REAL AND FRADULENT JOB POSTING PREDICTION USING MACHINE LEARNING TECHNIQUES

A PROJECT REPORT

Submitted By

KHIRAN G	**Reg.No: RA1711003010717**
MAGESHWAR.S	**Reg.No: RA1711003040087**
KRISHNA SAI.M	**Reg.No: RA1711003040088**

Under the guidance of

Mrs.K.Karthikayani, B.Tech, M.Tech.,

(Assistant Professor, Department of Computer Science & Engg.)

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGG.,

of

FACULTY OF ENGINEERING AND TECHNOLOGY

Vadapalani Campus, Chennai - 26.

MAY 2021

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Vadapalani Campus, Chennai - 26.

BONAFIDE CERTIFICATE

Certified that this project report titled **"Real and Fradulent Job Posting Prediction using Machine Learning Techniques"** is the bonafide work of **"KHIRAN.G (RA1711003010717), MAGESHWAR.S (RA1711003040087), KRISHNA SAI.M (RA1711003040088)"** who carried out the report work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE OF THE SUPERVISOR

Mrs.K.Karthikayani B.Tech, M.Tech.,

SUPERVISOR
Assistant Professor
Department of Computer Science and Engineering
SRM Institute of Science & Technology
Vadapalani Campus

SIGNATURE OF THE HOD

Dr.S.Prasanna Devi, B.E.,M.E., Ph.D., PGDHRM.,PDF(IISc)
Professor & Head
Department of Computer Science and Engineering
SRM Institute of Science & Technology
Vadapalani Campus

Signature of the Internal Examiner

Signature of the External Examiner

ABSTRACT

Due to Covid-19 Pandemic, Every sector has moved to virtual mode including recruitment for various job postings in public and private sector. This allows everyone to reduce their dependency on manual efforts and since all the job postings are posted online, it gives the recruiting company or agency a wide range of area to gather the talented candidates. People who are seeking for job can also know the information of the recruiting company /sector through the internet. But all job postings which are posted in the internet recruiting sites are not real and there are fraudulent job postings also. So we try to predict the fraudulent posting and real one. So the aim of this project is to use machine learning based techniques for real or fake job prediction results in best accuracy. Here we propose a machine learning-based method to accurately predict the fraudulent jobs in the form of whether the job is real or not.

ACKNOWLEDGEMENT

First and foremost, we would like to thank God Almighty for giving us the strength, knowledge, ability and opportunity to undertake this project work and to persevere and complete it to the best of our ability

.

We place our deep sense of gratitude and thank our management for providing us with a good infrastructure and learning environment throughout the course.

We take the opportunity to extend our heartfelt thanks to our respected Dean (E & T), Dr.K.Duraivelu, Ph.D., for his support throughout the four years of our BTech course.

We like to express our special thanks to our Head of the Department, Dr. S. Prasanna Devi,Ph.D, for encouraging and helping us throughout towards successful completion of our course.

We would also like to show our greatest appreciation to our Project Coordinators, Dr.P.Mohamad Fathimal,B.E.,M.E,PHD, and Mr.Manikannan, B.E., M.E. for the tremendous support and help offered every week when we had project discussion. Without their encouragement and guidance, this project would not have materialized.

We thank our Project Supervisor Mrs.Karthikayani, B.Tech, M.Tech. for her supervision, feedback, and mentoring. Without her support, it would have been tough for us to finish this project in a timely manner. Thus, we feel deeply obliged for her support.

We would like to thank the guidance and support received from all the members including our parents and friends who contributed to this project was vital for the success of the project. We are grateful for their constant support and help.

We would like to thank the guidance and support received from all the members including our parents and friends who contributed towards successful completion of this project

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER-1
INTRODUCTION

1.1 OVERVIEW

There are many work ads on the internet, including on well-known job posting pages, that never seem to be fraudulent. However, following the selection, the so-called employment agencies begin requesting money and financial account information. Lot of applicants slip into their hole and risk a significant amount of money as well as their present work. As a result, it is preferable to determine if a work application shared on the internet is genuine or not. It's very difficult to manually recognise it. For real and fake job prediction, we can use machine and deep learning algorithms.

1.2 OBJECTIVE

Employment scam is one of the serious issues in recent times addressed in the domain of Online Recruitment Frauds (ORF).So we predict the real or fake job posting in internet by implementing machine learning method and display the result

CHAPTER-2

SYSTEM ANALYSIS

2.1) EXISTING SYSTEM

In the OSN, a mathematical model was suggested to analyse the hierarchical spreading and regulating behaviours of message propagation. The proposed model uses differential equations to investigate the impact of user authentication and blocking, as well as message spread on OSNs. It explains how different misinformation refuting initiatives affect how misinformation spreads across communities. This model relies on a variety of scourge groups and features two levels of control instruments to keep the snitch as in laid-back neighbourhood under control. The fact implies that all clients are powerless, implying that everyone can manipulate a precariousness or untrustworthy post. Clients are affirmed using a verified class from the beginning for assertion. As a result, before accepting any user's request, the user authentication mechanism is used, and the message authenticity from this user is assessed in order to reduce malicious user activity on the OSN. If R0 is less than one (R0 1), the dissemination of false messages throughout the online network will be minimal; but, if R0 > 1, the rumour will continue in the OSN.

2.2) DRAWBACKS

1) This is just a mathematical model based on differential equations and it does not classify the fake news.
2) Accuracy, Recall F1 score metrics are not calculated and machine learning algorithms are not applied which enhances the model credibility to classify fake information in online social network.

2.3) PROPOSED SYSTEM

1. The proposed method is built a machine learning model to predict the real or fake job posting.
2. To implement this method we implement machine learning approach
3. The dataset is first preprocessed and the columns are analyzed to see the dependent and independent variable and then to extract patterns and produce the most accurate results, a variety of machine learning algorithms can be used.

2.4) PROBLEM STATEMENT

1. Employment scam is one of the serious issues in recent times addressed in the domain of Online Recruitment Frauds (ORF).
2. So we predict the real or fake job posting in internet by implementing machine learning method.

CHAPTER-3

LITERATURE SURVEY

3.1) TITLE: Fake News and Information in Online Social Networks

Authors: Gulshan Shrivastava , Prabhat Kumar, Rudra Pratap Ojha , Pramod Kumar Srivastava , Senthilkumar Mohan , Gautam Srivastava

Year: 2020(IEEE)

This paper proposes a mathematical model to analyse the complex spreading and regulating behaviours of message propagation in Online social Networks. The proposed model investigates the effects of user authentication and blocking, as well as the distribution of messages on Online social networks, using differential equations. The simple reproduction R0 expression is collected, and it is used to evaluate the position of gossip in the social network. According to the findings, if R0 is less than 1, rumours and false news are excluded, and OSNs are stabilised locally. The Jacobian matrix determines the rumorfree equilibrium'e local stability. The network would asymptotically stabilise locally in essence and will be free of rumours if the eigenvalues of the matrix are less than zero. The Lyapunov function was used to determine the social network's global asymptotic stable status.The social network has also been used to investigate the behaviour of various consumer groups. In the future, the latent and separation methods may be used to protect social networks from gossip dissemination and false news spreading. The topics discussed in this essay are of immediate importance, and the COVID-19 pandemic is causing a global crisis of misinformation and false news spreading widely on Online social networks, which will persist until the pandemic is cured/handled. Fake news dissemination can be disruptive to individuals, corporations, and many other aspects of culture, according to real-world statistics.

3.2) TITLE: Identify Fake News on Social Networking

AUTHORS: Nicollas R. de Oliveira, Dianne S. V. Medeiros

YEAR: 2020(IEEE)

This paper introduced a computational investigation, in view of regular language handling, productively applying solo learning calculations, for example, one-class SVM, in distinguishing counterfeit report in messages removed in web-based media. They proposed inorder to register to unique information as depth and width decrease procedure, by inactive meaningful investigation(LSA),information densification via our suggested technique. Three different news classification methodologies were

implemented – two employing cascading or unique configurations of learning algorithms and the other statistically evaluating the difference between the types of news. They proposed analysis based on natural language processing, efficiently applying machine learning algorithms to detect fake news in texts extracted from social media. The analysis considers news from Twitter, from which approximately 33,000 tweets were collected, assorted between real and proven false. In assessing the quality of detection, 86% accuracy, and 94% precision stand out even employing a dimensional reduction to one-sixth of the number of original features.

3.3) TITLE: Spammer Detection and Fake User Identification on Social Networks

AUTHORS: FAIZA MASOOD , GHANA AMMAD, AHMAD ALMOGREN ,ASSAD ABBAS , HASAN ALI KHATTAK

YEAR: 2019(IEEE)

They conducted a study of the methods used to identify spammers on Twitter. Furthermore, we implemented a scientific classification of Twitter spam detection approaches, dividing them into four categories: phoney substance venue, URL-based spam recognition, spam discovery in moving points, and phoney client discovery procedures. We also took a look at the newly implemented procedures. Client highlights, material highlights, map highlights, structure highlights, and time highlights are only a few examples. Furthermore, the procedures were predetermined priorities and databases were also taken into consideration. Analysts should be able to find data on best-in-class Twitter spam location tactics in a structured manner thanks to the newly launched audit. Amid the advancement of effective and productive, convincing methodologies for junk email discovery along with bogus client distinguishing proof on Twitter, there's still some untapped fields that need expert look. Currently, the following topics are highlighted: The genuine repercussions of false news distinguishing evidence from web-based media should be investigated as a result of the genuine repercussions of such news at both the individual and aggregate levels. The identification of gossip outlets from online media is a related topic worth investigating.

3.4) **TITLE: Job Failure in Cloud Based on Online Extreme Learning Machine**

AUTHOR: CHUNHONG LIU, JINGJING, YANLEI SHANG , CHUANCHANG LIU , BO CHENG

YEAR: 2017(IEEE)

A new position disappointment forecast strategy on the enormous distributed computing stage was on the basis of a digital limit training system, a new idea was introduced. The method of using a virtual incremental training system resulted in a rapid learning rate and a high level of speculation. The time and precision of the presentation of the proposed strategy were contrasted and those of some best-in-class strategies. The outcomes showed that the proposed model will guarantee device refreshing in less than 0.01 seconds and predict the working end with a 93 percent accuracy rate Hence, it can lessen the extra room overhead by keenly recognizing position disappointment, and fundamentally diminish the asset squander in the cloud. The current AI-based expectation strategies generally embrace disconnected working examples, which can't be utilized for online forecast in down-to-earth activities in which information shows up consecutively. To take care of this issue, This paper proposes the use to predict digital occupation end report.

3.5) **TITLE: Rumor Influence Minimization in Social Networks**

AUTHORS: Biao Wang, Ge Chen, Luoyi Fu, Li Song, and Xinbing Wang

YEAR-2017(IEEE)

To formulate the problem, we propose the complex gossip effect minimization with user interface model. Centered on the Ising model, a hierarchical rumour diffusion model is proposed that incorporates both global rumour popularity and person inclination. Then, to calculate the relationship between utility and blocking time, we suggest a tweaked version of the utility function. Then, under the constraint of user interface utility, we use survival theory to investigate the probability of nodes being triggered. To solve the optimization problem based on various node selection techniques, a greedy algorithm and

a dynamic blocking algorithm are suggested. Experiments on real-world social networks demonstrate the effectiveness of our approach. In our future work, we plan to design more sophisticated rumor blocking algorithms considering the connectivity of the social network topology and node properties. We intend to separate the entire social network into different communities with different user interests and then analyze the rumor propagation characteristics among communities. We are also interested in investigating how to prevent the rumor propagation effectively at a late stage.

3.6) TITLE: Detecting Fake News with Weak Social Supervision

AUTHORS: Kai Shu , Ahmed Hassan Awadallah , Susan Dumais , and Huan Liu

YEAR: 2020(IEEE)

In numerous AI applications, checked information seems to be insufficient, getting many names is quite tedious. We suggest another form of vulnerable government, feeble group administration, based on the exciting early delayed effects of violating inadequate oversight learning. We distinctively revolve around the use of internet platforms as an example of understanding fake news. Limited named data is getting maybe the greatest challange for coordinated research structures. That's also generally circumstance for some genuine endeavors where huge degree named models are either too expensive to even consider evening consider acquiring or blocked off caused by security, information access prerequisites. By using poor names or inserting targets from probing criteria and moreover outward data streams, delicate administration also seems to be incredible in easing the deficit of named data. Electronic communication has information, notwithstanding, has phenomenal credits that make it sensible for creating slight oversight, achieving another sort like ineffective administration, i.e., frail society administration. They explain how different forms of social communication may be seen as a poor social oversight in this post. They demonstrate that fragile community oversight is fair while standing up to the checked data lack issue by using the latest appraisal on false news as a use case, as moral conscience is plentiful but clarified models are scarce, to show that community oversight remains reasonable while standing up to the checked data lack issue.

CHAPTER-4

SYSTEM REQUIREMENTS SPECIFICATIONS

4.1) HARDWARE REQUIREMENTS

Processor : Pentium IV/III

Hard disk : minimum 80 GB

RAM : minimum 2 GB

4.2) Software Requirements:

Operating System : Windows

Tool : Anaconda with Jupyter Notebook

CHAPTER-5

SYSTEM DESIGN SPECIFICATION

5.1) SYSTEM ARCHITECTURE

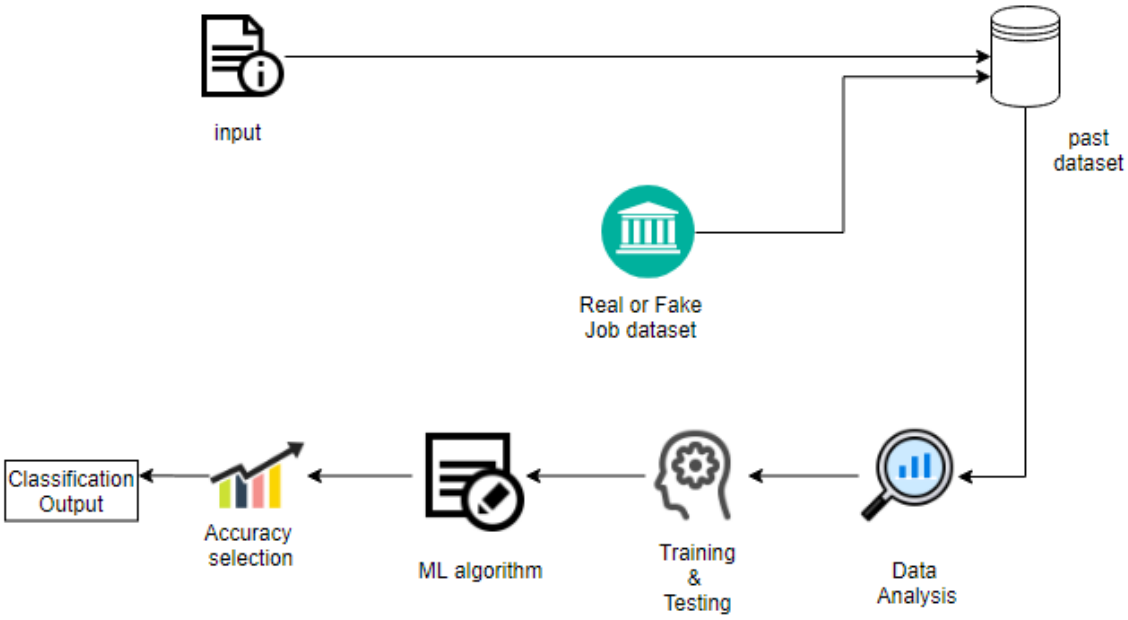

Fig 5.1.1) Architecture Diagram

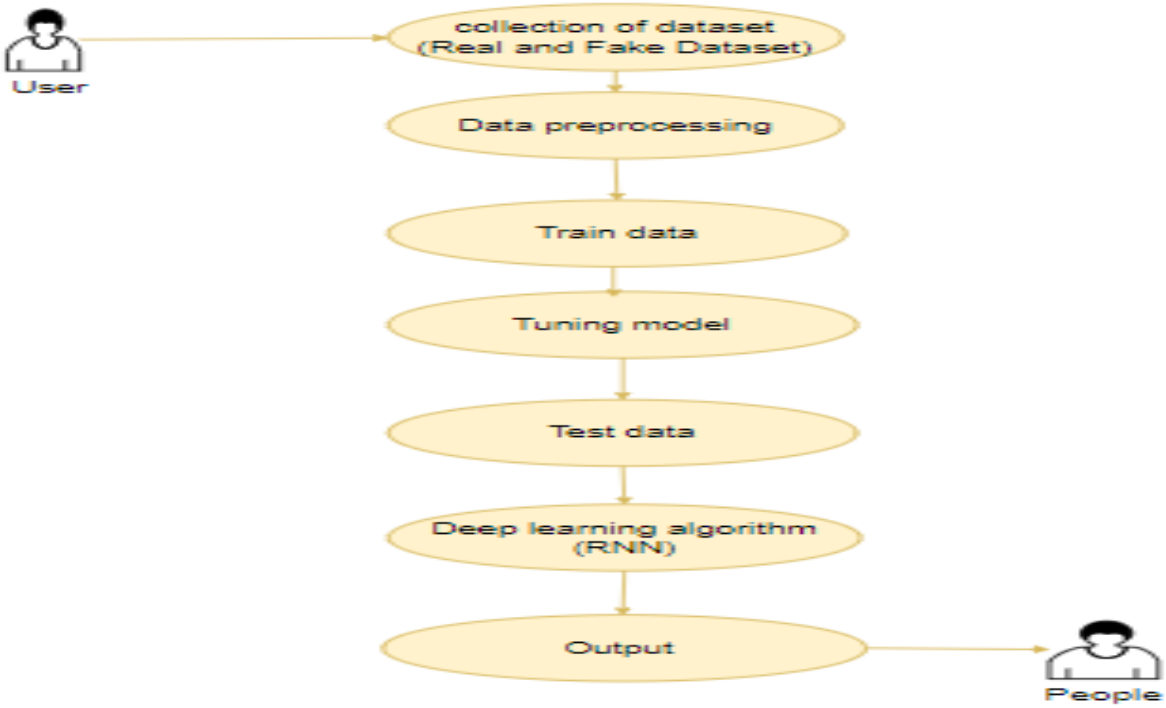

Fig-5.1.2) Use Case Diagram

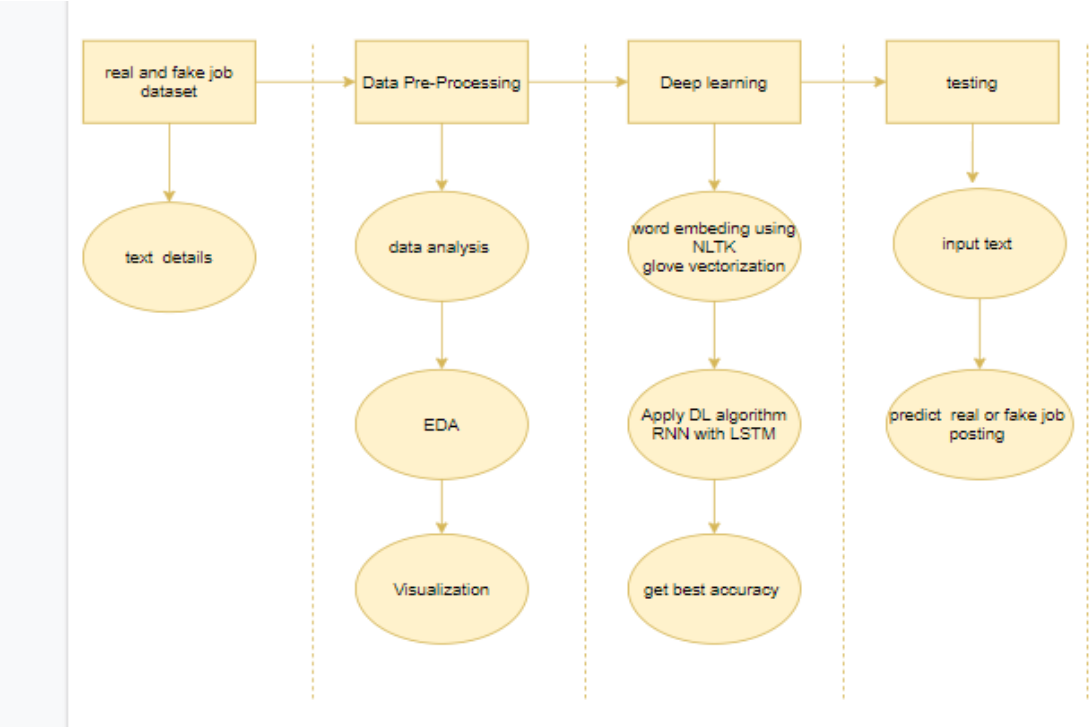

Fig-5.1.3) Activity Diagram

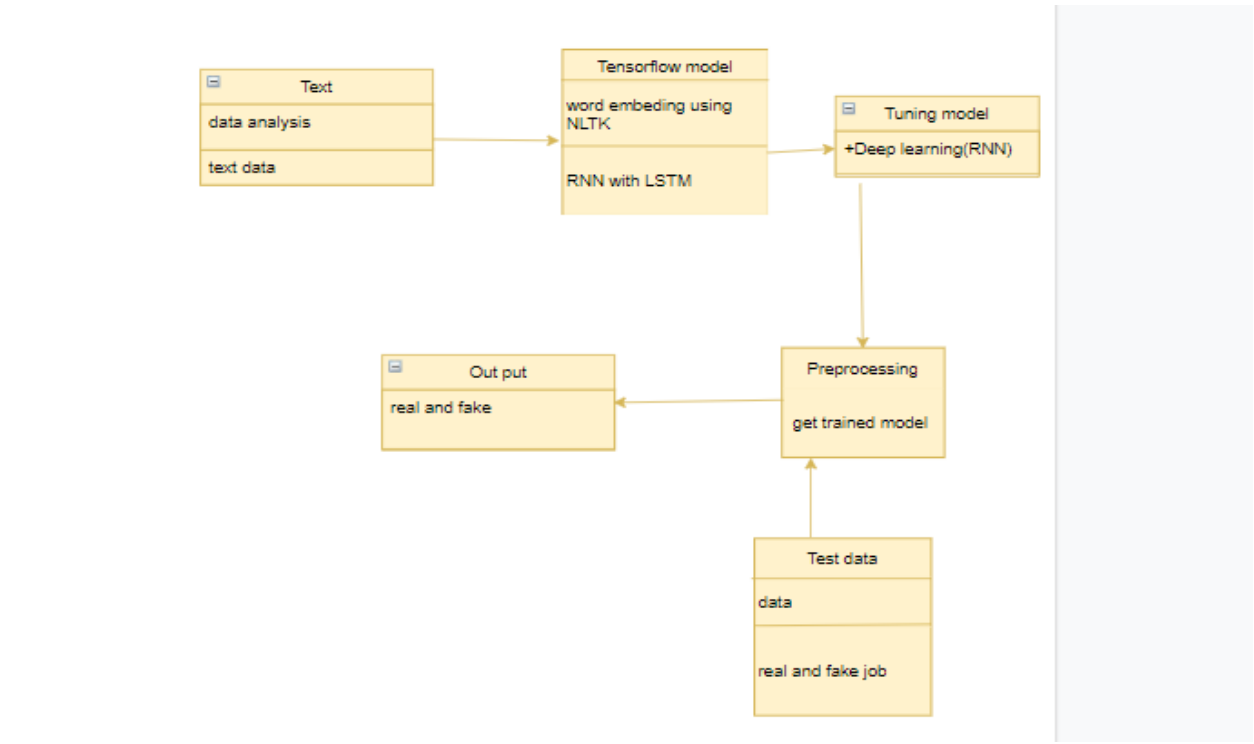

Fig-5.1.4) Class Diagram

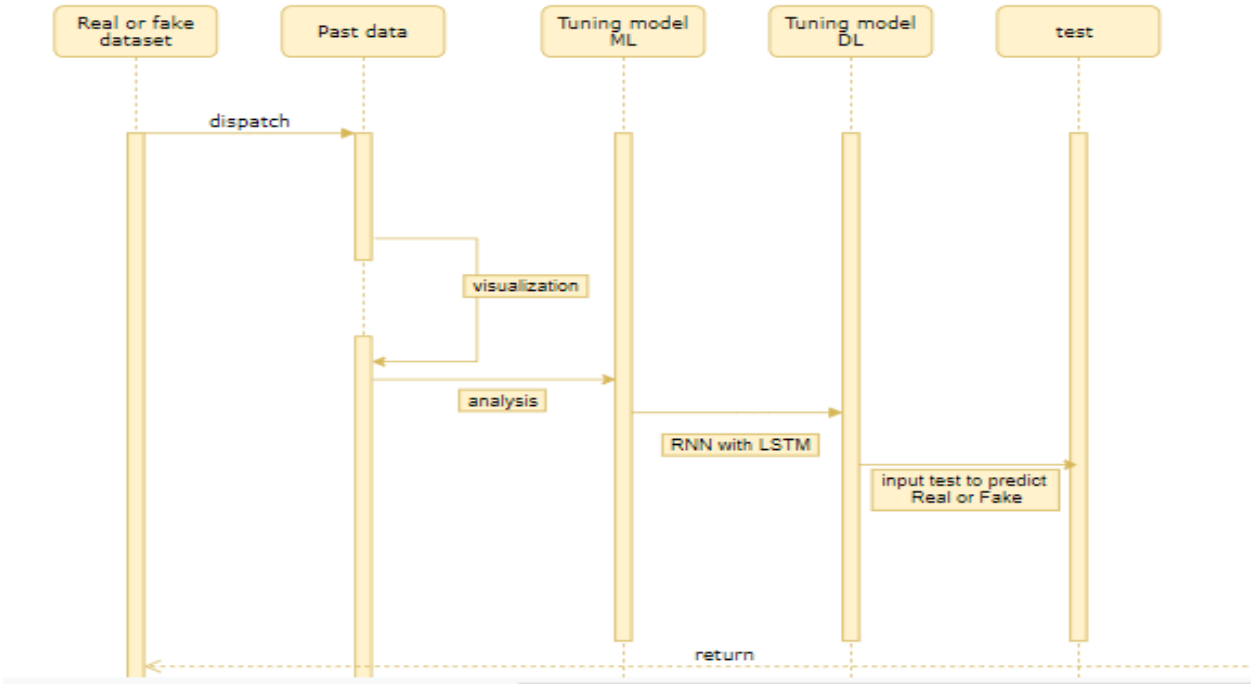

Fig-5.1.5) Sequence Diagram

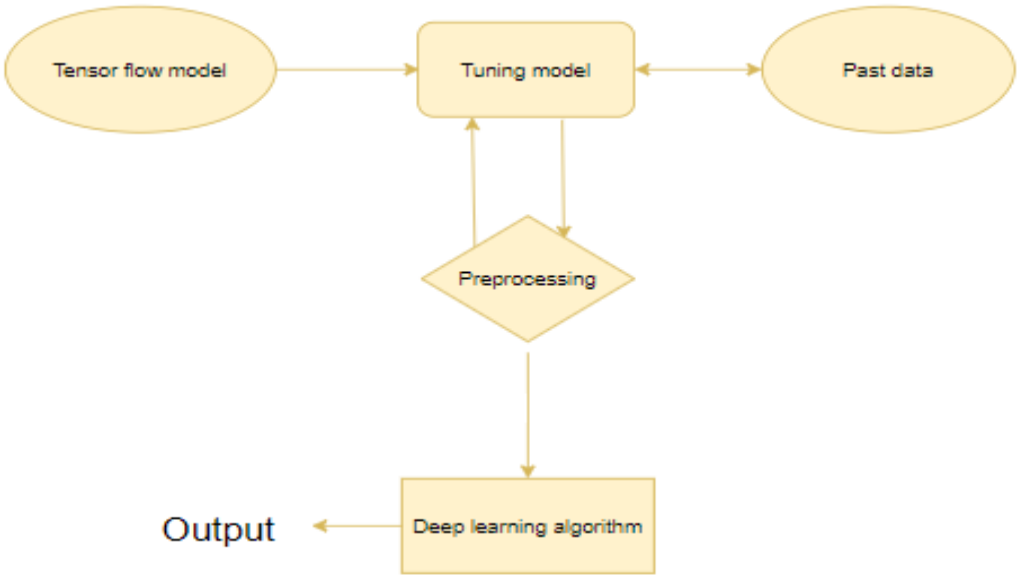

Fig-5.1.6) Entity-Relationship Diagram

5.2)MODULES

1. Data validation and pre-processing technique.
2. Data visualization and training a model by given attributes.
3. Performance measurements of ML algorithms.
4. Implementation of LSTM model for training and testing
5. Build a model for LSTM Neural Network.

5.3) MODULES DESCRIPTION

1) Data Validation and pre-processing technique:

Stacking the provided dataset to get in the library packages. Copy esteems are used to investigate the attribute ID by data shape, data type, and missing qualities. To break down the uni-variate, bi-variate, and multi-variate steps, change the given dataset and remove the section, among other things. The methods and techniques for cleaning data can differ in datsets. The main goal is to identify, eliminate blunders and get rid of bugs, irregularities in order to make information more useful in inquiry, dynamic.

2) Data visualization and training a model by given attributes:

In applied perspectives and AI, knowledge perception is a major skill. To be sure, insights focus on abstract representations and analyses of data. Data representation provides a substantial collection of tools for achieving subjective consensus. This can be useful for separating designs, degenerate details, exceptions, and much more when researching and becoming more familiar with a dataset. Knowledge expectations may be used to interact and display main associations in plots and outlines with some data

3) Performance measurements of ML algorithms:

It is a factual methodology for dissecting a measure of information in which at least one independent variable determines the outcome. A dichotomous variable is used to predict the outcome (where there are just two potential results). Calculated relapse's aim is to identify the suitable model for portraying a dichotomous pair's partnership attribute of interest (subordinate variable = reaction or outcome variable) and a set of autonomous (indicator or logical) variables. A Machine Learning arrangement equation called strategic relapse is used to estimate the probability of an absolute ward element.

4) **Implementation of LSTM model for training and testing**

Here we are building a LSTM network by using our word embedding technique. And we are going to split our data into training and testing purpose.

5) **Build a model for LSTM Neural Network.**

The built LSTM network is used to predict whether real or fake job based on the given data ie description.

CHAPTER-6

SYSTEM IMPLEMENTATION

6.1) VARIABLE IDENTIFICATION PROCESS

Variable identification is done with Uni-variate, Bi-variate and Multi-variate analysis:

- import libraries for access and functional purpose and read the given dataset
- General Properties of Analyzing the given dataset
- Display the given dataset in the form of data frame
- Checking for duplicate data

6.2) EXPLORATORY DATA ANALYSIS OF VISUALIZATION

In applied analytics and machine learning, data visualisation is a crucial capability. Statistics is concerned with objective data explanations and estimations. Data visualisation is a valuable collection of methods for obtaining a qualitative interpretation of data. This is useful for exploring and learning about a dataset, as well as finding trends, corrupt results, and outliers. Data visualisations should be used to express and explain core associations in graphs and maps that are more visceral and engaging for stakeholders than indicators of affiliation or meaning with a little domain information. Data visualisation and exploratory data processing are areas in and of themselves, and it would be recommended that you read any of the books mentioned at the end for more information.

6.3 CROSS VALIDATION

Cross-validation has three steps:

1. Set aside a portion of the sample data-set.
2. Train the model with the rest of the results.
3. Run the model on the data set's reserve data..

Advantages of train/test split:

1. K-fold cross-validation is K times better than Leave One Out cross-validation since the train/test break is repeated K times.
2. Examining the detailed findings of the assessment procedure is much easier.

Cross-validation has the following benefits:

1. More precise estimation of out-of-sample precision.

2. Data is used more "efficiently" so any observation is used for both preparation and research.

False Positives (FP): An individual who will pay anticipated as a defaulter. At the When the actual class is no and the expected class is yes. For example, if the genuine class indicates that this visitor did not endure, the expected class indicates that this visitor would endure.

False Negatives (FN): A person who defaults predicted as the payer. When the expected class is no but the real class is yes. For eg, if the traveler's real class value indicates that he or she survived but the expected class indicates that the traveller would die.

True Positives (TP): A person who will not pay predicted as a defaulter. These are the accurately expected positive values, indicating that the true class value is valid and the predicted class value is indeed valid. For example, if the real class value indicates that this commuter survived and the expected class also indicates that this commuter survived.

True Negatives (TN): A person who defaults predicted as the payer. These are the accurately estimated negative values, indicating that the true class value is no and the predicted class value is also no. For example, if the actual class reports that this passenger did not survive and the expected class reports the same.

In the example below 5 different algorithms are compared:

- Logistic Regression
- Random Forest
- K-Nearest Neighbors
- Decision tree
- Support Vector Machines

To test each algorithm, the K-fold cross validation protocol is used, which is programmed with the same random seed to ensure that the same splits to the training data are done and that each algorithm is tested in the same manner. Until matching algorithms, instal Scikit-Learn libraries and build a machine learning model.Preprocessing, linear model with logistic regression method, cross validation with KFold method, ensemble with random forest method, and tree with decision tree classifier are all included in this library package. Additionally, the train and test sets should be separated. By comparing precision, it is possible to forecast the outcome. .

6.4)PREDICTION RESULT BY ACCURACY

True Positive Rate(TPR) = TP / (TP + FN)

False Positive rate(FPR) = FP / (FP + TN)

Accuracy: The percentage of total predictions that are correct; otherwise, how much the model accurately forecasts defaulters and non-defaulters.

Accuracy calculation:

Accuracy = (TP + TN) / (TP + TN + FP + FN)

. It is essentially the number of correctly expected observations to all observations. One might believe that if our model is accurate, it is the best. Yes, consistency is an important parameter, also because you have datasets that are symmetric with almost identical for false positives and false negatives.

Precision: The percentage of optimistic forecasts that turn out to be right. (How much does the model get it right as it forecasts default?)

Precision = TP / (TP + FP)

Recall: The percentage of measured positive values that were accurately estimated. (The percentage of real defaulters predicted accurately by the model)

Recall = TP / (TP + FN)

Recall(Sensitivity) – Recall is defined as the proportion of correctly predicted positive observations to all observations in the actual class.

F1 Score : The weighted average of Precision and Recall is the F1 Score.. When false positives and false negatives have comparable costs, accuracy performs well.

General Formula:

F- Measure = 2TP / (2TP + FP + FN)

F1-Score Formula:

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

CHAPTER -7

RESULTS AND DISCUSSIONS

PERFORMANCE METRICS

7.1) Accuracy result of K-Nearest Neighbor is: 90.98712446351931

Classification report of K-Nearest Neighbor Results:

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       211
           1       0.56      0.23      0.32        22

    accuracy                           0.91       233
   macro avg       0.74      0.60      0.64       233
weighted avg       0.89      0.91      0.89       233

Confusion Matrix result of K-Nearest Neighbor is:
 [[207   4]
 [ 17   5]]

Sensitivity :  0.981042654028436

Specificity :  0.22727272727272727
```

7.2) Accuracy result of kmeans is: 45.06437768240343

Classification report of kmeans Results:

```
              precision    recall  f1-score   support

           0       0.85      0.48      0.61       211
           1       0.04      0.18      0.06        22

    accuracy                           0.45       233
   macro avg       0.44      0.33      0.34       233
weighted avg       0.77      0.45      0.56       233

Confusion Matrix result of kmeans is:
 [[101 110]
 [ 18   4]]

Sensitivity :  0.4786729857819905

Specificity :  0.18181818181818182
```

7.3) Accuracy result of Random Forest: is: 96.99570815450643

Classification report of Random Forest: Results:

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.98       211
           1       0.94      0.73      0.82        22

    accuracy                           0.97       233
   macro avg       0.96      0.86      0.90       233
weighted avg       0.97      0.97      0.97       233

Confusion Matrix result of Random Forest: is:
 [[210   1]
 [  6  16]]

Sensitivity :  0.995260663507109

Specificity :  0.7272727272727273
```

7.4) Accuracy result of Support Vector Machines is: 90.55793991416309

Classification report of Support Vector Machines: Results:

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.95       211
           1       0.00      0.00      0.00        22

    accuracy                           0.91       233
   macro avg       0.45      0.50      0.48       233
weighted avg       0.82      0.91      0.86       233

Confusion Matrix result of Support Vector Machines: is:
 [[211   0]
 [ 22   0]]

Sensitivity :  1.0

Specificity :  0.0
```

7.5) Accuracy result of Logistic Regression is: 90.55793991416309

Classification report of Logistic Regression : Results:

```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95       211
           1       0.50      0.18      0.27        22

    accuracy                           0.91       233
   macro avg       0.71      0.58      0.61       233
weighted avg       0.88      0.91      0.89       233

Confusion Matrix result of Logistic Regression : is:
```

```
 [[207   4]
 [ 18   4]]

Sensitivity :  0.981042654028436

Specificity :  0.18181818181818182
```

7.6) Accuracy result of Decision Tree Classifier is: 94.84978540772532

Classification report of Decision Tree Classifier : Results:

```
              precision    recall  f1-score   support

           0       0.99      0.95      0.97       211
           1       0.67      0.91      0.77        22

    accuracy                           0.95       233
   macro avg       0.83      0.93      0.87       233
weighted avg       0.96      0.95      0.95       233

Confusion Matrix result of Decision Tree Classifier : is:
 [[201  10]
 [  2  20]]

Sensitivity :  0.95260663507109

Specificity :  0.9090909090909091
```

The above results shows that Random forest algorithm has the highest accuracy rate of 96.5% and K means has the lowest accuracy rate of 45.06% to predict real and fake job.

7.7) Classification report of RNN-LSTM MODEL:Results:

1) Accuracy: 93.55%
2) Precision: 93.55%
3) Recall: 93.55%

Using RNN-LSTM algorithm, we can predict a job is real or fake based on its description with accuracy rate of 93.55%

CHAPTER-8
CONCLUSION AND FUTURE SCOPE

We executed the Machine Learning method with Python to distinguish phony and genuine occupation notices. Checked subtleties are pictured dependent on negative or positive qualities from the given arrangement of information. Information is anticipated with an AI calculation with applicable factual strategies to demonstrate the information accordingly

Future Work

- To simplify this step by displaying the forecast outcome in an internet or desktop software.
- To make the job easier and to apply in an Artificial Intelligence environment.

CHAPTER-9

SCREENSHOTS

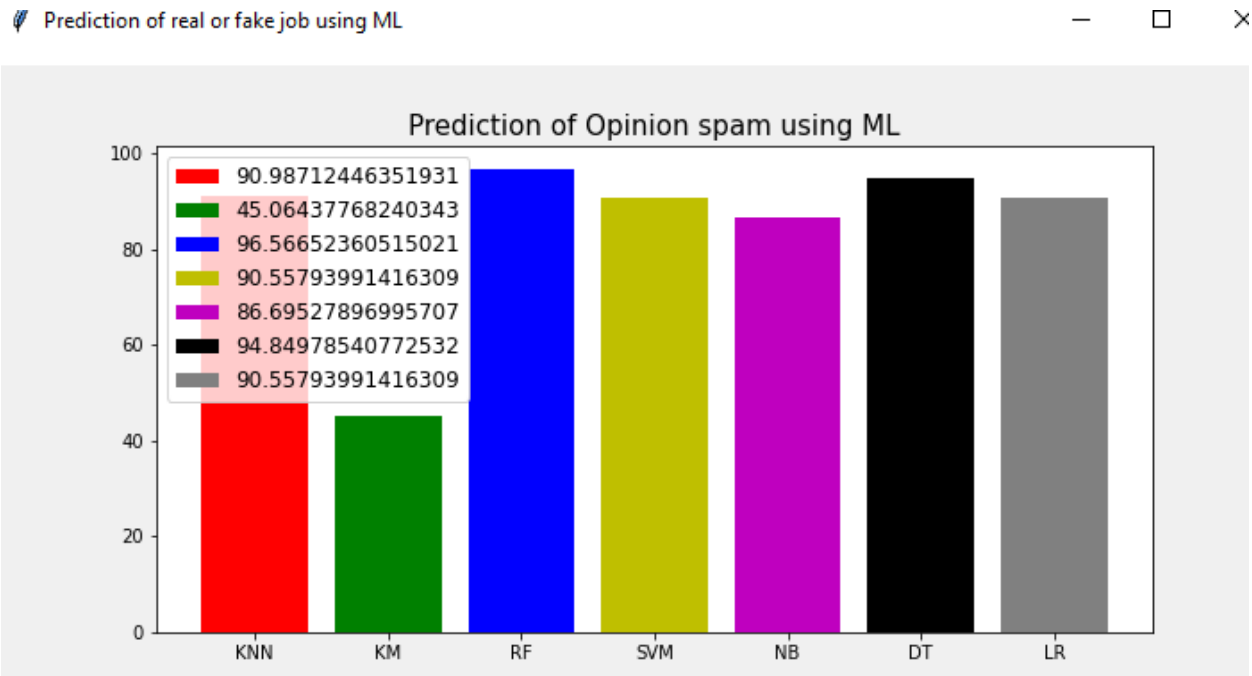

Fig-9.1) Comparing Accuracy of Machine learning algorithm

```
In [39]: print(f"[+] Accuracy: {accuracy*100:.2f}%")
         print(f"[+] Precision:   {precision*100:.2f}%")
         print(f"[+] Recall:   {recall*100:.2f}%")

         [+] Accuracy: 93.55%
         [+] Precision:   93.55%
         [+] Recall:   93.55%
```

Fig-9.2) Classification report of RNN-LSTM Algorithm

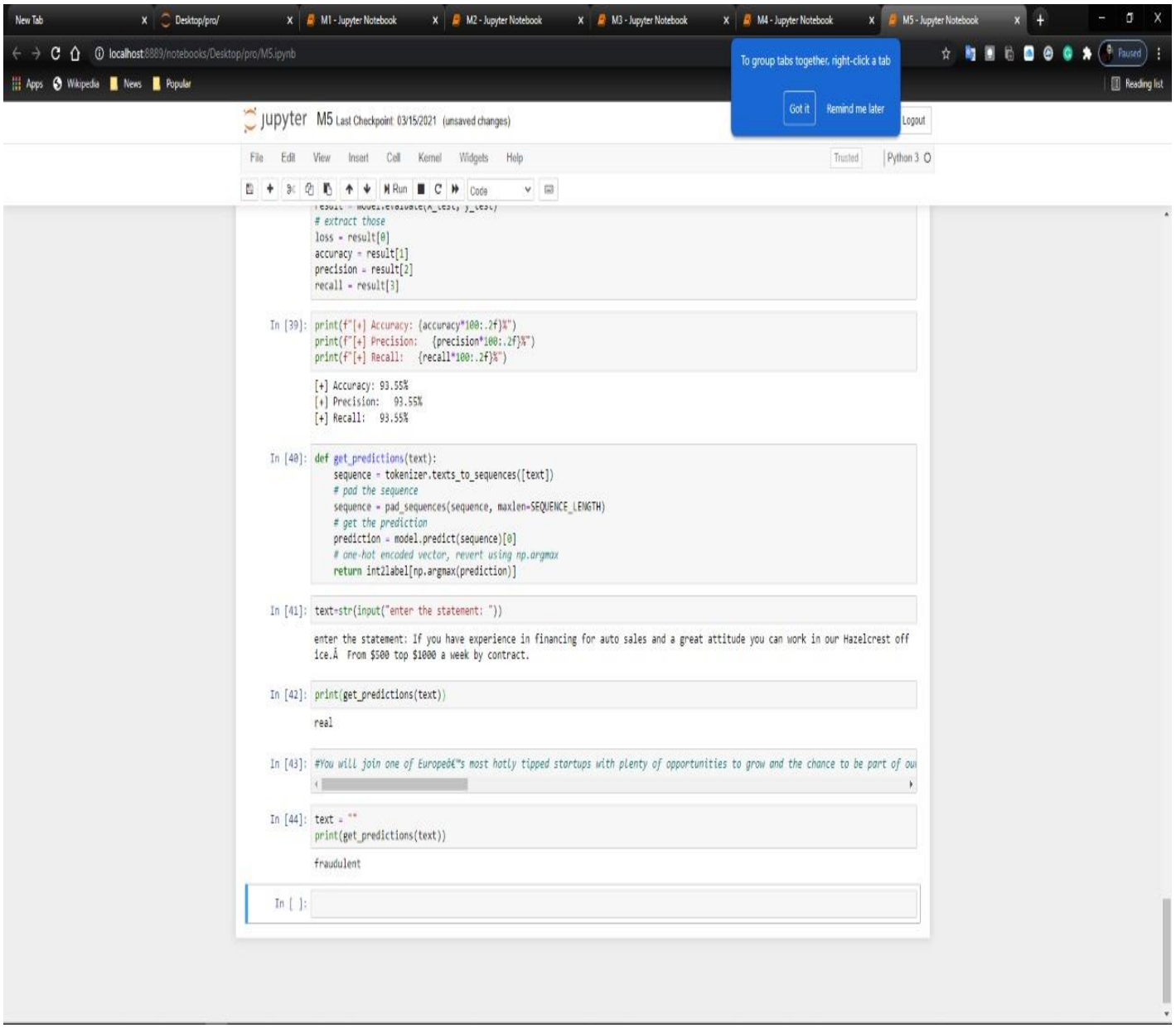

Fig-9.3) Prediction Output-Real

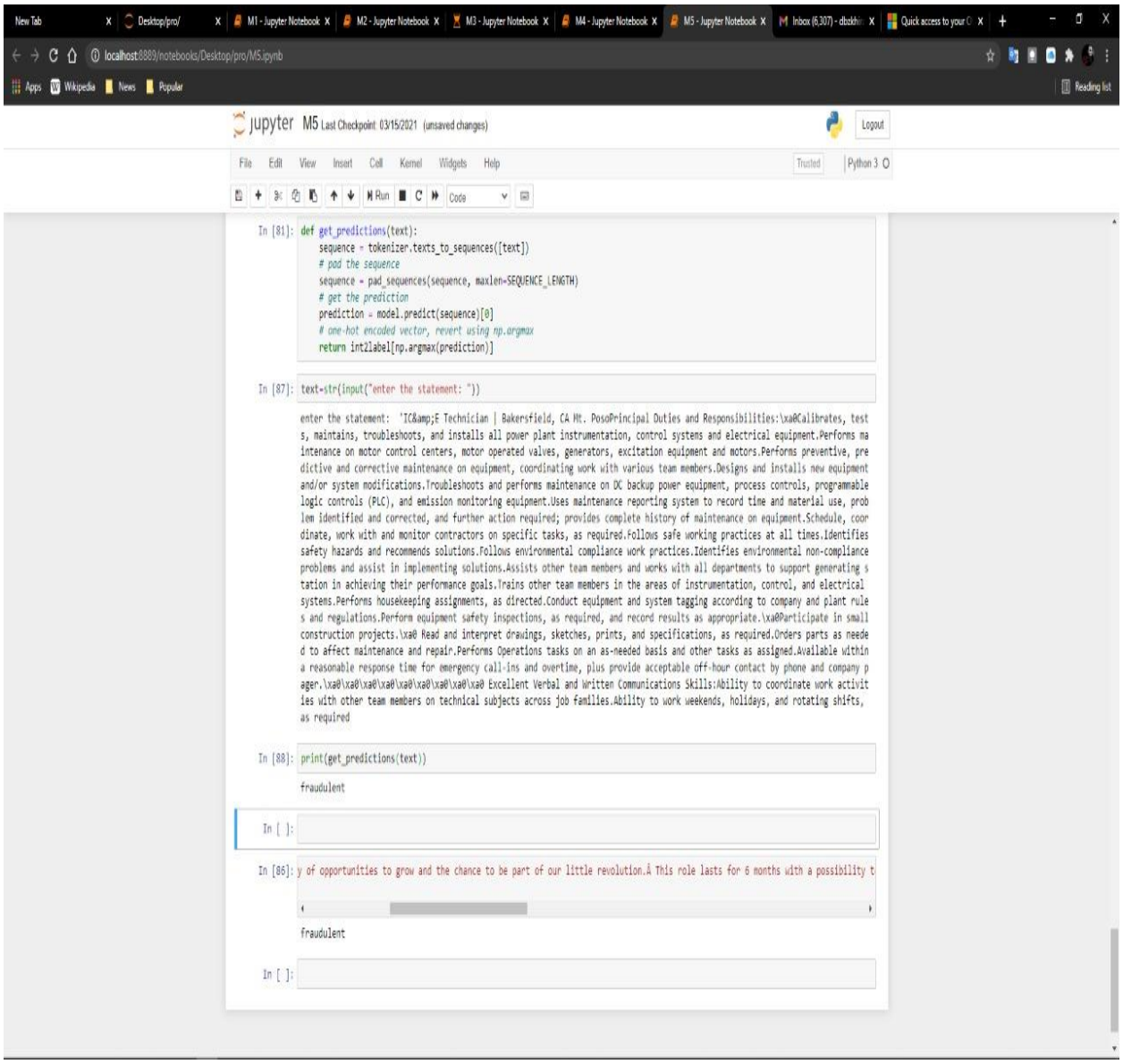

Fig-9.4) Prediction Output-Fraudulent

CHAPTER-10

SAMPLE SOURCE CODE

10.1) Data Validation and pre-processing technique:

```
#import library packages
import pandas as p
import numpy as n
import warnings
warnings.filterwarnings("ignore")
#Load given dataset
data = p.read_csv('fake_job_postings.csv')
data.head()
#shape
data.shape
df=data.dropna()
df.head()
df.shape
#show columns
df.columns
#To describe the dataframe
df.describe()
df.info()
df.duplicated()
#find sum of duplicate data
sum(df.duplicated())
#Checking sum of missing values
df.isnull().sum()
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='fraudulent', axis=1)
#Response variable
y = df.loc[:,'fraudulent']
```

```
#We'll use a test size of 30%. We also stratify the split on the response variable, which is very important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
print("Number of training dataset: ", len(X_train))
print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))
df.groupby('fraudulent').describe()
#plotting graph for distribution
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x = "fraudulent", data = df)
df.loc[:, 'fraudulent'].value_counts()
plt.title('Distribution of  fraudulent or NOT')
from collections import Counter
count1 = Counter(" ".join(df[df['fraudulent']==0]["description"]).split()).most_common(30)
data1 = p.DataFrame.from_dict(count1)
data1 = data1.rename(columns={0: "real", 1 : "count"})
count2 = Counter(" ".join(df[df['fraudulent']==1]["description"]).split()).most_common(30)
data2 = p.DataFrame.from_dict(count2)
data2 = data2.rename(columns={0: "fraud", 1 : "count_"})
from collections import Counter
count1 = Counter(" ".join(df[df['fraudulent']==0]["description"]).split()).most_common(30)
data1 = p.DataFrame.from_dict(count1)
data1 = data1.rename(columns={0: "real", 1 : "count"})
count2 = Counter(" ".join(df[df['fraudulent']==1]["description"]).split()).most_common(30)
data2 = p.DataFrame.from_dict(count2)
data2 = data2.rename(columns={0: "fraud", 1 : "count_"})
data1.plot.bar(legend = False, color = 'red',figsize = (20,15))
y_pos = n.arange(len(data1["real"]))
plt.xticks(y_pos, data1["real"])
```

```
plt.title('Top 30 words of Truthful')
plt.xlabel('words')
plt.ylabel('number')
plt.show()
#Graph for top 30 words of Opinion spam
data2.plot.bar(legend = False, color = 'green', figsize = (20,17))
y_pos = n.arange(len(data2["fraud"]))
plt.xticks(y_pos, data2["fraud"])
plt.title('Top 30 words of Opinion spam')
plt.xlabel('words')
plt.ylabel('number')
plt.show()
df.columns
from sklearn.preprocessing import LabelEncoder
var_mod = ['title', 'location', 'department', 'salary_range',
      'company_profile', 'requirements', 'benefits',
      'telecommuting', 'has_company_logo', 'has_questions', 'employment_type',
      'required_experience', 'required_education', 'industry', 'function']
le = LabelEncoder()
for i in var_mod:
   df[i] = le.fit_transform(df[i]).astype(str)
```

10.2) Data visualization and training a model by given attributes:

```
#import library packages
import pandas as p
import matplotlib.pyplot as plt
import seaborn as s
import numpy as n
import warnings
warnings.filterwarnings("ignore")
#Load given dataset
```

```
data = p.read_csv('fake_job_postings.csv')
df=data.dropna()
DF
#preprocessing, split test and dataset, split response variable
X = df.drop(labels='fraudulent', axis=1)
#Response variable
y = df.loc[:,'fraudulent']
#We'll use a test size of 30%. We also stratify the split on the response variable, which is very
important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
print("Number of training dataset: ", len(X_train))
print("Number of test dataset: ", len(X_test))
print("Total number of dataset: ", len(X_train)+len(X_test))
df.groupby('fraudulent').describe()
#plotting graph for distribution
import matplotlib.pyplot as plt
import seaborn as sns
sns.countplot(x = "fraudulent", data = df)
df.loc[:, 'fraudulent'].value_counts()
plt.title('Distribution of Opinion spam')
df['fraudulent'].unique()
# plotting graph by length.
truthful =df[df['fraudulent'] == 0]['description'].str.len()
sns.distplot(truthful, label='Real')
deceptive = df[df['fraudulent'] == 1]['description'].str.len()
sns.distplot(deceptive, label='Fake')
plt.title('Distribution by Length')
plt.legend()
#plotting graph by digits.
truthful1 = df[df['fraudulent'] == 0]['description'].str.replace(r'\D+', '').str.len()
```

```
sns.distplot(truthful1, label='Real')
deceptive1 = df[df['fraudulent'] == 1]['description'].str.replace(r'\D+', '').str.len()
sns.distplot(deceptive1, label='Fake')
plt.title('Distribution by Digits')
plt.legend()
#plotting graph for non-digits.
truthful2 = df[df['fraudulent'] == 0]['description'].str.replace(r'\w+', '').str.len()
sns.distplot(truthful2, label='Real')
deceptive2 = df[df['fraudulent'] == 1]['description'].str.replace(r'\w+', '').str.len()
sns.distplot(deceptive2, label='Fake')
plt.title('Distribution of Non-Digits')
plt.legend()
!pip install nltk
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re
import string
# remove whitespaces
df['description']=df['description'].str.strip()
# lowercase the text
df['description'] = df['description'].str.lower()
#remove punctuation
punc = string.punctuation
table = str.maketrans("",punc)
df['description']=df['description'].apply(lambda x: x.translate(table))
# tokenizing each message
df['word_tokens']=df.apply(lambda x: x['description'].split(' '),axis=1)
# removing stopwords
df['cleaned_text'] = df.apply(lambda x: [word for word in x['word_tokens'] if word not in
stopwords.words('english')],axis=1)
```

```
# stemming
ps = PorterStemmer()
df['stemmed']= df.apply(lambda x: [ps.stem(word) for word in x['cleaned_text']],axis=1)
# remove single letter words
df['final_text'] = df.apply(lambda x: ' '.join([word for word in x['stemmed'] if
len(word)>1]),axis=1)
# divide the set in training and test
from sklearn.model_selection import train_test_split
X,X_test,y,y_test = train_test_split(df.loc[:,'description':],df['fraudulent'],test_size=0.2)
# Now we'll create a vocabulary for the training set with word count
from collections import defaultdict
vocab=defaultdict(int)
for text in X['final_text'].values:
    for elem in text.split(' '):
        vocab[elem]+=1
from wordcloud import WordCloud
# Now we look at the types of words in ham and spam. We plot wordclouds for both
ham_text=' '.join(X.loc[y==0,'final_text'].values)
ham_wordcloud = WordCloud(background_color='white',max_words=2000).generate(ham_text)
spam_text=' '.join(X.loc[y==1,'final_text'].values)
spam_wordcloud =
WordCloud(background_color='white',max_words=2000).generate(spam_text)
plt.figure(figsize=[20,30])
plt.subplot(1,2,1)
plt.imshow(spam_wordcloud,interpolation='bilinear')
plt.title('Jobposting:fake')
plt.axis('off')
plt.subplot(1,2,2)
plt.imshow(ham_wordcloud, interpolation='bilinear')
plt.axis('off')
plt.title('Jobposting:real')
```

10.3) Performance measurements of ML algorithms:

```
#import library packages
import pandas as p
import numpy as n
import warnings
warnings.filterwarnings("ignore")
#Load given dataset
data = p.read_csv('deceptive-opinion.csv')
df=data.dropna()
df.columns
#Load given dataset
data = p.read_csv('fake_job_postings.csv')
df=data.dropna()
df.columns
from sklearn.preprocessing import LabelEncoder
var_mod = ['job_id', 'title', 'location', 'department', 'salary_range',
      'company_profile', 'description', 'requirements', 'benefits',
      'telecommuting', 'has_company_logo', 'has_questions', 'employment_type',
      'required_experience', 'required_education', 'industry', 'function',
      'fraudulent']
le = LabelEncoder()
for i in var_mod:
   df[i] = le.fit_transform(df[i]).astype(int)
#According to the cross-validated MCC scores, the random forest is the best-performing model,
so now let's evaluate its performance on the test set.
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
X = df.drop(labels='fraudulent', axis=1)
#Response variable
y = df.loc[:,'fraudulent']
#We'll use a test size of 30%. We also stratify the split on the response variable, which is very
```

```
important to do because there are so few fraudulent transactions.
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1, stratify=y)
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2, random_state=0)
kmeans.fit(X_train,y_train)
predictR = kmeans.predict(X_test)
print("")
x = (accuracy_score(y_test,predictR)*100)
print('Accuracy result of kmeans is:', x)
print("")
print("")
print('Classification report of kmeans Results:')
print("")
print(classification_report(y_test,predictR))
xkm = (accuracy_score(y_test,predictR)*100)
cm2=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of kmeans is:\n', confusion_matrix(y_test,predictR))
print("")
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(X_train,y_train)
predictR = rfc.predict(X_test)
print("")
x = (accuracy_score(y_test,predictR)*100)
print('Accuracy result of Random Forest: is:', x)
```

```
print("")
print("")
print('Classification report of Random Forest: Results:')
print("")
print(classification_report(y_test,predictR))
xrf = (accuracy_score(y_test,predictR)*100)
cm2=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Random Forest: is:\n', confusion_matrix(y_test,predictR))
print("")
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
from sklearn.svm import SVC
s = SVC()
s.fit(X_train,y_train)
predictR = s.predict(X_test)
print("")
x = (accuracy_score(y_test,predictR)*100)
print('Accuracy result of Support Vector Machines is:', x)
print("")
print("")
print('Classification report of Support Vector Machines: Results:')
print("")
print(classification_report(y_test,predictR))
xs = (accuracy_score(y_test,predictR)*100)
cm2=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Support Vector Machines: is:\n',
confusion_matrix(y_test,predictR))
print("")
```

```
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_train,y_train)
predictR = gnb.predict(X_test)
print("")
x = (accuracy_score(y_test,predictR)*100)
print('Accuracy result of Naive bayes is:', x)
print("")
print("")
print('Classification report of Naive bayes: Results:')
print("")
print(classification_report(y_test,predictR))
xnb = (accuracy_score(y_test,predictR)*100)
cm2=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Naive bayes: is:\n', confusion_matrix(y_test,predictR))
print("")
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
from sklearn.tree import DecisionTreeClassifier
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
predictR = dtree.predict(X_test)
print("")
```

```
x = (accuracy_score(y_test,predictR)*100)
print('Accuracy result ofDecision Tree Classifier  is:', x)
print("")
print("")
print('Classification report of Decision Tree Classifier : Results:')
print("")
print(classification_report(y_test,predictR))
xd = (accuracy_score(y_test,predictR)*100)
cm2=confusion_matrix(y_test,predictR)
print('Confusion Matrix result of Decision Tree Classifier : is:\n',
confusion_matrix(y_test,predictR))
print("")
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
from sklearn.linear_model import LogisticRegression
logR= LogisticRegression()
logR.fit(X_train,y_train)
predictR = logR.predict(X_test)
print("")
x = (accuracy_score(y_test,predictR)*100)
print('Accuracy result of Logistic Regression is:', x)
print("")
print("")
print('Classification report of Logistic Regression : Results:')
print("")
print(classification_report(y_test,predictR))
xl = (accuracy_score(y_test,predictR)*100)
cm2=confusion_matrix(y_test,predictR)
```

```
print('Confusion Matrix result of Logistic Regression : is:\n', confusion_matrix(y_test,predictR))
print("")
sensitivity1 = cm2[0,0]/(cm2[0,0]+cm2[0,1])
print('Sensitivity : ', sensitivity1 )
print("")
specificity1 = cm2[1,1]/(cm2[1,0]+cm2[1,1])
print('Specificity : ', specificity1)
def graph():
    import matplotlib.pyplot as plt
    data=[xkn,xkm,xrf,xs,xnb,xd,xl]
    alg="KNN","KM","RF","SVM","NB","DT","LR"
    plt.figure(figsize=(10,5))
    b=plt.bar(alg,data,color=("r","g","b","y","m","black","gray"))
    plt.title("Prediction of Opinion spam using ML",fontsize=15)
    plt.legend(b,data,fontsize=12)
    plt.savefig('comp.png')
import tkinter
from matplotlib.backends.backend_tkagg import (FigureCanvasTkAgg, NavigationToolbar2Tk)
from matplotlib.backend_bases import key_press_handler
from matplotlib.figure import Figure
import numpy as np
root = tkinter.Tk()
root.wm_title("Prediction of real or fake job using ML")
fig = Figure(figsize=(10,10),dpi=1)
canvas = FigureCanvasTkAgg(fig, master=root)
canvas.draw()
canvas.get_tk_widget().pack(side=tkinter.TOP, fill=tkinter.BOTH, expand=1)
icon=tkinter.PhotoImage(file='comp.png')
label=tkinter.Label(root,image=icon)
label.pack()
root.mainloop()
```

10.4) Implementation of LSTM model for training and testing

```
from tensorflow.keras.models import Sequential,load_model
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.layers import Dropout, Activation, Flatten
import warnings
warnings.filterwarnings("ignore")
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
data = pd.read_csv('fake_job_postings.csv', date_parser = True)
data=data.dropna()
data.columns
from sklearn.preprocessing import LabelEncoder
var_mod = ['job_id', 'title', 'location', 'department', 'salary_range',
      'company_profile', 'description', 'requirements', 'benefits',
      'telecommuting', 'has_company_logo', 'has_questions', 'employment_type',
      'required_experience', 'required_education', 'industry', 'function',
      'fraudulent']
le = LabelEncoder()
for i in var_mod:
   data[i] = le.fit_transform(data[i]).astype(int)
from sklearn.metrics import confusion_matrix, classification_report, matthews_corrcoef,
cohen_kappa_score, accuracy_score, average_precision_score, roc_auc_score
X = data.drop(labels='fraudulent', axis=1)
#Response variable
y = data.loc[:,'fraudulent']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=1, stratify=y)
scaler = MinMaxScaler()
```

```
data_training = scaler.fit_transform(X_train)
data_training
scaler = MinMaxScaler()
data_testing = scaler.fit_transform(X_test)
data_testing
data_training[0:10]
X_train1 = []
y_train1 = []
for i in range(60, data_training.shape[0]):
    X_train1.append(data_training[i-60:i])
    y_train1.append(data_training[i, 0])
X_train, y_train = np.array(X_train1), np.array(y_train1)
X_train.shape
X_test1 = []
y_test1 = []
for i in range(60, data_training.shape[0]):
    X_train1.append(data_training[i-60:i])
    y_train1.append(data_training[i, 0])
X_train, y_train = np.array(X_train1), np.array(y_train1)
X_train.shape
X_test1 = []
y_test1 = []
for i in range(60, data_testing.shape[0]):
    X_test1.append(data_testing[i-60:i])
    y_test1.append(data_testing[i, 0])
X_test, y_test = np.array(X_test1), np.array(y_test1)
X_test.shape
# Initialising the RNN
model = Sequential()
# Adding the first LSTM layer and some Dropout regularisation
model.add(LSTM(24, return_sequences=True,input_shape=(X_train.shape[1],
```

```
X_train.shape[2])))  # returns a sequence of vectors of dimension 64
model.add(Dropout(0.2))
# Adding a second LSTM layer and some Dropout regularisation
model.add(LSTM(units = 50))
#model.add(Dropout(0.2))
model.add(Dense(10,activation='relu'))
# Adding the output layer
#model.add(Dense(1, activation="linear"))
model.add(Dense(1))
#model.add(Dense(units = 1))
# Compiling the RNN
model.compile(optimizer = 'adam', loss = 'mean_squared_error',metrics=['accuracy'])
history=model.fit(X_train, y_train,batch_size=2, epochs=40)
# Model summary for number of parameters use in the algorithm
model.summary()
def graph():
    #Plot training & validation accuracy values
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['loss'])
    plt.title('Model accuracy')
    plt.ylabel('Accuracy')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
    plt.show()
    # Plot training & validation loss values
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['loss'])
    plt.title('Model loss')
    plt.ylabel('Loss')
    plt.xlabel('Epoch')
    plt.legend(['Train', 'Test'], loc='upper left')
```

```
    plt.show()
graph()
scores = model.predict(X_test)
import math, time
print("")
trainScore = model.evaluate(X_train, y_train, verbose=0)
print('Train Score: %.2f MSE (%.2f RMSE)' % (trainScore[0], math.sqrt(trainScore[0])))
print("")
testScore = model.evaluate(X_test, y_test, verbose=0)
print('Test Score: %.2f MSE (%.2f RMSE)' % (testScore[0], math.sqrt(testScore[0])))
```

10.5) Build a model for LSTM Neural Network.

```
import warnings
warnings.filterwarnings("ignore")
#!pip install keras-metrics
import tqdm
import numpy as np
import keras_metrics # for recall and precision metrics
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
from keras.layers import Embedding, LSTM, Dropout, Dense
from keras.models import Sequential
from keras.utils import to_categorical
from keras.callbacks import ModelCheckpoint, TensorBoard
from sklearn.model_selection import train_test_split
import time
import numpy as np
import pickle
SEQUENCE_LENGTH = 100 # the length of all sequences (number of words per sample)
EMBEDDING_SIZE = 100  # Using 100-Dimensional GloVe embedding vectors
```

```
TEST_SIZE = 0.20 # ratio of testing set
BATCH_SIZE = 64
EPOCHS =30 # number of epochs
label2int = {"real": 0, "fraudulent": 1} #added
int2label = {0: "real", 1: "fraudulent"} #added
import pandas as pd
data=pd.read_csv("fake_job_postings.csv")
data.isnull().sum()
df=data.dropna()
df.isnull().sum()
x=df['description']
X
x=list(x)
X
df['fraudulent'] = df['fraudulent'].replace({0: 'real',1:'fraudulent'})
y=df['fraudulent']
y. unique()
y=list(y)
Y
type(y)
# Text tokenization
# vectorizing text, turning each text into sequence of integers
tokenizer = Tokenizer(num_words=None,char_level=True,oov_token='UNK')
tokenizer.fit_on_texts(x)
# convert to sequence of integers
x = tokenizer.texts_to_sequences(x)
print(x[0])
# convert to numpy arrays
x = np.array(x)
y = np.array(y)
# pad sequences at the beginning of each sequence with 0's
```

```
# for example if SEQUENCE_LENGTH=4:
# [[5, 3, 2], [5, 1, 2, 3], [3, 4]]
# will be transformed to:
# [[0, 5, 3, 2], [5, 1, 2, 3], [0, 0, 3, 4]]
x = pad_sequences(x, maxlen=SEQUENCE_LENGTH)
# One Hot encoding labels
# [spam, ham, spam, ham, ham] will be converted to:
# [1, 0, 1, 0, 1] and then to:
# [[0, 1], [1, 0], [0, 1], [1, 0], [0, 1]]
y = [ label2int[label] for label in y ]
y = to_categorical(y)
print(y[0])
# split and shuffle
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=TEST_SIZE, random_state=7)
def get_embedding_vectors(tokenizer, dim=100):
    embedding_index = {}
    with open(f"glove.6B.{dim}d.txt", encoding='utf8') as f:
        for line in tqdm.tqdm(f, "Reading GloVe"):
            values = line.split()
            word = values[0]
            vectors = np.asarray(values[1:], dtype='float32')
            embedding_index[word] = vectors
    word_index = tokenizer.word_index
    embedding_matrix = np.zeros((len(word_index)+1, dim))
    for word, i in word_index.items():
        embedding_vector = embedding_index.get(word)
        if embedding_vector is not None:
            # words not found will be 0s
            embedding_matrix[i] = embedding_vector
    return embedding_matrix
def get_model(tokenizer, lstm_units):
```

```
    """
    Constructs the model,
    Embedding vectors => LSTM => 2 output Fully-Connected neurons with softmax activation
    """
    # get the GloVe embedding vectors
    embedding_matrix = get_embedding_vectors(tokenizer)
    model = Sequential()
    model.add(Embedding(len(tokenizer.word_index)+1,
              EMBEDDING_SIZE,
              weights=[embedding_matrix],
              trainable=False,
              input_length=SEQUENCE_LENGTH))
    model.add(LSTM(lstm_units, recurrent_dropout=0.2))
    model.add(Dropout(0.3))
    model.add(Dense(2, activation="softmax"))
    # compile as rmsprop optimizer
    # aswell as with recall metric
    model.compile(optimizer="rmsprop", loss="categorical_crossentropy",
                  metrics=["accuracy", 'Precision', 'Recall'])
    model.summary()
    return model
# constructs the model with 128 LSTM units
model = get_model(tokenizer=tokenizer, lstm_units=128)
# initialize our ModelCheckpoint and TensorBoard callbacks
# model checkpoint for saving best weights
model_checkpoint = ModelCheckpoint("real_and_fake_job_classifier_{val_loss:.2f}",
save_best_only=True,
                                   verbose=1)
# for better visualization
tensorboard = TensorBoard(f"real_and_fake_job_classifier_{time.time()}")
# print our data shapes
```

```
print("X_train.shape:", X_train.shape)
print("X_test.shape:", X_test.shape)
print("y_train.shape:", y_train.shape)
print("y_test.shape:", y_test.shape)
# train the model
model.fit(X_train, y_train, validation_data=(X_test, y_test),batch_size=BATCH_SIZE,
epochs=EPOCHS,callbacks=[tensorboard, model_checkpoint],verbose=1)
# get the loss and metrics
result = model.evaluate(X_test, y_test)
# extract those
loss = result[0]
accuracy = result[1]
precision = result[2]
recall = result[3]
print(f"[+] Accuracy: {accuracy*100:.2f}%")
print(f"[+] Precision:   {precision*100:.2f}%")
print(f"[+] Recall:   {recall*100:.2f}%")
def get_predictions(text):
    sequence = tokenizer.texts_to_sequences([text])
    # pad the sequence
    sequence = pad_sequences(sequence, maxlen=SEQUENCE_LENGTH)
    # get the prediction
    prediction = model.predict(sequence)[0]
    # one-hot encoded vector, revert using np.argmax
    return int2label[np.argmax(prediction)]
text=str(input("enter the statement: "))
print(get_predictions(text))
```

#You will join one of Europeâ€™s most hotly tipped startups with plenty of opportunities to grow and the chance to be part of our little revolution.Â This role lasts for 6 months with a possibility to become permanent depending on performance.Â You are invited to join our company holidaysÂ (which are completely insane), 2 days holiday per month, boxing yoga + our

team lunches every Friday.Â Oh, and coffee, thereâ€™s plenty of coffee. Just one last thing,Â you can use the office Sauna whenever you want too.#LDN

text = "You will join one of Europeâ€™s most hotly tipped startups with plenty of opportunities to grow and the chance to be part of our little revolution.Â This role lasts for 6 months with a possibility to become permanent depending on performance.Â You are invited to join our company holidaysÂ (which are completely insane), 2 days holiday per month, boxing yoga + our team lunches every Friday.Â Oh, and coffee, thereâ€™s plenty of coffee. Just one last thing,Â you can use the office Sauna whenever you want too.#LDN"

print(get_predictions(text))

CHAPTER-11
REFERENCES

- [1] Gulshan Shrivastava , Member, IEEE, Prabhat Kumar, Senior Member, IEEE, Rudra Pratap Ojha , Pramod Kumar Srivastava , Senthilkumar Mohan , and Gautam Srivastava(2020).” Defensive Modeling of Fake News Through Online Social Networks”. IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS,P.NO:1159-1167
- [2] Nicollas R. de Oliveira; Dianne S. V. Medeiros; Diogo M. F. Mattos.(2020) ” A Sensitive Stylistic Approach to Identify Fake News on Social Networking “ IEEE Signal Processing Letters’’, P.NO:1250 - 1254
- [3] FAIZA MASOOD , GHANA AMMAD, AHMAD ALMOGREN ,ASSAD ABBAS , HASAN ALI KHATTAK.(2019)” Spammer Detection and Fake User Identification on Social Networks “IEEE ACCESS’’.
- [4] CHUNHONG LIU, JINGJING , YANLEI SHANG , CHUANCHANG LIU , BO CHENG(2017) .” Predicting of Job Failure in Compute Cloud Based on Online Extreme Learning Machine: A Comparative Study. “IEEE ACCESS’’.
- [5] Biao Wang, Ge Chen, Luoyi Fu, Li Song, and Xinbing Wang(2017).” DRIMUX: Dynamic Rumor Influence Minimization with User Experience in Social Networks “IEEE Transactions on Knowledge and Data Engineering “.