PROJECT REPORT

SENTIMENTAL ANALYSIS OF **IMDb** MOVIE REVIEWS

BY

A.LOHIT - 180330004

K.SRI CHARITHA – 180330027

K.JAYATHI - 180330003

K.VAISHNAVI - 180330026

B.CHANDRIKA – 180330006

Table of contents:

1.Abstract :

Artificial intelligence (AI) is a relatively new branch of computer science. A tremendous amount of effort has been put into research associated with understanding biological systems, abstracting key principles of intelligent behaviour, and developing practical applications of AI since the year 2000. The ultimate goal of this science is reaching “Strong AI”. However, humanity must be careful with creation of artificial intelligence that is similar and, perhaps, to some extent, identical to the human being.

Today, the amount of data that is generated, by both humans and machines, far outpaces humans' ability to absorb, interpret, and make complex decisions based on that data. Artificial Intelligence forms the basis for all computer learning and is the future of all complex decision making.

It's clear that if we took AI away, our world would look vastly different in every way. As the current investments and research result in expanded and perfected uses of AI, we can expect the technology to become even more entangled into our daily existence, workplaces and society.

2.Acknowledgement :

All our team members efforts and dedication has made this project successful in achieving the goal. Dr.Marimganti Srinivas sir’s support and guidance made the implementation more easier and the progress of our task by all of us was perceptive.

3.Introduction :

SENTIMENT ANALYSIS ON IMDb MOVIE REVIEWS:

Sentiment analysis is a natural language processing problem where the intent of a movie is predicted using the reviews .

Here, a IMDB dataset containing around 25,000 reviews(good /bad)is taken and is trained and tested using different deep learning libraries

like tensor flow, keras, matplot lib(for plotting)etc. Here we use multi-layer perceptron model where it consists of 3 layers, Input layer for intialisation of data, hidden layer for computing the data and output layer to produce the results.

Tensorflow is an open source library especially used in neural networks for perception, prediction and creation of data .we can develop, train and build any complex

models using this platform .It is a math library and can perform different operations on multi-dimensional data arrays.

Keras is a neural network library which basically is designed for faster experimentation with the deep neural networks ,it is modular and extensible platform .This platform is user-friendly and allows us to try different ideas. it supports multiple back-end neural network computation.

4.Project Overview :

SENTIMENT ANALYSIS ON IMDb MOVIE REVIEWS:

we have framed this analysis as natural language accessing problem. The input layer consists of over 25,000 reviews, the output layer consists of the epoch values for every single data and gives the accuracy.

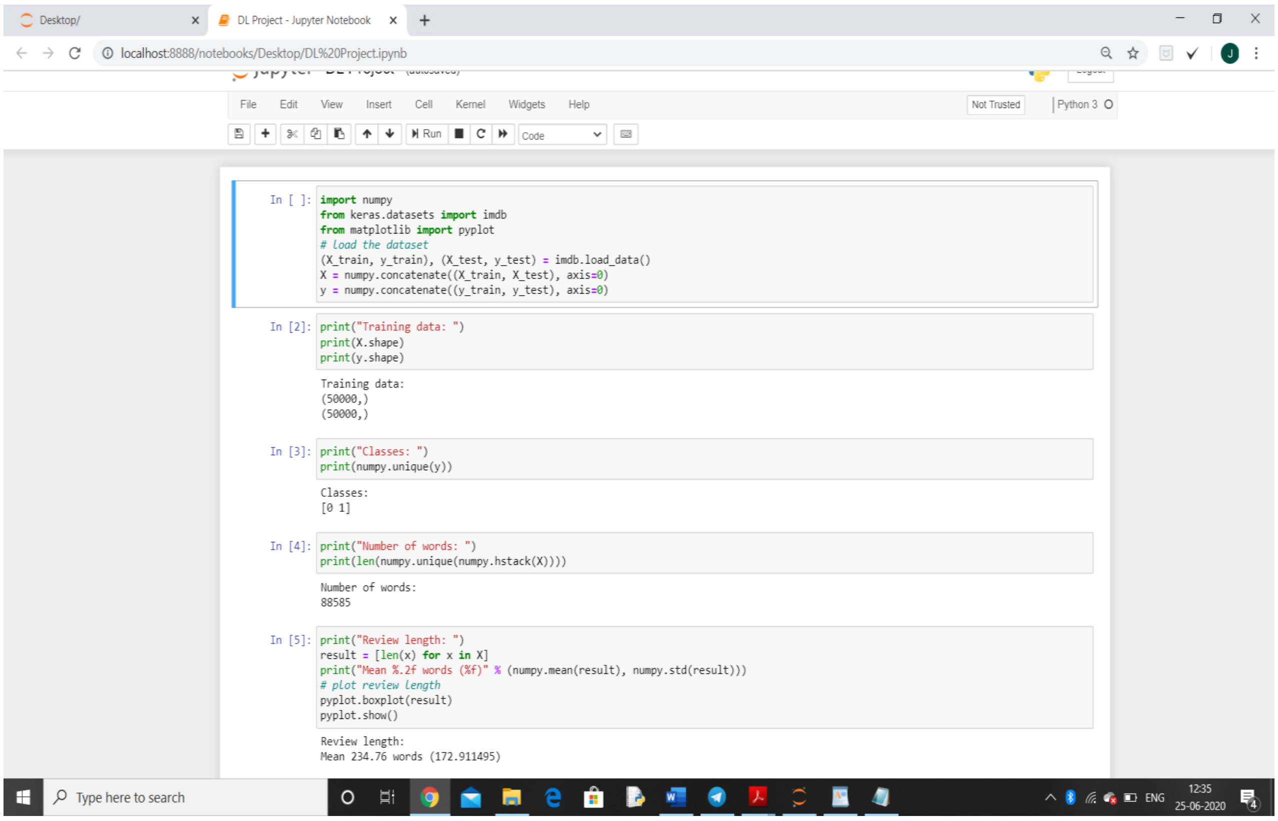
CODE:

At first ,we have trained the data, displayed the records and classifies the reviews as good(0) or bad(1).we have checked the average length of a review and plotted it.

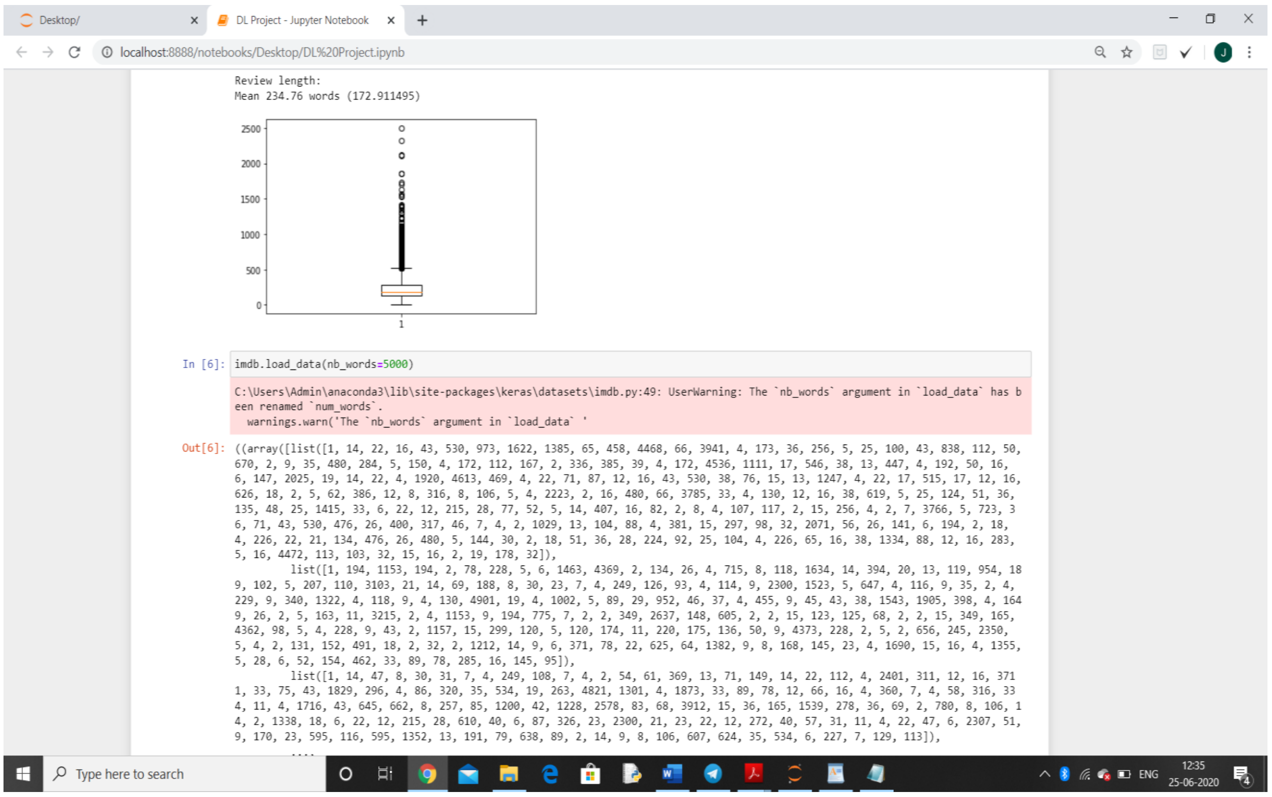
Using the keras library we have developed a multi-layer perceptron model with 250 hidden layers which compute the input layer data and 1 output layer which reviews the movie as good or bad

For final evaluation of the model, we have used the fit method by defining the batch size and number of epochs for the model.

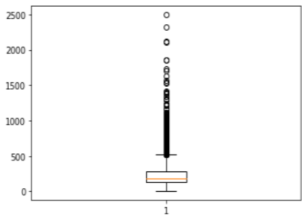
5.Tensorflow Code :



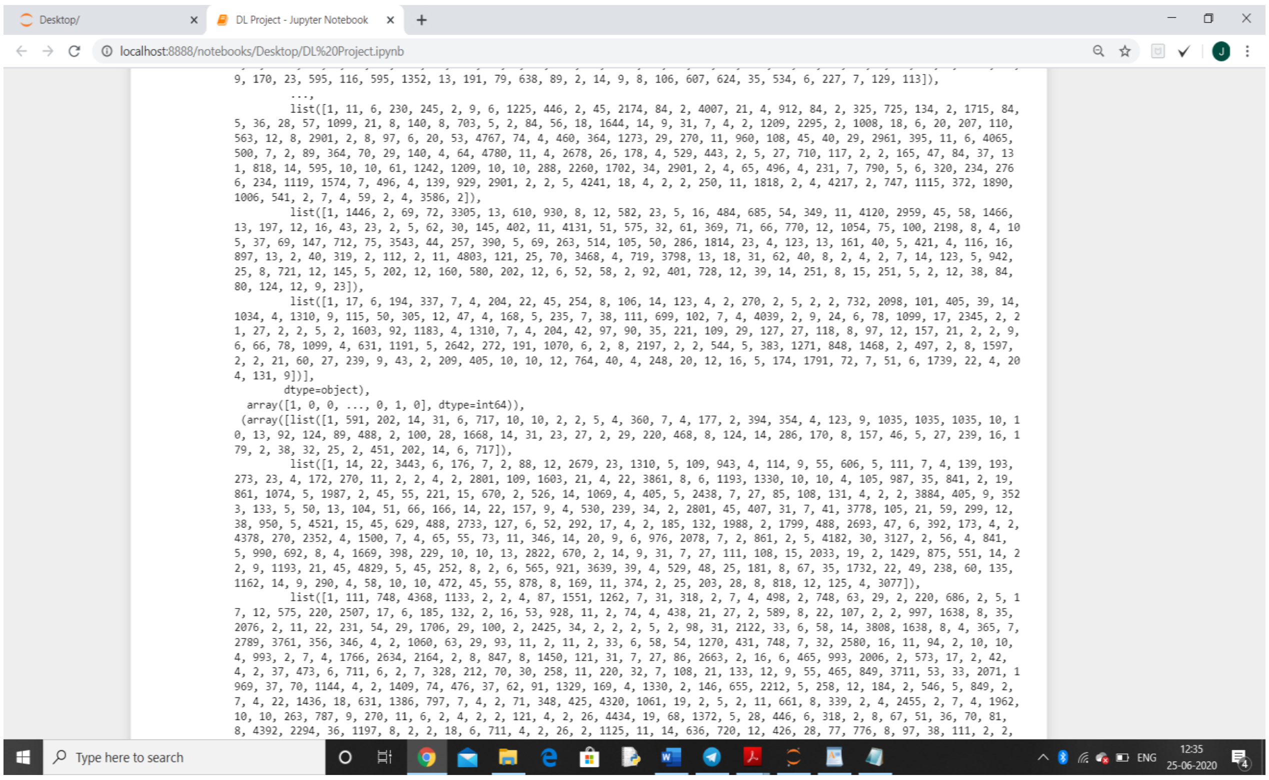
```
In [ ]: import numpy
        from keras.datasets import imdb
        from matplotlib import pyplot
        # load the dataset
        (X_train, y_train), (X_test, y_test) = imdb.load_data()
        X = numpy.concatenate((X_train, X_test), axis=0)
        y = numpy.concatenate((y_train, y_test), axis=0)

In [2]: print("Training data: ")
        print(X.shape)
        print(y.shape)

        Training data:
        (50000,)
        (50000,)

In [3]: print("Classes: ")
        print(numpy.unique(y))

        Classes:
        [0 1]

In [4]: print("Number of words: ")
        print(len(numpy.unique(numpy.hstack(X))))

        Number of words:
        88585

In [5]: print("Review length: ")
        result = [len(x) for x in X]
        print("Mean %.2f words (%f)" % (numpy.mean(result), numpy.std(result)))
        # plot review length
        pyplot.boxplot(result)
        pyplot.show()

        Review length:
        Mean 234.76 words (172.911495)
```



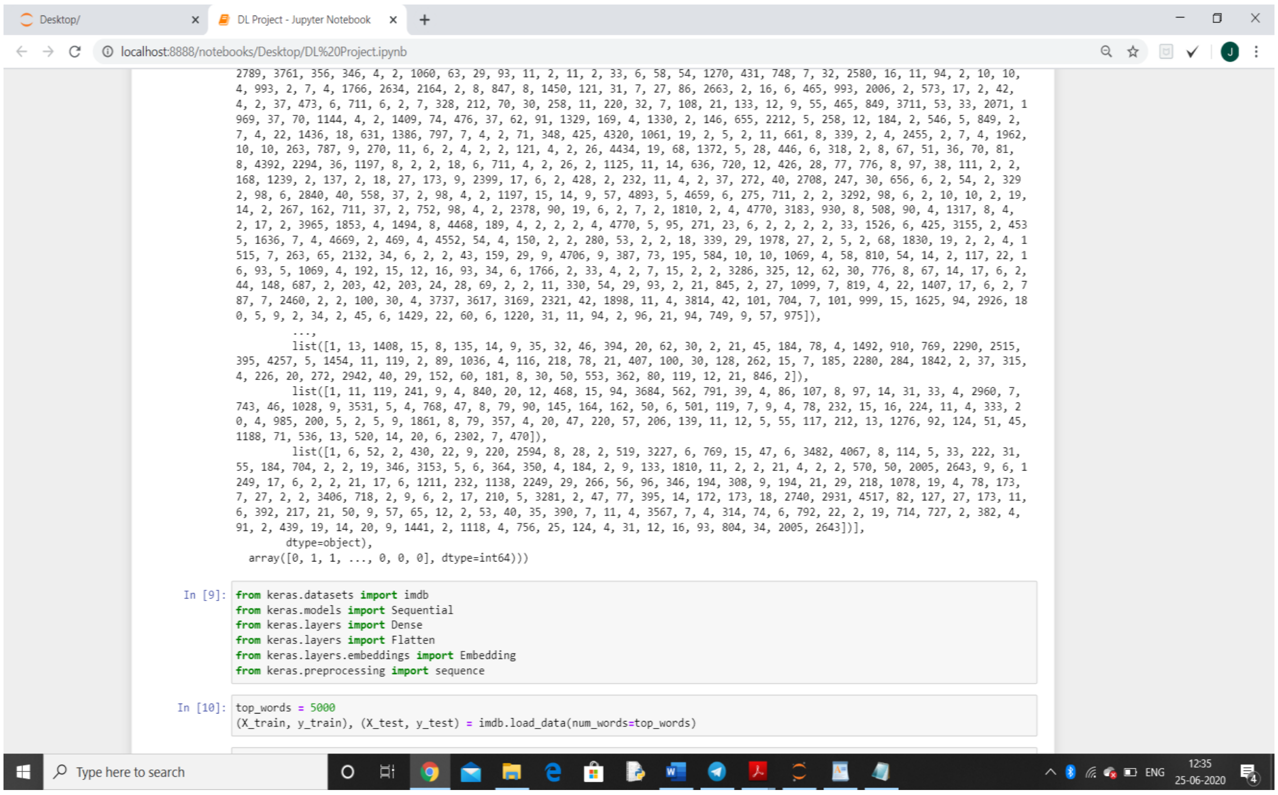
```
Review length:
Mean 234.76 words (172.911495)
```



```
In [6]: imdb.load_data(nb_words=5000)

C:\Users\Admin\anaconda3\lib\site-packages\keras\datasets\imdb.py:49: UserWarning: The `nb_words` argument in `load_data` has b
een renamed `num_words`.
  warnings.warn('The `nb_words` argument in `load_data` '

Out[6]: ((array([list([1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50,
670, 2, 9, 35, 480, 284, 5, 150, 4, 172, 112, 167, 2, 336, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4, 192, 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, 2, 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, 2, 15, 256, 4, 2, 7, 3766, 5, 723, 3
6, 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, 2, 18,
4, 226, 22, 21, 134, 476, 26, 480, 5, 144, 30, 2, 18, 51, 36, 28, 224, 92, 25, 104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283,
5, 16, 4472, 113, 103, 32, 15, 16, 2, 19, 178, 32]),
        list([1, 194, 1153, 194, 2, 78, 228, 5, 6, 1463, 4369, 2, 134, 26, 4, 715, 8, 118, 1634, 14, 394, 20, 13, 119, 954, 18
9, 102, 5, 207, 110, 3103, 21, 14, 69, 188, 8, 30, 23, 7, 4, 249, 126, 93, 4, 114, 9, 2300, 1523, 5, 647, 4, 116, 9, 35, 2, 4,
229, 9, 340, 1322, 4, 118, 9, 4, 130, 4901, 19, 4, 1002, 5, 89, 29, 952, 46, 37, 4, 455, 9, 45, 43, 38, 1543, 1905, 398, 4, 164
9, 26, 2, 5, 163, 11, 3215, 2, 4, 1153, 9, 194, 775, 7, 2, 2, 349, 2637, 148, 605, 2, 2, 15, 123, 125, 68, 2, 2, 15, 349, 165,
4362, 98, 5, 4, 228, 9, 43, 2, 1157, 15, 299, 120, 5, 120, 174, 11, 220, 175, 136, 50, 9, 4373, 228, 2, 5, 2, 656, 245, 2350,
5, 4, 2, 131, 152, 491, 18, 2, 32, 2, 1212, 14, 9, 6, 371, 78, 22, 625, 64, 1382, 9, 8, 168, 145, 23, 4, 1690, 15, 16, 4, 1355,
5, 28, 6, 52, 154, 462, 33, 89, 78, 285, 16, 145, 95]),
        list([1, 14, 47, 8, 30, 31, 7, 4, 249, 108, 7, 4, 2, 54, 61, 369, 13, 71, 149, 14, 22, 112, 4, 2401, 311, 12, 16, 371
1, 33, 75, 43, 1829, 296, 4, 86, 320, 35, 534, 19, 263, 4821, 1301, 4, 1873, 33, 89, 78, 12, 66, 16, 4, 360, 7, 4, 58, 316, 33
4, 11, 4, 1716, 43, 645, 662, 8, 257, 85, 1200, 42, 1228, 2578, 83, 68, 3912, 15, 36, 165, 1539, 278, 36, 69, 2, 780, 8, 106, 1
4, 2, 1338, 18, 6, 22, 12, 215, 28, 610, 40, 6, 87, 326, 23, 2300, 21, 23, 22, 12, 272, 40, 57, 31, 11, 4, 22, 47, 6, 2307, 51,
9, 170, 23, 595, 116, 595, 1352, 13, 191, 79, 638, 89, 2, 14, 9, 8, 106, 607, 624, 35, 534, 6, 227, 7, 129, 113]),
        ....
```


Desktop/
DL Project - Jupyter Notebook
localhost:8888/notebooks/Desktop/DL%20Project.ipynb
Type here to search
ENG
12:35
25-06-2020

```
9, 170, 23, 595, 116, 595, 1352, 13, 191, 79, 638, 89, 2, 14, 9, 8, 106, 607, 624, 35, 534, 6, 227, 7, 129, 113]),
         ...,
         list([1, 11, 6, 230, 245, 2, 9, 6, 1225, 446, 2, 45, 2174, 84, 2, 4007, 21, 4, 912, 84, 2, 325, 725, 134, 2, 1715, 84,
5, 36, 28, 57, 1099, 21, 8, 140, 8, 703, 5, 2, 84, 56, 18, 1644, 14, 9, 31, 7, 4, 2, 1209, 2295, 2, 1008, 18, 6, 20, 207, 110,
563, 12, 8, 2901, 2, 8, 97, 6, 20, 53, 4767, 74, 4, 460, 364, 1273, 29, 270, 11, 960, 108, 45, 40, 29, 2961, 395, 11, 6, 4065,
500, 7, 2, 89, 364, 70, 29, 140, 4, 64, 4780, 11, 4, 2678, 26, 178, 4, 529, 443, 2, 5, 27, 710, 117, 2, 2, 165, 47, 84, 37, 13
1, 818, 14, 595, 10, 10, 61, 1242, 1209, 10, 10, 288, 2260, 1702, 34, 2901, 2, 4, 65, 496, 4, 231, 7, 790, 5, 6, 320, 234, 276
6, 234, 1119, 1574, 7, 496, 4, 139, 929, 2901, 2, 2, 5, 4241, 18, 4, 2, 2, 250, 11, 1818, 2, 4, 4217, 2, 747, 1115, 372, 1890,
1006, 541, 2, 7, 4, 59, 2, 4, 3586, 2]),
         list([1, 1446, 2, 69, 72, 3305, 13, 610, 930, 8, 12, 582, 23, 5, 16, 484, 685, 54, 349, 11, 4120, 2959, 45, 58, 1466,
13, 197, 12, 16, 43, 23, 2, 5, 62, 30, 145, 402, 11, 4131, 51, 575, 32, 61, 369, 71, 66, 770, 12, 1054, 75, 100, 2198, 8, 4, 10
5, 37, 69, 147, 712, 75, 3543, 44, 257, 390, 5, 69, 263, 514, 105, 50, 286, 1814, 23, 4, 123, 13, 161, 40, 5, 421, 4, 116, 16,
897, 13, 2, 40, 319, 2, 112, 2, 11, 4803, 121, 25, 70, 3468, 4, 719, 3798, 13, 18, 31, 62, 40, 8, 2, 4, 2, 7, 14, 123, 5, 942,
25, 8, 721, 12, 145, 5, 202, 12, 160, 580, 202, 12, 6, 52, 58, 2, 92, 401, 728, 12, 39, 14, 251, 8, 15, 251, 5, 2, 12, 38, 84,
80, 124, 12, 9, 23]),
         list([1, 17, 6, 194, 337, 7, 4, 204, 22, 45, 254, 8, 106, 14, 123, 4, 2, 270, 2, 5, 2, 2, 732, 2098, 101, 405, 39, 14,
1034, 4, 1310, 9, 115, 50, 305, 12, 47, 4, 168, 5, 235, 7, 38, 111, 699, 102, 7, 4, 4039, 2, 9, 24, 6, 78, 1099, 17, 2345, 2, 2
1, 27, 2, 2, 5, 2, 1603, 92, 1183, 4, 1310, 7, 4, 204, 42, 97, 90, 35, 221, 109, 29, 127, 27, 118, 8, 97, 12, 157, 21, 2, 2, 9,
6, 66, 78, 1099, 4, 631, 1191, 5, 2642, 272, 191, 1070, 6, 2, 8, 2197, 2, 2, 544, 5, 383, 1271, 848, 1468, 2, 497, 2, 8, 1597,
2, 2, 21, 60, 27, 239, 9, 43, 2, 209, 405, 10, 10, 12, 764, 40, 4, 248, 20, 12, 16, 5, 174, 1791, 72, 7, 51, 6, 1739, 22, 4, 20
4, 131, 9])],
        dtype=object),
  array([1, 0, 0, ..., 0, 1, 0], dtype=int64)),
 (array([list([1, 591, 202, 14, 31, 6, 717, 10, 10, 2, 2, 5, 4, 360, 7, 4, 177, 2, 394, 354, 4, 123, 9, 1035, 1035, 1035, 10, 1
0, 13, 92, 124, 89, 488, 2, 100, 28, 1668, 14, 31, 23, 27, 2, 29, 220, 468, 8, 124, 14, 286, 170, 8, 157, 46, 5, 27, 239, 16, 1
79, 2, 38, 32, 25, 2, 451, 202, 14, 6, 717]),
         list([1, 14, 22, 3443, 6, 176, 7, 2, 88, 12, 2679, 23, 1310, 5, 109, 943, 4, 114, 9, 55, 606, 5, 111, 7, 4, 139, 193,
273, 23, 4, 172, 270, 11, 2, 2, 4, 2, 2801, 109, 1603, 21, 4, 22, 3861, 8, 6, 1193, 1330, 10, 10, 4, 105, 987, 35, 841, 2, 19,
861, 1074, 5, 1987, 2, 45, 55, 221, 15, 670, 2, 526, 14, 1069, 4, 405, 5, 2438, 7, 27, 85, 108, 131, 4, 2, 2, 3884, 405, 9, 352
3, 133, 5, 50, 13, 104, 51, 66, 166, 14, 22, 157, 9, 4, 530, 239, 34, 2, 2801, 45, 407, 31, 7, 41, 3778, 105, 21, 59, 299, 12,
38, 950, 5, 4521, 15, 45, 629, 488, 2733, 127, 6, 52, 292, 17, 4, 2, 185, 132, 1988, 2, 1799, 488, 2693, 47, 6, 392, 173, 4, 2,
4378, 270, 2352, 4, 1500, 7, 4, 65, 55, 73, 11, 346, 14, 20, 9, 6, 976, 2078, 7, 2, 861, 2, 5, 4182, 30, 3127, 2, 56, 4, 841,
5, 990, 692, 8, 4, 1669, 398, 229, 10, 10, 13, 2822, 670, 2, 14, 9, 31, 7, 27, 111, 108, 15, 2033, 19, 2, 1429, 875, 551, 14, 2
2, 9, 1193, 21, 45, 4829, 5, 45, 252, 8, 2, 6, 565, 921, 3639, 39, 4, 529, 48, 25, 181, 8, 67, 35, 1732, 22, 49, 238, 60, 135,
1162, 14, 9, 290, 4, 58, 10, 10, 472, 45, 55, 878, 8, 169, 11, 374, 2, 25, 203, 28, 8, 818, 12, 125, 4, 3077]),
         list([1, 111, 748, 4368, 1133, 2, 2, 4, 87, 1551, 1262, 7, 31, 318, 2, 7, 4, 498, 2, 748, 63, 29, 2, 220, 686, 2, 5, 1
7, 12, 575, 220, 2507, 17, 6, 185, 132, 2, 16, 53, 928, 11, 2, 74, 4, 438, 21, 27, 2, 589, 8, 22, 107, 2, 2, 997, 1638, 8, 35,
2076, 2, 11, 22, 231, 54, 29, 1706, 29, 100, 2, 2425, 34, 2, 2, 2, 5, 2, 98, 31, 2122, 33, 6, 58, 14, 3808, 1638, 8, 4, 365, 7,
2789, 3761, 356, 346, 4, 2, 1060, 63, 29, 93, 11, 2, 11, 2, 33, 6, 58, 54, 1270, 431, 748, 7, 32, 2580, 16, 11, 94, 2, 10, 10,
4, 993, 2, 7, 4, 1766, 2634, 2164, 2, 8, 847, 8, 1450, 121, 31, 7, 27, 86, 2663, 2, 16, 6, 465, 993, 2006, 2, 573, 17, 2, 42,
4, 2, 37, 473, 6, 711, 6, 2, 7, 328, 212, 70, 30, 258, 11, 220, 32, 7, 108, 21, 133, 12, 9, 55, 465, 849, 3711, 53, 33, 2071, 1
969, 37, 70, 1144, 4, 2, 1409, 74, 476, 37, 62, 91, 1329, 169, 4, 1330, 2, 146, 655, 2212, 5, 258, 12, 184, 2, 546, 5, 849, 2,
7, 4, 22, 1436, 18, 631, 1386, 797, 7, 4, 2, 71, 348, 425, 4320, 1061, 19, 2, 5, 2, 11, 661, 8, 339, 2, 4, 2455, 2, 7, 4, 1962,
10, 10, 263, 787, 9, 270, 11, 6, 2, 4, 2, 2, 121, 4, 2, 26, 4434, 19, 68, 1372, 5, 28, 446, 6, 318, 2, 8, 67, 51, 36, 70, 81,
8, 4392, 2294, 36, 1197, 8, 2, 2, 18, 6, 711, 4, 2, 26, 2, 1125, 11, 14, 636, 720, 12, 426, 28, 77, 776, 8, 97, 38, 111, 2, 2,
```

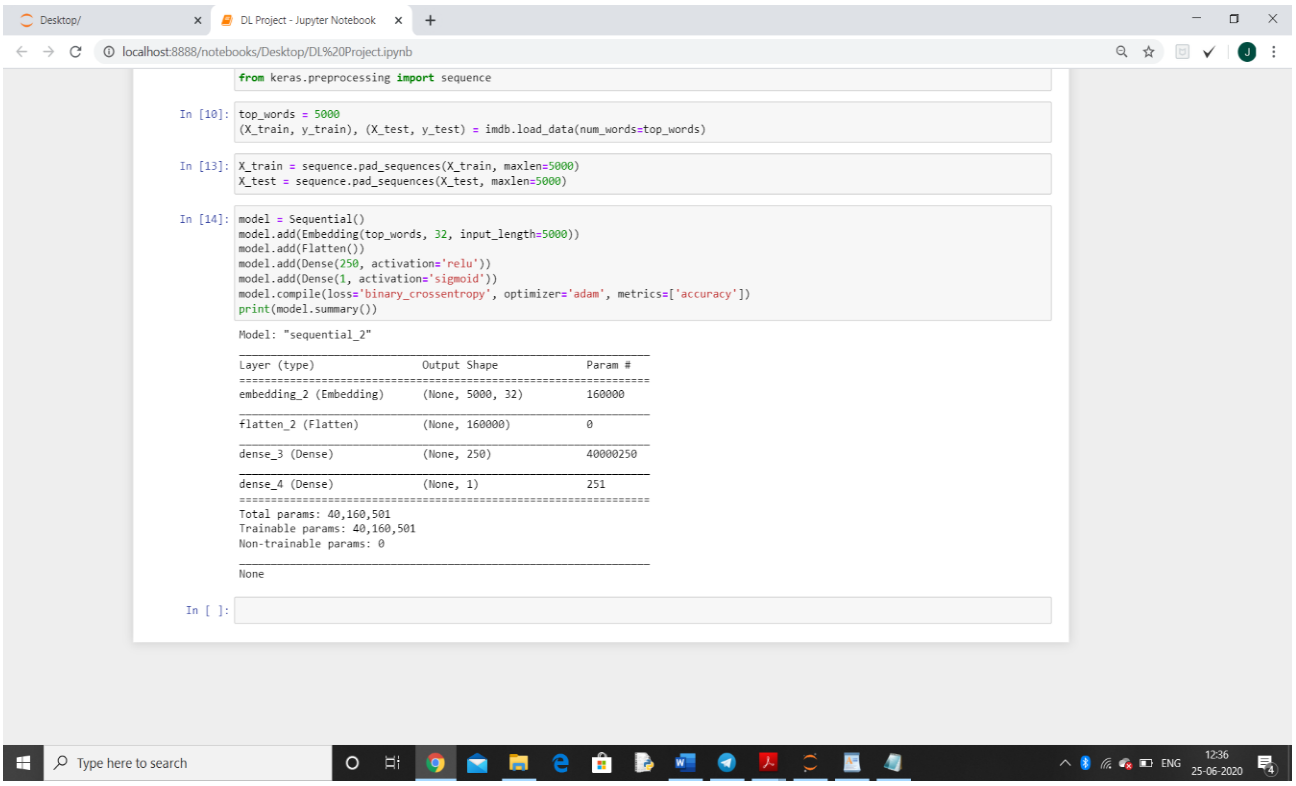

Desktop/
DL Project - Jupyter Notebook
localhost:8888/notebooks/Desktop/DL%20Project.ipynb
Type here to search
ENG
12:35
25-06-2020

```
2789, 3761, 356, 346, 4, 2, 1060, 63, 29, 93, 11, 2, 11, 2, 33, 6, 58, 54, 1270, 431, 748, 7, 32, 2580, 16, 11, 94, 2, 10, 10,
4, 993, 2, 7, 4, 1766, 2634, 2164, 2, 8, 847, 8, 1450, 121, 31, 7, 27, 86, 2663, 2, 16, 6, 465, 993, 2006, 2, 573, 17, 2, 42,
4, 2, 37, 473, 6, 711, 6, 2, 7, 328, 212, 70, 30, 258, 11, 220, 32, 7, 108, 21, 133, 12, 9, 55, 465, 849, 3711, 53, 33, 2071, 1
969, 37, 70, 1144, 4, 2, 1409, 74, 476, 37, 62, 91, 1329, 169, 4, 1330, 2, 146, 655, 2212, 5, 258, 12, 184, 2, 546, 5, 849, 2,
7, 4, 22, 1436, 18, 631, 1386, 797, 7, 4, 2, 71, 348, 425, 4320, 1061, 19, 2, 5, 2, 11, 661, 8, 339, 2, 4, 2455, 2, 7, 4, 1962,
10, 10, 263, 787, 9, 270, 11, 6, 2, 4, 2, 2, 121, 4, 2, 26, 4434, 19, 68, 1372, 5, 28, 446, 6, 318, 2, 8, 67, 51, 36, 70, 81,
8, 4392, 2294, 36, 1197, 8, 2, 2, 18, 6, 711, 4, 2, 26, 2, 1125, 11, 14, 636, 720, 12, 426, 28, 77, 776, 8, 97, 38, 111, 2, 2,
```

```
168, 1239, 2, 137, 2, 18, 27, 173, 9, 2399, 17, 6, 2, 428, 2, 232, 11, 4, 2, 37, 272, 40, 2708, 247, 30, 656, 6, 2, 54, 2, 329
2, 98, 6, 2840, 40, 558, 37, 2, 98, 4, 2, 1197, 15, 14, 9, 57, 4893, 5, 4659, 6, 275, 711, 2, 2, 3292, 98, 6, 2, 10, 10, 2, 19,
14, 2, 267, 162, 711, 37, 2, 752, 98, 4, 2, 2378, 90, 19, 6, 2, 7, 2, 1810, 2, 4, 4770, 3183, 930, 8, 508, 90, 4, 1317, 8, 4,
2, 17, 2, 3965, 1853, 4, 1494, 8, 4468, 189, 4, 2, 2, 2, 4, 4770, 5, 95, 271, 23, 6, 2, 2, 2, 2, 33, 1526, 6, 425, 3155, 2, 453
5, 1636, 7, 4, 4669, 2, 469, 4, 4552, 54, 4, 150, 2, 2, 280, 53, 2, 2, 18, 339, 29, 1978, 27, 2, 5, 2, 68, 1830, 19, 2, 2, 4, 1
515, 7, 263, 65, 2132, 34, 6, 2, 2, 43, 159, 29, 9, 4706, 9, 387, 73, 195, 584, 10, 10, 1069, 4, 58, 810, 54, 14, 2, 117, 22, 1
6, 93, 5, 1069, 4, 192, 15, 12, 16, 93, 34, 6, 1766, 2, 33, 4, 2, 7, 15, 2, 2, 3286, 325, 12, 62, 30, 776, 8, 67, 14, 17, 6, 2,
44, 148, 687, 2, 203, 42, 203, 24, 28, 69, 2, 2, 11, 330, 54, 29, 93, 2, 21, 845, 2, 27, 1099, 7, 819, 4, 22, 1407, 17, 6, 2, 7
87, 7, 2460, 2, 2, 100, 30, 4, 3737, 3617, 3169, 2321, 42, 1898, 11, 4, 3814, 42, 101, 704, 7, 101, 999, 15, 1625, 94, 2926, 18
0, 5, 9, 2, 34, 2, 45, 6, 1429, 22, 60, 6, 1220, 31, 11, 94, 2, 96, 21, 94, 749, 9, 57, 975]),
         ...,
         list([1, 13, 1408, 15, 8, 135, 14, 9, 35, 32, 46, 394, 20, 62, 30, 2, 21, 45, 184, 78, 4, 1492, 910, 769, 2290, 2515,
395, 4257, 5, 1454, 11, 119, 2, 89, 1036, 4, 116, 218, 78, 21, 407, 100, 30, 128, 262, 15, 7, 185, 2280, 284, 1842, 2, 37, 315,
4, 226, 20, 272, 2942, 40, 29, 152, 60, 181, 8, 30, 50, 553, 362, 80, 119, 12, 21, 846, 2]),
         list([1, 11, 119, 241, 9, 4, 840, 20, 12, 468, 15, 94, 3684, 562, 791, 39, 4, 86, 107, 8, 97, 14, 31, 33, 4, 2960, 7,
743, 46, 1028, 9, 3531, 5, 4, 768, 47, 8, 79, 90, 145, 164, 162, 50, 6, 501, 119, 7, 9, 4, 78, 232, 15, 16, 224, 11, 4, 333, 2
0, 4, 985, 200, 5, 2, 5, 9, 1861, 8, 79, 357, 4, 20, 47, 220, 57, 206, 139, 11, 12, 5, 55, 117, 212, 13, 1276, 92, 124, 51, 45,
1188, 71, 536, 13, 520, 14, 20, 6, 2302, 7, 470]),
         list([1, 6, 52, 2, 430, 22, 9, 220, 2594, 8, 28, 2, 519, 3227, 6, 769, 15, 47, 6, 3482, 4067, 8, 114, 5, 33, 222, 31,
55, 184, 704, 2, 2, 19, 346, 3153, 5, 6, 364, 350, 4, 184, 2, 9, 133, 1810, 11, 2, 2, 21, 4, 2, 2, 570, 50, 2005, 2643, 9, 6, 1
249, 17, 6, 2, 2, 21, 17, 6, 1211, 232, 1138, 2249, 29, 266, 56, 96, 346, 194, 308, 9, 194, 21, 29, 218, 1078, 19, 4, 78, 173,
7, 27, 2, 2, 3406, 718, 2, 9, 6, 2, 17, 210, 5, 3281, 2, 47, 77, 395, 14, 172, 173, 18, 2740, 2931, 4517, 82, 127, 27, 173, 11,
6, 392, 217, 21, 50, 9, 57, 65, 12, 2, 53, 40, 35, 390, 7, 11, 4, 3567, 7, 4, 314, 74, 6, 792, 22, 2, 19, 714, 727, 2, 382, 4,
91, 2, 439, 19, 14, 20, 9, 1441, 2, 1118, 4, 756, 25, 124, 4, 31, 12, 16, 93, 804, 34, 2005, 2643])],
        dtype=object),
  array([0, 1, 1, ..., 0, 0, 0], dtype=int64)))
```

```
In [9]: from keras.datasets import imdb
        from keras.models import Sequential
        from keras.layers import Dense
        from keras.layers import Flatten
        from keras.layers.embeddings import Embedding
        from keras.preprocessing import sequence
```

```
In [10]: top_words = 5000
         (X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
```


Desktop/
DL Project - Jupyter Notebook
localhost:8888/notebooks/Desktop/DL%20Project.ipynb
Type here to search
ENG
12:36
25-06-2020

```
from keras.preprocessing import sequence
```

```
In [10]: top_words = 5000
         (X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
```

```
In [13]: X_train = sequence.pad_sequences(X_train, maxlen=5000)
         X_test = sequence.pad_sequences(X_test, maxlen=5000)
```

```
In [14]: model = Sequential()
         model.add(Embedding(top_words, 32, input_length=5000))
         model.add(Flatten())
         model.add(Dense(250, activation='relu'))
         model.add(Dense(1, activation='sigmoid'))
         model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
         print(model.summary())
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_2 (Embedding)      (None, 5000, 32)          160000
_________________________________________________________________
flatten_2 (Flatten)          (None, 160000)            0
_________________________________________________________________
dense_3 (Dense)              (None, 250)               40000250
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 251
=================================================================
Total params: 40,160,501
Trainable params: 40,160,501
Non-trainable params: 0
_________________________________________________________________
None
```

```
In [ ]:
```

6.Final Result :

We discovered the IMDB sentiment analysis dataset for natural language processing.

we learned how to develop deep learning models for sentiment analysis including:

- How to load and review the IMDB dataset within Keras.
- How to develop a large neural network model for sentiment analysis.
- How to develop a one-dimensional convolutional neural network model for sentiment analysis.