Aim: Classification using Deep Neural Network

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```
from keras.datasets import imdb
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=10000)
data = np.concatenate((X_train, X_test), axis=0)
label = np.concatenate((y_train, y_test), axis=0)
```

In [3]:
```
print("Review is ",X_train[5])
print("Review is ",y_train[5])
```

```
Review is  [1, 778, 128, 74, 12, 630, 163, 15, 4, 1766, 7982, 1051, 2, 32, 8
5, 156, 45, 40, 148, 139, 121, 664, 665, 10, 10, 1361, 173, 4, 749, 2, 16, 3
804, 8, 4, 226, 65, 12, 43, 127, 24, 2, 10, 10]
Review is  0
```

In [4]:
```
vocab=imdb.get_word_index()
print(vocab)
```

```
{'fawn': 34701, 'tsukino': 52006, 'nunnery': 52007, 'sonja': 16816, 'van
i': 63951, 'woods': 1408, 'spiders': 16115, 'hanging': 2345, 'woody': 228
9, 'trawling': 52008, "hold's": 52009, 'comically': 11307, 'localized': 4
0830, 'disobeying': 30568, "'royale": 52010, "harpo's": 40831, 'canet': 5
2011, 'aileen': 19313, 'acurately': 52012, "diplomat's": 52013, 'rickma
n': 25242, 'arranged': 6746, 'rumbustious': 52014, 'familiarness': 52015,
"spider'": 52016, 'hahahah': 68804, "wood'": 52017, 'transvestism': 4083
3, "hangin'": 34702, 'bringing': 2338, 'seamier': 40834, 'wooded': 34703,
'bravora': 52018, 'grueling': 16817, 'wooden': 1636, 'wednesday': 16818,
"'prix": 52019, 'altagracia': 34704, 'circuitry': 52020, 'crotch': 11585,
'busybody': 57766, "tart'n'tangy": 52021, 'burgade': 14129, 'thrace': 520
23, "tom's": 11038, 'snuggles': 52025, 'francesco': 29114, 'complainers':
52027, 'templarios': 52125, '272': 40835, '273': 52028, 'zaniacs': 52130,
'275': 34706, 'consenting': 27631, 'snuggled': 40836, 'inanimate': 15492,
'uality': 52030, 'bronte': 11926, 'errors': 4010, 'dialogs': 3230, "yomad
a's": 52031, "madman's": 34707, 'dialoge': 30585, 'usenet': 52033, 'video
drome': 40837, "kid'": 26338, 'pawed': 52034, "'girlfriend'": 30569, "'pl
easure": 52035, "'reloaded'": 52036, "kazakos'": 40839, 'rocque': 52037,
'mailings': 52038, 'brainwashed': 11927, 'mcanally': 16819, "tom''": 5203
```

In [5]:
```
X_train.shape
```

Out[5]:
```
(25000,)
```

In [6]:
```
X_test.shape
```

Out[6]:
```
(25000,)
```

In [7]:
```
y_train
```

Out[7]:
```
array([1, 0, 0, ..., 0, 1, 0], dtype=int64)
```

In [8]:
```
y_test
```

Out[8]:
```
array([0, 1, 1, ..., 0, 0, 0], dtype=int64)
```

In [9]:
```
def vectorize(sequences, dimension = 10000):
  # Create an all-zero matrix of shape (len(sequences), dimension)
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results
```

In [10]:
```
test_x = data[:10000]
test_y = label[:10000]
train_x = data[10000:]
train_y = label[10000:]
```

In [11]:
```
test_y
```

Out[11]:
```
array([1, 0, 0, ..., 1, 0, 0], dtype=int64)
```

In [12]:
```
print("Label:", label[0])
```

```
Label: 1
```

In [13]:
```
print(data[0])
```

```
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 3
6, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480, 284, 5, 150, 4, 17
2, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4, 1
92, 50, 16, 6, 147, 2025, 19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12,
16, 43, 530, 38, 76, 15, 13, 1247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2,
5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2223, 5244, 16, 480, 66, 3785, 33, 4,
130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25, 1415, 33, 6, 22, 12,
215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 5952, 15, 256, 4, 2,
7, 3766, 5, 723, 36, 71, 43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 1029, 13,
104, 88, 4, 381, 15, 297, 98, 32, 2071, 56, 26, 141, 6, 194, 7486, 18, 4, 22
6, 22, 21, 134, 476, 26, 480, 5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25,
104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 1
5, 16, 5345, 19, 178, 32]
```

In [14]:
```
index = imdb.get_word_index()
reverse_index = dict([(value, key) for (key, value) in index.items()])
decoded = " ".join( [reverse_index.get(i - 3, "#") for i in data[0]] )
print(decoded)
```

```
# this film was just brilliant casting location scenery story direction ever
yone's really suited the part they played and you could just imagine being t
here robert # is an amazing actor and now the same being director # father c
ame from the same scottish island as myself so i loved the fact there was a
real connection with this film the witty remarks throughout the film were gr
eat it was just brilliant so much that i bought the film as soon as it was r
eleased for # and would recommend it to everyone to watch and the fly fishin
g was amazing really cried at the end it was so sad and you know what they s
ay if you cry at a film it must have been good and this definitely was also
# to the two little boy's that played the # of norman and paul they were jus
t brilliant children are often left out of the # list i think because the st
ars that play them all grown up are such a big profile for the whole film bu
t these children are amazing and should be praised for what they have done d
on't you think the whole story was so lovely because it was true and was som
eone's life after all that was shared with us all
```

In [15]:
```
index
```

Out[15]:
```
{'fawn': 34701,
 'tsukino': 52006,
 'nunnery': 52007,
 'sonja': 16816,
 'vani': 63951,
 'woods': 1408,
 'spiders': 16115,
 'hanging': 2345,
 'woody': 2289,
 'trawling': 52008,
 "hold's": 52009,
 'comically': 11307,
 'localized': 40830,
 'disobeying': 30568,
 "'royale": 52010,
 "harpo's": 40831,
 'canet': 52011,
 'aileen': 19313,
 'acurately': 52012,
 "diplomat's": 52013
```

In [16]:
```
reverse_index
```

Out[16]:
```
{34701: 'fawn',
 52006: 'tsukino',
 52007: 'nunnery',
 16816: 'sonja',
 63951: 'vani',
 1408: 'woods',
 16115: 'spiders',
 2345: 'hanging',
 2289: 'woody',
 52008: 'trawling',
 52009: "hold's",
 11307: 'comically',
 40830: 'localized',
 30568: 'disobeying',
 52010: "'royale",
 40831: "harpo's",
 52011: 'canet',
 19313: 'aileen',
 52012: 'acurately',
```

In [17]:
```
decoded
```

Out[17]:
```
"# this film was just brilliant casting location scenery story direction eve
ryone's really suited the part they played and you could just imagine being
there robert # is an amazing actor and now the same being director # father
came from the same scottish island as myself so i loved the fact there was a
real connection with this film the witty remarks throughout the film were gr
eat it was just brilliant so much that i bought the film as soon as it was r
eleased for # and would recommend it to everyone to watch and the fly fishin
g was amazing really cried at the end it was so sad and you know what they s
ay if you cry at a film it must have been good and this definitely was also
# to the two little boy's that played the # of norman and paul they were jus
t brilliant children are often left out of the # list i think because the st
ars that play them all grown up are such a big profile for the whole film bu
t these children are amazing and should be praised for what they have done d
on't you think the whole story was so lovely because it was true and was som
eone's life after all that was shared with us all"
```

In [18]:
```
data = vectorize(data)
label = np.array(label).astype("float32")
```

In [19]:
```
# Creating train and test data set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data,label, test_size=0.2
```

In [20]:
```
X_train.shape
```

Out[20]:
```
(40000, 10000)
```

In [21]:
```
X_test.shape
```

Out[21]:
```
(10000, 10000)
```

In [22]:
```
from keras.utils import to_categorical
from keras import models
from keras import layers
```

In [23]:
```
model = models.Sequential()
# Input - Layer
model.add(layers.Dense(50, activation = "relu", input_shape=(10000, )))
# Hidden - Layers
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation = "relu")) #ReLU" stands for Rectified
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation = "relu"))
# Output- Layer
model.add(layers.Dense(1, activation = "sigmoid")) #adds another Dense layer
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                500050

 dropout (Dropout)           (None, 50)                0

 dense_1 (Dense)             (None, 50)                2550

 dropout_1 (Dropout)         (None, 50)                0

 dense_2 (Dense)             (None, 50)                2550

 dense_3 (Dense)             (None, 1)                 51

=================================================================
Total params: 505,201
Trainable params: 505,201
Non-trainable params: 0
_________________________________________________________________
```

In [24]:
```
import tensorflow as tf
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
```

In [25]:
```
model.compile(
 optimizer = "adam",
 loss = "binary_crossentropy",
 metrics = ["accuracy"]
)
```

In [26]:
```
results = model.fit(
 X_train, y_train,
 epochs= 2,
 batch_size = 500,
 validation_data = (X_test, y_test),
 callbacks=[callback]
)
```

```
Epoch 1/2
80/80 [==============================] - 5s 46ms/step - loss: 0.4119 - accur
acy: 0.8105 - val_loss: 0.2616 - val_accuracy: 0.8968
Epoch 2/2
80/80 [==============================] - 3s 38ms/step - loss: 0.2191 - accur
acy: 0.9158 - val_loss: 0.2548 - val_accuracy: 0.8989
```

In [27]:
```
print(np.mean(results.history["val_accuracy"]))
```

```
0.897849977016449
```

In [28]:
```
model.predict(X_test)
```

```
313/313 [==============================] - 1s 3ms/step
```

Out[28]:
```
array([[0.1319962 ],
       [0.9938973 ],
       [0.8705446 ],
       ...,
       [0.95068085],
       [0.9893535 ],
       [0.99133193]], dtype=float32)
```