**COMPARATIVE ANALYSIS OF VARIOUS ALGORITHMS FOR FAKE NEWS DETECTION**

***Thesis submitted in partial fulfilment***

***of the requirements for the degree of***

**Bachelor of Technology**

**In**

**Computer Science and Engineering**

**by**

**Prithwiraj Samanta**

***Under the guidance of***

**Dr. Rashmi Panda**

****

**भारतीय सूचना प्रौद्योगिकी संस्थान राँची**

**INDIAN INSTITUTE OF INFORMATION TECHNOLOGY, RANCHI**

(An Institution of National importance under act of Parliament)

(Ranchi - 834010), Jharkhand

***Department of Computer Science and Engineering***

****

**Indian Institute of Information Technology**

**Ranchi**

**Certificate**

This is to certify that the thesis entitled ”**Comparative Analysis of various Algorithms for Fake New Detection**” is a Bonafede record of work carried out by **Prithwiraj Samanta,** under my supervision and guidance, for the partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology (Honours)** in Computer Science and Engineering at the Indian Institute of Information Technology, Ranchi. The thesis has fulfilled all the requirements as per the regulations of the institute and in my opinion reached the standard for submission.

**Dr. Rashmi Panda**

**Assistant Professor**

**Indian Institute of Information Technology**

**Date: December 2022**

**Place: IIIT Ranchi**

Acknowledgement

I am greatly indebted to my project supervisor Prof. Rashmi Panda for her invaluable guidance and encouragement throughout the course of this project. I would like to take this opportunity to express my sincere and profound gratitude to him.

I would also like to gratefully acknowledge the suggestions and discussions received from my teammate Satya.

My sincere thanks also go to all the faculty of CSE for their kind co-operation in all spheres during my project work. I would also like to thank all others whose direct or indirect help has benefited me during my stay at IIIT Ranchi.

Furthermore, I would like to thank all other friends and fellow students of IIIT Ranchi and particularly my CSE mates for sparing great time throughout my B.Tech and very good support I received from them during my stay.

Last but not the least, I thank my parents and all my family members for their support and motivation they have provided throughout my project.

**Prithwiraj Samanta**

Abstract

The advent of the World Wide Web and the rapid adoption of social media platforms (such as Facebook and Twitter) paved the way for information dissemination that has never been witnessed in the human history before. With the current usage of social media platforms, consumers are creating and sharing more information than ever before, some of which are misleading with no relevance to reality. Automated classification of a text article as misinformation or disinformation is a challenging task. Even an expert in a particular domain has to explore multiple aspects before giving a verdict on the truthfulness of an article. In this work, we have created an analysis report of various algorithms (particularly LSTM + CNN + Attention + Transformer Models) for automated classification of news articles. Our study explores different textual properties that can be used to distinguish fake contents from real. By using those properties, we train a combination of different machine learning algorithms using various methods and evaluate their performance on real world datasets.

Content

**Acknowledgement ii**

**Abstract iii**

**List of Figures iv**

**List of Tables v**

**1 Introduction 7**

1.1 Background …………………………………………………………………………………. 7

1.2 LSTM Models …………………………………………………………………………….... 7

1.3 CNN Models …………………………………………………………………………………. 8

1.4 LSTM + CNN Models ……………………………………………………………………. 8

1.5 RNN + Attention Models ……………………..………………………………………. 9

1.6 Transformer Models ……………………………………………………………………. 9

**2 Review of Literature 11**

2.1 LSTM + CNN Models ………………………………………………………………….. 11

2.2 RNN + Attention Models ………………………………………..………………….. 12

2.3 Transformer Models ………………………………………………………………….. 12

**3 Implementation 11**

3.1 Data Pre-processing ………………………………………………………….……… 11

3.2 LSTM + CNN Models …………………………………………………………...…... 11

3.3 RNN + Attention Models ………………………..……………………………….… 11

3.4 Transformer Models ……………………………………………………………….… 11

**4 Results and Discussion 11**

4.1 Result ……….…………………………………………………………………………..…… 11

4.2 Discussion …...…………………………………………………………………………..… 11

**5 Conclusion and Future Work 11**

5.1 Conclusion ….………………………………………………………………………..…… 11

5.2 Future Work ………………………………………………………………………..……. 11

**List of Abbreviation vi**

**References vii**

List of Figures

1.1 Graph of Loss of Model-1 ……………………………………………………….…… 14

1.2 Graph of Accuracy of Model-1 …………………………….………………….…… 14

1.3 Graph of Loss of Model-2 ……………………………………………………….…… 15

1.4 Graph of Accuracy of Model-2 …………………………….………………….…… 15

1.5 Graph of Loss of Model-3 ……………………………………………………….…… 16

1.6 Graph of Accuracy of Model-3 …………………………….………………….…… 16

1.7 Graph of Loss of Model-4 ……………………………………………………….…… 17

1.8 Graph of Accuracy of Model-4 …………………………….………………….…… 17

1.9 Graph of Loss of Model-5 ……………………………………………………….…… 18

1.10 Graph of Accuracy of Model-5 …………………………….………….…...….…… 18

List of Tables

4.1 Summary of Result of all models …………………………………………………… 10

**1**

Introduction

* 1. **Background**

With the advancement of technology, digital news is more widely exposed to users globally and contributes to the increment of spreading hoaxes and disinformation online. Fake news can be found through popular platforms such as social media and the Internet. There have been multiple solutions and efforts in the detection of fake news where it even works with artificial intelligence tools. However, fake news intends to convince the reader to believe false information which deems these articles difficult to perceive. The rate of producing digital news is large and quick, running daily at every second, thus it is challenging for machine learning to effectively detect fake news.

In the discourse of not being able to detect fake news, the world would no longer hold value in truth. Fake news paves the way for deceiving others and promoting ideologies. These people who produce the wrong information benefit by earning money with the number of interactions on their publications. Spreading disinformation holds various intentions, in particular, to gain favour in political elections, for business and products, done out of spite or revenge. Humans can be gullible and fake news is challenging to differentiate from the normal news. Most are easily influenced especially by the sharing of friends and family due to relations and trust. We tend to base our emotions from the news, which makes accepting not difficult when it is relevant and stance from our own beliefs. Therefore, we become satisfied with what we want to hear and fall into these traps.

* 1. **LSTM Model**

In theory, classic  RNNs can keep track of arbitrary long-term dependencies in the input sequences. The problem with vanilla RNNs is computational (or practical) in nature: when training a vanilla RNN using back-propagation, the long-term gradients which are back-propagated can "vanish" (that is, they can tend to zero) or "explode" (that is, they can tend to infinity), because of the computations involved in the process, which use finite-precision numbers. RNNs using LSTM units partially solve the vanishing gradient problem, because LSTM units allow gradients to also flow unchanged. However, LSTM networks can still suffer from the exploding gradient problem.

A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. LSTM networks are well-suited to classifying, processing and [making predictions](https://en.wikipedia.org/wiki/Predict) based on time series data, since there can be lags of unknown duration between important events in a time series.

* 1. **CNN Model**

CNNs are [regularized](https://en.wikipedia.org/wiki/Regularization_(mathematics)) versions of [multilayer perceptron](https://en.wikipedia.org/wiki/Multilayer_perceptron). Multilayer perceptron usually mean fully connected networks, that is, each neuron in one [layer](https://en.wikipedia.org/wiki/Layer_(deep_learning)) is connected to all neurons in the next [layer](https://en.wikipedia.org/wiki/Layer_(deep_learning)). The "full connectivity" of these networks makes them prone to [overfitting](https://en.wikipedia.org/wiki/Overfitting) data. Typical ways of regularization, or preventing overfitting, include: penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme.

A convolutional neural network consists of an input layer, [hidden layers](https://en.wikipedia.org/wiki/Multilayer_perceptron#Layers) and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final [convolution](https://en.wikipedia.org/wiki/Convolution). In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically, this includes a layer that performs a [dot product](https://en.wikipedia.org/wiki/Dot_product) of the convolution kernel with the layer's input matrix. This product is usually the [Frobenius inner product](https://en.wikipedia.org/wiki/Frobenius_inner_product), and its activation function is commonly [ReLU](https://en.wikipedia.org/wiki/Rectifier_(neural_networks)). As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

* 1. **LSTM + CNN Models**

This involves using Convolutional Neural Network (CNN) layers for feature extraction on input data combined with LSTMs to support sequence prediction.

CNN LSTMs were developed for visual time series prediction problems and the application of generating textual descriptions from sequences of images or sequence of text document.

This architecture is appropriate for problems that:

* Have spatial structure in their input such as the 2D structure or pixels in an image or the 1D structure of words in a sentence, paragraph, or document.
* Have a temporal structure in their input such as the order of images in a video or words in text, or require the generation of output with temporal structure such as words in a textual description.
  1. **RNN + Attention Models**

Improved RNN models such as Long Short-Term Memory networks (LSTMs) enable training on long sequences overcoming problems like vanishing gradients. However, even the more advanced models have their limitations and researchers had a hard time developing high-quality models when working with long data sequences. In machine translation, for example, the RNN has to find connections between long input and output sentences composed of dozens of words. It seemed that the existing RNN architectures needed to be changed and adapted to better deal with such tasks.

Attention is a mechanism combined in the RNN allowing it to focus on certain parts of the input sequence when predicting a certain part of the output sequence, enabling easier learning and of higher quality. Combination of attention mechanisms enabled improved performance in many tasks making it an integral part of modern RNN networks.

**1.6 Transformer Models**

Attention mechanisms let a model draw from the state at any preceding point along the sequence. The attention layer can access all previous states and weights them according to a learned measure of relevancy, providing relevant information about far-away tokens.

A clear example of the value of attention is in [language translation](https://en.wikipedia.org/wiki/Language_translation), where context is essential to assigning the meaning of a word in a sentence. In an English-to-French translation system, the first word of the French output most probably depends heavily on the first few words of the English input. However, in a classic LSTM model, in order to produce the first word of the French output, the model is given only the state vector of the last English word. Theoretically, this vector can encode information about the whole English sentence, giving the model all necessary knowledge. In practice this information is often poorly preserved by the LSTM. An attention mechanism can be added to address this problem: the decoder is given access to the state vectors of every English input word, not just the last, and can learn attention weights that dictate how much to attend to each English input state vector.

When added to RNNs, attention mechanisms increase performance. The development of the Transformer architecture revealed that attention mechanisms were powerful in themselves, and that sequential recurrent processing of data was not necessary to achieve the performance gains of RNNs with attention. Transformers use an attention mechanism without an RNN, processing all tokens at the same time and calculating attention weights between them in successive layers.

**2**

Review of Literature

* 1. **LSTM + CNN Models**

this

* 1. **RNN + Attention Models**

this opportunity to express my sincere and profound gratitude to him.

* 1. **Transformer Models**

this opportunity to express my sincere and profound gratitude to him.

**3**

Implementation

* 1. **Data Pre-processing**

The text data which is available to us for using in the fake news detection is full of noisy information and present in format that can’t be directly used by our algorithms. So, we have done the data pre-processing to transform the data into a useable form.

We started with removing the null values present in the dataset. We have dropped the entire row in which any column is null.

We have expanded the contaction(short forms and shorthands) used in the english text using contractions library of python. We converted the text into lowercase and have split the text into word list.

We have removed all form of puctuation and stop words from the processed word list.

We have removed all special charecters and numbers from the processed word list.

We have removed all non-english words from the word list and then again convert it back to text.

Till now we have removed the noise and unwanted data. Now we will be formatting the text to be used by models. We have use tfidf vectorizer to extract the features from the corpus(processed text) and convert it to word vector.

We have also created a separate form of dataset using tokenizer instead of vectorizer.

At last, we save both the format using pickle library of python. We have generated both the format used by common ML algorithms

We tested the performance of our datasets using Multinomial Naïve Bayes Classifiers.

We selected one of the datasets among 3 based on the performance. Now we are ready to proceed with preparing the selected dataset for Deep Learning.

For preparing the data for Deep Learning, we will be using stemming instead of vectorization or tokenization. We will be using a python library, Port Stemmer.

We used one-hot encoder to encode the words and then generate the embedding matrix.

*pickle.dump(embedded\_doc, open("../Dataset/embedded\_doc.pickle", "wb"))*

*pickle.dump(embedded\_doc\_test, open("../Dataset/embedded\_doc\_test.pickle", "wb"))*

We convert the matrix into NumPy array before feeding it to network.

*X\_final=np.array(embedded\_doc)*

*y\_final=np.array(y\_train)*

*X\_final\_test=np.array(embedded\_doc\_test)*

*y\_final\_test=np.array(y\_test)*

* 1. **LSTM + CNN Models**

We will be analysing performance of different architecture of combination of CNN + LSTM networks.

**Model-1**

*embedding\_vector\_feature = 100*

*model=Sequential()*

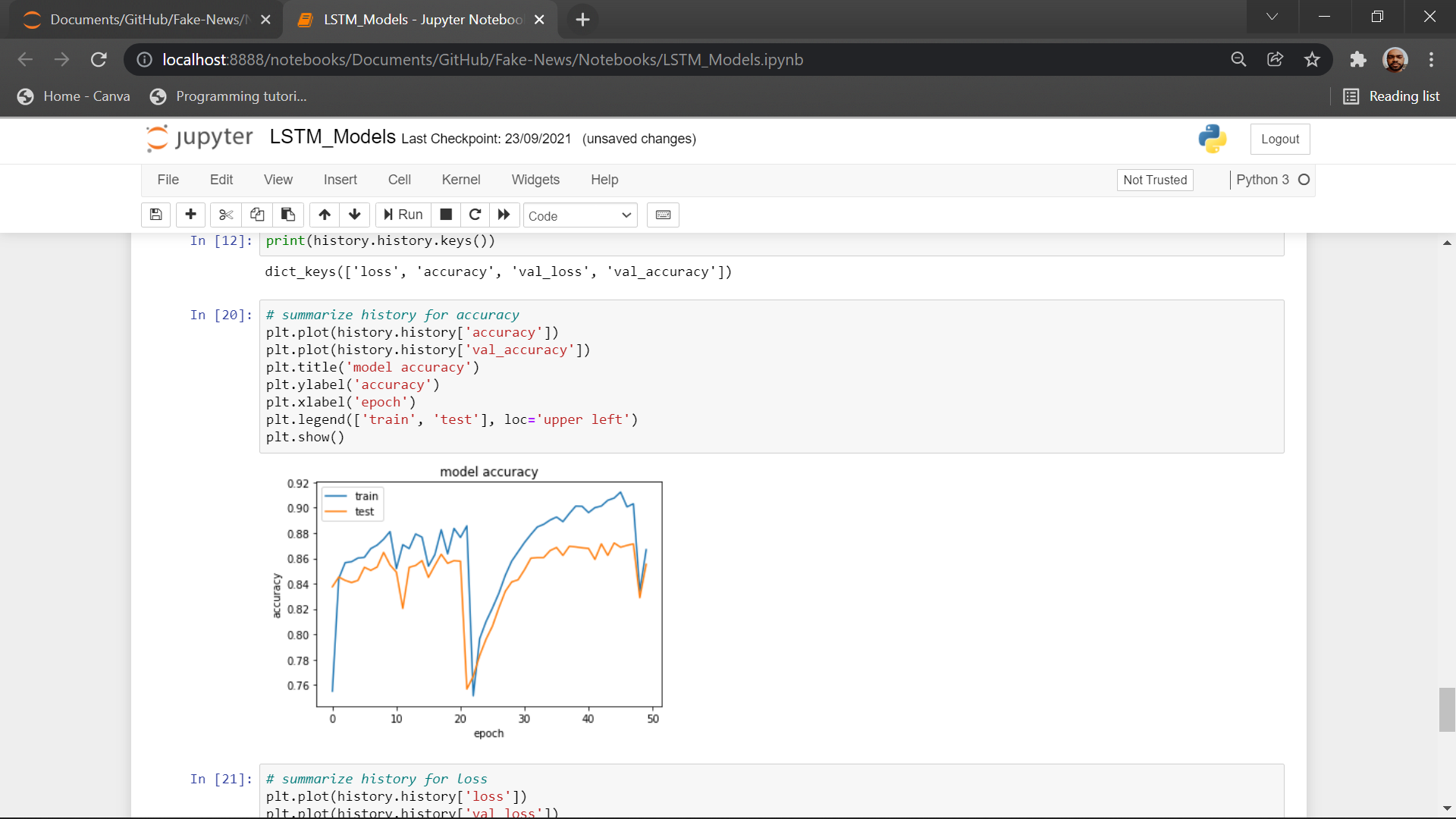
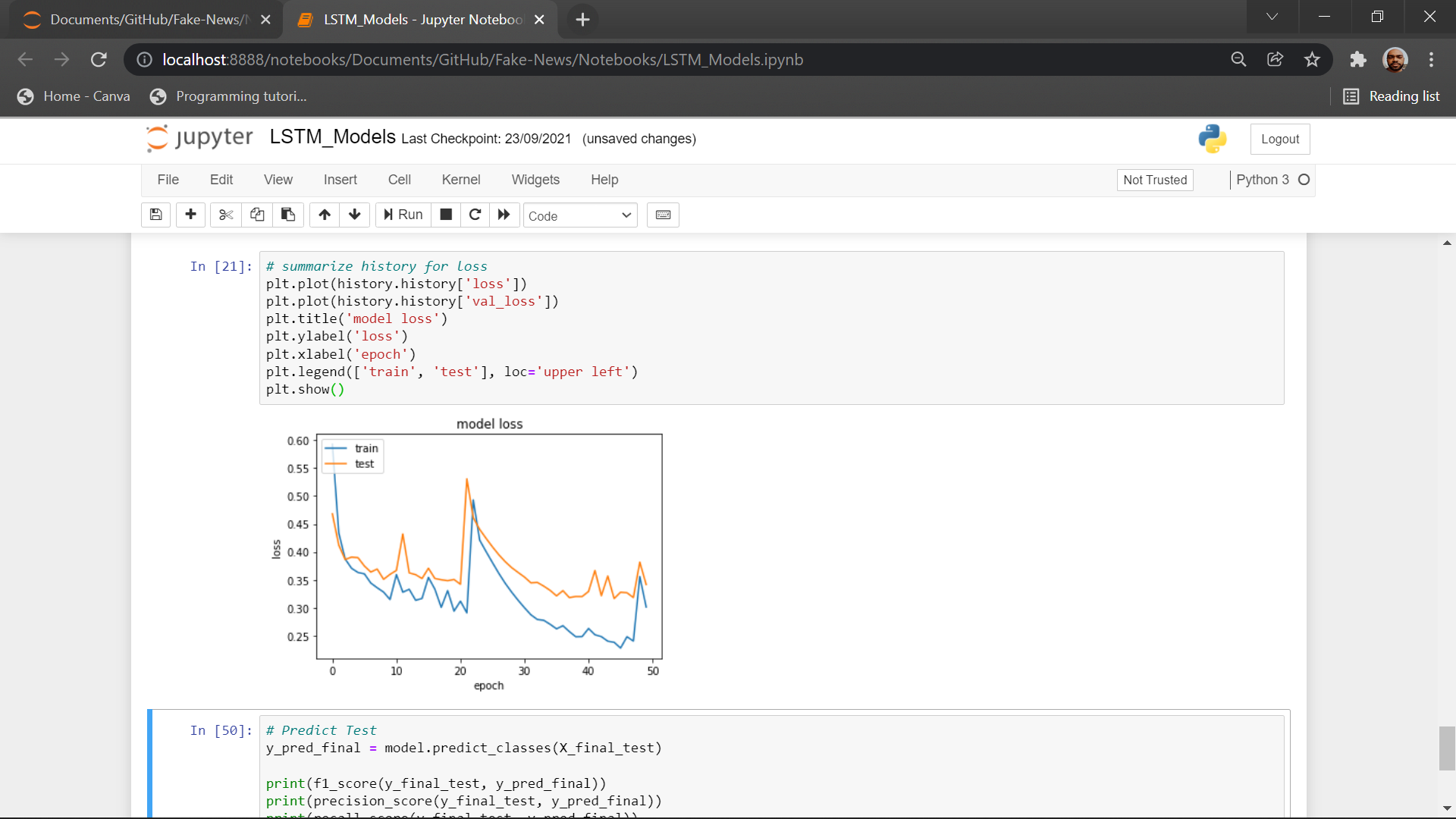
*model.add(Embedding(vo\_size,embedding\_vector\_feature,input\_length=sent\_length))*

*model.add(LSTM(10))*

*model.add(Dense(1,activation='sigmoid'))*

*model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])*

*print(model.summary())*

**

*Figure 1.1 Figure 1.2*

It is a simple LSTM model with 100 features. We observe that its validation accuracy is 86% though we have the training accuracy 2-3% higher. There is a good amount of oscillation in the accuracy during training and there is steep change in curve at few points.

**Model-2**

*embedding\_vector\_feature = 300*

*model=Sequential()*

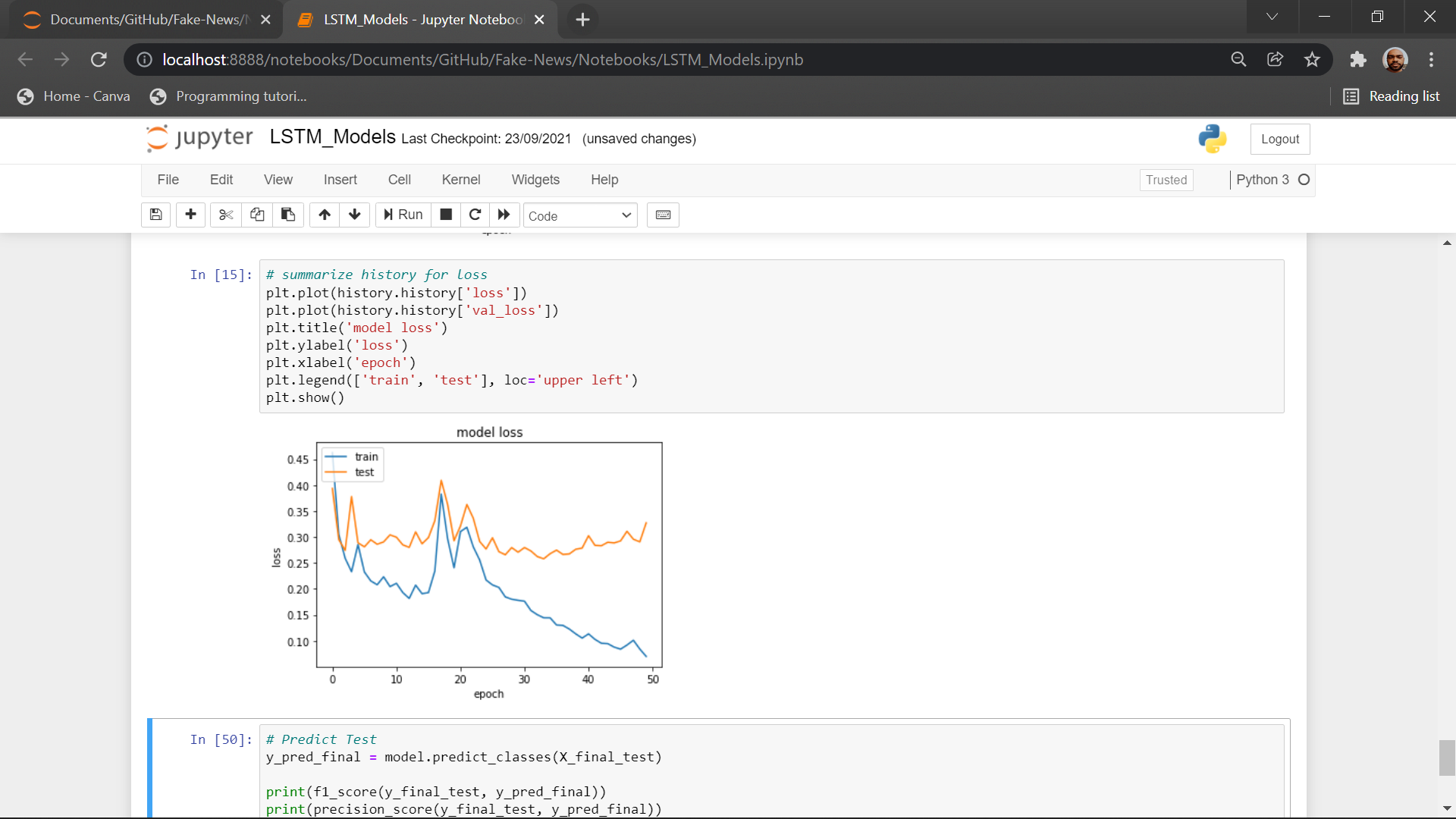
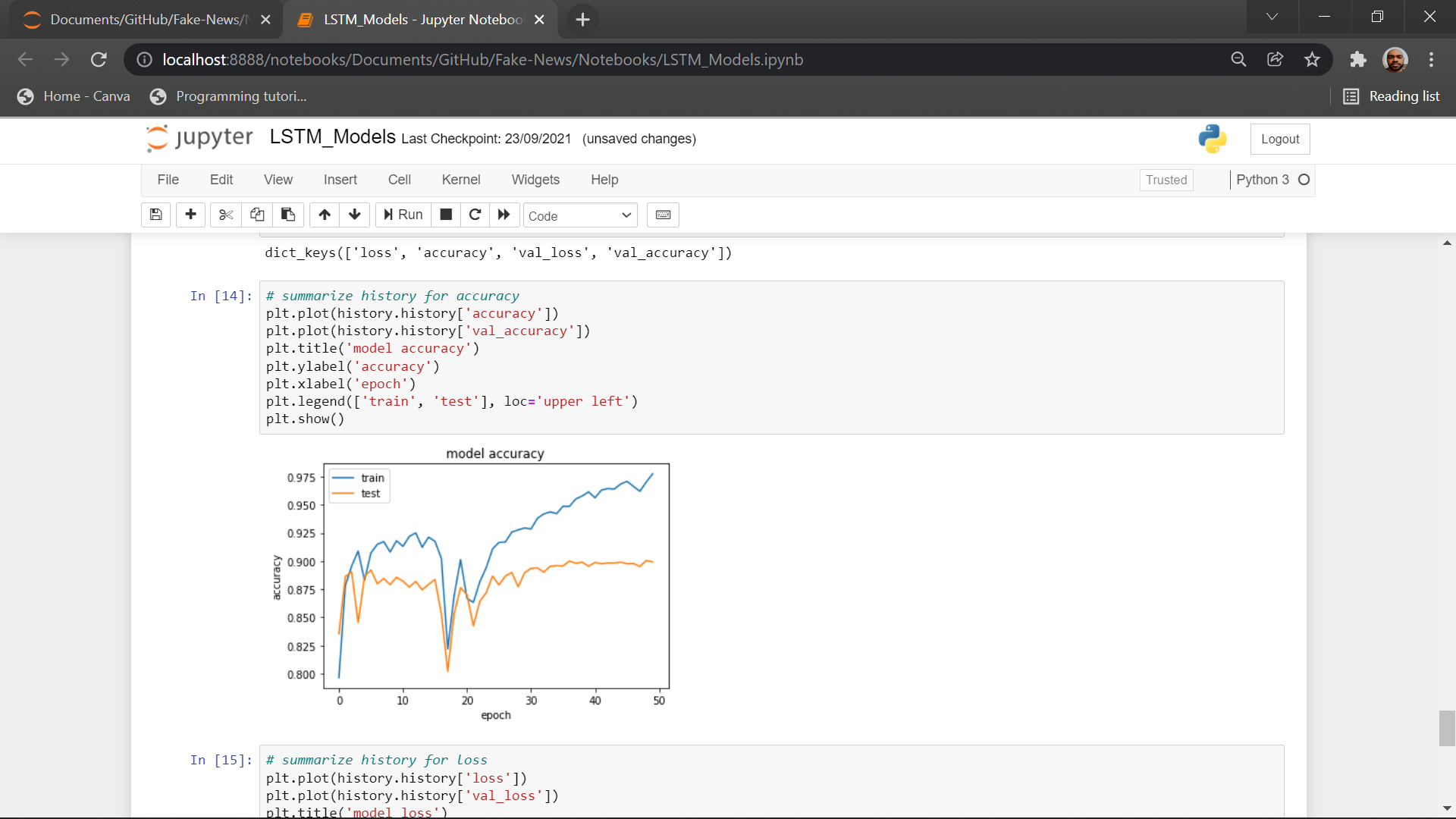
*model.add(Embedding(vo\_size,embedding\_vector\_feature,input\_length=sent\_length))*

*model.add(LSTM(50))*

*model.add(Dense(1,activation='sigmoid'))*

*model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])*

*print(model.summary())*

**

*Figure 1.3 Figure 1.4*

This is an upgrade of previous model with a greater number of features and neurons. It has a significant increase in the accuracy but takes a large amount of computation power due to large network.

**Model-3**

*embedding\_vector\_feature = 300*

*model=Sequential()*

*model.add(Embedding(vo\_size,embedding\_vector\_feature,input\_length=sent\_length))*

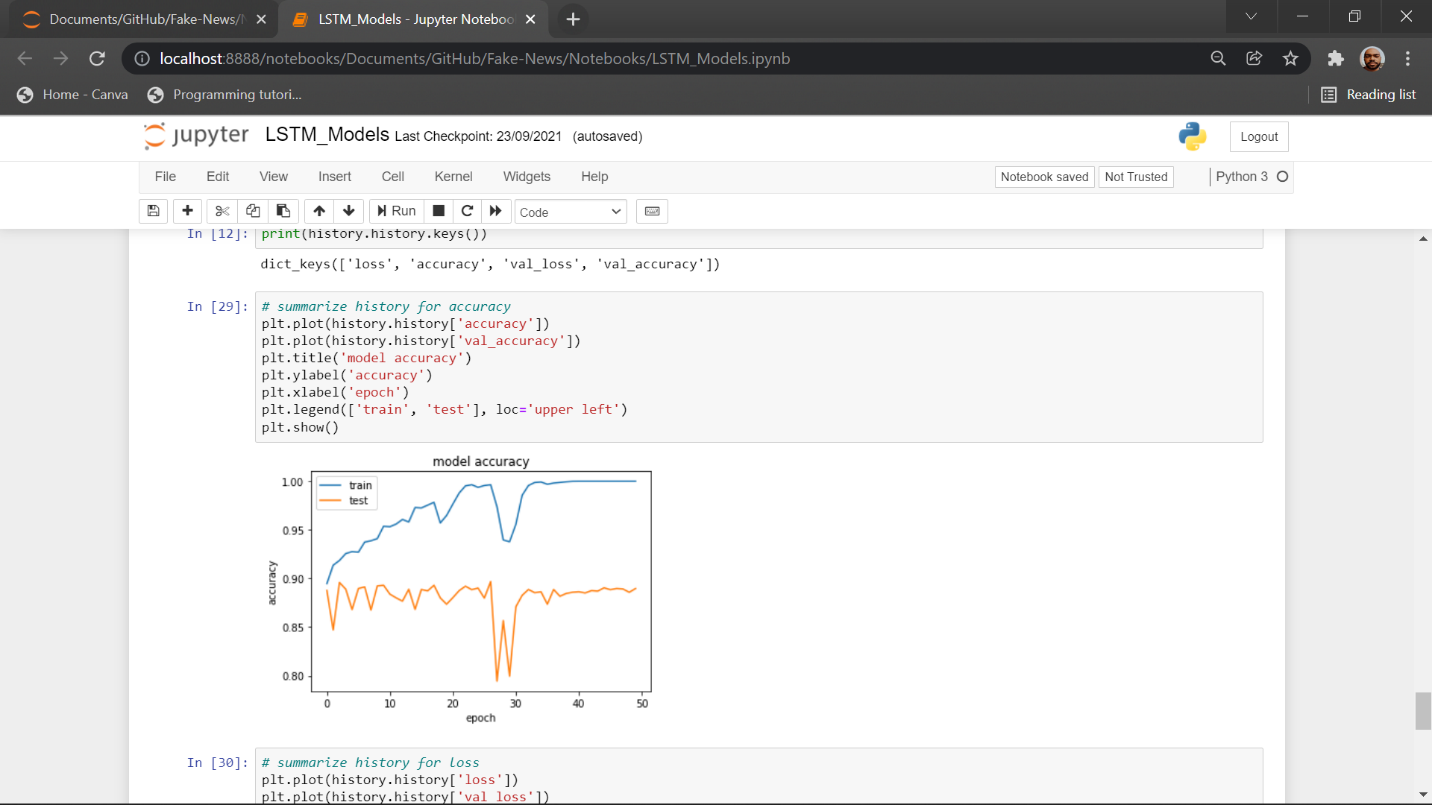
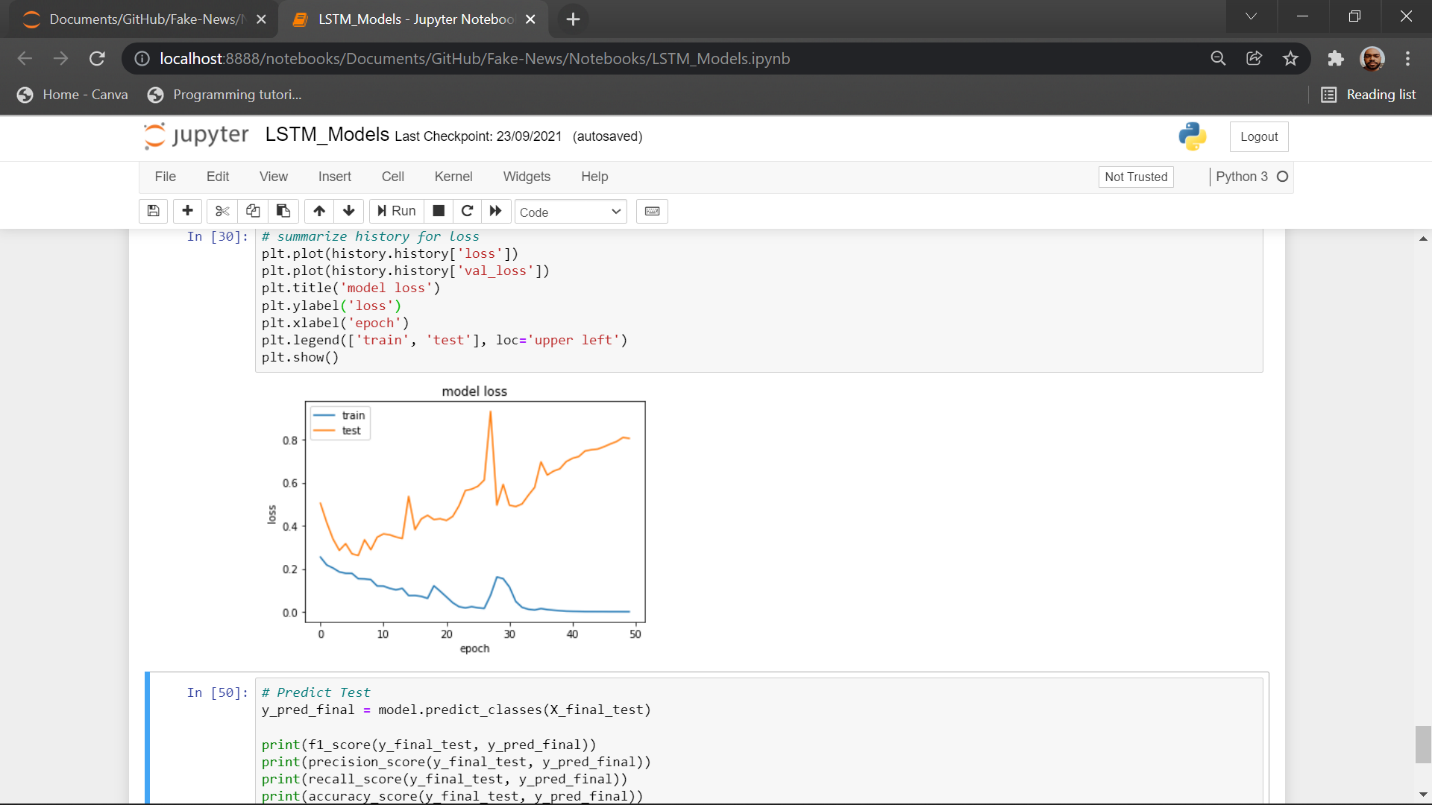
*model.add(LSTM(50))*

*model.add(BatchNormalization())*

*model.add(Dense(1,activation='sigmoid'))*

*model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])*

*print(model.summary())*

**

*Figure 1.5 Figure 1.6*

It is an upgrade of previous model with additional batch normalization layer. On adding batch normalization, it has boosted the overall accuracy of the previous model. It is overfitting on the training set that can be seen in the Figure 1.6.

**Model-4**

*embedding\_vector\_feature = 300*

*model=Sequential()*

*model.add(Embedding(vo\_size,embedding\_vector\_feature,input\_length=sent\_length))*

*model.add(Conv1D(filters=25, kernel\_size=5, padding='same', activation='relu'))*

*model.add(MaxPooling1D(pool\_size=2))*

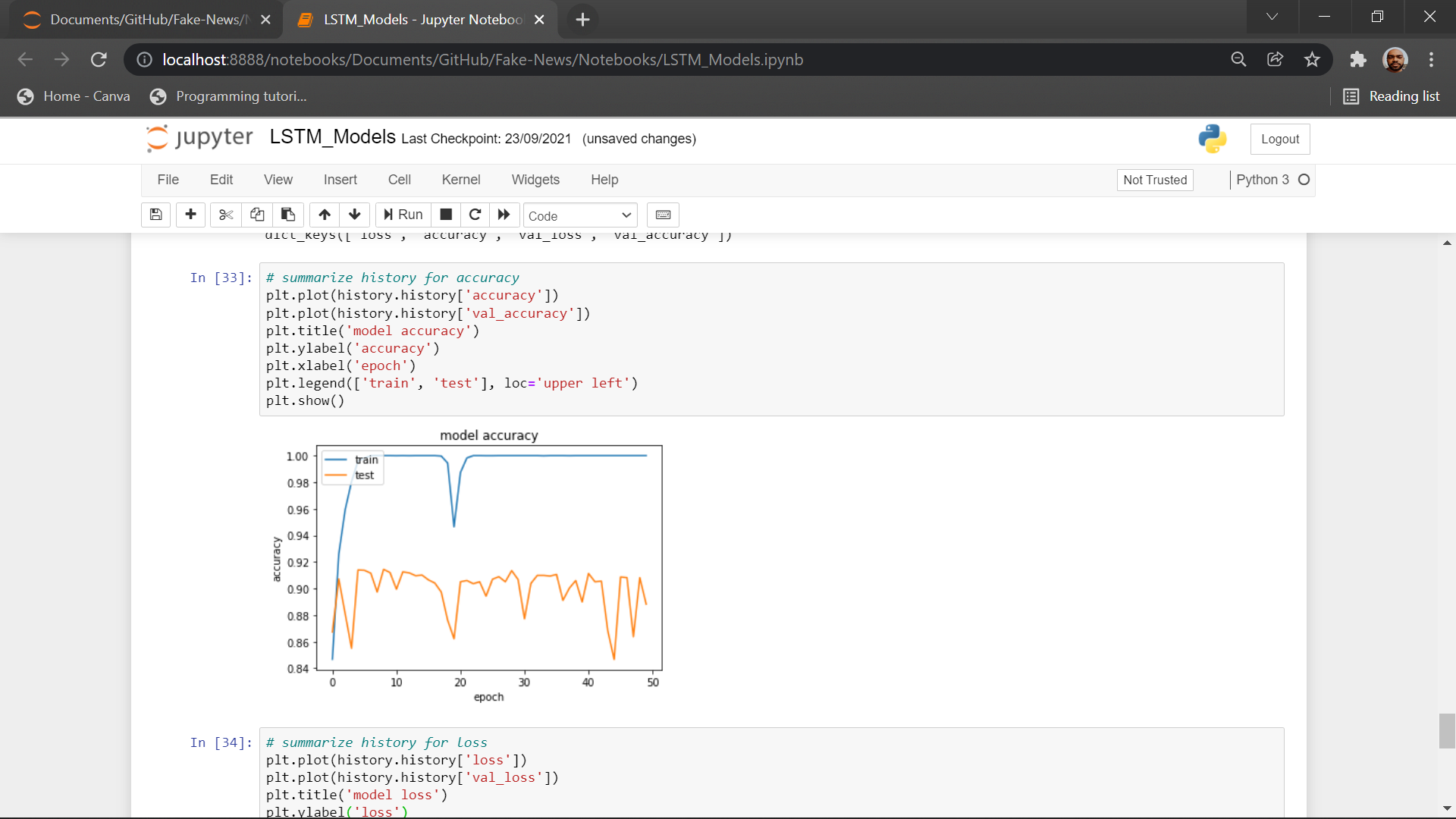
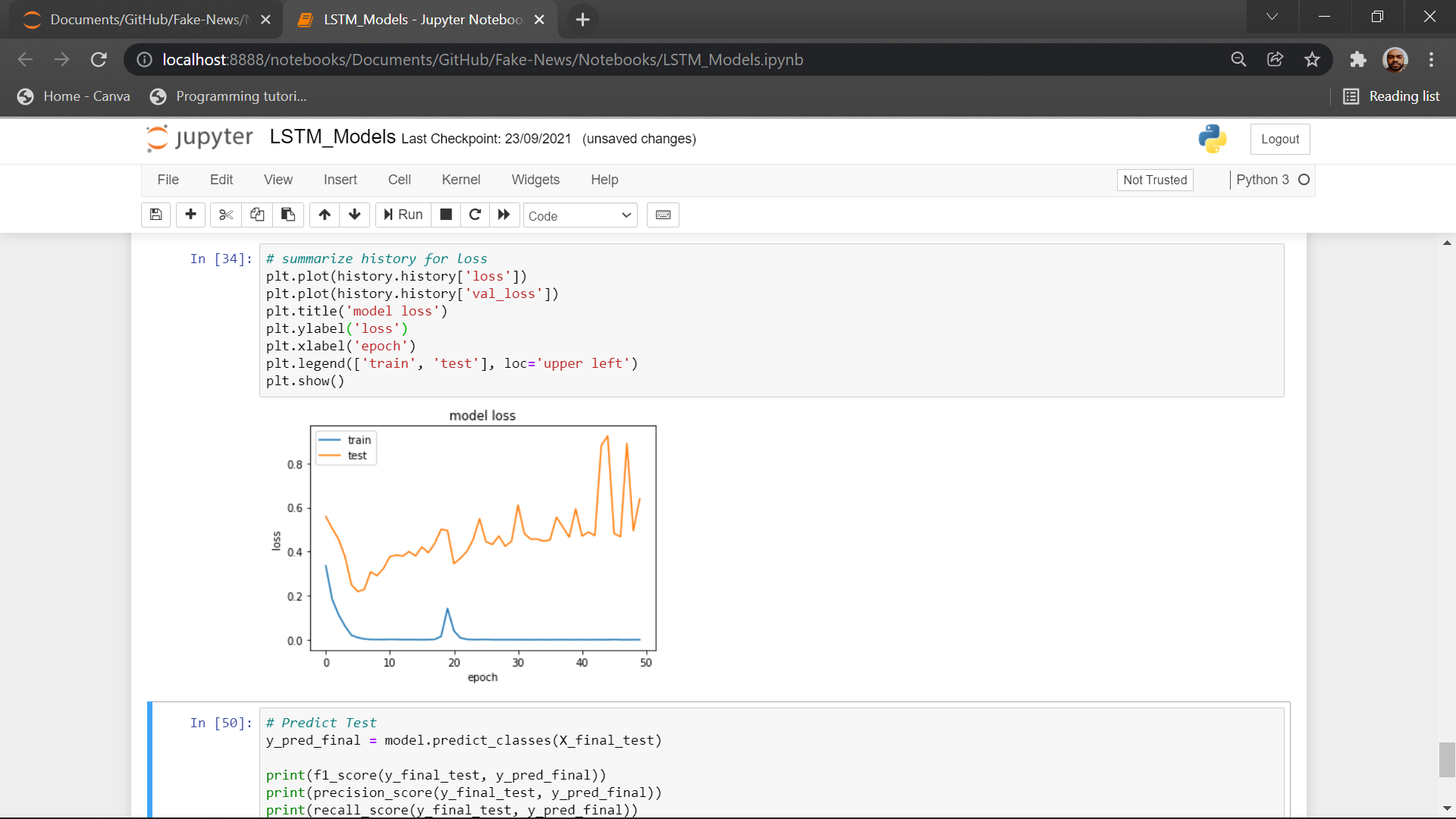
*model.add(LSTM(50))*

*model.add(BatchNormalization())*

*model.add(Dense(1,activation='sigmoid'))*

*model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])*

*print(model.summary())*

**

*Figure 1.7 Figure 1.8*

It is an upgrade of previous model with additional CNN network. CNN input the is used for detecting the features and then more refined details is passed to LSTM boosting its accuracy to 91%. Though we have used batch normalization its is not required that will be clear from our final model.

**Model-5**

*embedding\_vector\_feature = 100*

*model=Sequential()*

*model.add(Embedding(vo\_size,embedding\_vector\_feature,input\_length=sent\_length))*

*model.add(Conv1D(filters=25, kernel\_size=5, padding='same', activation='relu'))*

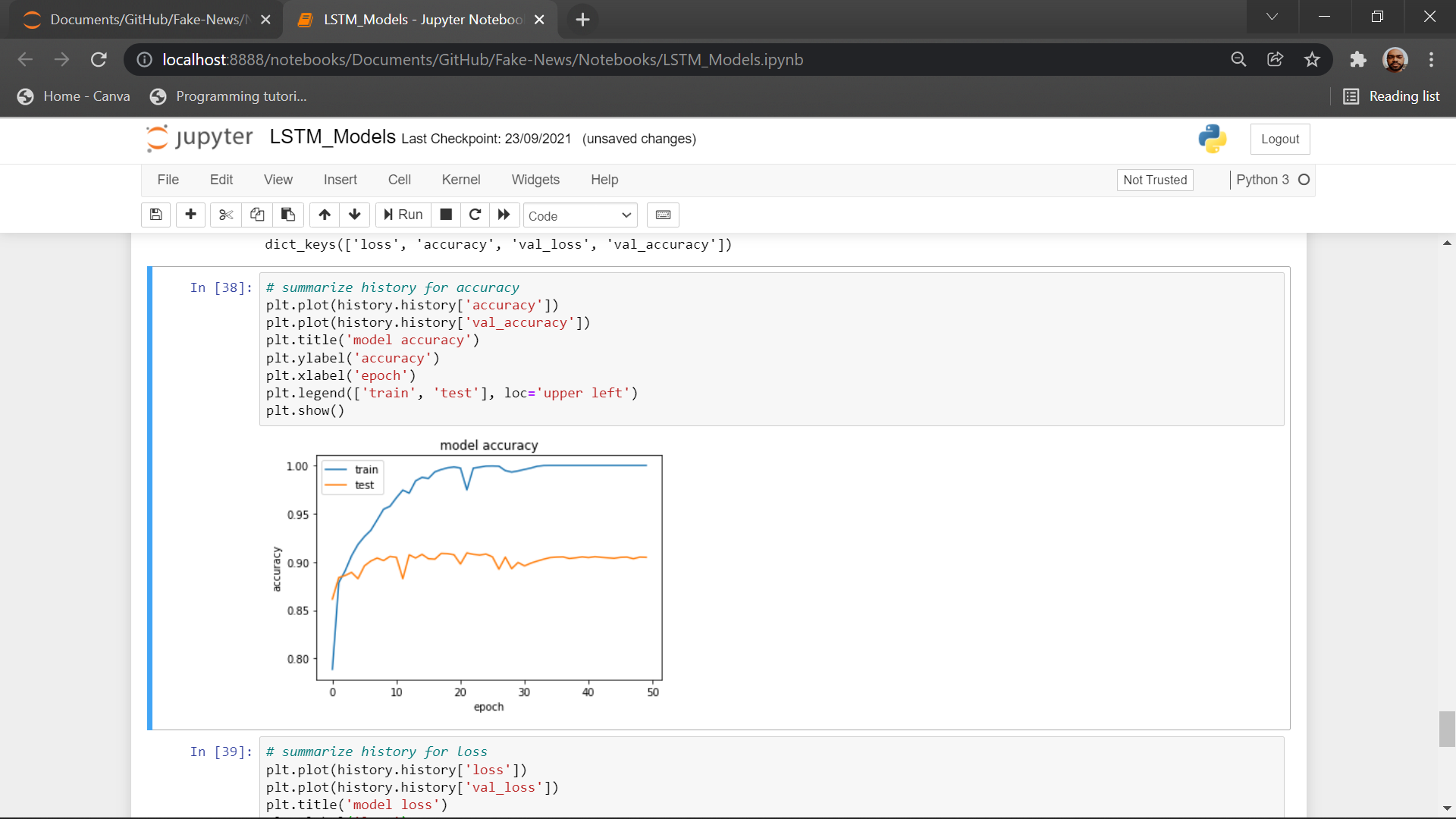
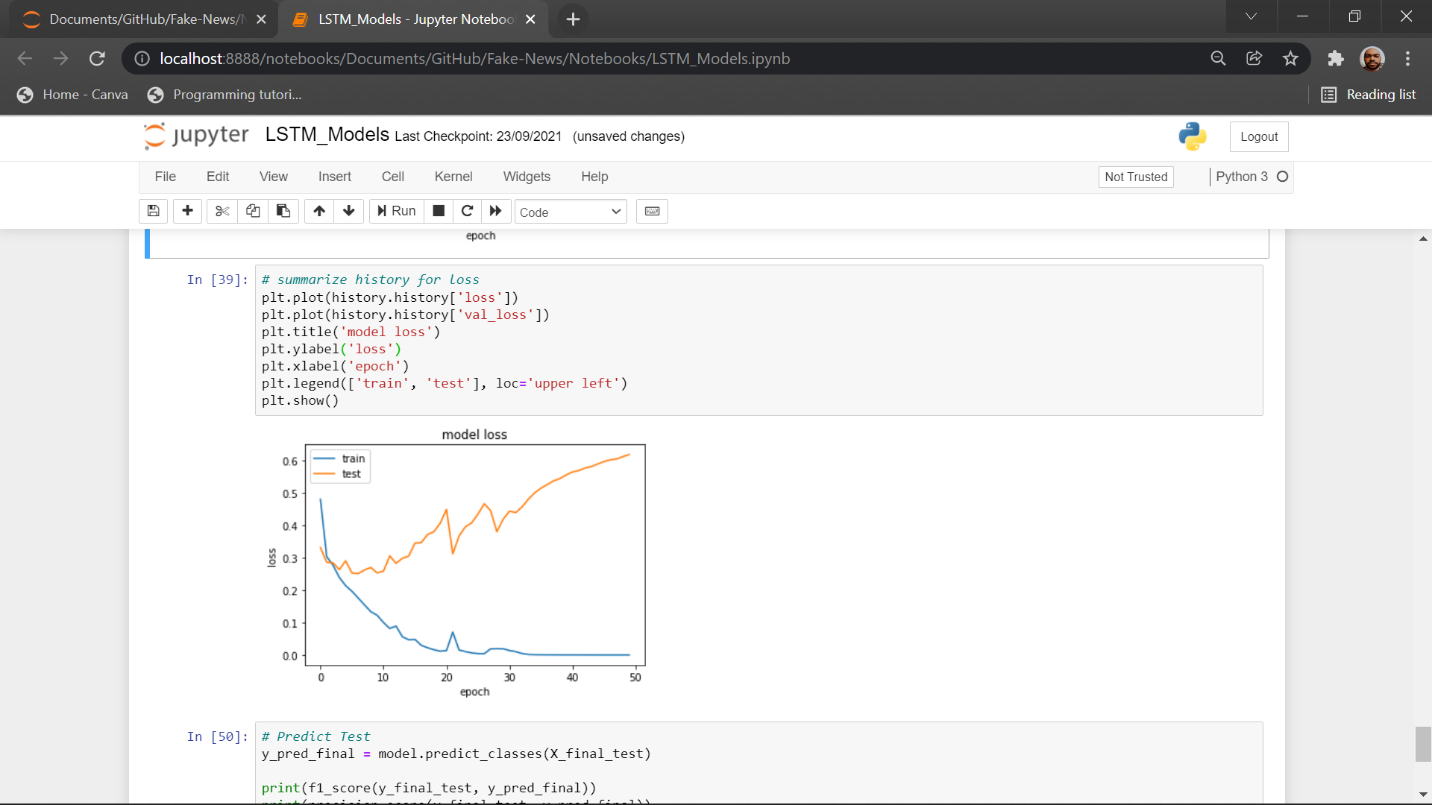
*model.add(MaxPooling1D(pool\_size=5))*

*model.add(LSTM(25))*

*model.add(Dense(1,activation='sigmoid'))*

*model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])*

*print(model.summary())*



*Figure 1.9 Figure 1.10*

It is a downgrade of previous model with decrease in number of neurons and features and removal of batch normalization. We can observe that our final model uses much less computing power and features for same accuracy(as of only LSTM Models). Batch normalization is not required as the LSTM layer is already stabilized by the CNN layer.

* 1. **RNN + Attention Models**

**Model-6**

this opportunity to express my sincere and profound gratitude to him.

* 1. **Transformer Models**

**Model-6**

this opportunity to express my sincere and profound gratitude to him.

**4**

Results and Discussion

* 1. **Result**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Val. Acc.** | **Train Acc.** | **Val. Loss** | **Train Loss** |
| LSTM only(Model-1) (100 features) | 86% | 88% | 0.30 | 0.35 |
| LSTM only(Model-2) (300 features) | 89% | 94% | 0.12 | 0.26 |
| LSTM only(Model-3) (300 features)(Batch Normalization) | 89% | 98% | 0.50 | 0.10 |
| LSTM + CNN (Model-4) (300 features) (Batch Normalization) | 91% | 99% | 0.50 | 0.05 |
| LSTM + CNN (Model-4) (100 features) | 90% | 99% | 0.45 | 0.05 |
|  |  |  |  |  |

*Table 4.1*

* 1. **Discussion**

this opportunity to express my sincere and profound gratitude to him

**5**

Conclusion and Future Work

* 1. **Conclusion**

In our study we started with simple LSTM models and tested their performance which was proportional to the number of features and the size of network. But still it produced maximum accuracy up to 89%. We observed that LSTM + CNN models are capable of producing accuracy up to 90% with much smaller network compared to simple LSTM model. But there was a problem of overfitting indicated of accuracy of training set(99%). We used RNN + Attention model which has accuracy about 95%. Then while studying attention model further we found that attention model is itself sufficient to give the result. Then we switched to transformer models which has accuracy up to 99%. In our study we have able to find out how gradually we progressed from LSTM models to transformer models. Our study will help the future researcher to understand the how these models are derived from its predecessor models and what improved its performance from its predecessor.

* 1. **Future Work**

We are planning to work on detecting fake news shown in form of videos. We will be using our knowledge of text-based fake news detection and speech to text conversion. We will convert the speech in video into text, and then try to predict whether the news is fake or not. We may develop algorithm to identify the fake speaker and then warn the users against him/her. We have not yet created a data pipeline for our models. Our next work will contain data pipeline to automate the entire process of fetching data and converting it to required form. We will need to create the dataset as we have not found a suitable dataset for detecting fake news shown in form of videos.

List of Abbreviations

**NN** Neural Network

**LSTM** Long Short-Term Memory

**GRU** Gated Recurrent Unit

**Val.** Validation

**Acc.** Accuracy

References

[1] Xinyi Zhou and Reza Zafarani. 2020. A Survey of Fake News: Fundamental Theories, Detection Methods, and Opportunities. ACM Comput. Surv. 53, 5, Article 109 (September 2020). https://doi.org/10.1145/3395046

[2] Ayat Abedalla, Aisha AI-Sadi, and Malak Abdullah. 2019. A Closer Look at Fake News Detection: A Deep Learning Perspective. ICAAI Proceedings of the 2019 3rd International Conference in Artificial Intelligence (October 2019). https://doi.org/10.1145/3369114.3369149

[3] Jibran Fawaid, Asiyah Awalina, and Rifky Yunus Krishnabayu . 2021. Indonesia’s Fake News Detection using Transformer Network. SIET. 6th International Conference on Sustainable Information Engineering and Technology (September 2021) <https://doi.org/10.1145/3479645.3479666>

[4] Oluwaseun Ajao, Deepayan Bhowmik, and Shahrzad Zargari. 2018. Fake News Identification on Twitter with Hybrid CNN and RNN Models. SMSociety. 9th International Conference on Social Media and Society. (July 2018). https:// [doi.org/10.1145/3217804.3217917](https://doi.org/10.1145/3217804.3217917)