Imitation Box

Prepared by
Abhishek Vyas(16IT146)

Under the supervision of
Mr. Ravi Patel

A Report Submitted to
Charotar University of Science and Technology
for Partial Fulfillment of the Requirements for the
Degree of Bachelor of Technology
in Information Technology
IT345 Software Group Project-II (5th sem)

Submitted at

DEPARTMENT OF INFORMATION TECHNOLOGY
Chandubhai S. Patel Institute of Technology
At: Changa, Dist: Anand – 388421
November 2018

CHARUSAT
CHAROTAR UNIVERSITY OF SCIENCE AND TECHNOLOGY

CERTIFICATE

This is to certify that the report entitled "**Imitation Box**" is a bonafied work carried out by **Mr. Abhishek V Vyas (16IT146)** under the guidance and supervision of **Mr. Ravi Patel** for the subject **Software Group Project-II(IT345)** of **5th** Semester of Bachelor of Technology in **Information Technology** at Faculty of Technology & Engineering – CHARUSAT, Gujarat.

To the best of my knowledge and belief, this work embodies the work of candidate himself, has duly been completed, and fulfills the requirement of the ordinance relating to the B.Tech. Degree of the University and is up to the standard in respect of content, presentation and language for being referred to the examiner.

Under supervision of,

Mr. Ravi Patel
Assistant Professor
Dept. of Information Technology
CSPIT, Changa, Gujarat.

Prof. Parth Shah
Head & Associate Professor
Department of Information Technology
CSPIT, Changa, Gujarat.

Chandubhai S Patel Institute of Technology

At: Changa, Ta. Petlad, Dist. Anand, PIN: 388 421. Gujarat

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

I am very thankful to God blessing helped us a lot to make our effort successful, Thank to my parents who inspired us to go ahead in my work, it is our pleasure to take this opportunity to make all those who directly or indirectly helped us in my project.

I are extremely grateful to Dr. Parth Shah, head of department of information technology for his excellent guidance, valuable suggestion and encouragement to carry out this project.

I extend my sincere thanks to the project guide Mr. Ravi Patel for his constructive criticism valuable suggestion and guidance to bring out the best from this project.

Finally, I would like to express my deep sense of gratitude to my entire classmate and many other friends of us who have supported and enriched me by sharing their ideas and storing out our doubts having discussion with us. I thank one and all.

ABSTRACT

This Project comes up with the applications of f Natural Language Processing techniques and Recurrent Neural Networks for detecting the 'fake news', that is, misleading news stories that comes from the non-reputable sources. Only by building a model based on a count vectorizer (using word tallies) or a (Term Frequency Inverse Document Frequency) tfidf matrix, (word tallies relative to how often they're used in other articles in your dataset) can only get you so far. But these models do not consider the important qualities like word ordering and context. It is very possible that two articles that are similar in their word count will be completely different in their meaning. The data science community has responded by taking actions against the problem. There is a Kaggle competition called as the "Fake News Challenge" and Facebook is employing AI to filter fake news stories out of users' feeds. Combatting the fake news is a classic text classification project with a straight forward proposition. Is it possible for you to build a model that can differentiate between "Real "news and "Fake" news? So a proposed work on assembling a dataset of both fake and real news and employ a Naive Bayes classifier in order to create a model to classify an article into fake or real based on its words and phrases.

CHAPTER:-1

INTRODUCTION

1.1 The Pilot

- In this project I investigate whether machine learning methods and deep learning methods can help to detect fake news.
- The New York Times narrowly defines fake news as "a made-up story with an intention to deceive".
- As a further matter, fake news is often the result of an interest to mislead people for some secondary gain.
- For example, spreading false information about opponents can strengthen a certain political interest.
- Experts claim that made-up stories about Hillary Clinton were one of the reasons why she lost against Donald Trump in the last US presidential election.
- Misinformation and influence campaigns have been reported in other countries as well.
- Reportedly Russia attempted to influence the UK referendum about leaving the European Union by posting more than 45,000 messages on Twitter.
- The trend of wide-spread use of mobile devices lead to an increasing proportion of the population receiving their news online, often in near real-time and often from novel news sources.
- As a consequence, there is a high risk of falling victim to fake news today.
- Many people are not aware that the news they read is fabricated as it is often a tedious and complex task to determine the veracity of a news story.
- Indeed, this can be a challenging task even for trained fact-checkers.
- In a research poll made in the US, about 64% of the adults answered that made-up news stories cause "a great deal of confusion about the basic facts of current events".
- According to experts, the larger problem caused by fake news is that people start doubting real news.

1.2 Project Summary

- In the project Imitation Box, we explore the application of Natural Language Processing techniques and Recurrent Neural Networks to identify when a news source may be producing fake news.
- We use a corpus of labeled real and fake new articles to build a classifier that can make decisions about information based on the content from the corpus.
- We use a text classification approach, using four different classification models, and analyze the results. The best performing model was the LSTM implementation.

1.3 Purpose

- The model focuses on identifying fake news sources, based on multiple articles originating from a source. Once a source is labeled as a producer of fake news, we can predict with high confidence that any future articles from that source will also be fake news. Focusing on sources widens our article misclassification tolerance, because we then have multiple data points coming from each source.

1.4 Scope

- This is project prepared Imitation Box to classify and detect fake news on the social media.
- It can be used to detect any social media news i.e. from Buzzfeed, Twitter, Facebook, Reddit or another.

1.5 Objective

❖ In this project, I investigate the efficiency of representing news articles using Natural Language Processing techniques and Recurrent Neural Networks for the problem of detecting the relationship between a headline and its corresponding body text. Studying this, I train and evaluate a classifier on the dataset provided by Kaggle. More specifically, I aim for the classifier to accomplish the following tasks:
  • Determine whether a news article headline and the corresponding body text is related to each other or not. That is, classify a headline and article pair into one of the two categories unrelated or related.
  • Given a headline and an article that is related to each other, classify the stance of the article relative to the claim stated in the headline into one of the three categories agree, disagree or discuss.

1.6 Technology Used

➢ Python
➢ Python Libraries: keras, matplotlib, scikitplot, numpy, sklearn
➢ DataSet: drawn from Kaggle
➢ Software : Spyder

1.7 Drawback of Existing System:

- The models through which I have detected Fake News are not 100% accurate.
- The maximum accuracy we get is 92%.
- Now a days, People post the fake news in such a way that it seems to be true news, So it may happen that the predicted model provides you the wrong information.

CHAPTER:-2
THEROY

2.1 Getting your machine ready

- Lets implement basic components in a step by step manner in order to create a text classification framework in python. To start with, import all the required libraries.
- You would need requisite libraries to run this code – you can install them at their individual official links
  - → Scikit-learn
  - → Scikitplot
  - → Matplotlib
  - → Keras
  - → Numpy

2.1.1 Dataset preparation

- For the purpose of this project, I am the using dataset of Kaggle.
- The training dataset has about 16600 rows of data from various articles on the internet.
- I had to do quite a bit of pre-processing of the data, as is evident from our source code, in order to train our models.
- A full training dataset has the following attributes:
  1. id: unique id for a news article
  2. title: the title of a news article
  3. author: author of the news article
  4. text: the text of the article; incomplete in some cases
  5. label: a label that marks the article as potentially unreliable
     1: unreliable
     0: reliable

```
# load the dataset
data = open('data/corpus').read()
labels, texts = [], []
for i, line in enumerate(data.split("\n")):
    content = line.split()
    labels.append(content[0])
    texts.append(content[1:])

# create a dataframe using texts and lables
trainDF = pandas.DataFrame()
trainDF['text'] = texts
trainDF['label'] = labels
```

- Next, I will split the dataset into training and validation sets so that I can train and test classifier. Also, I will encode our target column so that it can be used in machine learning models.

```
# split the dataset into training and validation datasets
train_x, valid_x, train_y, valid_y =
model_selection.train_test_split(trainDF['text'], trainDF['label'])
# label encode the target variable
encoder = preprocessing.LabelEncoder()
train_y = encoder.fit_transform(train_y)
valid_y = encoder.fit_transform(valid_y)
```

2.2 Feature Engineering

- The next step is the feature engineering step. In this step, raw text data will be transformed into feature vectors and new features will be created using the existing dataset. I will implement the following different ideas in order to obtain relevant features from our dataset.

  2.2.1 Count Vectors as features
  2.2.2 TF-IDF Vectors as features
    Word level
    N-Gram level
    Character level
  2.2.3 Word Embeddings as features
  2.2.4 Text / NLP based features
  2.2.5 Topic Models as features

- Let's look at the implementation of these ideas in detail.

2.2.1 Count Vectors as features

- Count Vector is a matrix notation of the dataset in which every row represents a document from the corpus, every column represents a term from the corpus, and every cell represents the frequency count of a particular term in a particular document.

```
# create a count vectorizer object
count_vect = CountVectorizer(analyzer='word', token_pattern=r'\w{1,}')
count_vect.fit(trainDF['text'])

# transform the training and validation data using count vectorizer object
```

```
xtrain_count =  count_vect.transform(train_x)
xvalid_count =  count_vect.transform(valid_x)
```

2.2.2 TF-IDF Vectors as features

- Count Vector is a matrix notation of the dataset in which every row represents a document from the corpus, every column reprTF-IDF score represents the relative importance of a term in the document and the entire corpus. TF-IDF score is composed by two terms: the first computes the normalized Term Frequency (TF), the second term is the Inverse Document Frequency (IDF), computed as the logarithm of the number of the documents in the corpus divided by the number of documents where the specific term appears.
- TF(t) = (Number of times term t appears in a document) / (Total number of terms in the document)
- IDF(t) = log_e(Total number of documents / Number of documents with term t in it)
- TF-IDF Vectors can be generated at different levels of input tokens (words, characters, n-grams)
  a. Word Level TF-IDF : Matrix representing tf-idf scores of every term in different documents
  b. N-gram Level TF-IDF : N-grams are the combination of N terms together. This Matrix representing tf-idf scores of N-grams
  c. Character Level TF-IDF : Matrix representing tf-idf scores of character level n-grams in the corpus

```
# word level tf-idf
tfidf_vect = TfidfVectorizer(analyzer='word', token_pattern=r'\w{1,}',
max_features=5000)
tfidf_vect.fit(trainDF['text'])
xtrain_tfidf =  tfidf_vect.transform(train_x)
xvalid_tfidf =  tfidf_vect.transform(valid_x)

# ngram level tf-idf
tfidf_vect_ngram = TfidfVectorizer(analyzer='word',
token_pattern=r'\w{1,}', ngram_range=(2,3), max_features=5000)
tfidf_vect_ngram.fit(trainDF['text'])
xtrain_tfidf_ngram =  tfidf_vect_ngram.transform(train_x)
xvalid_tfidf_ngram =  tfidf_vect_ngram.transform(valid_x)

# characters level tf-idf
tfidf_vect_ngram_chars = TfidfVectorizer(analyzer='char',
token_pattern=r'\w{1,}', ngram_range=(2,3), max_features=5000)
tfidf_vect_ngram_chars.fit(trainDF['text'])
xtrain_tfidf_ngram_chars =  tfidf_vect_ngram_chars.transform(train_x)
xvalid_tfidf_ngram_chars =  tfidf_vect_ngram_chars.transform(valid_x)
```

2.2.3 Word Embeddings

- A word embedding is a form of representing words and documents using a dense vector representation. The position of a word within the vector space is learned from text and is based on the words that surround the word when it is used. Word embeddings can be trained using the input corpus itself or can be generated using pre-trained word embeddings such as Glove, FastText, and Word2Vec. Any one of them can be downloaded and used as transfer learning. One can read more about word embeddings here.
- Following snnipet shows how to use pre-trained word embeddings in the model. There are four essential steps:
- Loading the pretrained word embeddings
- Creating a tokenizer object
- Transforming text documents to sequence of tokens and pad them
- Create a mapping of token and their respective embeddings

```
# load the pre-trained word-embedding vectors
embeddings_index = {}
for i, line in enumerate(open('data/wiki-news-300d-1M.vec')):
    values = line.split()
    embeddings_index[values[0]] = numpy.asarray(values[1:], dtype='float32')

# create a tokenizer
token = text.Tokenizer()
token.fit_on_texts(trainDF['text'])
word_index = token.word_index

# convert text to sequence of tokens and pad them to ensure equal length
vectors
train_seq_x = sequence.pad_sequences(token.texts_to_sequences(train_x),
maxlen=70)
valid_seq_x = sequence.pad_sequences(token.texts_to_sequences(valid_x),
maxlen=70)

# create token-embedding mapping
embedding_matrix = numpy.zeros((len(word_index) + 1, 300))
for word, i in word_index.items():
    embedding_vector = embeddings_index.get(word)
    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector
```

2.2.4 Text / NLP based features

- A number of extra text based features can also be created which sometimes are helpful for improving text classification models.
- Some examples are:
  1. Word Count of the documents – total number of words in the documents
  2. Character Count of the documents – total number of characters in the documents
  3. Average Word Density of the documents – average length of the words used in the documents
  4. Puncutation Count in the Complete Essay – total number of punctuation marks in the documents
  5. Upper Case Count in the Complete Essay – total number of upper count words in the documents
  6. Title Word Count in the Complete Essay – total number of proper case (title) words in the documents
  7. Frequency distribution of Part of Speech Tags:
     - → Noun Count
     - → Verb Count
     - → Adjective Count
     - → Adverb Count
     - → Pronoun Count
- These features are highly experimental ones and should be used according to the problem statement only.

```
trainDF['char_count'] = trainDF['text'].apply(len)
trainDF['word_count'] = trainDF['text'].apply(lambda x: len(x.split()))
trainDF['word_density'] = trainDF['char_count'] / (trainDF['word_count']+1)
trainDF['punctuation_count'] = trainDF['text'].apply(lambda x: len("".join(_
for _ in x if _ in string.punctuation)))
trainDF['title_word_count'] = trainDF['text'].apply(lambda x: len([wrd for
wrd in x.split() if wrd.istitle()]))
trainDF['upper_case_word_count'] = trainDF['text'].apply(lambda x: len([wrd
for wrd in x.split() if wrd.isupper()]))
```

```
pos_family = {
    'noun' : ['NN','NNS','NNP','NNPS'],
    'pron' : ['PRP','PRP$','WP','WP$'],
    'verb' : ['VB','VBD','VBG','VBN','VBP','VBZ'],
    'adj' :  ['JJ','JJR','JJS'],
    'adv' : ['RB','RBR','RBS','WRB']
}

# function to check and get the part of speech tag count of a words in a
given sentence
```

```
def check_pos_tag(x, flag):
    cnt = 0
    try:
        wiki = textblob.TextBlob(x)
        for tup in wiki.tags:
            ppo = list(tup)[1]
            if ppo in pos_family[flag]:
                cnt += 1
    except:
        pass
    return cnt

trainDF['noun_count']  =  trainDF['text'].apply(lambda  x:  check_pos_tag(x,
'noun'))
trainDF['verb_count']  =  trainDF['text'].apply(lambda  x:  check_pos_tag(x,
'verb'))
trainDF['adj_count']  =  trainDF['text'].apply(lambda  x:  check_pos_tag(x,
'adj'))
trainDF['adv_count']  =  trainDF['text'].apply(lambda  x:  check_pos_tag(x,
'adv'))
trainDF['pron_count']  =  trainDF['text'].apply(lambda  x:  check_pos_tag(x,
'pron'))
```

2.2.5 Topic Models as features

- Topic Modelling is a technique to identify the groups of words (called a topic) from a collection of documents that contains best information in the collection.
- I have used Latent Dirichlet Allocation for generating Topic Modelling Features.
- LDA is an iterative model which starts from a fixed number of topics.
- Each topic is represented as a distribution over words, and each document is then represented as a distribution over topics.
- Although the tokens themselves are meaningless, the probability distributions over words provided by the topics provide a sense of the different ideas contained in the documents.

```
# train a LDA Model
lda_model = decomposition.LatentDirichletAllocation(n_components=20,
learning_method='online', max_iter=20)
X_topics = lda_model.fit_transform(xtrain_count)
topic_word = lda_model.components_
vocab = count_vect.get_feature_names()

# view the topic models
n_top_words = 10
topic_summaries = []
for i, topic_dist in enumerate(topic_word):
    topic_words = numpy.array(vocab)[numpy.argsort(topic_dist)][:-
(n_top_words+1):-1]
    topic_summaries.append(' '.join(topic_words))
```

2.3 Model Building

- The final step in the text classification framework is to train a classifier using the features created in the previous step. There are many different choices of machine learning models which can be used to train a final model. I will implement following different classifiers for this purpose:
  1. Naive Bayes Classifier
  2. Linear Classifier
  3. Support Vector Machine
  4. Bagging Models
  5. Boosting Models
  6. Shallow Neural Networks
  7. Deep Neural Networks
     - Convolutional Neural Network (CNN)
     - Long Short Term Modelr (LSTM)
     - Gated Recurrent Unit (GRU)
     - Bidirectional RNN
     - Recurrent Convolutional Neural Network (RCNN)
     - Other Variants of Deep Neural Networks

- Lets implement these models and understand their details. The following function is a utility function which can be used to train a model. It accepts the classifier, feature_vector of training data, labels of training data and feature vectors of valid data as inputs. Using these inputs, the model is trained and accuracy score is computed.

2.4 Improving Text Classification Models

- While the above framework can be applied to a number of text classification problems, but to achieve a good accuracy some improvements can be done in the overall framework. For example, following are some tips to improve the performance of text classification models and this framework.

  1. **Text Cleaning:** text cleaning can help to reducue the noise present in text data in the form of stopwords, punctuations marks, suffix variations etc. This article can help to understand how to implement text classification in detail.

2. **Hstacking Text / NLP features with text feature vectors:** In the feature engineering section, I generated a number of different feature vectros, combining them together can help to improve the accuracy of the classifier.

3. **Hyperparamter Tuning in modelling:** Tuning the paramters is an important step, a number of parameters such as tree length, leafs, network paramters etc can be fine tuned to get a best fit model.

4. **Ensemble Models:** Stacking different models and blending their outputs can help to further improve the results.

CHAPTER-3
MODELS

3.1 Naive Bayes Classifier

- In order to get a baseline accuracy rate for our data, I implemented a Naive Bayes classifier.
- Specifically, I used the scikit-learn implementation of Gaussian Naive Bayes. This is one of the simplest approaches to classification, in which a probabilistic approach is used, with the assumption that all features are conditionally independent given the class label. As with the other models, I used the Doc2Vec embeddings described above. The Naive Bayes Rule is based on the Bayes' theorem Our system will be used by two types of users they are as follows:

$$P(c|x) = P(x|c)P(c)/P(x)$$

- Parameter estimation for naive Bayes models uses the method of maximum likelihood. The advantage here is that it requires only a small amount of training data to estimate the parameters.

3.2 Support Vector Machine

- The original Support Vector Machine (SVM) was proposed by Vladimir N. Vapnik and Alexey Ya. Chervonenkis in 1963.
- But that model can only do linear classification so it doesn't suit for most of the practical problems.
- Later in 1992, Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik introduced the kernel trick which enables the SVM for non-linear classification.
- That makes the SVM much powerful.
- I use the Radial Basis Function kernel in our project. The reason I use this kernel is that two Doc2Vec feature vectors will be close to each other if their corresponding documents are similar, so the distance computed by the kernel function should still represent the original distance.
- Since the Radial Basis Function is

$$K(x, x') = \exp\left(|x - x'|^2 / 2\sigma^2 \right)$$

- It correctly represents the relationship I desire and it is a common kernel for SVM.
- I use the theory introduced in [6] to implement the SVM. The main idea of the SVM is to separate different classes of data by the widest "street". This goal can be represented as the optimization problem.
- Finally I solve this optimization problem using the convex optimization tools provided by Python package CVXOPT.

3.3 Feed-forward Neural Network

- I implemented two feed-forward neural network models, one using Tensorflow and one using Keras.
- Neural networks are commonly used in modern NLP applications, in contrast to older approaches which primarily focused on linear models such as SVM's and logistic regression.
- Our neural network implementations use three hidden layers.
- In the Tensorflow implementation, all layers had 300 neurons each, and in the Keras implementation I used layers of size 256, 256, and 80, interspersed with dropout layers to avoid overfitting.
- For our activation function, I chose the Rectified Linear Unit (ReLU), which has been found to perform well in NLP applications.
- This has a fixed-size input $x \in R^{1\times 300}$

$h_1 = \text{ReLU}(W_1x + b_1)$
$h_2 = \text{ReLU}(W_2h_1 + b_2)$
$y = \text{Logits}(W_3h_2 + b_3)$

3.4 Long Short-Term Memory

- The Long-Short Term Memory (LSTM) unit was proposed by Hochreiter and Schmidhuber.
- It is good at classifying serialized objects because it will selectively memorize the previous input and use that, together with the current input, to make prediction.
- The news content (text) in our problem is inherently serialized.
- The order of the words carries the important information of the sentence.
- So the LSTM model suits for our problem.
- Since the order of the words is important for the LSTM unit, we cannot use the Doc2Vec for preprocessing because it will transfer the entire document into one vector and lose the order information.
- To prevent that, I use the word embedding instead.
- I first clean the text data by removing all characters which are not letters nor numbers.
- Then I count the frequency of each word appeared in our training dataset to find 5000 most common words and give each one an unique integer ID.
- For example, the most common word will have ID 0, and the second most common one will have 1, etc.
- After that I replace each common word with its assigned ID and delete all uncommon words.
- Notice that the 5000 most common words cover the most of the text, so I only lose little information but transfer the string to a list of integers.

- Since the LSTM unit requires a fixed input vector length, I truncate the list longer than 500 numbers because more than half of the news is longer than 500 words.
- Then for those list shorter than 500 words, I pad 0's at the beginning of the list.
- I also delete the data with only a few words since they don't carry enough information for training.
- By doing this, I transfer the original text string to a fixed length integer vector while preserving the words order information.
- Finally I use word embedding to transfer each word ID to a 32-dimension vector.
- The word embedding will train each word vector based on word similarity.
- If two words frequently appear together in the text, they are thought to be more similar and the distance of their corresponding vectors is small.
- The pre-processing transfers each news in raw text into a fixed size matrix.
- Then I feed the processed training data into the LSTM unit to train the model.
- The LSTM is still a neural network.
- But different from the fully connected neural network, it has cycle in the neuron connections.
- So the previous state (or memory) of the LSTM unit c_t will play a role in new prediction h_t.

$$h_t = o_t \cdot \tanh(c_t)$$

$$c_t = f_t \cdot c_t{-}1 + i_t{\cdot}\tilde{c}$$

$$\tilde{c}_t = \tanh(x_tW_c + h_{t-1}U_c + b_c)$$

$$o_t = \sigma(x_tW_o + h_{t-1}U_o + b_o)$$

$$i_t = \sigma(x_tW_i + h_{t-1}U_i + b_i)$$

$$f_t = \sigma(x_tW_f + h_{t-1}U_f + b_f)$$

CHAPTER:-4
PROJECT MANAGEMENT

4.1 Project Planning

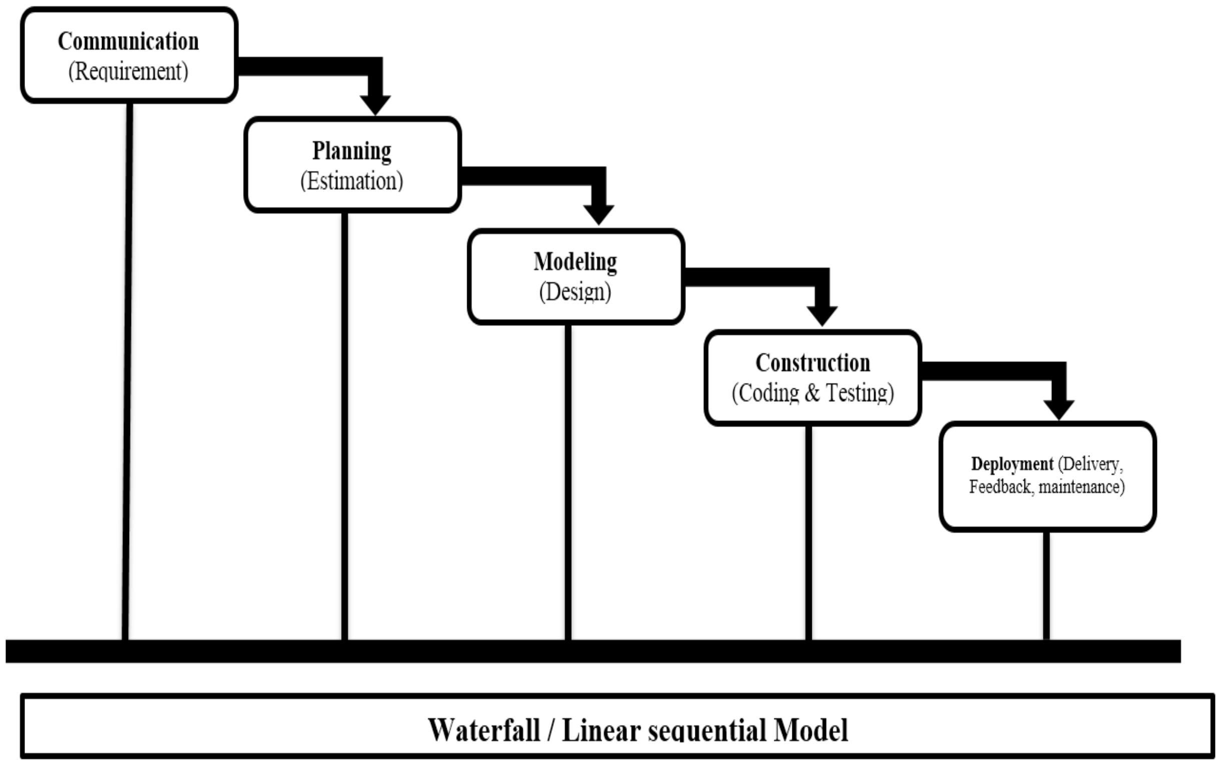
- Effective management of a software project depends on thoroughly planning the progress of the project.
- A well-planned strategy leads to the best and optimal use of the resources available and ensures completion of project on time.

4.1.1 Project Development Approach and Justification

- A system development methodology refers to the framework that is used to structure, plan, and control the process of developing an information system.
- Each of the available methodologies is best suited to specific kinds of projects, based on various technical, organizational, project and team considerations.
- For developing the “Imitation Box”, I have used the “Waterfall Model / Sequential Model”.
- The product is designed, implemented, integrated and tested In a Sequential That’s why it’s Call Sequential Model.

What is Waterfall model?

- The **waterfall model** is a sequential design process, often used in software development processes.
- The Waterfall Model was first Process Model to be introduced. It is also referred to as a **linear-sequential life cycle model**.
- It is very simple to understand and use.
- This systematic and sequential model is divided into five phases. The phases are always in order and not overlap. In a waterfall model, each phase must be completed fully before the next phase can begin. That why it is also known as waterfall model.



[Figure 4.1.1 Waterfall Model]

The sequential phases in Waterfall model are:

<u>Communication Phase:</u>

- Requirements are gathered.
- The problem is specified along with the desired service objectives (goals).
- The Constrains are identified.
- The system specification is produced from the requirement analysis. This document should clearly define the product function.

<u>Planning Phase:</u>

- Planning is critical activity in software development. A good plan is based on the requirements of system and should be done before later phase begins.
- Planning includes activities like:
  - Cost estimation to find best affordable cost.
  - Task scheduling and timeline management.
  - Team structure designing.

Modeling Phase:

- Software Engineer at this phase concern with:
    - Data Structure.
    - Software Architecture.
    - Algorithmic Details.
    - Interface Representations.
- By the end of this phase software engineers should be able to understand the relationship between the hardware, software and the associated interfaces.

Construction Phase:

- Coding is done.
- The Implementation and testing is done.
- The designs are translated into the software domain.
- Detailed documentation from the design phase can significantly reduce the coding effort.
- Testing at this stage focuses on making sure that errors are identified and software meets its required specification.
- All the program units are integrated and tested to ensure that the complete system meets the software requirements.

Deployment Phase:

- After construction the software is delivered to customer
- All the problems that arise during the previous phase will be solved in this last stage.
- Maintenance part is very large part and never ending part of software development.
- According to customers feedback the correction is made into the model.
- When changes are made in any of the phase, the relevant documentation should be updated to reflect the change.

Advantages of waterfall model:

- Simple and easy to understand and use.
- Stages and activities are well defined.
- Helps to plan and schedule projects.
- Testing is inherent to every phase.
- Documentation is produced at every phase.
- Works well for smaller projects where requirements are very well understood.

4.1.2 Project Plan

- Project planning is basically concerned with identifying and measuring activities, milestones and deliverables produced by project.
- All this is done by our project team members.
- Thus in this section we cover following:-

Estimating some basic attributes of the project:

A) Duration:

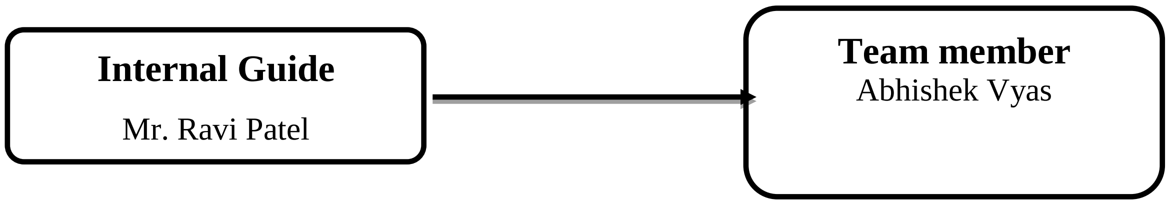
- How long will it take to complete the development? The Client for which the website is to be developed is not clear with his requirements and not very rigid with the demands due to the unawareness of possibilities.
- It is necessary to understand his requirements in details, do a detailed research of what combination of technologies would be best for the project, get acclimatized to them, do the rest of the planning and go ahead with the work. Hence the duration of the Project can be roughly estimated to 4.5 months.

B) Efforts:

- How much efforts would be required? Since it is a one person group and as it is a very hazy picture ahead of how to go about the project, figuring that out and considering the pros and cons would take up a lot of effort in itself. Thus efforts can be estimated to 3 hours of weekdays work for 4.5 months.

4.1.3 Group Dependency

As described above we have done our project in team work. And these working dependencies showed below:



[Figure 4.1.3 Group Dependency]

4.2 Project Scheduling

- Project scheduling is one of the main key aspects of any project. Any project must be schedule before developing it when project developer works on scheduled project it is more advantageous for him/her to compare to unscheduled project.
- It gives us timeline for finishing the particular activity. Scheduling gives us idea about project length, its cost, its normal duration of completion and we can also find out the shortest way to complete the project with less overall cost of project.
- Project schedule describes dependency between activities. The estimated time required to reach each milestones and allocation of people to activities.

4.2.1 Basic Principle

"Software project scheduling is an activity that distributes estimated effort across the planned project duration by allocating the effort to specific software engineering tasks."

Proper scheduling requires:

- Effort and timing are intelligently allocated to each task.
- Interdependencies between tasks are properly indicated.
- Resources are allocated for the work to be done.

4.3 Risk management

- Software is a difficult undertaking. Lots of things can go wrong and frankly, many often do. It is for this reason that being prepared understanding the risks and taking proactive measure to avoid or manage them is a key element of good software project management. Recognizing what can go wrong is the first step called, Risk Identification. Next each risk is analysed to determine the likelihood that it will occur and the damage that it will do if it does occur. Once this information is established, risks are ranked, by probability and impact. Finally a plan is developed to manage those risks with high probability.

4.3.1 Risk Identification

- Risk management is an extremely important task in all projects and so it is in this project as well. There are certain risks that if run true, it can not only run the whole effort and foil the project plans, can cause some serious danger such as copyright in fragment. In sections below, risk identification, their analysis and planning is taken care of each.

Risk Type	Description
Project Risks	➢ Implementation lethargies ➢ Unclear and yet very wide set of requirements
Technical Risks	➢ Performance issues ➢ Security Issues ➢ Scalability problem
Business Risks	➢ User permissions need to be implemented tightly else it can cause losses

Table 4.3 Risk Type

4.3.2 Risk Planning

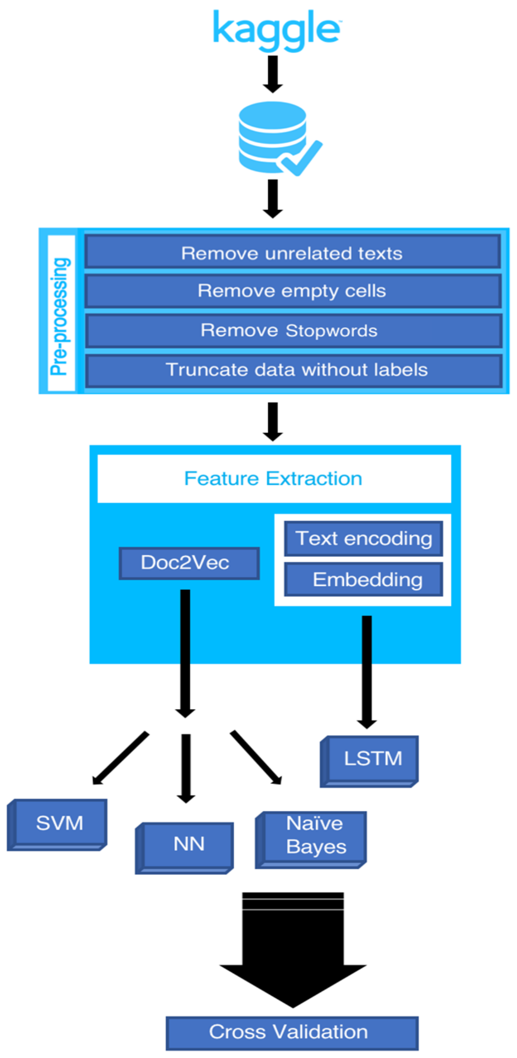
- Risk planning process is considered when each of the key risk has been identified. Risk reduction Strategy is used as abatement procedure. This involves planning ways to contain the damage due to a risk.

Risk	Strategy
Requirement Changes	Derive traceability information to access requirements, change impact and maximize information hiding.
Schedule risk	To reduce this risk, we are going to complete our project according to our schedule.
Performance	Investigate Database which can effectively process.

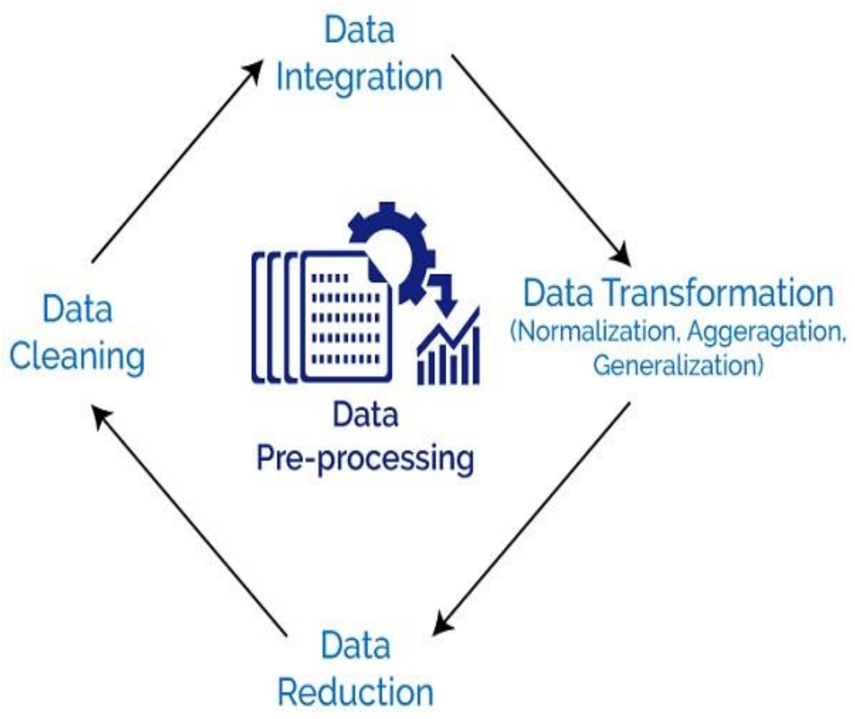
Table 4.4 Risk Planning

CHAPTER-5
SYSTEM DESIGN

5.1 Flow Chart of System



(Fig 5.1.1: Flow Chart)

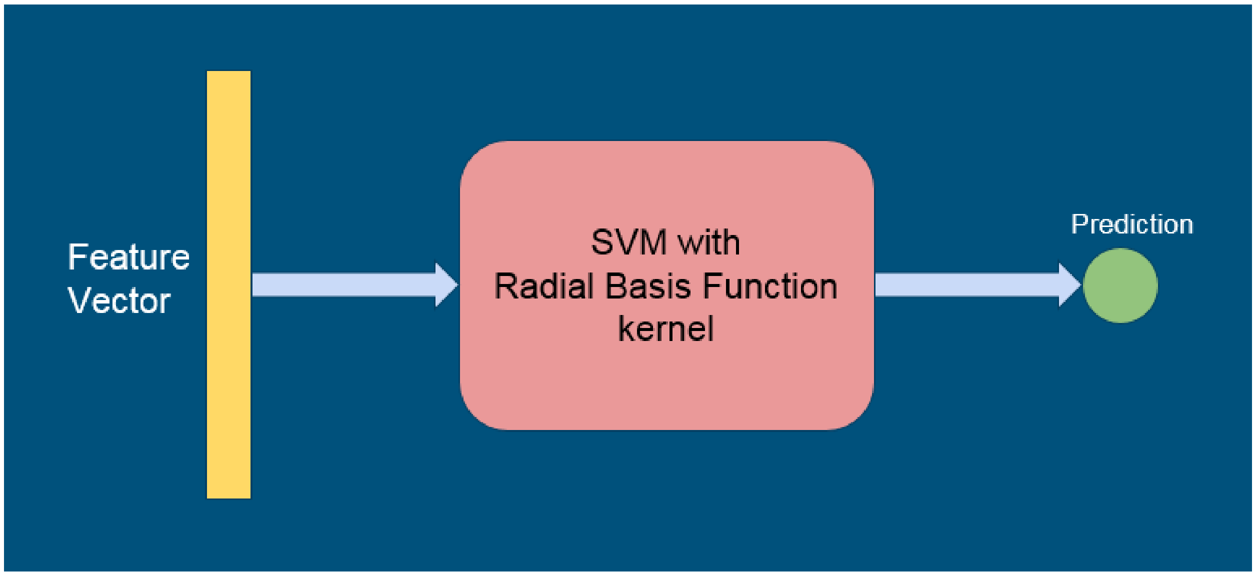

Data Integration
Data Transformation (Normalization, Aggeragation, Generalization)
Data Reduction
Data Cleaning
Data Pre-processing

(Fig 5.1.2: Training a Model)

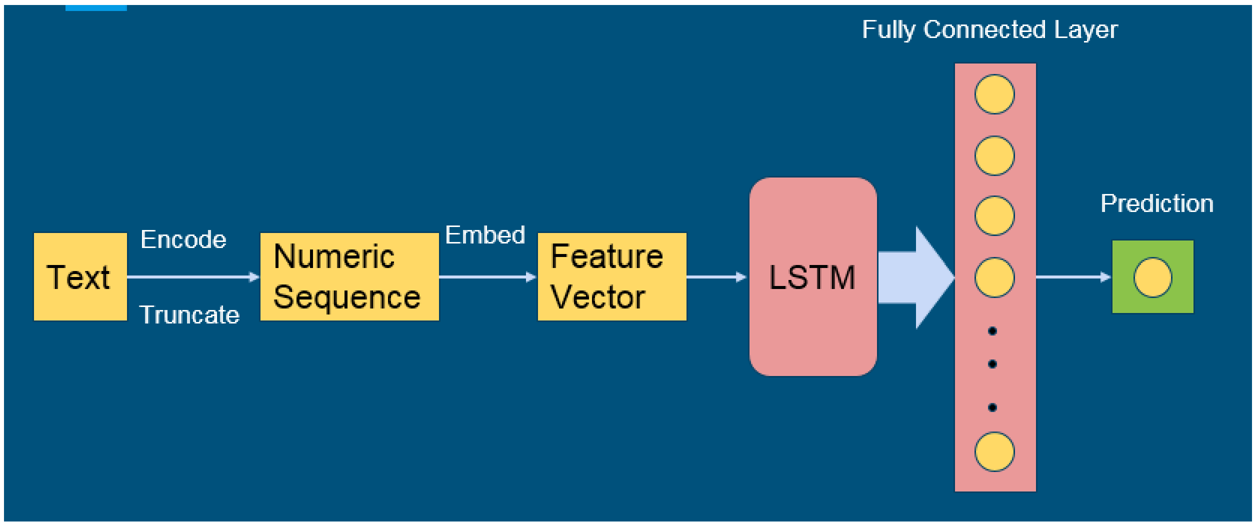
$$\underbrace{P(c\mid x)}_{\text{Posterior Probability}} = \frac{\overbrace{P(x\mid c)}^{\text{Likelihood}}\,\overbrace{P(c)}^{\text{Class Prior Probability}}}{\underbrace{P(x)}_{\text{Predictor Prior Probability}}}$$

$$P(c\mid X) = P(x_1\mid c)\times P(x_2\mid c)\times\cdots\times P(x_n\mid c)\times P(c)$$

(Fig 5.1.3: Naïve Bayes)



(Fig 5.1.4: SVM)



(Fig 5.1.2: LSTM)

5.2 Major Functionality

- Detects Fake News more accurately in LSTM up to 94% which is good.
- It gets helpful to the news channel to detect whether the news are real or fake, so It would be helpful to the news channel.
- And if we publish the application then it would be helpful to the people also.

CHAPTER:-6 IMPLEMENTATION PLANNING AND DETAILS

6.1 IMPLEMENTATION ENVIRONMENT:

- The project was a result of solo hardwork. Decision on the problem was made mutually by me and my guide. Communication among us was horizontal.

6.1.1 Implementation Planning:

- Implementation phase requires precise planning and monitoring mechanism in order to ensure schedule and completeness. I implemented various model to check which one is better one.

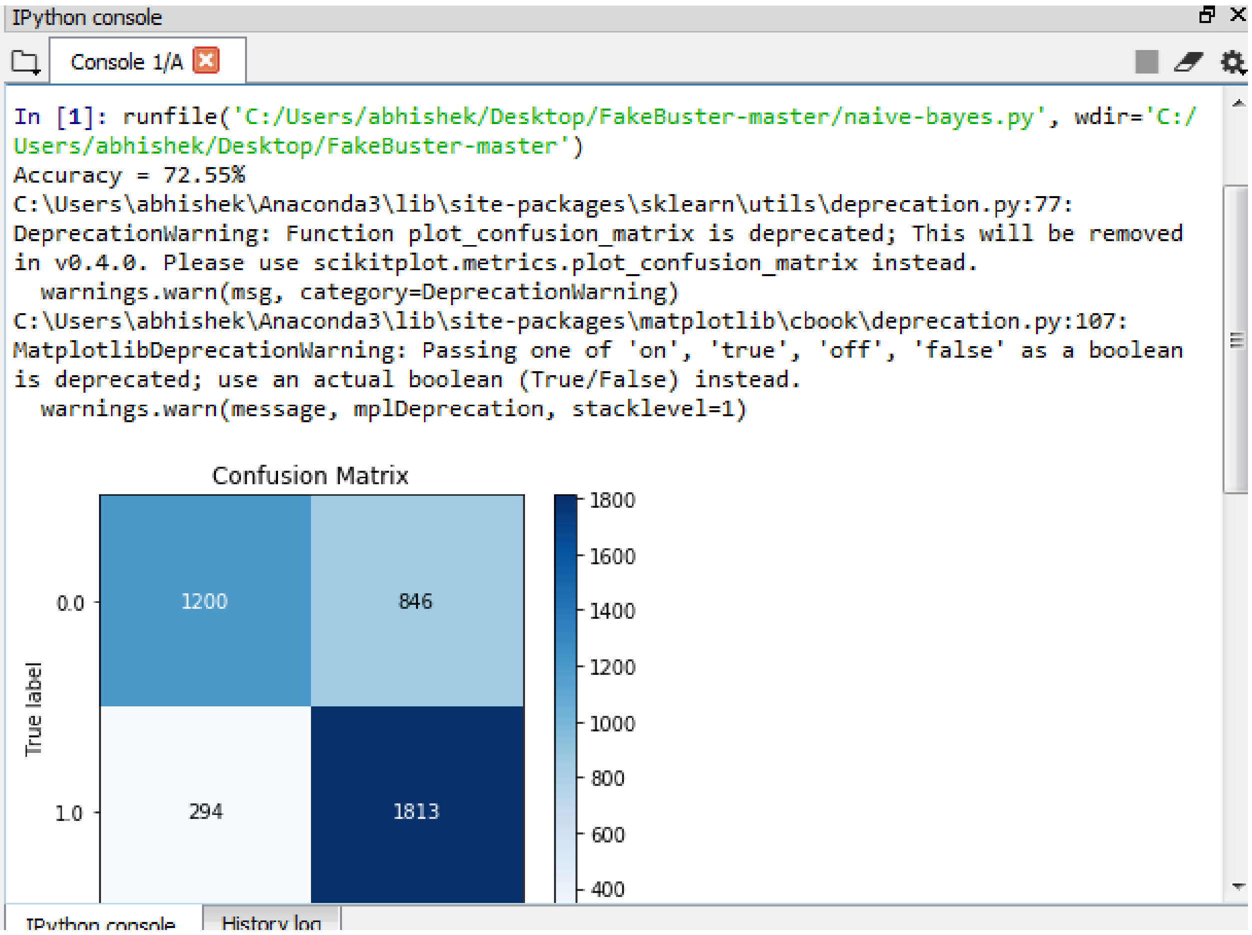
6.2 CODING STANDARD:

- I have used various python libraries and accordingly their models i.e. Natural Language Processing techniques and Recurrent Neural Networks.

CHAPTER-7 SCREENSHOTS

7.1 Naïve Bayes:

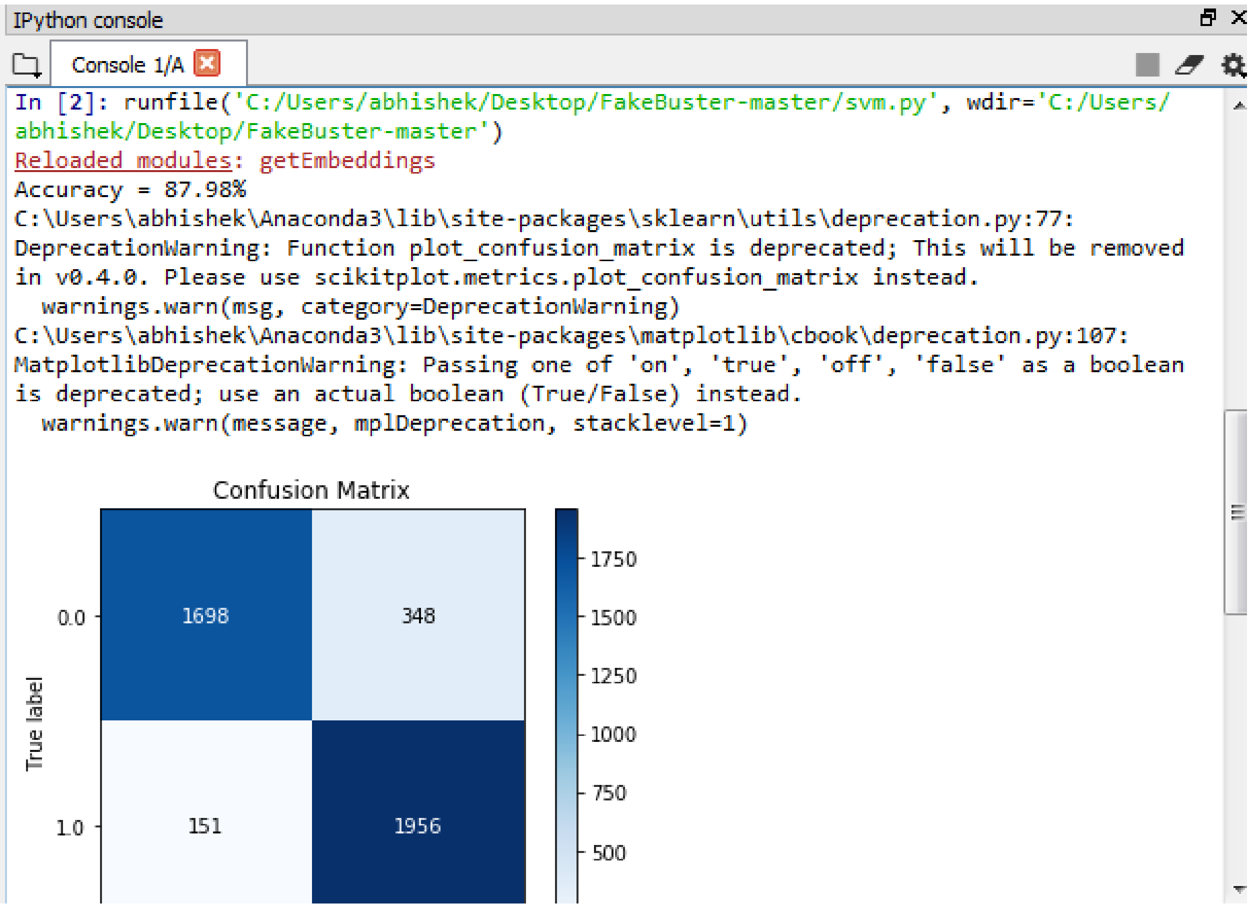
7.1.1 Terminal ScreenShot:



```
In [1]: runfile('C:/Users/abhishek/Desktop/FakeBuster-master/naive-bayes.py', wdir='C:/
Users/abhishek/Desktop/FakeBuster-master')
Accuracy = 72.55%
C:\Users\abhishek\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:77:
DeprecationWarning: Function plot_confusion_matrix is deprecated; This will be removed
in v0.4.0. Please use scikitplot.metrics.plot_confusion_matrix instead.
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\abhishek\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py:107:
MatplotlibDeprecationWarning: Passing one of 'on', 'true', 'off', 'false' as a boolean
is deprecated; use an actual boolean (True/False) instead.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```

(Fig 7.1.1: Naïve Bayes)

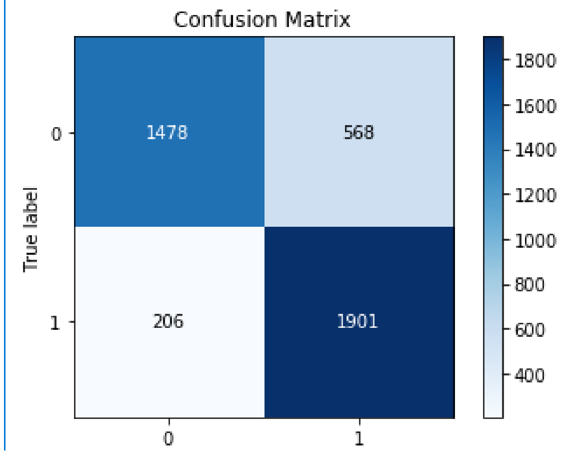
7.2 SVM:
7.2.1 Terminal ScreenShot:



```
In [2]: runfile('C:/Users/abhishek/Desktop/FakeBuster-master/svm.py', wdir='C:/Users/
abhishek/Desktop/FakeBuster-master')
Reloaded modules: getEmbeddings
Accuracy = 87.98%
C:\Users\abhishek\Anaconda3\lib\site-packages\sklearn\utils\deprecation.py:77:
DeprecationWarning: Function plot_confusion_matrix is deprecated; This will be removed
in v0.4.0. Please use scikitplot.metrics.plot_confusion_matrix instead.
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\abhishek\Anaconda3\lib\site-packages\matplotlib\cbook\deprecation.py:107:
MatplotlibDeprecationWarning: Passing one of 'on', 'true', 'off', 'false' as a boolean
is deprecated; use an actual boolean (True/False) instead.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```

(Fig 7.2.1: SVM)

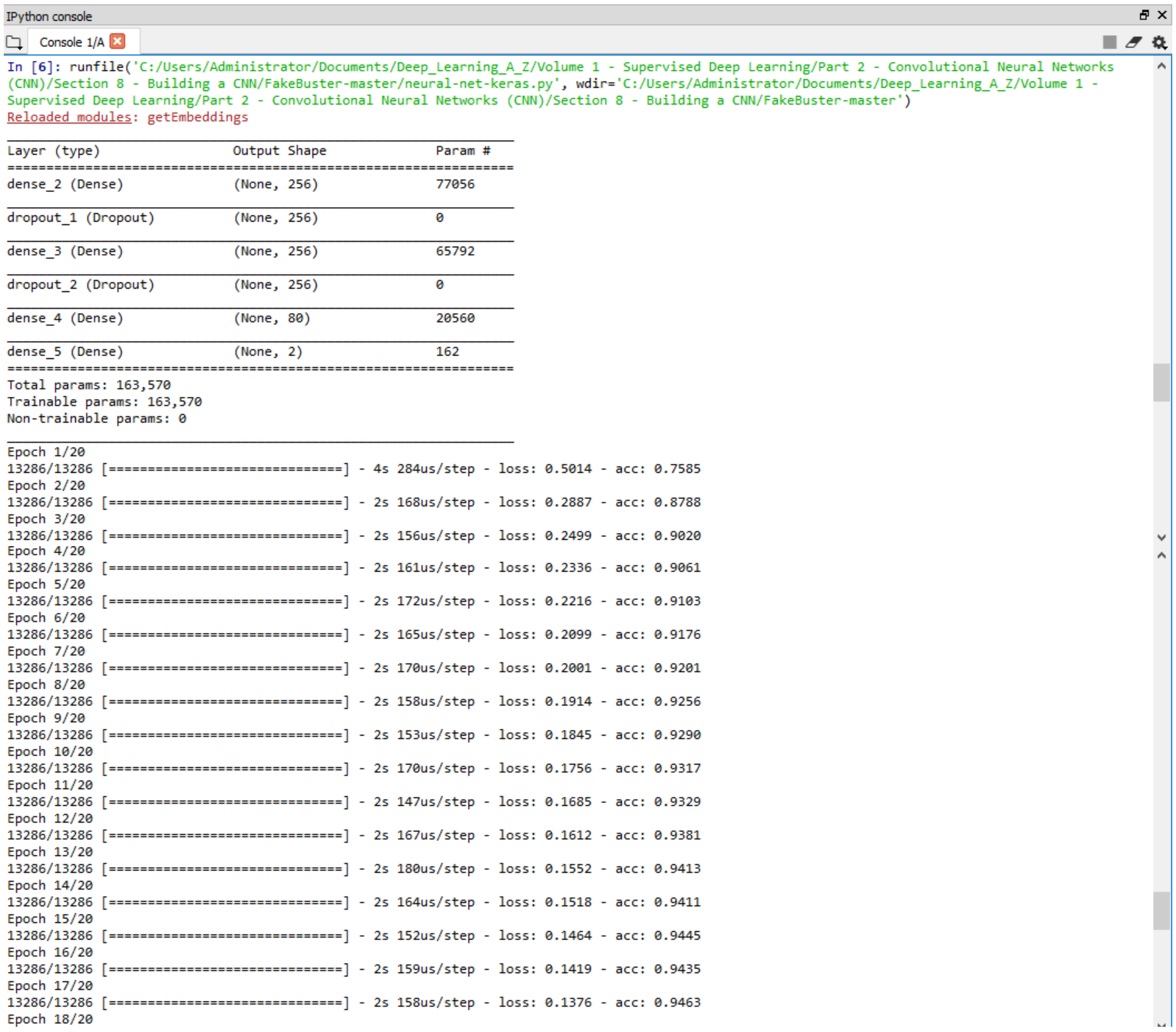
7.3 Neural Network using TensorFlow:
7.3.1 Terminal ScreenShot:



(Fig 7.3.1: Neural Network Using TensorFlow)

7.4 Neural Network using Keras:

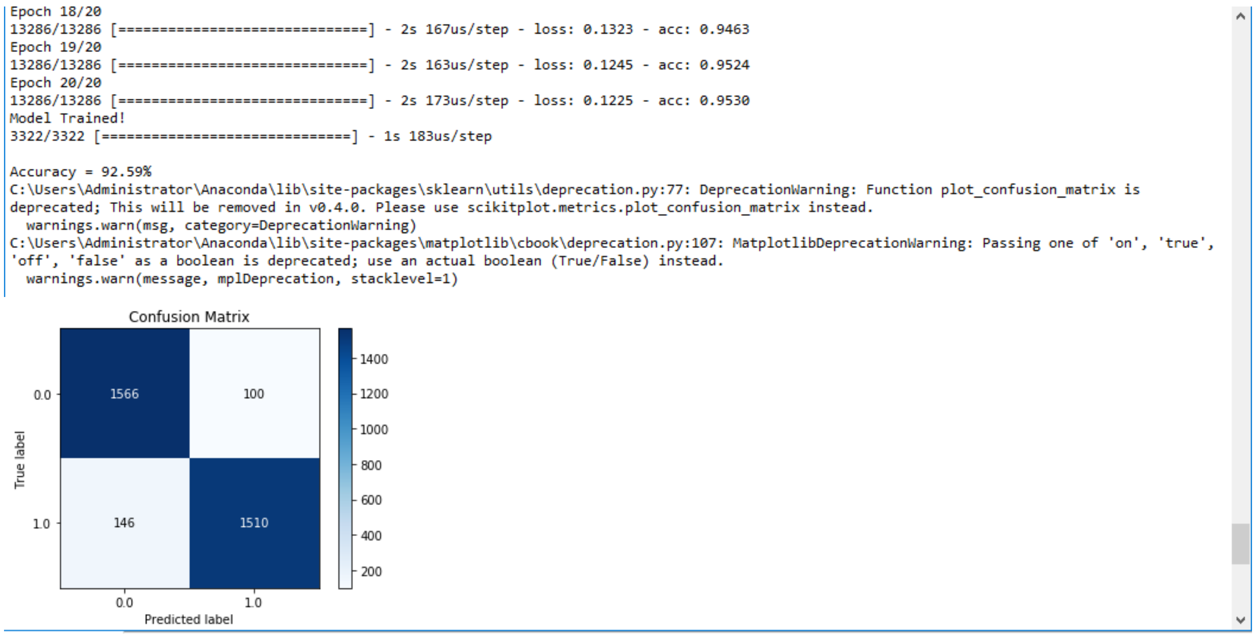
7.4.1 Terminal ScreenShot:



IPython console

Console 1/A

```
In [6]: runfile('C:/Users/Administrator/Documents/Deep_Learning_A_Z/Volume 1 - Supervised Deep Learning/Part 2 - Convolutional Neural Networks
(CNN)/Section 8 - Building a CNN/FakeBuster-master/neural-net-keras.py', wdir='C:/Users/Administrator/Documents/Deep_Learning_A_Z/Volume 1 -
Supervised Deep Learning/Part 2 - Convolutional Neural Networks (CNN)/Section 8 - Building a CNN/FakeBuster-master')
Reloaded modules: getEmbeddings
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_2 (Dense)              (None, 256)               77056
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               65792
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 80)                20560
_________________________________________________________________
dense_5 (Dense)              (None, 2)                 162
=================================================================
Total params: 163,570
Trainable params: 163,570
Non-trainable params: 0
_________________________________________________________________
Epoch 1/20
13286/13286 [==============================] - 4s 284us/step - loss: 0.5014 - acc: 0.7585
Epoch 2/20
13286/13286 [==============================] - 2s 168us/step - loss: 0.2887 - acc: 0.8788
Epoch 3/20
13286/13286 [==============================] - 2s 156us/step - loss: 0.2499 - acc: 0.9020
Epoch 4/20
13286/13286 [==============================] - 2s 161us/step - loss: 0.2336 - acc: 0.9061
Epoch 5/20
13286/13286 [==============================] - 2s 172us/step - loss: 0.2216 - acc: 0.9103
Epoch 6/20
13286/13286 [==============================] - 2s 165us/step - loss: 0.2099 - acc: 0.9176
Epoch 7/20
13286/13286 [==============================] - 2s 170us/step - loss: 0.2001 - acc: 0.9201
Epoch 8/20
13286/13286 [==============================] - 2s 158us/step - loss: 0.1914 - acc: 0.9256
Epoch 9/20
13286/13286 [==============================] - 2s 153us/step - loss: 0.1845 - acc: 0.9290
Epoch 10/20
13286/13286 [==============================] - 2s 170us/step - loss: 0.1756 - acc: 0.9317
Epoch 11/20
13286/13286 [==============================] - 2s 147us/step - loss: 0.1685 - acc: 0.9329
Epoch 12/20
13286/13286 [==============================] - 2s 167us/step - loss: 0.1612 - acc: 0.9381
Epoch 13/20
13286/13286 [==============================] - 2s 180us/step - loss: 0.1552 - acc: 0.9413
Epoch 14/20
13286/13286 [==============================] - 2s 164us/step - loss: 0.1518 - acc: 0.9411
Epoch 15/20
13286/13286 [==============================] - 2s 152us/step - loss: 0.1464 - acc: 0.9445
Epoch 16/20
13286/13286 [==============================] - 2s 159us/step - loss: 0.1419 - acc: 0.9435
Epoch 17/20
13286/13286 [==============================] - 2s 158us/step - loss: 0.1376 - acc: 0.9463
Epoch 18/20
```

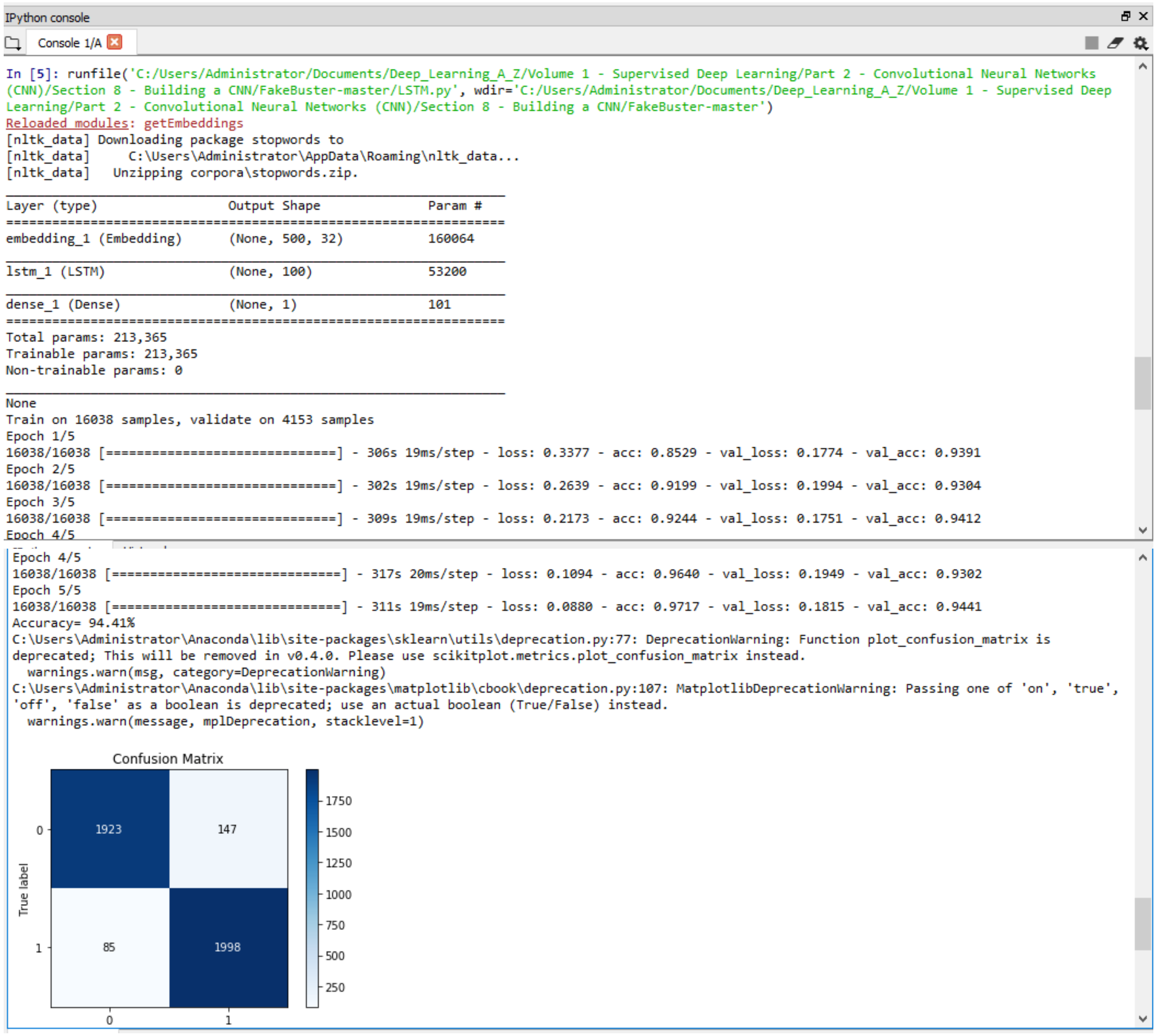


```
Epoch 18/20
13286/13286 [==============================] - 2s 167us/step - loss: 0.1323 - acc: 0.9463
Epoch 19/20
13286/13286 [==============================] - 2s 163us/step - loss: 0.1245 - acc: 0.9524
Epoch 20/20
13286/13286 [==============================] - 2s 173us/step - loss: 0.1225 - acc: 0.9530
Model Trained!
3322/3322 [==============================] - 1s 183us/step

Accuracy = 92.59%
C:\Users\Administrator\Anaconda\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning: Function plot_confusion_matrix is
deprecated; This will be removed in v0.4.0. Please use scikitplot.metrics.plot_confusion_matrix instead.
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\Administrator\Anaconda\lib\site-packages\matplotlib\cbook\deprecation.py:107: MatplotlibDeprecationWarning: Passing one of 'on', 'true',
'off', 'false' as a boolean is deprecated; use an actual boolean (True/False) instead.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```

(Fig 7.4.1: Neural Network Using Keras)

7.5 LSTM:
7.5.1 Terminal ScreenShot:



```
In [5]: runfile('C:/Users/Administrator/Documents/Deep_Learning_A_Z/Volume 1 - Supervised Deep Learning/Part 2 - Convolutional Neural Networks
(CNN)/Section 8 - Building a CNN/FakeBuster-master/LSTM.py', wdir='C:/Users/Administrator/Documents/Deep_Learning_A_Z/Volume 1 - Supervised Deep
Learning/Part 2 - Convolutional Neural Networks (CNN)/Section 8 - Building a CNN/FakeBuster-master')
Reloaded modules: getEmbeddings
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Administrator\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding_1 (Embedding)      (None, 500, 32)           160064
_________________________________________________________________
lstm_1 (LSTM)                (None, 100)               53200
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 101
=================================================================
Total params: 213,365
Trainable params: 213,365
Non-trainable params: 0
_________________________________________________________________
None
Train on 16038 samples, validate on 4153 samples
Epoch 1/5
16038/16038 [==============================] - 306s 19ms/step - loss: 0.3377 - acc: 0.8529 - val_loss: 0.1774 - val_acc: 0.9391
Epoch 2/5
16038/16038 [==============================] - 302s 19ms/step - loss: 0.2639 - acc: 0.9199 - val_loss: 0.1994 - val_acc: 0.9304
Epoch 3/5
16038/16038 [==============================] - 309s 19ms/step - loss: 0.2173 - acc: 0.9244 - val_loss: 0.1751 - val_acc: 0.9412
Epoch 4/5
```

```
Epoch 4/5
16038/16038 [==============================] - 317s 20ms/step - loss: 0.1094 - acc: 0.9640 - val_loss: 0.1949 - val_acc: 0.9302
Epoch 5/5
16038/16038 [==============================] - 311s 19ms/step - loss: 0.0880 - acc: 0.9717 - val_loss: 0.1815 - val_acc: 0.9441
Accuracy= 94.41%
C:\Users\Administrator\Anaconda\lib\site-packages\sklearn\utils\deprecation.py:77: DeprecationWarning: Function plot_confusion_matrix is
deprecated; This will be removed in v0.4.0. Please use scikitplot.metrics.plot_confusion_matrix instead.
  warnings.warn(msg, category=DeprecationWarning)
C:\Users\Administrator\Anaconda\lib\site-packages\matplotlib\cbook\deprecation.py:107: MatplotlibDeprecationWarning: Passing one of 'on', 'true',
'off', 'false' as a boolean is deprecated; use an actual boolean (True/False) instead.
  warnings.warn(message, mplDeprecation, stacklevel=1)
```

(Fig 7.5.1: LSTM)

CHAPTER- 8
FUTURE ENHANCEMENT

8.1 Future Enhancement:

- For fake news detection, we can add as features the source of the news, including any associated URLs, the topic (e.g., science, politics, sports, etc.), publishing medium (blog, print, social media), country or geographic reg on of origin, publication year, as well as linguistic features not exploited in this exercise use of capitalization, fraction of words that are proper nouns (using gazetteers), and others.
- Besides, we can also aggregate the well-performed classifiers to achieve better accuracy.
- For example, using bootstrap aggregating for the Neural Netwrok, LSTM and SVM models to get better prediction result.
- An ambitious work would be to search the news on the Internet and compare the search results with the original news.
- Since the search result is usually reliable, this method should be more accurate, but also involves natural language understanding because the search results will not be exactly the same as the original news.
- So we will need to compare the meaning of two contents and decide whether they mean the same thing.

CHAPTER- 9 CONCLUSION

9.1 CONCLUSION

Model	Accuracy
Naive Bayes	72.94%
SVM	88.42%
Neural Network using TensorFlow	81.42%
Neural Network using Keras	92.62%
LSTM	94.53%

- We observed that LSTM gave us the best results.
- We had to use a different set of embeddings for preprocessing the data to be fed to our LSTM model.
- It uses ordered set of Word2Vec representations.
- The LSTM achieves the highest F1 score in comparison to all the other models, followed by the Neural Network model using Keras.
- One of the reasons that LSTM performs so well is because the text is inherently a serialized object.
- All the other models use the Doc2Vec to get their feature vectors and hence, they rely on the Doc2Vec to extract the order information and perform a classification on it.
- On the other hand, the LSTM model preserves the order using a different pre-processing method and makes prediction based on both the words and their order.
- This is how it outperform others.

9.2 REFERENCES

https://arxiv.org/pdf/1708.01967.pdf
http://cs229.stanford.edu/proj2017/final-reports/5244348.pdf
https://www.kaggle.com/c/fake-news/data
https://www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/
https://www.analyticsvidhya.com/blog/2016/10/tutorial-optimizing-neural-networks-using-keras-with-image-recognition-case-study/
https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/
https://www.analyticsvidhya.com/blog/2017/03/tensorflow-understanding-tensors-and-graphs/#
https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/