets = [' '.join([j for j in i if c[j] > 1]) for i in v]

total_word = 0
for x,word in enumerate(tweets):
    num_word = len(word.split())
```

```
    total_word = total_word + num_word
print(total_word)
```

```
296750
```

```
X = np.array(tweets[: len(train)])
y = label
```

```
from sklearn.model_selection import train_test_split

X_train,X_val, y_train, y_val = train_test_split(X,y, stratify=y, test_size=0.3, random_state
X_train.shape, y_train.shape, X_val.shape,y_val.shape
```

```
((22373,), (22373,), (9589,), (9589,))
```

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer_tfidf = TfidfVectorizer(stop_words='english', max_df=0.7, min_df=0.01)
train_tfIdf = vectorizer_tfidf.fit_transform(X_train.astype('U'))
val_tfIdf = vectorizer_tfidf.transform(X_val.astype('U'))
print(vectorizer_tfidf.get_feature_names()[:5])
```

```
['affirmation', 'amazing', 'beautiful', 'best', 'blog']
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
  warnings.warn(msg, category=FutureWarning)
```

```
train_tfIdf.shape,  val_tfIdf.shape
```

```
((22373, 45), (9589, 45))
```

```
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=5).fit(train_tfIdf, y_train)
print(model.score(train_tfIdf, y_train))
```

```
0.9304071872346131
```

```
y_pred = model.predict(val_tfIdf)
print(model.score(val_tfIdf, y_val))
```

```
0.9313797059130253
```

```
from sklearn.metrics import confusion_matrix

print(confusion_matrix(y_val, y_pred))
```

```
[[8881   35]
 [ 623   50]]
```

```
from sklearn.metrics import classification_report
print(classification_report(y_val, y_pred))
```

```
                  precision    recall  f1-score   support

               0       0.93      1.00      0.96      8916
               1       0.59      0.07      0.13       673

        accuracy                           0.93      9589
       macro avg       0.76      0.54      0.55      9589
    weighted avg       0.91      0.93      0.91      9589
```

Machine Learning Data Science Laboratory (410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.6

Title: Yelp reviews polarity

Objectives:

1. To handle given data for Text Classification
2. To analyze Text using python programming language.

Software Requirement:

Windows /Linux

Theory:

The Yelp reviews polarity dataset is constructed by considering stars 1 and 2 negative, and 3 and 4 positive. For each polarity 280,000 training samples and 19,000 testing samples are take randomly. In total there are 560,000 trainig samples and 38,000 testing samples. Negative polarity is class 1, and positive class 2. Definition:

Information processing model that is inspired by the way biological nervous system (i.e. the brain) process information, is called Neural Network.
Neural Network has the ability to learn by examples. It is not designed to perform fix /, rather task which need thinking (e.g. Predictions).

ANN is composed of large number of highly interconnected processing elements(neurons)working in unison to solve problems. It mimic human brain.

With advanced in deep learning, you can now visualize the entire deep learning process or just the Convolutional Neural Network you've built.

We are going to build simple neural network using Keras and then use ANN visualizer to visualize our neural network.

Conclusion

Handled given data for Text Classification and analyzed Text using python programming language.

Machine Learning And Data Science Laboratory(410501)

BE Sem I Honors in ML&DS

Academic Year: 2022-23

Lab Assignment No.6

NAME : TANAYA PESHAVE

ROLL NO : 42465

PRN NO : 72017327C

ABC ID : 931-253-282-048

BRANCH : E&TC

COLLEGE : PICT

```
  from tensorflow.keras.models import Sequential
  from tensorflow.keras.layers import Dense, Embedding, GRU, LSTM, Dropout
  from tensorflow.keras import utils
  from tensorflow.keras.preprocessing.sequence import pad_sequences
  from tensorflow.keras.preprocessing.text import Tokenizer
  from tensorflow.keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
  import pandas as pd
  import numpy as np
  import matplotlib.pyplot as plt


num_words = 10000
max_review_len = 200


train = pd.read_csv('/content/test.csv',header=None,names=['Label', 'Review'])
test = pd.read_csv('/content/test.csv',header=None,names=['Label', 'Review'])


train
```

	Label	Review
0	2	Contrary to other reviews, I have zero complai...
1	1	Last summer I had an appointment to get new ti...
2	2	Friendly staff, same starbucks fair you get an...
3	1	The food is good. Unfortunately the service is...
4	2	Even when we didn't have a car Filene's Baseme...
...	...	...
37995	1	If I could give 0...I would. Don't do it.
37996	2	Items Selected:\nChocolate Cinnamon Horn\nSmal...
37997	1	Expensive lunch meals. Fried pickles were goo...

```
y_train, y_test = train['Label'] - 1, test['Label'] - 1
```

```
reviews = train['Review']
reviews[:5]
```

```
0    Contrary to other reviews, I have zero complai...
1    Last summer I had an appointment to get new ti...
2    Friendly staff, same starbucks fair you get an...
3    The food is good. Unfortunately the service is...
4    Even when we didn't have a car Filene's Baseme...
Name: Review, dtype: object
```

```
tokenizer = Tokenizer(num_words=num_words)
tokenizer.fit_on_texts(reviews)
tokenizer.word_index
```

```
{'the': 1,
 'and': 2,
 'i': 3,
 'to': 4,
 'a': 5,
 'was': 6,
 'of': 7,
 'it': 8,
 'for': 9,
 'in': 10,
 'is': 11,
 'n': 12,
 'that': 13,
 'my': 14,
 'we': 15,
 'this': 16,
 'but': 17,
 'with': 18,
```

```
 'they': 19,
 'you': 20,
 'on': 21,
 'not': 22,
 'have': 23,
 'had': 24,
 'at': 25,
 'were': 26,
 'so': 27,
 'are': 28,
 'food': 29,
 'be': 30,
 'place': 31,
 'me': 32,
 'there': 33,
 'good': 34,
 'as': 35,
 'out': 36,
 'like': 37,
 'if': 38,
 'just': 39,
 'all': 40,
 'our': 41,
 'very': 42,
 'get': 43,
 'here': 44,
 'one': 45,
 'time': 46,
 'great': 47,
 'when': 48,
 'up': 49,
 'or': 50,
 'would': 51,
 'from': 52,
 'service': 53,
 'their': 54,
 'back': 55,
 'about': 56,
 'no': 57,
 'go': 58
```

```
sequences = tokenizer.texts_to_sequences(reviews)


index = 42
print(reviews[index])
print(sequences[index])
```

```
This Valentines Day I ordered a pizza for my boyfriend and asked that they make a heart
[16, 6105, 145, 3, 91, 5, 173, 9, 14, 781, 2, 148, 13, 19, 123, 5, 1327, 21, 8, 36, 7, 5
```

```
print(tokenizer.word_index['some'])
print(tokenizer.word_index['of'])
```

```
print(tokenizer.word_index['the'])
print(tokenizer.word_index['worst'])
```

```
70
7
1
380
```

```
x_train = pad_sequences(sequences, maxlen=max_review_len)
x_train[0]
```

```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0, 9779,    4,   82,  316,    3,   23, 1340, 1659,   56,
          1,   53,   50,    1,  257,    3,   23,   68,  285, 1997,   53,
         44,    9,    1,  539,  149,  277,  160,    2, 1301,    4,   14,
        139,   18,  293,   37, 7947, 2175,  262,  516,   28, 1173,    2,
        112,   60,  567,  532, 1435,   16,   11,   45,   31,   13,    3,
         80,   22,  255,   37,    3,  144,  171,  694, 1955,    7,   39,
         73,    7,   14,   82, 2748, 5598,   23,   68,    9,   21,   14,
          7, 1569,    2,   23, 1787,   14, 2060, 1732,  470,   17,   44,
         14,   53,    2, 1403, 5752,   95,   40,   68,   89, 1039,    2,
        313,   49,    4,   32,    4, 1498, 1003,   19,   39, 3981,    1,
        308,  129,    8,  572,    5,  202,  114,  102,    8,   90,   10,
       1179,  277], dtype=int32)
```

```
model = Sequential()
model.add(Embedding(num_words, 64, input_length=max_review_len))
model.add(GRU(128))
model.add(Dense(1, activation='sigmoid'))
```

```
model.compile(optimizer='adam',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
model_save_path = '/content/best_model.h5'
checkpoint_callback = ModelCheckpoint(model_save_path,
                                      monitor='val_accuracy',
                                      save_best_only=True,
                                      verbose=1)
```

```
history = model.fit(x_train,
                    y_train,
                    epochs=5,
                    batch_size=128,
```

```
                    validation_split=0.1,
                    callbacks=[checkpoint_callback])
```

```
Epoch 1/5
267/268 [============================>.] - ETA: 0s - loss: 0.4554 - accuracy: 0.7795
Epoch 1: val_accuracy improved from -inf to 0.93289, saving model to /content/best_model
268/268 [==============================] - 11s 17ms/step - loss: 0.4552 - accuracy: 0.77
Epoch 2/5
267/268 [============================>.] - ETA: 0s - loss: 0.1890 - accuracy: 0.9283
Epoch 2: val_accuracy improved from 0.93289 to 0.93895, saving model to /content/best_mo
268/268 [==============================] - 4s 15ms/step - loss: 0.1889 - accuracy: 0.928
Epoch 3/5
268/268 [==============================] - ETA: 0s - loss: 0.1416 - accuracy: 0.9499
Epoch 3: val_accuracy did not improve from 0.93895
268/268 [==============================] - 4s 15ms/step - loss: 0.1416 - accuracy: 0.949
Epoch 4/5
265/268 [============================>.] - ETA: 0s - loss: 0.1086 - accuracy: 0.9636
Epoch 4: val_accuracy did not improve from 0.93895
268/268 [==============================] - 4s 15ms/step - loss: 0.1084 - accuracy: 0.963
Epoch 5/5
265/268 [============================>.] - ETA: 0s - loss: 0.0830 - accuracy: 0.9726
Epoch 5: val_accuracy did not improve from 0.93895
268/268 [==============================] - 4s 15ms/step - loss: 0.0828 - accuracy: 0.972
```

```
model.load_weights(model_save_path)
```

```
test_sequences = tokenizer.texts_to_sequences(test['Review'])
x_test = pad_sequences(test_sequences, maxlen=max_review_len)
x_test[0]
```

```
array([   0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0,    0,    0,    0,    0,    0,    0,    0,    0,    0,
          0,    0, 9779,    4,   82,  316,    3,   23, 1340, 1659,   56,
          1,   53,   50,    1,  257,    3,   23,   68,  285, 1997,   53,
         44,    9,    1,  539,  149,  277,  160,    2, 1301,    4,   14,
        139,   18,  293,   37, 7947, 2175,  262,  516,   28, 1173,    2,
        112,   60,  567,  532, 1435,   16,   11,   45,   31,   13,    3,
         80,   22,  255,   37,    3,  144,  171,  694, 1955,    7,   39,
         73,    7,   14,   82, 2748, 5598,   23,   68,    9,   21,   14,
          7, 1569,    2,   23, 1787,   14, 2060, 1732,  470,   17,   44,
         14,   53,    2, 1403, 5752,   95,   40,   68,   89, 1039,    2,
        313,   49,    4,   32,    4, 1498, 1003,   19,   39, 3981,    1,
        308,  129,    8,  572,    5,  202,  114,  102,    8,   90,   10,
       1179,  277], dtype=int32)
```

```
scores = model.evaluate(x_test, y_test, verbose=1)
```

```
print("The percent of correct answers:", round(scores[1] * 100, 4))
```

```
1188/1188 [==============================] - 7s 6ms/step - loss: 0.1294 - accuracy: 0.95
The percent of correct answers: 95.7053
```