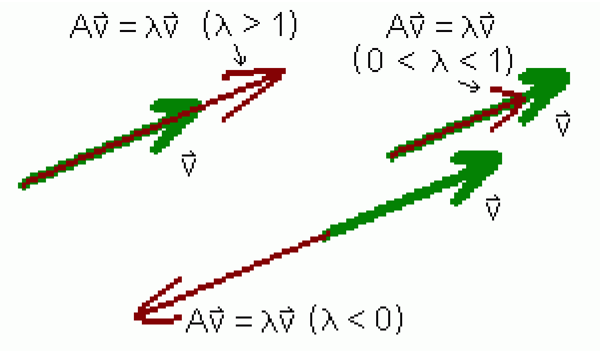
Practical 1

MATRIX MULTIPLICATION, EIGEN VECTORS, EIGENVALUE COMPUTATION USING TENSORFLOW

Definitions



Let AA be an n×nn×n matrix. The number λλ is an **eigenvalue** of AA if there exists a non-zero vector vv such that

Av=λv.Av=λv.

In this case, vector vv is called an **eigenvector** of AA corresponding to λλ.

```
import tensorflow as tf
print("Matrix Multiplication Demo")
```

```
Matrix Multiplication Demo
```

```
x=tf.constant([1,2,3,4,5,6],shape=[2,3])
print("Matrix A \n{}".format(x))
#print(x)
```

```
Matrix A
[[1 2 3]
 [4 5 6]]
```

```
y=tf.constant([7,8,9,10,11,12],shape=[3,2])
print("Matrix B \n{}".format(y))
```

```
Matrix B
[[ 7  8]
 [ 9 10]
 [11 12]]
```

```
z=tf.matmul(x,y)
print("Product: \n{}".format(z))
```

```
Product:
[[ 58  64]
 [139 154]]
```

```
e_matrix_A=tf.random.uniform([2,2],minval=3,maxval=10,dtype=tf.float32,name="matrixA")
```

```
e_matrix_A=tf.random.uniform([2,2],minval=3,maxval=10,dtype=tf.float32,name="matrixA")
print("Matrix A:\n{}".format(e_matrix_A))
```

```
Matrix A:
[[6.027209  5.4556627]
 [8.2146015 3.625847 ]]
```

```
eigen_values_A,eigen_vectors_A=tf.linalg.eigh(e_matrix_A)
print("Eigen Vectors:\n{}\n\nEigen values:\n{}\n".format(eigen_vectors_A, eigen_values_A))
```

```
Eigen Vectors:
[[-0.6539773   0.75651425]
 [ 0.75651425  0.6539773 ]]

Eigen values:
[-3.4753585 13.128415 ]
```

Practical 2

Deep Forward Network For XOR

Deep feedforward networks, also often called feedforward neural networks, or multilayer perceptron's (MLPs), are the quintessential deep learning models. The goal of a feedforward network is to approximate some function f.

For example, for a classifier, y = f(x) maps an input x to a category y. A feedforward network defines a mapping y = f (x; θ) and learns the value of the parameters θ that result in the best function approximation. These models are called feedforward because information flows through the function being evaluated from x, through the intermediate computations used to define f, and finally to the output y. There are no feedback connections in which outputs of the model are fed back into itself.

XOR Truth Table:

Inputs		Output
A	**B**	**X**
0	0	0
0	1	1
1	0	1
1	1	0

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

```
model=Sequential()
model.add(Dense(units=2,activation='relu',input_dim=2))
model.add(Dense(units=1,activation='sigmoid'))
```

```
#loss function binary_crossentropy
```

```
print(model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 2)                 6

 dense_1 (Dense)             (None, 1)                 3

=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
None
```

```
print(model.get_weights())
```

```
[array([[ 0.03003848,  0.12337577],
       [-1.1043283 ,  0.74570227]], dtype=float32), array([0., 0.], dtype=float32), array([[-0
       [ 0.2811495 ]], dtype=float32), array([0.], dtype=float32)]
```

```
X=np.array([[0.,0.],[0.,1.],[1.,0.],[1.,1.]])
Y=np.array([0.,1.,1.,0.])

model.fit(X,Y,epochs=1000,batch_size=4)
```

```
Epoch 62/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6941 - accuracy: 0.5000
Epoch 63/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6941 - accuracy: 0.5000
Epoch 64/1000
1/1 [==============================] - 0s 7ms/step - loss: 0.6940 - accuracy: 0.5000
Epoch 65/1000
1/1 [==============================] - 0s 7ms/step - loss: 0.6940 - accuracy: 0.5000
Epoch 66/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6940 - accuracy: 0.5000
Epoch 67/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6940 - accuracy: 0.5000
Epoch 68/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6940 - accuracy: 0.5000
Epoch 69/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6940 - accuracy: 0.5000
Epoch 70/1000
1/1 [==============================] - 0s 12ms/step - loss: 0.6940 - accuracy: 0.5000
Epoch 71/1000
1/1 [==============================] - 0s 7ms/step - loss: 0.6940 - accuracy: 0.5000
Epoch 72/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 73/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 74/1000
1/1 [==============================] - 0s 7ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 75/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 76/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 77/1000
```

```
1/1 [==============================] - 0s 7ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 78/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 79/1000
1/1 [==============================] - 0s 7ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 80/1000

1/1 [==============================] - 0s 7ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 81/1000
1/1 [==============================] - 0s 6ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 82/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.6939 - accuracy: 0.5000
Epoch 83/1000
1/1 [==============================] - 0s 6ms/step - loss: 0.6938 - accuracy: 0.5000
Epoch 84/1000
1/1 [==============================] - 0s 6ms/step - loss: 0.6938 - accuracy: 0.5000
Epoch 85/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.6938 - accuracy: 0.5000
Epoch 86/1000
1/1 [==============================] - 0s 8ms/step - loss: 0.6938 - accuracy: 0.5000
Epoch 87/1000
1/1 [==============================] - 0s 6ms/step - loss: 0.6938 - accuracy: 0.5000
Epoch 88/1000
1/1 [==============================] - 0s 7ms/step - loss: 0.6938 - accuracy: 0.5000
Epoch 89/1000
1/1 [==============================] - 0s 7ms/step - loss: 0.6938 - accuracy: 0.5000
Epoch 90/1000
1/1 [==============================] - 0s 5ms/step - loss: 0.6938 - accuracy: 0.5000
```

```
print(model.get_weights())
```

```
[array([[-0.07455742,  1.0847937 ],
       [-0.27809477, -0.10743535]], dtype=float32), array([0., 0.], dtype=float32), array([[ 1
       [-0.57224935]], dtype=float32), array([0.], dtype=float32)]
```

```
print(model.predict(X,batch_size=0))
```

```
[[0.55189294]
 [0.55189294]
 [0.55189294]
 [0.32152754]]
```

PRACTICAL 3A

CLASSIFICATION USING DNN

Classification neural networks used for feature categorization are very similar to fault-diagnosis networks, except that they only allow one output response for any input pattern, instead of allowing multiple faults to occur for a given set of operating conditions. The classification network selects the category based on which output response has the highest output value.

Problem statement:
The given dataset comprises health information about diabetic women patients. We need to create a deep feed forward network that will classify women suffering from diabetes mellitus as 1.

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

```
# diabetes.csv download link
# https://drive.google.com/file/d/1V-J7IqOFvYK4zyjUvFGr96Y67bO-7Ydm/view?usp=sharing
```

```
dataset=loadtxt('/content/sample_data/diabetes.csv',delimiter=',', skiprows=1)
print(dataset)
```

```
[[  6.    148.     72.    ...   0.627  50.      1.   ]
 [  1.     85.     66.    ...   0.351  31.      0.   ]
 [  8.    183.     64.    ...   0.672  32.      1.   ]
 ...
 [  5.    121.     72.    ...   0.245  30.      0.   ]
 [  1.    126.     60.    ...   0.349  47.      1.   ]
 [  1.     93.     70.    ...   0.315  23.      0.   ]]
```

```
X=dataset[:,0:8]
Y=dataset[:,8]
print(X)
```

```
[[  6.    148.     72.    ...  33.6     0.627  50.   ]
 [  1.     85.     66.    ...  26.6     0.351  31.   ]
 [  8.    183.     64.    ...  23.3     0.672  32.   ]
 ...
 [  5.    121.     72.    ...  26.2     0.245  30.   ]
 [  1.    126.     60.    ...  30.1     0.349  47.   ]
 [  1.     93.     70.    ...  30.4     0.315  23.   ]]
```

```
print(Y)
```

```
[1. 0. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 1. 1. 1. 1. 1. 0. 1. 0. 0. 1. 1.
 1. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0.
 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0.
 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 1. 1. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0.
 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0.
 1. 1. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 1.
 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 1. 1. 1. 1. 0.
 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 0. 0.
 1. 0. 1. 0. 0. 1. 0. 1. 0. 1. 1. 1. 0. 0. 1. 0. 1. 0. 0. 0. 1. 0. 0. 0.
 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 1. 1. 1. 0. 1.
 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 0. 0.
 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0. 1. 0. 1. 0.
 1. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 0.
 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0.
 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1. 1. 0. 0. 0. 0. 0.
 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.
 0. 1. 0. 1. 1. 0. 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 1. 0. 0. 1. 0.
 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 0. 1. 1. 0. 1. 0. 0. 0. 1. 1. 0. 0. 0. 0.
```

```
 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 0. 1. 1.
 1. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 1. 0. 1. 1. 1. 1. 0. 1. 1. 0. 0. 0. 0.
 0. 0. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1. 0. 1. 0. 1. 0. 1.
 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 1. 0. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0. 1.
 0. 0. 1. 0. 1. 1. 1. 0. 0. 1. 1. 1. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 1. 0.]
```

```
#Creating model
model = Sequential()
#Dense means hidden layer
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
print(model.summary())
```

```
Model: "sequential_10"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_24 (Dense)            (None, 12)                108

 dense_25 (Dense)            (None, 8)                 104

 dense_26 (Dense)            (None, 1)                 9

=================================================================
Total params: 221
Trainable params: 221
Non-trainable params: 0
_________________________________________________________________
None
```

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

model.fit(X, Y, epochs=150, batch_size=10)
```

```
Epoch 1/150
77/77 [==============================] - 1s 2ms/step - loss: 20.3858 - accuracy: 0.6510
Epoch 2/150
77/77 [==============================] - 0s 2ms/step - loss: 2.2423 - accuracy: 0.6328
Epoch 3/150
77/77 [==============================] - 0s 2ms/step - loss: 1.2760 - accuracy: 0.6224
Epoch 4/150
77/77 [==============================] - 0s 2ms/step - loss: 1.0711 - accuracy: 0.6341
Epoch 5/150
77/77 [==============================] - 0s 2ms/step - loss: 0.9242 - accuracy: 0.6393
Epoch 6/150
77/77 [==============================] - 0s 2ms/step - loss: 0.8099 - accuracy: 0.6628
Epoch 7/150
77/77 [==============================] - 0s 2ms/step - loss: 0.7526 - accuracy: 0.6549
Epoch 8/150
77/77 [==============================] - 0s 2ms/step - loss: 0.7073 - accuracy: 0.6615
Epoch 9/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6887 - accuracy: 0.6719
Epoch 10/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6782 - accuracy: 0.6706
Epoch 11/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6780 - accuracy: 0.6693
Epoch 12/150
```

```
77/77 [==============================] - 0s 2ms/step - loss: 0.6533 - accuracy: 0.6654
Epoch 13/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6380 - accuracy: 0.6849
Epoch 14/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6317 - accuracy: 0.6771
Epoch 15/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6409 - accuracy: 0.6849
Epoch 16/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6257 - accuracy: 0.6979
Epoch 17/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6578 - accuracy: 0.6901
Epoch 18/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6230 - accuracy: 0.6875
Epoch 19/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6201 - accuracy: 0.6927
Epoch 20/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6094 - accuracy: 0.6979
Epoch 21/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6127 - accuracy: 0.7096
Epoch 22/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6012 - accuracy: 0.7018
Epoch 23/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6056 - accuracy: 0.6979
Epoch 24/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6104 - accuracy: 0.7005
Epoch 25/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5954 - accuracy: 0.7083
Epoch 26/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6031 - accuracy: 0.6940
Epoch 27/150
77/77 [==============================] - 0s 2ms/step - loss: 0.6315 - accuracy: 0.6862
Epoch 28/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5798 - accuracy: 0.7122
Epoch 29/150
77/77 [==============================] - 0s 2ms/step - loss: 0.5828 - accuracy: 0.7031
```

```
predictions = model.predict(X)
rounded = [round(x[0]) for x in predictions]
print(rounded)
```

```
[1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0,
```

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

```
24/24 [==============================] - 0s 1ms/step - loss: 0.4776 - accuracy: 0.7656

accuracy: 76.56%
```

PRACTICAL 3B

BINARY CLASSIFICATION USING MLP

Multilayer Perceptron falls under the category of feedforward algorithms, because inputs are combined with the initial weights in a weighted sum and subjected to the activation function, just like in the Perceptron. But the difference is that each linear combination is propagated to the next layer. Each layer is

feeding the next one with the result of their computation, their internal representation of the data. This goes all the way through the hidden layers to the output layer.
Binary classification, which looks at an input and predicts which of two possible classes it belongs to. Practical uses include sentiment analysis, spam detection, and credit-card fraud detection. Such models are trained with datasets labelled with 1s and 0s representing the two classes, employ popular learning algorithms such as logistic regression and Naïve Bayes, and are frequently built with libraries such as Scikit-learn.

```
# mlp for binary classification
from pandas import read_csv
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

```
# Ionosphere.csv download link
# https://drive.google.com/file/d/1youDnAIZdWTybK9gAkSpsWp5zgismVjS/view?usp=sharing
```

```
# load the dataset
path = '/content/sample_data/Ionosphere.csv'
df = read_csv(path, header=None, skiprows=1)
```

```
# split into input and output columns
X, y = df.values[:, :-1], df.values[:, -1]
# ensure all data are floating point values
X = X.astype('float32')
# encode strings to integer
y = LabelEncoder().fit_transform(y)
# split into train and test datasets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33)
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
# determine the number of input features
n_features = X_train.shape[1]
# define model
model = Sequential()
model.add(Dense(10, activation='relu', kernel_initializer='he_normal', input_shape=(n_features,)))
model.add(Dense(8, activation='relu', kernel_initializer='he_normal'))
model.add(Dense(1, activation='sigmoid'))
# compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
# fit the model
model.fit(X_train, y_train, epochs=150, batch_size=32, verbose=0)
```

```
(235, 34) (116, 34) (235,) (116,)
<keras.callbacks.History at 0x7ff913826610>
```

```
# evaluate the model
loss, acc = model.evaluate(X_test, y_test, verbose=0)
print('Test Accuracy: %.3f' % acc)
```

```
Test Accuracy: 0.966
```

```
# make a prediction
row = [1,0,0.99539,-0.05889,0.85243,0.02306,0.83398,-0.37708,1,0.03760,0.85243,-0.17755,0.59755,-0.4
yhat = model.predict([row])
print('Predicted: %.3f' % yhat)
```

```
Predicted: 0.985
```

PRACTICAL 4

PREDICTING THE PROBABILITY OF THE CLASS

```
from keras.models import Sequential
from keras.layers import Dense
from sklearn.datasets import make_blobs
from sklearn.preprocessing import MinMaxScaler
```

```
X,Y=make_blobs(n_samples=100,centers=2,n_features=2,random_state=1)
scalar=MinMaxScaler()
scalar.fit(X)
X=scalar.transform(X)
```

```
model=Sequential()
model.add(Dense(4,input_dim=2,activation='relu'))
model.add(Dense(4,activation='relu'))
model.add(Dense(1,activation='sigmoid'))
model.compile(loss='binary_crossentropy',optimizer='adam')
model.fit(X,Y,epochs=500)
```

```
Epoch 1/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7692
Epoch 2/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7651
Epoch 3/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7616
Epoch 4/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7577
Epoch 5/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7544
Epoch 6/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7509
Epoch 7/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7478
Epoch 8/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7450
Epoch 9/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7417
Epoch 10/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7388
Epoch 11/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7358
Epoch 12/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7326
Epoch 13/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7297
Epoch 14/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7268
```

```
Epoch 15/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7240
Epoch 16/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7209
Epoch 17/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7177
Epoch 18/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7144
Epoch 19/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7114
Epoch 20/500
4/4 [==============================] - 0s 3ms/step - loss: 0.7076
Epoch 21/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7043
Epoch 22/500
4/4 [==============================] - 0s 2ms/step - loss: 0.7005
Epoch 23/500
4/4 [==============================] - 0s 2ms/step - loss: 0.6967
Epoch 24/500
4/4 [==============================] - 0s 2ms/step - loss: 0.6931
Epoch 25/500
4/4 [==============================] - 0s 2ms/step - loss: 0.6898
Epoch 26/500
4/4 [==============================] - 0s 2ms/step - loss: 0.6864
Epoch 27/500
4/4 [==============================] - 0s 3ms/step - loss: 0.6832
Epoch 28/500
4/4 [==============================] - 0s 2ms/step - loss: 0.6804
Epoch 29/500
4/4 [==============================] - 0s 2ms/step - loss: 0.6785
```

```
Xnew,Yreal=make_blobs(n_samples=3,centers=2,n_features=2,random_state=1)
Xnew=scalar.transform(Xnew)
Yclass=model.predict(Xnew)
Ynew=model.predict(Xnew)
```

```
for i in range(len(Xnew)):
  print("X=%s,Predicted_probability=%s,Predicted_class=%s"%(Xnew[i],Ynew[i],Yclass[i]))
```

```
X=[0.89337759 0.65864154],Predicted_probability=[0.00376934],Predicted_class=[0.00376934]
X=[0.29097707 0.12978982],Predicted_probability=[0.81591594],Predicted_class=[0.81591594]
X=[0.78082614 0.75391697],Predicted_probability=[0.00787392],Predicted_class=[0.00787392]
```

PRACTICAL 5A

CNN FOR CIFAR10 IMAGES

A neural network in which at least one layer is a convolutional layer. A typical convolutional neural network consists of some combination of the following layers:

- convolutional layers
- pooling layers
- dense layers
  Convolutional neural networks have had great success in certain kinds of problems, such as image recognition.

```
import tensorflow as tf
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

```
(train_images, train_labels), (test_images, test_labels) = datasets.cifar10.load_data()
# Normalize pixel values to be between 0 and 1
train_images, test_images = train_images / 255.0, test_images / 255.0
```

```
Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz
170500096/170498071 [==============================] - 2s 0us/step
170508288/170498071 [==============================] - 2s 0us/step
```

```
class_names = ['airplane', 'automobile', 'bird', 'cat', 'deer',
               'dog', 'frog', 'horse', 'ship', 'truck']
plt.figure(figsize=(10,10))
for i in range(25):
    plt.subplot(5,5,i+1)
    plt.xticks([])
    plt.yticks([])
    plt.grid(False)
    plt.imshow(train_images[i])
    # The CIFAR labels happen to be arrays,
    # which is why you need the extra index
    plt.xlabel(class_names[train_labels[i][0]])
plt.show()
```

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

=================================================================
Total params: 56,320
Trainable params: 56,320
Non-trainable params: 0
_________________________________________________________________
```

```
model.add(layers.Flatten())
model.add(layers.Dense(64, activation='relu'))
model.add(layers.Dense(10))
model.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 64)          36928

 flatten (Flatten)           (None, 1024)              0

 dense_9 (Dense)             (None, 64)                65600

 dense_10 (Dense)            (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
```

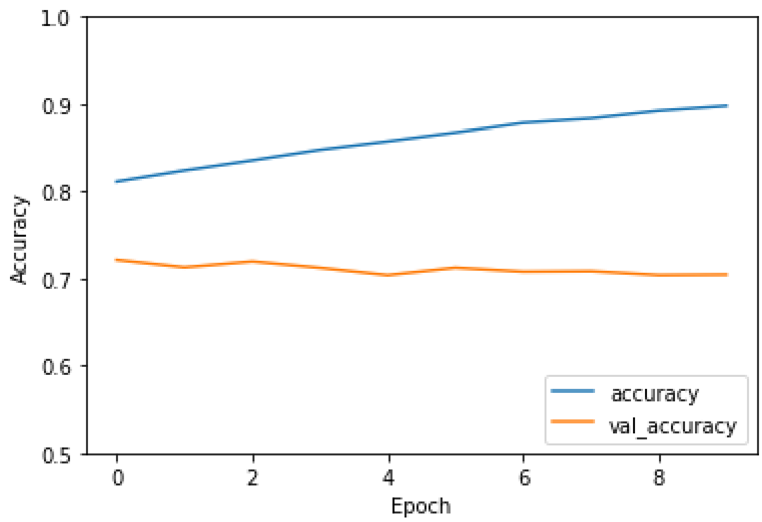
```
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
history = model.fit(train_images, train_labels, epochs=10,
                    validation_data=(test_images, test_labels))
```

```
Epoch 1/10
1563/1563 [==============================] - 43s 28ms/step - loss: 0.5349 - accuracy: 0.8113 -
Epoch 2/10
1563/1563 [==============================] - 44s 28ms/step - loss: 0.4958 - accuracy: 0.8240 -
Epoch 3/10
1563/1563 [==============================] - 43s 28ms/step - loss: 0.4615 - accuracy: 0.8355 -
Epoch 4/10
1563/1563 [==============================] - 43s 28ms/step - loss: 0.4303 - accuracy: 0.8475 -
Epoch 5/10
1563/1563 [==============================] - 44s 28ms/step - loss: 0.4006 - accuracy: 0.8571 -
Epoch 6/10
1563/1563 [==============================] - 43s 28ms/step - loss: 0.3716 - accuracy: 0.8672 -
Epoch 7/10
1563/1563 [==============================] - 44s 28ms/step - loss: 0.3449 - accuracy: 0.8784 -
Epoch 8/10
1563/1563 [==============================] - 43s 28ms/step - loss: 0.3240 - accuracy: 0.8835 -
Epoch 9/10
1563/1563 [==============================] - 42s 27ms/step - loss: 0.3001 - accuracy: 0.8922 -
Epoch 10/10
1563/1563 [==============================] - 43s 27ms/step - loss: 0.2817 - accuracy: 0.8977 -
```

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label ='val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5,1])
plt.legend(loc='lower right')
test_loss,test_acc=model.evaluate(test_images, test_labels, verbose=2)
```

```
313/313 - 2s - loss: 1.2522 - accuracy: 0.7043 - 2s/epoch - 7ms/step
```




```
print(test_acc)
```

```
0.7042999863624573
```

PRACTICAL 5B

IMAGE CLASSIFICATION

[ ] ↳ *27 cells hidden*

PRACTICAL 5C

DATA AUGMENTATION

Data augmentation is a strategy that enables practitioners to significantly increase the diversity of data available for training models, without actually collecting new data. Data augmentation techniques such as cropping, padding, and horizontal flipping are commonly used to train large neural networks.

```
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
import tensorflow_datasets as tfds
```

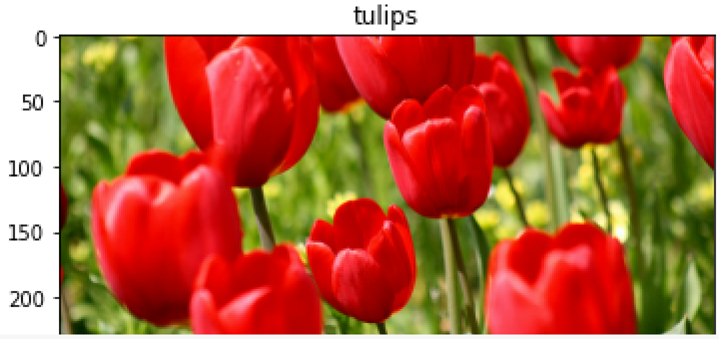
```
from tensorflow.keras import layers
(train_ds, val_ds, test_ds), metadata = tfds.load(
    'tf_flowers',
    split=['train[:80%]', 'train[80%:90%]', 'train[90%:]'],
    with_info=True,
    as_supervised=True,
)
```

```
Downloading and preparing dataset 218.21 MiB (download: 218.21 MiB, generated: 221.83 MiB, tot
DI Completed...: 100%                                   5/5 [00:03<00:00, 1.57 file/s]
Dataset tf_flowers downloaded and prepared to ~/tensorflow_datasets/tf_flowers/3.0.1. Subseque
```

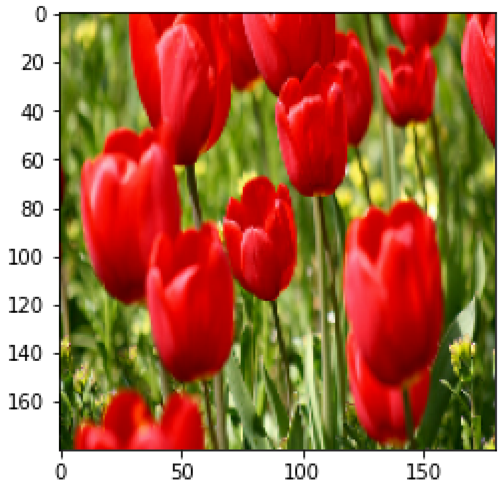
```
num_classes = metadata.features['label'].num_classes
print(num_classes)
```

```
5
```

```
get_label_name = metadata.features['label'].int2str

image, label = next(iter(train_ds))
_ = plt.imshow(image)
_ = plt.title(get_label_name(label))
```




```
IMG_SIZE = 180

resize_and_rescale = tf.keras.Sequential([
  layers.Resizing(IMG_SIZE, IMG_SIZE),
  layers.Rescaling(1./255)
])
result = resize_and_rescale(image)
_ = plt.imshow(result)
```

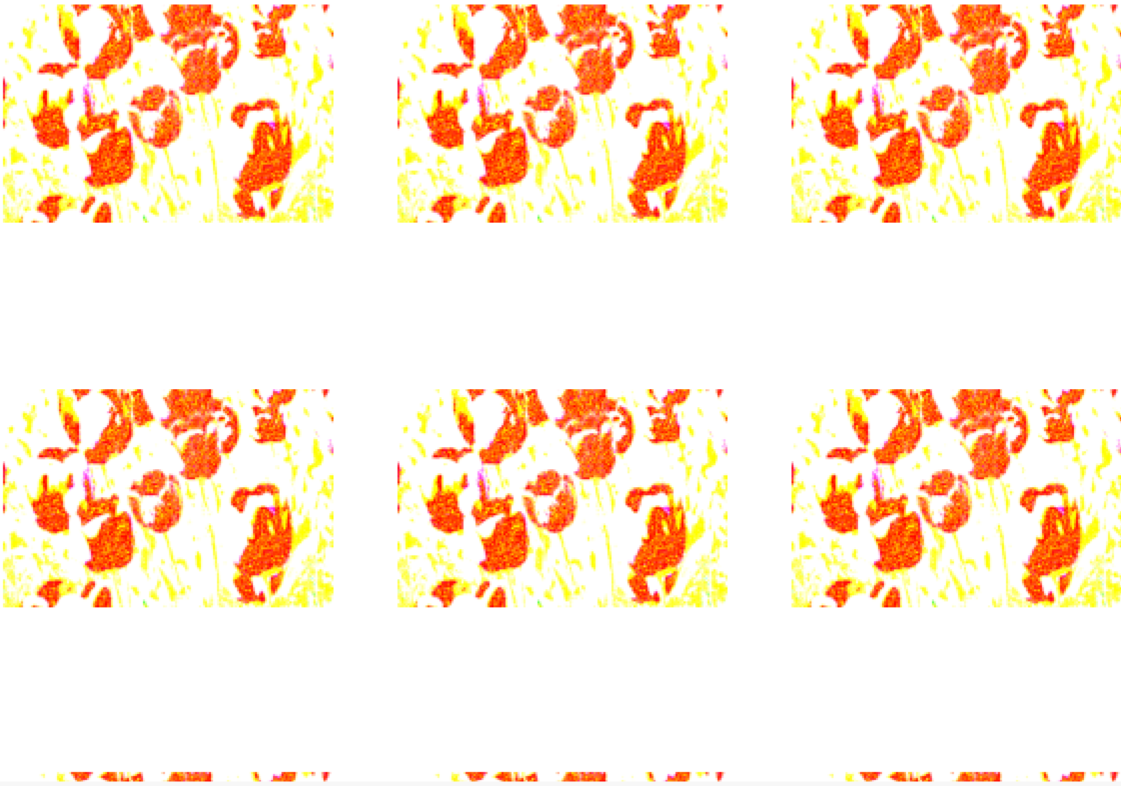



```
print("Min and max pixel values:", result.numpy().min(), result.numpy().max())
```

```
Min and max pixel values: 0.0 1.0
```

```
data_augmentation = tf.keras.Sequential([
  layers.RandomFlip("horizontal_and_vertical"),
  layers.RandomRotation(0.2),
])
# Add the image to a batch.
image = tf.expand_dims(image, 0)
plt.figure(figsize=(10, 10))
for i in range(9):
  augmented_image = data_augmentation(image)
  ax = plt.subplot(3, 3, i + 1)
  plt.imshow(augmented_image[0])
  plt.axis("off")
```

```
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..
WARNING:matplotlib.image:Clipping input data to the valid range for imshow with RGB data ([0..
```



```
model = tf.keras.Sequential([
  # Add the preprocessing layers you created earlier.
  resize_and_rescale,
  data_augmentation,
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  # Rest of your model.
])
```

```
aug_ds = train_ds.map(
  lambda x, y: (resize_and_rescale(x, training=True), y))
batch_size = 32
AUTOTUNE = tf.data.AUTOTUNE

def prepare(ds, shuffle=False, augment=False):
  # Resize and rescale all datasets.
  ds = ds.map(lambda x, y: (resize_and_rescale(x), y),
              num_parallel_calls=AUTOTUNE)

  if shuffle:
    ds = ds.shuffle(1000)

  # Batch all datasets.
  ds = ds.batch(batch_size)

  # Use data augmentation only on the training set.
  if augment:
    ds = ds.map(lambda x, y: (data_augmentation(x, training=True), y),
```

```
                num_parallel_calls=AUTOTUNE)

  # Use buffered prefetching on all datasets.
  return ds.prefetch(buffer_size=AUTOTUNE)
```

```
train_ds = prepare(train_ds, shuffle=True, augment=True)
val_ds = prepare(val_ds)
test_ds = prepare(test_ds)

model = tf.keras.Sequential([
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(num_classes)
])
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
epochs=5
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/5
92/92 [==============================] - 65s 680ms/step - loss: 1.4067 - accuracy: 0.3971 - va
Epoch 2/5
92/92 [==============================] - 61s 652ms/step - loss: 1.1268 - accuracy: 0.5364 - va
Epoch 3/5
92/92 [==============================] - 62s 663ms/step - loss: 1.0204 - accuracy: 0.5950 - va
Epoch 4/5
92/92 [==============================] - 61s 652ms/step - loss: 0.9607 - accuracy: 0.6213 - va
Epoch 5/5
92/92 [==============================] - 61s 659ms/step - loss: 0.8926 - accuracy: 0.6543 - va
```

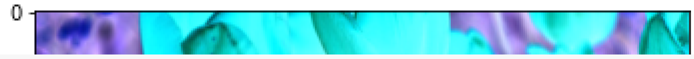
```
loss, acc = model.evaluate(test_ds)
print("Accuracy", acc)
```

```
12/12 [==============================] - 3s 199ms/step - loss: 0.8842 - accuracy: 0.6512
Accuracy 0.6512261629104614
```

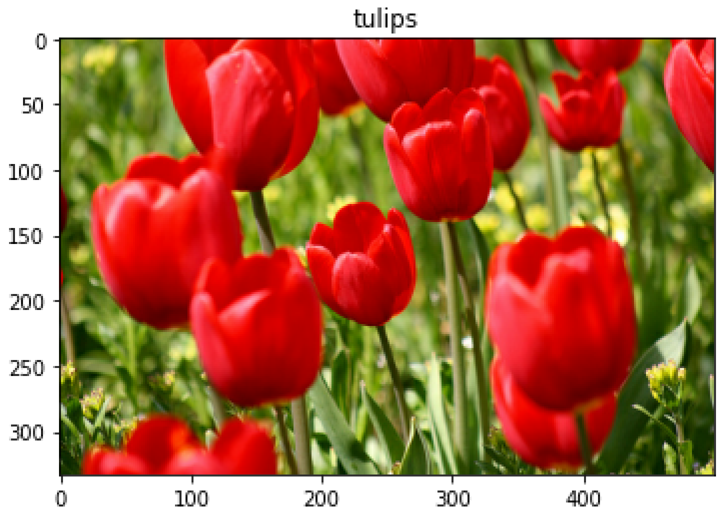
```
def random_invert_img(x, p=0.5):
  if  tf.random.uniform([]) < p:
    x = (255-x)
  else:
    x
  return x
def random_invert(factor=0.5):
  return layers.Lambda(lambda x: random_invert_img(x, factor))
```

```
random_invert = random_invert()
plt.figure(figsize=(10, 10))
for i in range(9):
  augmented_image = random_invert(image)
  ax = plt.subplot(3, 3, i + 1)
  plt.imshow(augmented_image[0].numpy().astype("uint8"))
  plt.axis("off")
```

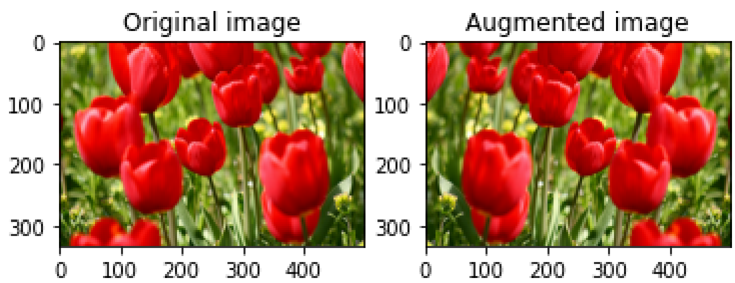
```
class RandomInvert(layers.Layer):
  def __init__(self, factor=0.5, **kwargs):
    super().__init__(**kwargs)
    self.factor = factor

  def call(self, x):
    return random_invert_img(x)
_ = plt.imshow(RandomInvert()(image)[0])
```



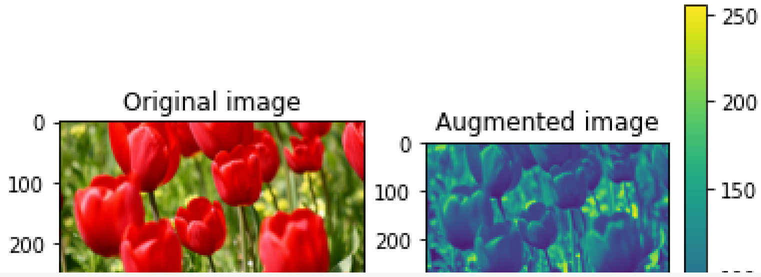

```
(train_ds, val_ds, test_ds), metadata = tfds.load(
    'tf_flowers',
    split=['train[:80%]', 'train[80%:90%]', 'train[90%:]'],
    with_info=True,
    as_supervised=True,
)
image, label = next(iter(train_ds))
_ = plt.imshow(image)
_ = plt.title(get_label_name(label))
```



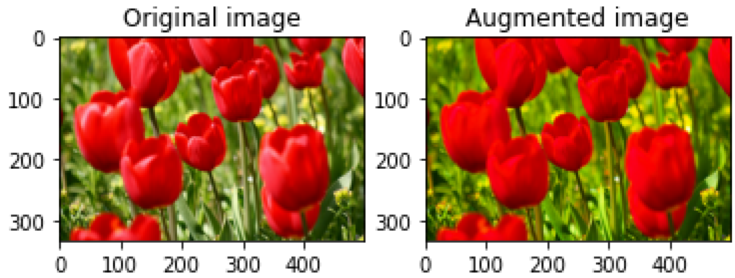

```
def visualize(original, augmented):
  fig = plt.figure()
  plt.subplot(1,2,1)
  plt.title('Original image')
  plt.imshow(original)

  plt.subplot(1,2,2)
  plt.title('Augmented image')
  plt.imshow(augmented)
flipped = tf.image.flip_left_right(image)
visualize(image, flipped)
```



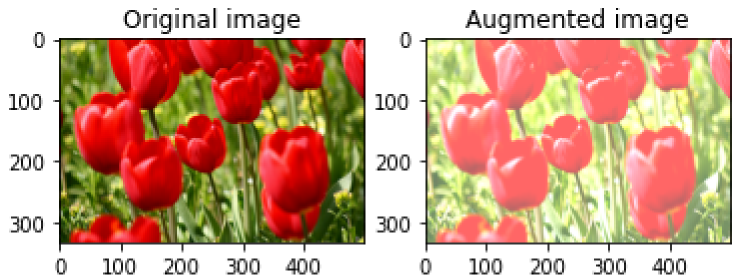

```
grayscaled = tf.image.rgb_to_grayscale(image)
visualize(image, tf.squeeze(grayscaled))
_ = plt.colorbar()
```



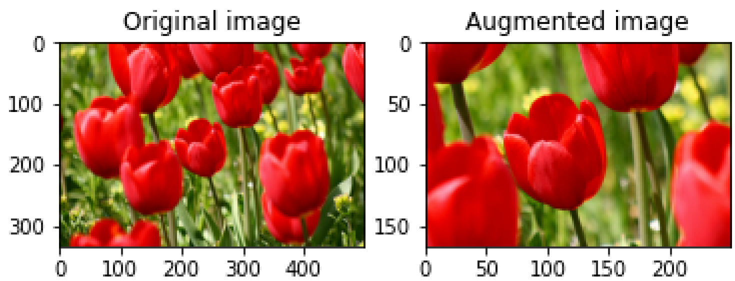

```
saturated = tf.image.adjust_saturation(image, 3)
visualize(image, saturated)
```




```
bright = tf.image.adjust_brightness(image, 0.4)
visualize(image, bright)
```




```
cropped = tf.image.central_crop(image, central_fraction=0.5)
visualize(image,cropped)
```




```
rotated = tf.image.rot90(image)
visualize(image, rotated)
```

PRACTICAL 6

BUILDING RNN USING SINGLE NEURON

Recurrent neural networks (RNN) are a class of neural networks that is powerful for modeling sequence data such as time series or natural language. Schematically, a RNN layer uses a for loop to iterate over the timesteps of a sequence, while maintaining an internal state that encodes information about the timesteps it has seen so far. The Keras RNN API is designed with a focus on:
Ease of use: the built-in keras.layers.RNN, keras. layers.LSTM, keras.layers.GRU layers enable you to quickly build recurrent models without having to make difficult configuration choices.

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
```

```
model = keras.Sequential()
# Add an Embedding layer expecting input vocab of size 1000, and
# output embedding dimension of size 64.
model.add(layers.Embedding(input_dim=1000, output_dim=64))
# Add a LSTM layer with 128 internal units.
model.add(layers.LSTM(128))
# Add a Dense layer with 10 units.
model.add(layers.Dense(10))

model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, None, 64)          64000

 lstm (LSTM)                 (None, 128)               98816

 dense_17 (Dense)            (None, 10)                1290

=================================================================
Total params: 164,106
Trainable params: 164,106
Non-trainable params: 0
_________________________________________________________________
```

```
model = keras.Sequential()
model.add(layers.Embedding(input_dim=1000, output_dim=64))

# The output of GRU will be a 3D tensor of shape (batch_size, timesteps, 256)
model.add(layers.GRU(256, return_sequences=True))
```

```
# The output of SimpleRNN will be a 2D tensor of shape (batch_size, 128)
model.add(layers.SimpleRNN(128))

model.add(layers.Dense(10))

model.summary()
```

```
Model: "sequential_12"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, None, 64)          64000

 gru (GRU)                   (None, None, 256)         247296

 simple_rnn (SimpleRNN)      (None, 128)               49280

 dense_18 (Dense)            (None, 10)                1290

=================================================================
Total params: 361,866
Trainable params: 361,866
Non-trainable params: 0
_________________________________________________________________
```

```
encoder_vocab = 1000
decoder_vocab = 2000

encoder_input = layers.Input(shape=(None,))
encoder_embedded = layers.Embedding(input_dim=encoder_vocab, output_dim=64)(
    encoder_input
)
```

```
# Return states in addition to output
output, state_h, state_c = layers.LSTM(64, return_state=True, name="encoder")(
    encoder_embedded
)
```

```
encoder_state = [state_h, state_c]

decoder_input = layers.Input(shape=(None,))
decoder_embedded = layers.Embedding(input_dim=decoder_vocab, output_dim=64)(
    decoder_input
)
```

```
# Pass the 2 states to a new LSTM layer, as initial state
decoder_output = layers.LSTM(64, name="decoder")(
    decoder_embedded, initial_state=encoder_state
)
output = layers.Dense(10)(decoder_output)
```

```
model = keras.Model([encoder_input, decoder_input], output)
model.summary()
```

```
Model: "model"
```

```
Layer (type)                 Output Shape         Param #     Connected to
==================================================================================
input_1 (InputLayer)         [(None, None)]       0           []

input_2 (InputLayer)         [(None, None)]       0           []

embedding_2 (Embedding)      (None, None, 64)     64000       ['input_1[0][0]']

embedding_3 (Embedding)      (None, None, 64)     128000      ['input_2[0][0]']

encoder (LSTM)               [(None, 64),         33024       ['embedding_2[0][0]']
                              (None, 64),
                              (None, 64)]

decoder (LSTM)               (None, 64)           33024       ['embedding_3[0][0]',
                                                               'encoder[0][1]',
                                                               'encoder[0][2]']

dense_19 (Dense)             (None, 10)           650         ['decoder[0][0]']

==================================================================================
Total params: 258,698
Trainable params: 258,698
Non-trainable params: 0
__________________________________________________________________________________
```

PRACTICAL 7

NLP CORPUS

GUTENBERG

```
import nltk
```

```
nltk.__file__
```

```
'/usr/local/lib/python3.7/dist-packages/nltk/__init__.py'
```

```
nltk.download('popular')
```

```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package gazetteers to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/gazetteers.zip.
[nltk_data]    | Downloading package genesis to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/genesis.zip.
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/gutenberg.zip.
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/inaugural.zip.
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
```

```
[nltk_data]    |   Unzipping corpora/movie_reviews.zip.
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/names.zip.
[nltk_data]    | Downloading package shakespeare to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/shakespeare.zip.
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/stopwords.zip.
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/treebank.zip.
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/twitter_samples.zip.
[nltk_data]    | Downloading package omw to /root/nltk_data...
[nltk_data]    | Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]    | Downloading package wordnet to /root/nltk_data...
[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 maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Unzipping tokenizers/punkt.zip.
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data]    |
[nltk_data]  Done downloading collection popular
True
```

```
nltk.download('gutenberg')
```

```
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
True
```

```
from nltk.corpus import gutenberg
```

```
nltk.corpus.gutenberg.fileids()
```

```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
```

```
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

```
#read single file
gutenberg.raw('austen-emma.txt')
```

```
'[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER I\n\n\nEmma Woodhouse, handsome, clever, a
nd rich, with a comfortable home\nand happy disposition, seemed to unite some of the best ble
ssings\nof existence; and had lived nearly twenty-one years in the world\nwith very little to
distress or vex her.\n\nShe was the youngest of the two daughters of a most affectionate,\nin
dulgent father; and had, in consequence of her sister\'s marriage,\nbeen mistress of his hous
e from a very early period.  Her mother\nhad died too long ago for her to have more than an i
ndistinct\nremembrance of her caresses; and her place had been supplied\nby an excellent woma
n as governess, who had fallen little short\nof a mother in affection.\n\nSixteen years had M
iss Taylor been in Mr. Woodhouse\'s family,\nless as a governess than a friend, very fond of
both daughters,\nbut particularly of Emma.  Between _them_ it was more the intimacy\nof siste
```

```
#read multiple files
gutenberg.raw(['austen-emma.txt','austen-sense.txt'])
```


```
'[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER I\n\n\nEmma Woodhouse, handsome, clever, a
nd rich, with a comfortable home\nand happy disposition, seemed to unite some of the best ble
ssings\nof existence; and had lived nearly twenty-one years in the world\nwith very little to
distress or vex her.\n\nShe was the youngest of the two daughters of a most affectionate,\nin
dulgent father; and had, in consequence of her sister\'s marriage,\nbeen mistress of his hous
e from a very early period.  Her mother\nhad died too long ago for her to have more than an i
ndistinct\nremembrance of her caresses; and her place had been supplied\nby an excellent woma
n as governess, who had fallen little short\nof a mother in affection.\n\nSixteen years had M
iss Taylor been in Mr. Woodhouse\'s family,\nless as a governess than a friend, very fond of
both daughters,\nbut particularly of Emma.  Between _them_ it was more the intimacy\nof siste
```


```
len(gutenberg.raw(['austen-emma.txt','austen-sense.txt']))
```

```
1560093
```

```
nltk.corpus.gutenberg.words(fileids=['austen-emma.txt','whitman-leaves.txt'])
```

```
['[', 'Emma', 'by', 'Jane', 'Austen', '1816', ']', ...]
```

```
len(nltk.corpus.gutenberg.words(fileids=['austen-emma.txt','whitman-leaves.txt']))
```

```
347310
```

```
gutenberg.raw(fileids=['austen-emma.txt','whitman-leaves.txt'])
```


```
'[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER I\n\n\nEmma Woodhouse, handsome, clever, a
nd rich, with a comfortable home\nand happy disposition, seemed to unite some of the best ble
ssings\nof existence; and had lived nearly twenty-one years in the world\nwith very little to
distress or vex her.\n\nShe was the youngest of the two daughters of a most affectionate,\nin
dulgent father; and had, in consequence of her sister\'s marriage,\nbeen mistress of his hous
e from a very early period.  Her mother\nhad died too long ago for her to have more than an i
ndistinct\nremembrance of her caresses; and her place had been supplied\nby an excellent woma
n as governess, who had fallen little short\nof a mother in affection.\n\nSixteen years had M
iss Taylor been in Mr. Woodhouse\'s family,\nless as a governess than a friend, very fond of
both daughters,\nbut particularly of Emma.  Between _them_ it was more the intimacy\nof siste
```


```
len(gutenberg.raw(fileids=['austen-emma.txt','whitman-leaves.txt']))
```

```
1598286
```

```
len(gutenberg.sents())
```

```
98503
```

```
gutenberg.root
```

```
FileSystemPathPointer('/root/nltk_data/corpora/gutenberg')
```

```
gutenberg.encoding('austen-emma.txt')
```

```
'latin1'
```

```
gutenberg.readme()
```

```
'Project Gutenberg Selections\nhttp://gutenberg.net/\n\nThis corpus contains etexts from from
Project Gutenberg,\nby the following authors:\n\n* Jane Austen (3)\n* William Blake (2)\n* Th
ornton W. Burgess\n* Sarah Cone Bryant\n* Lewis Carroll\n* G. K. Chesterton (3)\n* Maria Edge
worth\n* King James Bible\n* Herman Melville\n* John Milton\n* William Shakespeare (3)\n* Wal
t Whitman\n\nThe beginning of the body of each book could not be identified automatically,\ns
o the semi-generic header of each file has been removed, and included below.\nSome source fil
es ended with a line "End of The Project Gutenberg Etext...",\nand this has been deleted.\n\n
Information about Project Gutenberg (one page)\n\nWe produce about two million dollars for ea
ch hour we work.  The\nfifty hours is one conservative estimate for how long it we take\nto g
et any etext selected, entered, proofread, edited, copyright\nsearched and analyzed, the copy
```

BROWN

```
nltk.download('brown')
```

```
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Unzipping corpora/brown.zip.
True
```

```
from nltk.corpus import brown
```

```
# files in a corpus - fileids()
nltk.corpus.brown.fileids()
```

```
['ca01',
 'ca02',
 'ca03',
 'ca04',
 'ca05',
 'ca06',
 'ca07',
 'ca08',
 'ca09',
 'ca10',
 'ca11',
 'ca12',
 'ca13',
 'ca14',
 'ca15',
 'ca16',
 'ca17',
 'ca18',
```

```
 'ca19',
 'ca20',
 'ca21',
 'ca22',
 'ca23',
 'ca24',
 'ca25',
 'ca26',
 'ca27',
 'ca28',
 'ca29',
 'ca30',
 'ca31',
 'ca32',
 'ca33',
 'ca34',
 'ca35',
 'ca36',
 'ca37',
 'ca38',
 'ca39',
 'ca40',
 'ca41',
 'ca42',
 'ca43',
 'ca44',
 'cb01',
 'cb02',
 'cb03',
 'cb04',
 'cb05',
 'cb06',
 'cb07',
 'cb08',
 'cb09',
 'cb10',
 'cb11',
 'cb12',
 'cb13',
 'cb14',
```

```
# list categories of the corpus - categories()
nltk.corpus.brown.categories()
```

```
['adventure',
 'belles_lettres',
 'editorial',
 'fiction',
 'government',
 'hobbies',
 'humor',
 'learned',
 'lore',
 'mystery',
 'news',
 'religion',
 'reviews',
 'romance',
 'science_fiction']
```

```
# files of the corpus corresponding to categories - fileids(['category1', 'category2'])
nltk.corpus.brown.fileids(['adventure'])
```

```
['cn01',
 'cn02',
 'cn03',
 'cn04',
 'cn05',
 'cn06',
 'cn07',
 'cn08',
 'cn09',
 'cn10',
 'cn11',
 'cn12',
 'cn13',
 'cn14',
 'cn15',
 'cn16',
 'cn17',
 'cn18',
 'cn19',
 'cn20',
 'cn21',
 'cn22',
 'cn23',
 'cn24',
 'cn25',
 'cn26',
 'cn27',
 'cn28',
 'cn29']
```

```
# files of the corpus corresponding to categories
nltk.corpus.brown.fileids(['adventure', 'lore'])
```

```
['cf01',
 'cf02',
 'cf03',
 'cf04',
 'cf05',
 'cf06',
 'cf07',
 'cf08',
 'cf09',
 'cf10',
 'cf11',
 'cf12',
 'cf13',
 'cf14',
 'cf15',
 'cf16',
 'cf17',
 'cf18',
 'cf19',
 'cf20',
 'cf21',
 'cf22',
 'cf23',
 'cf24',
 'cf25',
 'cf26',
 'cf27',
 'cf28',
 'cf29',
```

```
 'cf30',
 'cf31',
 'cf32',
 'cf33',
 'cf34',
 'cf35',
 'cf36',
 'cf37',
 'cf38',
 'cf39',
 'cf40',
 'cf41',
 'cf42',
 'cf43',
 'cf44',
 'cf45',
 'cf46',
 'cf47',
 'cf48',
 'cn01',
 'cn02',
 'cn03',
 'cn04',
 'cn05',
 'cn06',
 'cn07',
 'cn08',
 'cn09',
 'cn10',
```

```
# categories of the corpus corresponding to fileids - categories(['fileids1', 'fileids2'])
nltk.corpus.brown.categories(['cn29', 'cn24'])
```

```
['adventure']
```

```
nltk.corpus.brown.categories(['ca29'])
```

```
['news']
```

```
# raw content of the corpus - raw()
nltk.corpus.brown.raw()
```

```
'\n\n\tThe/at Fulton/np-tl County/nn-tl Grand/jj-tl Jury/nn-tl said/vbd Friday/nr an/at inves
tigation/nn of/in Atlanta's/np$ recent/jj primary/nn election/nn produced/vbd ``/`` no/at evi
dence/nn ''/'' that/cs any/dti irregularities/nns took/vbd place/nn ./.\n\n\n\tThe/at jury/nn
further/rbr said/vbd in/in term-end/nn presentments/nns that/cs the/at City/nn-tl Executive/j
j-tl Committee/nn-tl ,/, which/wdt had/hvd over-all/jj charge/nn of/in the/at election/nn ,/,
``/`` deserves/vbz the/at praise/nn and/cc thanks/nns of/in the/at City/nn-tl of/in-tl Atlant
a/np-tl ''/'' for/in the/at manner/nn in/in which/wdt the/at election/nn was/bedz conducted/v
bn ./.\n\n\n\tThe/at September-October/np term/nn jury/nn had/hvd been/ben charged/vbn by/in
Fulton/np-tl Superior/jj-tl Court/nn-tl Judge/nn-tl Durwood/np Pye/np to/to investigate/vb re
ports/nns of/in possible/jj ``/`` irregularities/nns ''/'' in/in the/at hard-fought/jj primar
```

```
len(nltk.corpus.brown.raw())
```

```
9964284
```

```
# raw content of the specified file - raw(fileids=['f1','f2'])
nltk.corpus.brown.raw(fileids=['ca01','cn23'])
```

```
'\n\n\tThe/at Fulton/np-tl County/nn-tl Grand/jj-tl Jury/nn-tl said/vbd Friday/nr an/at inves
tigation/nn of/in Atlanta's/np$ recent/jj primary/nn election/nn produced/vbd ``/`` no/at evi
dence/nn ''/'' that/cs any/dti irregularities/nns took/vbd place/nn ./.\n\n\n\tThe/at jury/nn
further/rbr said/vbd in/in term-end/nn presentments/nns that/cs the/at City/nn-tl Executive/j
j-tl Committee/nn-tl ,/, which/wdt had/hvd over-all/jj charge/nn of/in the/at election/nn ,/,
``/`` deserves/vbz the/at praise/nn and/cc thanks/nns of/in the/at City/nn-tl of/in-tl Atlant
a/np-tl ''/'' for/in the/at manner/nn in/in which/wdt the/at election/nn was/bedz conducted/v
bn ./.\n\n\n\tThe/at September-October/np term/nn jury/nn had/hvd been/ben charged/vbn by/in
Fulton/np-tl Superior/jj-tl Court/nn-tl Judge/nn-tl Durwood/np Pye/np to/to investigate/vb re
```

```
len(nltk.corpus.brown.raw(fileids=['ca01','cn23']))
```

```
39286
```

```
# raw content of the specified category - raw(categories=['c1','c2'])
nltk.corpus.brown.raw(categories=['news','lore'])
```

```
'\n\n\tThe/at Fulton/np-tl County/nn-tl Grand/jj-tl Jury/nn-tl said/vbd Friday/nr an/at inves
tigation/nn of/in Atlanta's/np$ recent/jj primary/nn election/nn produced/vbd ``/`` no/at evi
dence/nn ''/'' that/cs any/dti irregularities/nns took/vbd place/nn ./.\n\n\n\tThe/at jury/nn
further/rbr said/vbd in/in term-end/nn presentments/nns that/cs the/at City/nn-tl Executive/j
j-tl Committee/nn-tl ,/, which/wdt had/hvd over-all/jj charge/nn of/in the/at election/nn ,/,
``/`` deserves/vbz the/at praise/nn and/cc thanks/nns of/in the/at City/nn-tl of/in-tl Atlant
a/np-tl ''/'' for/in the/at manner/nn in/in which/wdt the/at election/nn was/bedz conducted/v
bn ./.\n\n\n\tThe/at September-October/np term/nn jury/nn had/hvd been/ben charged/vbn by/in
Fulton/np-tl Superior/jj-tl Court/nn-tl Judge/nn-tl Durwood/np Pye/np to/to investigate/vb re
ports/nns of/in possible/jj ``/`` irregularities/nns ''/'' in/in the/at hard-fought/jj primar
```

```
len(nltk.corpus.brown.raw(categories=['news','lore']))
```

```
1832464
```

```
# words of the corpus - words()
nltk.corpus.brown.words()
```

```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

```
len(nltk.corpus.brown.words())
```

```
1161192
```

```
# words of the specified file - words(fileids=['f1','f2'])
nltk.corpus.brown.words(fileids=['ca03','cn23'])
```

```
['Several', 'defendants', 'in', 'the', 'Summerdale', ...]
```

```
len(nltk.corpus.brown.words(fileids=['ca01','cn23']))
```

```
4591
```

```
# words of the specified category - words(categories=['c1','c2'])
nltk.corpus.brown.words(categories=['news','lore'])
```

```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

```
len(nltk.corpus.brown.words(categories=['news','lore']))
```

```
210853
```

```
# sents of the corpus - sents()
nltk.corpus.brown.sents()
```

```
[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of',
"Atlanta's", 'recent', 'primary', 'election', 'produced', '``', 'no', 'evidence', "''",
'that', 'any', 'irregularities', 'took', 'place', '.'], ['The', 'jury', 'further', 'said',
'in', 'term-end', 'presentments', 'that', 'the', 'City', 'Executive', 'Committee', ',',
'which', 'had', 'over-all', 'charge', 'of', 'the', 'election', ',', '``', 'deserves', 'the',
'praise', 'and', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta', "''", 'for', 'the',
'manner', 'in', 'which', 'the', 'election', 'was', 'conducted', '.'], ...]
```

```
len(nltk.corpus.brown.sents())
```

```
57340
```

```
# sents of the specified file - sents(fileids=['f1','f2'])
nltk.corpus.brown.sents(fileids=['ca03','cn23'])
```

```
[['Several', 'defendants', 'in', 'the', 'Summerdale', 'police', 'burglary', 'trial', 'made',
'statements', 'indicating', 'their', 'guilt', 'at', 'the', 'time', 'of', 'their', 'arrest',
',', 'Judge', 'James', 'B.', 'Parsons', 'was', 'told', 'in', 'Criminal', 'court',
'yesterday', '.'], ['The', 'disclosure', 'by', 'Charles', 'Bellows', ',', 'chief', 'defense',
'counsel', ',', 'startled', 'observers', 'and', 'was', 'viewed', 'as', 'the', 'prelude',
'to', 'a', 'quarrel', 'between', 'the', 'six', 'attorneys', 'representing', 'the', 'eight',
'former', 'policemen', 'now', 'on', 'trial', '.'], ...]
```

```
len(nltk.corpus.brown.sents(fileids=['ca01','cn23']))
```

```
227
```

```
# sents of the specified category - words(categories=['c1','c2'])
nltk.corpus.brown.sents(categories=['news','lore'])
```

```
[['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', 'Friday', 'an', 'investigation', 'of',
"Atlanta's", 'recent', 'primary', 'election', 'produced', '``', 'no', 'evidence', "''",
'that', 'any', 'irregularities', 'took', 'place', '.'], ['The', 'jury', 'further', 'said',
'in', 'term-end', 'presentments', 'that', 'the', 'City', 'Executive', 'Committee', ',',
'which', 'had', 'over-all', 'charge', 'of', 'the', 'election', ',', '``', 'deserves', 'the',
'praise', 'and', 'thanks', 'of', 'the', 'City', 'of', 'Atlanta', "''", 'for', 'the',
'manner', 'in', 'which', 'the', 'election', 'was', 'conducted', '.'], ...]
```

```
len(nltk.corpus.brown.sents(categories=['news','lore']))
```

```
9504
```

```
# location of given file on disk
nltk.corpus.brown.abspath('ca01')
```

```
FileSystemPathPointer('/root/nltk_data/corpora/brown/ca01')
```

```
# show encoding of the file
nltk.corpus.brown.encoding('cn23')
```

```
'ascii'
```

```
# open stream to read the file
nltk.corpus.brown.open('ca03')
```

```
<nltk.data.SeekableUnicodeStreamReader at 0x7fb7b2c0ea90>
```

```
# path of root of locally installed corpus
nltk.corpus.brown.root
```

```
FileSystemPathPointer('/root/nltk_data/corpora/brown')
```

```
# readme file
nltk.corpus.brown.readme()
```

```
'BROWN CORPUS\n\nA Standard Corpus of Present-Day Edited American\nEnglish, for use with Digi
tal Computers.\n\nby W. N. Francis and H. Kucera (1964)\nDepartment of Linguistics, Brown Uni
versity\nProvidence, Rhode Island, USA\n\nRevised 1971, Revised and Amplified 1979\n\nhttp://
www.hit.uib.no/icame/brown/bcm.html\n\nDistributed with the permission of the copyright holde
r,\nredistribution permitted.\n'
```

INAUGURAL

[ ] ↳ *9 cells hidden*

REUTERS

```
nltk.download('reuters')
from nltk.corpus import reuters
```

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

```
nltk.corpus.reuters.fileids()
```

```
['test/14826',
 'test/14828',
 'test/14829',
 'test/14832',
 'test/14833',
 'test/14839',
 'test/14840',
 'test/14841',
 'test/14842',
 'test/14843',
 'test/14844',
 'test/14849',
 'test/14852',
 'test/14854',
 'test/14858',
 'test/14859',
 'test/14860',
 'test/14861',
 'test/14862',
 'test/14863',
 'test/14865',
 'test/14867',
 'test/14872',
 'test/14873',
```

```
 'test/14875',
 'test/14876',
 'test/14877',
 'test/14881',
 'test/14882',
 'test/14885',
 'test/14886',
 'test/14888',
 'test/14890',
 'test/14891',
 'test/14892',
 'test/14899',
 'test/14900',
 'test/14903',
 'test/14904',
 'test/14907',
 'test/14909',
 'test/14911',
 'test/14912',
 'test/14913',
 'test/14918',
 'test/14919',
 'test/14921',
 'test/14922',
 'test/14923',
 'test/14926',
 'test/14928',
 'test/14930',
 'test/14931',
 'test/14932',
 'test/14933',
 'test/14934',
 'test/14941',
```

```
nltk.corpus.reuters.categories()
```

```
['acq',
 'alum',
 'barley',
 'bop',
 'carcass',
 'castor-oil',
 'cocoa',
 'coconut',
 'coconut-oil',
 'coffee',
 'copper',
 'copra-cake',
 'corn',
 'cotton',
 'cotton-oil',
 'cpi',
 'cpu',
 'crude',
 'dfl',
 'dlr',
 'dmk',
 'earn',
 'fuel',
 'gas',
 'gnp',
 'gold',
 'grain',
```

```
 'groundnut',
 'groundnut-oil',
 'heat',
 'hog',
 'housing',
 'income',
 'instal-debt',
 'interest',
 'ipi',
 'iron-steel',
 'jet',
 'jobs',
 'l-cattle',
 'lead',
 'lei',
 'lin-oil',
 'livestock',
 'lumber',
 'meal-feed',
 'money-fx',
 'money-supply',
 'naphtha',
 'nat-gas',
 'nickel',
 'nkr',
 'nzdlr',
 'oat',
 'oilseed',
 'orange',
 'palladium',
 'palm-oil',
```

```
reuters.raw('test/16587')
```

```
'HONEYWELL INC &lt;HON> 1ST QTR OPER NET\n  Oper shr 96 cts vs 79 cts\n      Oper net 43.7 ml
n vs 36.4 mln\n      Sales 1.48 billion vs 1.15 billion\n      NOTE: 1987 sales includes oper
ations of Sperry Aerospace.\n      1986 operating net excludes a charge from discontinued\n
operations of 10.2 mln dlrs or 22 cts a share.\n  \n\n'
```

```
reuters.raw(['test/16587','test/16588'])
```

```
'HONEYWELL INC &lt;HON> 1ST QTR OPER NET\n  Oper shr 96 cts vs 79 cts\n      Oper net 43.7 ml
n vs 36.4 mln\n      Sales 1.48 billion vs 1.15 billion\n      NOTE: 1987 sales includes oper
ations of Sperry Aerospace.\n      1986 operating net excludes a charge from discontinued\n
operations of 10.2 mln dlrs or 22 cts a share.\n  \n\nWALL STREET STOCKS/BROWNING FERRIS &lt;
BFI>\n  The Environmental Protection Agency\'s\n  five to 10 mln dlr suit against a Browning-
Ferris Industries\n  Inc &lt;BFI> unit, CECOS International Inc, caused the stock to\n  drop
today, analysts said.\n      The stock has fallen 2-1/4 to 56-1/8 so far today, after\n  the
news about the suit was released this morning.\n      "It\'s potentially a big suit and inves
tors feel that its\n  not good to go against regulators," Kenneth Ch\'u-K\'ai Leung, a\n  Smi
th Barney analyst said.\n      "What investors are actually saying by selling off some BFI\n
```

```
nltk.corpus.reuters.words(fileids=['test/16587','test/16588'])
```

```
['HONEYWELL', 'INC', '&', 'lt', ';', 'HON', '>', '1ST', ...]
```

```
len(reuters.sents())
```

```
54711
```

```
len(reuters.raw(['test/16587','test/16588']))
```

```
1483
```

```
len(nltk.corpus.reuters.words(fileids=['test/16587','test/16588']))
```

```
303
```

```
reuters.raw(categories=['tin','trade'])
```

```
'ASIAN EXPORTERS FEAR DAMAGE FROM U.S.-JAPAN RIFT\n  Mounting trade friction between the\n
U.S. And Japan has raised fears among many of Asia\'s exporting\n  nations that the row could
inflict far-reaching economic\n  damage, businessmen and officials said.\n      They told Reu
ter correspondents in Asian capitals a U.S.\n  Move against Japan might boost protectionist s
entiment in the\n  U.S. And lead to curbs on American imports of their products.\n      But s
ome exporters said that while the conflict would hurt\n  them in the long-run, in the short-t
erm Tokyo\'s loss might be\n  their gain.\n      The U.S. Has said it will impose 300 mln dlr
s of tariffs on\n  imports of Japanese electronics goods on April 17, in\n  retaliation for J
apan\'s alleged failure to stick to a pact not\n  to sell semiconductors on world markets at
below cost.\n      Unofficial Japanese estimates put the impact of the tariffs\n  at 10 billi
```

```
reuters.root
```

```
ZipFilePathPointer('/root/nltk_data/corpora/reuters.zip', 'reuters/')
```

```
reuters.encoding('test/16587')
```

```
'ISO-8859-2'
```

```
reuters.readme()
```

```
'\n      The Reuters-21578 benchmark corpus, ApteMod version\n\nThis is a publically availabl
e version of the well-known Reuters-21578\n"ApteMod" corpus for text categorization.  It has
been used in\npublications like these:\n\n * Yiming Yang and X. Liu. "A re-examination of tex
t categorization\n   methods".  1999.  Proceedings of 22nd Annual International SIGIR.\n   ht
tp://citeseer.nj.nec.com/yang99reexamination.html\n\n * Thorsten Joachims. "Text categorizati
on with support vector\n   machines: learning with many relevant features".  1998. Proceeding
s\n   of ECML-98, 10th European Conference on Machine Learning.\n   http://citeseer.nj.nec.co
m/joachims98text.html\n\nApteMod is a collection of 10,788 documents from the Reuters financi
al\nnewswire service, partitioned into a training set with 7769 documents\nand a test set wit
h 3019 documents.  The total size of the corpus is\nabout 43 MB.  It is also available for do
```

Practical 7 Part 2 Create Your Own Corpora

scrape from rj college site about page

```
import requests
from bs4 import BeautifulSoup
import nltk
```

```
page = requests.get("https://www.rjcollege.edu.in/about-us/")
soup = BeautifulSoup(page.content,'html.parser')
str3 = soup.find_all('p')[0].get_text()
#strall = [x.get_text() for x in soup.find_all('p')]
str3
```

```
'On the auspicious day of Shri Krishna Janmashtami, 15th August 1938, the people of Ghatkopar
and the surrounding suburbs witnessed the birth of Hindi Vidya Prachar Samiti, a brain child
of a visionary Late Shri Nandkishore Singh Jairamji. The Samiti was established with the obje
ctives of catering to the educational needs of the Hindi speaking community. It made a humble
beginning by starting a primary school, which gradually expanded into a full-fledged secondar
y school.\nThe Hindi High School with its high academic standards has carved for itself a pla
ce not only among leading secondary schools in Mumbai but also educational institutions impar
ting instructions in Hindi throughout Maharashtra. With its primary objectives achieved the S
```

```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
from nltk.tokenize import sent_tokenize, word_tokenize
sents = sent_tokenize(str3)
sents
```

```
['On the auspicious day of Shri Krishna Janmashtami, 15th August 1938, the people of
Ghatkopar and the surrounding suburbs witnessed the birth of Hindi Vidya Prachar Samiti, a
brain child of a visionary Late Shri Nandkishore Singh Jairamji.',
 'The Samiti was established with the objectives of catering to the educational needs of the
Hindi speaking community.',
 'It made a humble beginning by starting a primary school, which gradually expanded into a
full-fledged secondary school.',
 'The Hindi High School with its high academic standards has carved for itself a place not
only among leading secondary schools in Mumbai but also educational institutions imparting
instructions in Hindi throughout Maharashtra.',
 'With its primary objectives achieved the Samiti decided to extend its frontiers and broaden
its horizons.',
 'As a result, Ramniranjan Jhunjhunwala College came into existence in 1963, enabling a
larger section of the society to take advantage of the facilities provided for higher
education.',
 'In 1976 the Junior College section was introduced and in 1981 the Commerce faculty
commenced both at the Junior and Degree College level.',
 'From 1999-2000 the College has added a number of self-financing courses like BMS, B.B.I,
B.Sc.',
 'in C.S., I.T., Biotechnology, M.Sc.',
 'in Computer Science and Biotechnology as well as add on courses, which further hone the
special skills of the students.',
 'In 2014 saw a change in education system with greater emphasis being given to employability
of youth.',
 'As an effort to realize the dream of Make in India, Digital India, Clean and Green India,
we have started skill based program supported by University Grants commission known as
Bachelor in Vocation.',
 'The college has been reaccredited with ‘A’ Grade by NAAC in 2014 with a CGPA 3.50 and
received the Best College Award (2007-2008) of the University of Mumbai.',
 'The College has been bestowed with IMC RAMKRISHNA BAJAJ PERFORMANCE EXCELLENCE TROPHY,
2010.',
 'The Principal of the college was awarded “Best Teacher” by Government of Maharashtra in
2011.',
 'Government of Maharashtra conferred the college with “JAAGAR JAANIVANCHA” (First in Mumbai
Suburban- in 2013 and Second in Mumbai Suburban- in 2014) for safety of girls.']
```

rjc addmission page as table

```
page1 = requests.get("https://www.rjcollege.edu.in/pgadmission2022-23/")
soup1 = BeautifulSoup(page1.content, 'html.parser')
```

```
table1 = soup1.find('table', id='tablepress-57')
table1
```

```
<table class="tablepress tablepress-id-57" id="tablepress-57">
<thead>
<tr class="row-1 odd">
<th class="column-1">Program</th><th class="column-2">FULL</th><th class="column-3">I ST
INSTALLMENT</th><th class="column-4">II ND INSTALLMENT</th><th class="column-5">SC</th><th
class="column-6">ST</th>
</tr>
</thead>
<tbody class="row-hover">
<tr class="row-2 even">
<td class="column-1">MSC I  (Botany, Zoology, Chemistry-Physical, Organic and Inorganic)
</td><td class="column-2">15465</td><td class="column-3">10100</td><td class="column-
4">5365</td><td class="column-5">590</td><td class="column-6">3740</td>
</tr>
<tr class="row-3 odd">
<td class="column-1">MA I (HINDI / ENGLISH)</td><td class="column-2">11615</td><td
class="column-3">7500</td><td class="column-4">4115</td><td class="column-5">590</td><td
class="column-6">3740</td>
</tr>
<tr class="row-4 even">
<td class="column-1">MCOM I  </td><td class="column-2">15415</td><td class="column-
3">10000</td><td class="column-4">5415</td><td class="column-5">590</td><td class="column-
6">3740</td>
</tr>
<tr class="row-5 odd">
<td class="column-1">MSC PHYSICS I</td><td class="column-2">30615</td><td class="column-
3">19800</td><td class="column-4">10815</td><td class="column-5">590</td><td class="column-
6">3740</td>
</tr>
<tr class="row-6 even">
<td class="column-1">MSC I BT</td><td class="column-2">45015</td><td class="column-
3">29100</td><td class="column-4">15915</td><td class="column-5">590</td><td class="column-
6">3740</td>
</tr>
<tr class="row-7 odd">
<td class="column-1">MSC I CHEM ANALYTICAL</td><td class="column-2">45015</td><td
class="column-3">29200</td><td class="column-4">15815</td><td class="column-5">590</td><td
class="column-6">3740</td>
</tr>
<tr class="row-8 even">
<td class="column-1">MSC I COMP SCI</td><td class="column-2">44615</td><td class="column-
3">29000</td><td class="column-4">15615</td><td class="column-5">590</td><td class="column-
6">3740</td>
</tr>
<tr class="row-9 odd">
<td class="column-1">MSC I INFORMATION TECH</td><td class="column-2">44615</td><td
class="column-3">29000</td><td class="column-4">15615</td><td class="column-5">590</td><td
class="column-6">3740</td>
</tr>
<tr class="row-10 even">
<td class="column-1">MA EMA I  </td><td class="column-2">52375</td><td class="column-
3">34000</td><td class="column-4">18375</td><td class="column-5">2200</td><td
class="column-6">5350</td>
</tr>
<tr class="row-11 odd">
<td class="column-1">MSC I ENVIRONMENTAL SCIENCE &amp; DISASTER MANAGEMENT  </td><td
class="column-2">45015</td><td class="column-3">29200</td><td class="column-4">15815</td>
<td class="column-5">-</td><td class="column-6">-</td>
```

scrape images

```python
page3 = requests.get('https://www.rjcollege.edu.in/gallery/')
soup3 = BeautifulSoup(page3.content, 'html.parser')

images_list = []
images = soup3.select('img')
for image in images:
    src = image.get('src')
    alt = image.get('alt')
    images_list.append({"src": src, "alt": alt})

for image in images_list:
    print(image)
```


```
{'src': 'data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAAWQAAABjAQMAAACWp+hvAAAAA1BMVEUAAACn
{'src': 'https://www.rjcollege.edu.in/wp-content/uploads/2021/12/website-logo-1.png', 'alt':
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
```

{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi
{'src': 'data:image/svg+xml,%3Csvg%20xmlns%3D%22http%3A%2F%2Fwww.w3.org%2F2000%2Fsvg%22%20wi

scrape for xml

```
# https://www.w3schools.com/xml/note.xml
page4 = requests.get("https://www.w3schools.com/xml/note.xml")
soup4 = BeautifulSoup(page4.content, 'xml')
to = soup4.find_all('to')
to
```

```
[<to>Tove</to>]
```

scrape for json

```
# https://medium.com/@cmukesh8688/web-scraping-json-dictionary-and-pandas-part-2-4d34432281d9
page4 = requests.get("https://maps2.dcgis.dc.gov/dcgis/rest/services/FEEDS/MPD/MapServer/2/query?whe
data_json = page4.json()
print(type(data_json))
```

```
<class 'dict'>
```

```
data_json
```

{'displayFieldName': 'CCN',
 'fieldAliases': {'CCN': 'CCN',
  'REPORT_DAT': 'REPORT_DATE',
  'SHIFT': 'SHIFT',
  'METHOD': 'METHOD',
  'OFFENSE': 'OFFENSE',
  'BLOCK': 'BLOCK',
  'XBLOCK': 'XBLOCK',
  'YBLOCK': 'YBLOCK',
  'WARD': 'WARD',
  'ANC': 'ANC',
  'DISTRICT': 'DISTRICT',
  'PSA': 'PSA',
  'NEIGHBORHOOD_CLUSTER': 'NEIGHBORHOOD_CLUSTER',
  'BLOCK_GROUP': 'BLOCK_GROUP',
  'CENSUS_TRACT': 'CENSUS_TRACT',
  'VOTING_PRECINCT': 'VOTING_PRECINCT',
  'LATITUDE': 'LATITUDE',
  'LONGITUDE': 'LONGITUDE',
  'BID': 'BID',
  'START_DATE': 'START_DATE',
  'END_DATE': 'END_DATE',
  'OBJECTID': 'OBJECTID',
  'OCTO_RECORD_ID': 'OCTO_RECORD_ID'},
 'geometryType': 'esriGeometryPoint',
 'spatialReference': {'wkid': 4326, 'latestWkid': 4326},

```
 'fields': [{'name': 'CCN',
   'type': 'esriFieldTypeString',
   'alias': 'CCN',
   'length': 8},
  {'name': 'REPORT_DAT',
   'type': 'esriFieldTypeDate',
   'alias': 'REPORT_DATE',
   'length': 8},
  {'name': 'SHIFT',
   'type': 'esriFieldTypeString',
   'alias': 'SHIFT',
   'length': 50},
  {'name': 'METHOD',
   'type': 'esriFieldTypeString',
   'alias': 'METHOD',
   'length': 50},
  {'name': 'OFFENSE',
   'type': 'esriFieldTypeString',
   'alias': 'OFFENSE',
   'length': 250},
  {'name': 'BLOCK',
   'type': 'esriFieldTypeString',
   'alias': 'BLOCK',
   'length': 100},
  {'name': 'XBLOCK', 'type': 'esriFieldTypeDouble', 'alias': 'XBLOCK'},
  {'name': 'YBLOCK', 'type': 'esriFieldTypeDouble', 'alias': 'YBLOCK'},
  {'name': 'WARD',
   'type': 'esriFieldTypeString',
   'alias': 'WARD',
   'length': 1},
  {'name': 'ANC', 'type': 'esriFieldTypeString', 'alias': 'ANC', 'length': 5},
  {'name': 'DISTRICT',
```


```
for x in data_json:
    print(x)
```

```
displayFieldName
fieldAliases
geometryType
spatialReference
fields
features
exceededTransferLimit
```

```
print(data_json["features"])
```


```
[{'attributes': {'CCN': '20155350', 'REPORT_DAT': 1604088714000, 'SHIFT': 'EVENING', 'METHOD':
```


```
df = pd.DataFrame(data_json['features'])
df.head()
```

PRACTICAL 8

Lemmatization, Stemming, Tokenization, Stopwords

```
# https://www.nltk.org/book/ch08.html
import nltk
```

```
groucho_grammar = nltk.CFG.fromstring("""
S -> NP VP
PP -> P NP
NP -> Det N | Det N PP | 'I'
VP -> V NP | VP PP
Det -> 'an' | 'my'
N -> 'elephant' | 'pajamas'
V -> 'shot'
P -> 'in'
""")
```

```
sent = ['I', 'shot', 'an', 'elephant', 'in', 'my', 'pajamas']
parser = nltk.ChartParser(groucho_grammar)
for tree in parser.parse(sent):
  print(tree)
```

```
(S
  (NP I)
  (VP
    (VP (V shot) (NP (Det an) (N elephant)))
    (PP (P in) (NP (Det my) (N pajamas)))))
(S
  (NP I)
  (VP
    (V shot)
    (NP (Det an) (N elephant) (PP (P in) (NP (Det my) (N pajamas))))))
```

```
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
True
```

```
from nltk.tokenize import word_tokenize
from nltk.stem.wordnet import WordNetLemmatizer

def wordtokenization():
    content = """Stemming is funnier than a bummer says the sushi loving computer scientist.
    She really wants to buy cars. She told me angrily. It is better for you.
```

```
    Man is walking. We are meeting tomorrow. You really don't know..!"""
    print(word_tokenize(content))

def wordlemmatization():
    wordlemma = WordNetLemmatizer()
    print(wordlemma.lemmatize('cars'))
    print(wordlemma.lemmatize('walking',pos='v'))
    print(wordlemma.lemmatize('meeting',pos='n'))
    print(wordlemma.lemmatize('meeting',pos='v'))
    print(wordlemma.lemmatize('better',pos='a'))
    print(wordlemma.lemmatize('is',pos='v'))
    print(wordlemma.lemmatize('funnier',pos='a'))
    print(wordlemma.lemmatize('expected',pos='v'))
    print(wordlemma.lemmatize('fantasized',pos='v'))

if __name__ =="__main__":
    wordtokenization()
    print("\n")
    print("----------Word Lemmatization----------")
    wordlemmatization()
```

```
['Stemming', 'is', 'funnier', 'than', 'a', 'bummer', 'says', 'the', 'sushi', 'loving', 'comput


----------Word Lemmatization----------
car
walk
meeting
meet
good
be
funny
expect
fantasize
```

```
text = """Wikis are enabled by wiki
software, otherwise known as wiki engines. A wiki engine, being a
form of a content management system, differs from other web-based systems such as blog software, in
content is created without any defined owner or leader, and wikis have little inherent structure, al
 to emerge according to the needs of the users.[1] Wiki engines usually allow content to be written
  markup language and sometimes edited with the help of a rich-text editor.[2] There are dozens of d
   in use, both standalone and part of other software, such as bug tracking systems. Some wiki engin
   whereas others are proprietary. Some permit control over different functions (levels of access);
may permit access without enforcing access control. Other rules may be imposed to organize content.'

data = text.split('.')
for i in data:
  print(i)
```

```
Wikis are enabled by wiki
software, otherwise known as wiki engines
 A wiki engine, being a
form of a content management system, differs from other web-based systems such as blog softwar
content is created without any defined owner or leader, and wikis have little inherent structu
 to emerge according to the needs of the users
[1] Wiki engines usually allow content to be written using a simplified
  markup language and sometimes edited with the help of a rich-text editor
[2] There are dozens of different wiki engines
   in use, both standalone and part of other software, such as bug tracking systems
```

```
 Some wiki engines are open-source,
   whereas others are proprietary
 Some permit control over different functions (levels of access); for example, editing rights
 Others
may permit access without enforcing access control
 Other rules may be imposed to organize content
```

```
import nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize

str = "I love to study Natuaral Languague Processing in Python"
print("--------Sent tokenize-------------")
print(sent_tokenize(str))
print("\n")
print("--------Word tokenize-------------")
print(word_tokenize(str))
```

```
--------Sent tokenize-------------
['I love to study Natuaral Languague Processing in Python']


--------Word tokenize-------------
['I', 'love', 'to', 'study', 'Natuaral', 'Languague', 'Processing', 'in', 'Python']
```

```
import nltk
from nltk.tokenize import RegexpTokenizer
tk = RegexpTokenizer('\s+', gaps = True)
str = "I love to study Natuaral Languague Processing in Python. Wikis are enabled by wiki software,
tokens = tk.tokenize(str)
print(tokens)
```

```
['I', 'love', 'to', 'study', 'Natuaral', 'Languague', 'Processing', 'in', 'Python.', 'Wikis',
```

```
import nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
word_tokenize("can't")
```

```
['ca', "n't"]
```

```
from nltk.tokenize import TreebankWordTokenizer
tokenizer = TreebankWordTokenizer()
tokenizer.tokenize('Hello World.')
```

```
['Hello', 'World', '.']
```

```
tokenizer.tokenize("can't")
```

```
['ca', "n't"]
```

```
word_tokenize("Hello World.")
```

```
['Hello', 'World', '.']
```

```
from nltk.tokenize import WordPunctTokenizer
tokenizer = WordPunctTokenizer()
tokenizer.tokenize("Can't is a contradiction")
```

```
['Can', "'", 't', 'is', 'a', 'contradiction']
```

```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer("[\w']+")
tokenizer.tokenize("Can't is a contradiction")
```

```
["Can't", 'is', 'a', 'contradiction']
```

```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer("[\w ']+")
print(tokenizer.tokenize("abc@gmail.com"))
print(tokenizer.tokenize("xyz@rediffmail.com"))
print(tokenizer.tokenize("rjc@rjcollege.edu.in"))
```

```
['abc', 'gmail', 'com']
['xyz', 'rediffmail', 'com']
['rjc', 'rjcollege', 'edu', 'in']
```

```
address = ['abc@gmail.com', 'xyz@rediffmail.com', 'rjc@rjcollege.edu.in']
ls1 = []
for i in address:
  wr = tokenizer.tokenize(i)
  ls1.append(wr)

ls1
```

```
[['abc', 'gmail', 'com'],
 ['xyz', 'rediffmail', 'com'],
 ['rjc', 'rjcollege', 'edu', 'in']]
```

```
text4 = ('The students of MSc IT are using historical data for data for data warehousing projects')
from nltk.stem import PorterStemmer as ps
print(text4)
print("---------------------")
lst = []
for w in text4:
  rootWord=ps().stem(w)
  print(rootWord)
  lst.append(rootWord)
```

```
['The', 'students', 'of', 'MSc', 'IT', 'are', 'using', 'historical', 'data', 'for', 'data', 'f
---------------------
the
student
of
```

```
msc
it
are
use
histor
data
for
data
for
data
wareh
project
```

```
print(type(lst))
lst
```

```
<class 'list'>
['the',
 'student',
 'of',
 'msc',
 'it',
 'are',
 'use',
 'histor',
 'data',
 'for',
 'data',
 'for',
 'data',
 'wareh',
 'project']
```

```
words = ['Unexpected','disagreement','disagee','agreement','quirkiness','historical','canonical']
for w in words:
  stemPrint = ps().stem(w)
  print(w," -Stem- ", stemPrint)
```

```
Unexpected  -Stem-  unexpect
disagreement  -Stem-  disagr
disagee  -Stem-  disage
agreement  -Stem-  agreement
quirkiness  -Stem-  quirki
historical  -Stem-  histor
canonical  -Stem-  canon
```

```
import nltk
nltk.__file__
nltk.download('popular')
nltk.download('gutenberg')
from nltk.corpus import gutenberg
```

```
[nltk_data] Downloading collection 'popular'
[nltk_data]    |
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Package cmudict is already up-to-date!
[nltk_data]    | Downloading package gazetteers to /root/nltk_data...
[nltk_data]    |   Package gazetteers is already up-to-date!
[nltk_data]    | Downloading package genesis to /root/nltk_data...
```

```
[nltk_data]    |   Package genesis is already up-to-date!
[nltk_data]    | Downloading package gutenberg to /root/nltk_data...
[nltk_data]    |   Package gutenberg is already up-to-date!
[nltk_data]    | Downloading package inaugural to /root/nltk_data...
[nltk_data]    |   Package inaugural is already up-to-date!
[nltk_data]    | Downloading package movie_reviews to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package movie_reviews is already up-to-date!
[nltk_data]    | Downloading package names to /root/nltk_data...
[nltk_data]    |   Package names is already up-to-date!
[nltk_data]    | Downloading package shakespeare to /root/nltk_data...
[nltk_data]    |   Package shakespeare is already up-to-date!
[nltk_data]    | Downloading package stopwords to /root/nltk_data...
[nltk_data]    |   Package stopwords is already up-to-date!
[nltk_data]    | Downloading package treebank to /root/nltk_data...
[nltk_data]    |   Package treebank is already up-to-date!
[nltk_data]    | Downloading package twitter_samples to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package twitter_samples is already up-to-date!
[nltk_data]    | Downloading package omw to /root/nltk_data...
[nltk_data]    |   Package omw is already up-to-date!
[nltk_data]    | Downloading package omw-1.4 to /root/nltk_data...
[nltk_data]    |   Package omw-1.4 is already up-to-date!
[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]    |   Package wordnet2021 is already up-to-date!
[nltk_data]    | Downloading package wordnet31 to /root/nltk_data...
[nltk_data]    |   Package wordnet31 is already up-to-date!
[nltk_data]    | Downloading package wordnet_ic to /root/nltk_data...
[nltk_data]    |   Package wordnet_ic is already up-to-date!
[nltk_data]    | Downloading package words to /root/nltk_data...
[nltk_data]    |   Package words is already up-to-date!
[nltk_data]    | Downloading package maxent_ne_chunker to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package maxent_ne_chunker is already up-to-date!
[nltk_data]    | Downloading package punkt to /root/nltk_data...
[nltk_data]    |   Package punkt is already up-to-date!
[nltk_data]    | Downloading package snowball_data to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package snowball_data is already up-to-date!
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Package averaged_perceptron_tagger is already up-
[nltk_data]    |       to-date!
[nltk_data]    |
[nltk_data]  Done downloading collection popular
[nltk_data] Downloading package gutenberg to /root/nltk_data...
[nltk_data]   Package gutenberg is already up-to-date!
```

```
import nltk
from nltk import sent_tokenize
from nltk import word_tokenize

text = gutenberg.raw('austen-emma.txt')
text
```

```
'[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER I\n\n\nEmma Woodhouse, handsome, clever, a
nd rich, with a comfortable home\nand happy disposition, seemed to unite some of the best ble
ssings\nof existence; and had lived nearly twenty-one years in the world\nwith very little to
distress or vex her.\n\nShe was the youngest of the two daughters of a most affectionate,\nin
```

```
textBlob = '''Emma Woodhouse, handsome, clever, and rich, with a comfortable home
and happy disposition, seemed to unite some of the best blessings
of existence; and had lived nearly twenty-one years in the world
with very little to distress or vex her.'''.split()
```

```
both daughters,\nbut particularly of Emma.  Between  them  it was more the intimacy\nof siste
```

```
from nltk.stem import PorterStemmer as ps
for k in textBlob:
  wordStem = ps().stem(k)
  print(k," -Stem- ", wordStem)
```

```
Emma  -Stem-  emma
Woodhouse,  -Stem-  woodhouse,
handsome,  -Stem-  handsome,
clever,  -Stem-  clever,
and  -Stem-  and
rich,  -Stem-  rich,
with  -Stem-  with
a  -Stem-  a
comfortable  -Stem-  comfort
home  -Stem-  home
and  -Stem-  and
happy  -Stem-  happi
disposition,  -Stem-  disposition,
seemed  -Stem-  seem
to  -Stem-  to
unite  -Stem-  unit
some  -Stem-  some
of  -Stem-  of
the  -Stem-  the
best  -Stem-  best
blessings  -Stem-  bless
of  -Stem-  of
existence;  -Stem-  existence;
and  -Stem-  and
had  -Stem-  had
lived  -Stem-  live
nearly  -Stem-  nearli
twenty-one  -Stem-  twenty-on
years  -Stem-  year
in  -Stem-  in
the  -Stem-  the
world  -Stem-  world
with  -Stem-  with
very  -Stem-  veri
little  -Stem-  littl
to  -Stem-  to
distress  -Stem-  distress
or  -Stem-  or
vex  -Stem-  vex
her.  -Stem-  her.
```

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()

tex = '''Emma Woodhouse, handsome, clever, and rich, with a comfortable home
```

```
and happy disposition, seemed to unite some of the best blessings
of existence; and had lived nearly twenty-one years in the world
with very little to distress or vex her.'''
word_list = nltk.word_tokenize(tex)
print(word_list)
```

```
['Emma', 'Woodhouse', ',', 'handsome', ',', 'clever', ',', 'and', 'rich', ',', 'with', 'a', 'c
```

```
for w in word_list:
  lem = lemmatizer.lemmatize(w)
  print(lem)
```

```
Emma
Woodhouse
,
handsome
,
clever
,
and
rich
,
with
a
comfortable
home
and
happy
disposition
,
seemed
to
unite
some
of
the
best
blessing
of
existence
;
and
had
lived
nearly
twenty-one
year
in
the
world
with
very
little
to
distress
or
vex
her
.
```

Q. Perform Tokenization, Stemming, Lemmatization and POS Tagging for a textblob

```
!pip install textblob
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/sim
Requirement already satisfied: textblob in /usr/local/lib/python3.7/dist-packages (0.15.3)
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.7/dist-packages (from textb
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from nltk>=3.1-
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from nltk>=3.
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from nltk>=3.1
```

```
import textblob
from textblob import TextBlob
```

```
text = '''Emma Woodhouse, handsome, clever, and rich, with a comfortable home
and happy disposition, seemed to unite some of the best blessings
of existence; and had lived nearly twenty-one years in the world
with very little to distress or vex her.'''
```

```
TextBlob(text).words
```

```
WordList(['Emma', 'Woodhouse', 'handsome', 'clever', 'and', 'rich', 'with', 'a',
'comfortable', 'home', 'and', 'happy', 'disposition', 'seemed', 'to', 'unite', 'some', 'of',
'the', 'best', 'blessings', 'of', 'existence', 'and', 'had', 'lived', 'nearly', 'twenty-one',
'years', 'in', 'the', 'world', 'with', 'very', 'little', 'to', 'distress', 'or', 'vex',
'her'])
```

```
import nltk
from nltk import sent_tokenize
from nltk import word_tokenize
```

```
tokens_words = nltk.word_tokenize(text)
print(tokens_words)
```

```
['Emma', 'Woodhouse', ',', 'handsome', ',', 'clever', ',', 'and', 'rich', ',', 'with', 'a', 'c
```

```
from nltk.stem import PorterStemmer
```

```
ps = PorterStemmer()
word = ("civilization")
ps.stem(word)
```

```
'civil'
```

```
from nltk.stem.snowball import SnowballStemmer
```

```
stemmer = SnowballStemmer(language = "english")
word = "civilization"
stemmer.stem(word)
```

```
'civil'
```

```
import nltk
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
```

```
# Lemmatize single word

print(lemmatizer.lemmatize("workers"))
print(lemmatizer.lemmatize("beeches"))
```

```
worker
beech
```

```
text = "Let’s lemmatize a simple sentence. We first tokenize the sentence into words using nltk.word
word_list = nltk.word_tokenize(text)
print(word_list)
```

```
['Let', '’', 's', 'lemmatize', 'a', 'simple', 'sentence', '.', 'We', 'first', 'tokenize', 'the
```

```
lemmatized_output = ' '.join([lemmatizer.lemmatize(w) for w in word_list])
print(lemmatized_output)
```

```
Let ’ s lemmatize a simple sentence . We first tokenize the sentence into word using nltk.word
```

```
# pip install textblob

from textblob import TextBlob, Word
```

```
word = 'stripes'
w = Word(word)
w.lemmatize()
```

```
'stripe'
```

```
text = "The striped bats are hanging on their feet for best"
sent = TextBlob(text)
" ". join([w.lemmatize() for w in sent.words])
```

```
'The striped bat are hanging on their foot for best'
```

```
import nltk
from nltk import word_tokenize
```

```
text = "The striped bats are hanging on their feet for best"
tokens = nltk.word_tokenize(text)
print("Parts of Speech: ",nltk.pos_tag(tokens))
```

```
Parts of Speech:  [('The', 'DT'), ('striped', 'JJ'), ('bats', 'NNS'), ('are', 'VBP'), ('hangin
```

PRACTICAL 9

One-Hot Encoding, Bag of Words, N-grams, TF-IDF

One-Hot Encoding

```
import pandas as pd
from sklearn.feature_extraction import DictVectorizer

df = pd.DataFrame([['rick','young'],['phil','old']],columns=['name','age-group'])
print(df)
print("\n----By using Panda ----\n")
print(pd.get_dummies(df))

X = pd.DataFrame({'income': [100000,110000,90000,30000,14000,50000],
                  'country':['US', 'CAN', 'US', 'CAN', 'MEX', 'US'],
                  'race':['White', 'Black', 'Latino', 'White', 'White', 'Black']})

print("\n----By using Sikit-learn ----\n")
v = DictVectorizer()
qualitative_features = ['country']
X_qual = v.fit_transform(X[qualitative_features].to_dict('records'))
print(v.vocabulary_)
print(X_qual.toarray())
```

```
   name age-group
0  rick     young
1  phil       old

----By using Panda ----

   name_phil  name_rick  age-group_old  age-group_young
0          0          1              0                1
1          1          0              1                0

----By using Sikit-learn ----

{'country=US': 2, 'country=CAN': 0, 'country=MEX': 1}
[[0. 0. 1.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Bag of Words

```
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np

ngram_vectorizer = CountVectorizer(analyzer='char_wb', ngram_range=(2, 2), min_df=1)

# List is noumber of document here there are two document and each has only one word
# we are considering n_gram = 2 on chapracter unit leve
```

```
counts = ngram_vectorizer.fit_transform(['words', 'wprds'])

# this check weather the given word character is present in the above teo word which are documents h
ngram_vectorizer.get_feature_names() == ([' w', 'ds', 'or', 'pr', 'rd', 's ', 'wo', 'wp'])
print(counts.toarray().astype(int))
```

```
[[1 1 1 0 1 1 1 0]
 [1 1 0 1 1 1 0 1]]
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Functio
  warnings.warn(msg, category=FutureWarning)
```

N-Grams

```
from nltk import ngrams
sentence = 'this is a foo bar sentences and i want to ngramize it'
n = 4 # you can give 4, 5, 1 or any number less than sentences length
ngramsres = ngrams(sentence.split(), n)
for grams in ngramsres:
  print(grams)
```

```
('this', 'is', 'a', 'foo')
('is', 'a', 'foo', 'bar')
('a', 'foo', 'bar', 'sentences')
('foo', 'bar', 'sentences', 'and')
('bar', 'sentences', 'and', 'i')
('sentences', 'and', 'i', 'want')
('and', 'i', 'want', 'to')
('i', 'want', 'to', 'ngramize')
('want', 'to', 'ngramize', 'it')
```

TF-IDF

```
!pip3 install -U textblob
!python3 -m textblob.download_corpora
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/sim
Requirement already satisfied: textblob in /usr/local/lib/python3.7/dist-packages (0.15.3)
Collecting textblob
  Downloading textblob-0.17.1-py2.py3-none-any.whl (636 kB)
     |████████████████████████████████| 636 kB 4.0 MB/s
Requirement already satisfied: nltk>=3.1 in /usr/local/lib/python3.7/dist-packages (from textb
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from nltk>=3.
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from nltk>=3.1
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.7/dist-packages (from
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from nltk>=3.1-
Installing collected packages: textblob
  Attempting uninstall: textblob
    Found existing installation: textblob 0.15.3
    Uninstalling textblob-0.15.3:
      Successfully uninstalled textblob-0.15.3
Successfully installed textblob-0.17.1
[nltk_data] Downloading package brown to /root/nltk_data...
[nltk_data]   Unzipping corpora/brown.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[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 conll2000 to /root/nltk_data...
[nltk_data]   Unzipping corpora/conll2000.zip.
[nltk_data] Downloading package movie_reviews to /root/nltk_data...
[nltk_data]   Unzipping corpora/movie_reviews.zip.
Finished.
```

```
from __future__ import division
from textblob import TextBlob
import math
```

```
text = 'tf idf, short form of term frequency, inverse document frequency'
text2 = 'is a numerical statistic that is intended to reflect how important'
text3 = 'a word is to a document in a collection or corpus'
```

```
def tf(word, blob):
       return blob.words.count(word) / len(blob.words)

def n_containing(word, bloblist):
    return 1 + sum(1 for blob in bloblist if word in blob)

def idf(word, bloblist):
    x = n_containing(word, bloblist)
    return math.log(len(bloblist) / (x if x else 1))

def tfidf(word, blob, bloblist):
   return tf(word, blob) * idf(word, bloblist)

blob = TextBlob(text)
blob2 = TextBlob(text2)
blob3 = TextBlob(text3)
bloblist = [blob, blob2, blob3]
tf_score = tf('short', blob)
idf_score = idf('short', bloblist)
tfidf_score = tfidf('short', blob, bloblist)
print("tf score for word short = "+ str(tf_score)+"\n")
print("idf score for word short = "+ str(idf_score)+"\n")
print("tf - idf score of word short = "+str(tfidf_score))
```

```
tf score for word short = 0.1

idf score for word short = 0.4054651081081644

tf - idf score of word short = 0.04054651081081644
```

PRACTICAL 10

Word Embedding

```
from gensim.models import Word2Vec
```

```
import numpy as np

sentences = [['drink','not','good'],
             ['felt','superb'],
             ['just','good','ambience'],
             ['bad','taste'],
             ['parking','problem'],
             ['fantastic','food']]
y = np.array([0,1,1,0,0,1])
```

```
model = Word2Vec(sentences, min_count=1,size=100)
```

```
WARNING:gensim.models.base_any2vec:under 10 jobs per worker: consider setting a smaller `batch
```

```
print(model)
```

```
Word2Vec(vocab=13, size=100, alpha=0.025)
```

```
words = list(model.wv.vocab)
print(words)
```

```
['drink', 'not', 'good', 'felt', 'superb', 'just', 'ambience', 'bad', 'taste', 'parking', 'pro
```

```
print(model['drink'])
print(model['fantastic'])
```

```
[-5.0050818e-04  2.4672786e-03 -7.9926918e-04 -4.3498487e-03
  8.9418690e-04 -2.7653528e-03  4.7444564e-04 -4.3859198e-03
 -1.1973906e-03  8.7700860e-04 -4.3210834e-03  4.5646173e-03
 -1.4640016e-03 -3.8874540e-03 -2.3239923e-03 -8.3687960e-04
  2.6024114e-03  1.8356381e-03  4.4276826e-03  3.3546719e-03
  1.5427787e-03 -2.9658033e-03  4.3613068e-03 -4.5249020e-03
  1.1769844e-04  2.3500354e-03  4.9169124e-03  4.6623610e-03
 -5.9600483e-04  4.2867553e-03  2.8040679e-03 -9.5843535e-04
  6.5640768e-04  4.4129933e-03 -2.8350889e-03 -1.7969299e-03
 -3.6473530e-03  3.1940362e-03  2.1483670e-03 -3.4526661e-03
  2.7186791e-03  3.9265989e-03 -2.0560392e-03 -6.0720218e-04
 -2.4248050e-03 -1.0135595e-03  8.0119917e-04  6.5606751e-04
 -5.9487724e-05  4.3260008e-03  3.3480325e-03 -4.9368972e-03
 -3.7443740e-03 -3.6723157e-03  4.3429574e-03  2.3899982e-03
 -6.5888424e-04  2.1906642e-03 -1.4874450e-03 -4.7439621e-03
  2.5053741e-04  4.5042736e-03 -4.3766606e-03 -2.0952218e-03
 -2.9876768e-03  3.9986446e-03 -2.5275715e-03 -4.2584146e-04
 -2.6159394e-03 -3.9486034e-04  2.8907224e-03 -4.7935704e-03
 -1.7911082e-04 -3.0656364e-03 -4.5300885e-03  4.7503825e-04
 -5.9390627e-04 -3.5241202e-03 -2.2918228e-03 -3.6969481e-03
 -1.8111392e-03  2.5434752e-03  3.4643838e-03 -1.3447943e-03
  9.7135972e-04  1.0736240e-03 -3.1336928e-03 -2.0245414e-03
  4.3270946e-03 -2.9920582e-03 -7.2518102e-04  3.8299570e-03
  1.1019234e-03 -1.0992186e-03  2.1397523e-03  1.8650386e-03
 -4.9404930e-03  9.4359642e-04 -2.3236845e-03 -2.3845474e-03]
[ 3.2091551e-03  4.8351043e-04  2.3154723e-03  2.3536545e-03
  2.0127015e-03  2.8162207e-03 -3.0615588e-03  9.9767069e-04
  3.1230166e-03  2.6377521e-03  4.6836352e-03 -1.2763324e-03
 -3.9963010e-03 -3.1762065e-03 -2.1095972e-03  4.8815594e-03
  4.6088551e-03  1.2343933e-03 -4.4568647e-03 -2.7567144e-03
```

```
 -2.0010697e-03  2.4485593e-03 -2.5457975e-03  3.0205532e-03
 -2.1982386e-03  7.2195876e-04  5.1726896e-04 -2.7886149e-03
 -9.9975057e-04 -4.2017037e-03  4.3885307e-03  2.0536608e-03
 -1.3098851e-03 -1.2111312e-03 -4.3849843e-03 -1.3156281e-03
  1.0674889e-03 -8.7614125e-04 -8.3966099e-04  2.5931298e-04
  3.4646194e-03 -6.0989312e-04 -1.1076453e-03  8.4693060e-04
 -3.9437166e-03 -2.3501546e-03 -4.2097187e-03 -2.9950307e-03
  1.6589048e-03 -4.9967770e-03  1.6021567e-03  2.5338894e-03
  3.5506985e-03  3.6188818e-03  3.7043677e-03  1.1846009e-03
  4.9893567e-03 -4.2518871e-03  3.9963713e-03  9.6573582e-04
 -2.3562626e-03  3.3991094e-04  1.2320660e-03  3.1653976e-03
 -2.6492898e-03 -1.9942648e-03 -3.6559079e-03  4.2768288e-03
 -4.5010503e-03 -4.0102550e-03  3.0403894e-03  4.6649147e-03
 -4.1440446e-03  4.8272875e-03 -4.7174515e-03  3.4112388e-03
  3.5911656e-03  1.6063412e-03 -6.5607886e-04  4.1648536e-03
  4.0290840e-03 -3.9226939e-03  9.0192603e-05  1.7243662e-03
  4.7389958e-03 -2.9569168e-03  6.3476537e-04 -3.6375304e-03
  5.4094032e-04  2.8263549e-03 -9.1645046e-04 -3.0580326e-03
  4.7163079e-03  4.5088520e-03  3.9135879e-03 -2.8747665e-03
  4.0506413e-03  2.0309053e-03 -1.4043089e-03 -3.9382428e-03]
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to de
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: DeprecationWarning: Call to de
```

```
means_0 = np.mean(model[sentences[0]],axis=0)
means = []
for i in sentences :
    row_means = np.mean(model[i],axis=0)
    means.append(row_means)
means = np.array(means)
X = means
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: DeprecationWarning: Call to de
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: DeprecationWarning: Call to de
  after removing the cwd from sys.path.
```

```
from sklearn.ensemble import RandomForestClassifier
model_rf = RandomForestClassifier(random_state=1211,
                                  n_estimators=100,oob_score=True)
model_rf.fit( X , y )
test_sentences = [['bad','food'],['just','fantastic']]
test_means = []
for i in test_sentences :
    row_means = np.mean(model[i],axis=0)
    test_means.append(row_means)
num_test_means = np.array(test_means)
X_test = num_test_means
y_pred = model_rf.predict(X_test)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:8: DeprecationWarning: Call to de
```

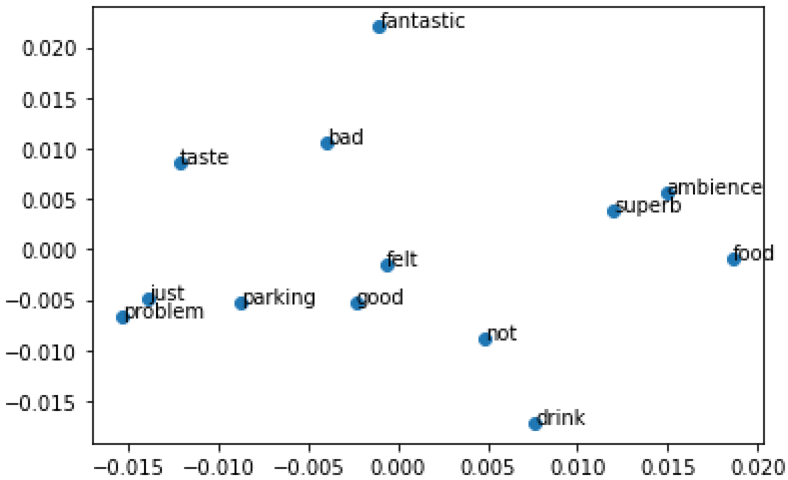
```
model.save('model.bin')
```

```
new_model = Word2Vec.load('model.bin')
print(new_model)
```

```
Word2Vec(vocab=13, size=100, alpha=0.025)
```

```
from sklearn.decomposition import PCA
from matplotlib import pyplot
X = model[model.wv.vocab]
pca = PCA(n_components=2)
result = pca.fit_transform(X)
pyplot.scatter(result[:, 0], result[:, 1])
vwords = list(model.wv.vocab)
for j, word in enumerate(vwords):
  pyplot.annotate(word,xy=(result[j, 0],result[j, 1]))
pyplot.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DeprecationWarning: Call to de
  This is separate from the ipykernel package so we can avoid doing imports until
```



other ipynb files

cfg - https://colab.research.google.com/drive/1y9yIwn6X8ZyN52y8tA6M2v8mDeoDFwOz?usp=sharing
text to speech - https://colab.research.google.com/drive/1mR6gv2Yr5IYJpb6T_MdQfFlwJcjkPZ-l?usp=sharing