A

Major Project

On

FAKE OR REAL JOB POSTING PREDICTION

(Submitted in partial fulfillment of the requirements for the award of Degree)

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

By

K.Likhitha	(177R1A0523)
G.Vineetha	(177R1A0517)
C.Prathik Reddy	(177R1A0518)

Under the Guidance of

J. NARASIMHA RAO

(Associate Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CMR TECHNICAL CAMPUS

UGC AUTONOMOUS

(Accredited by NAAC, NBA, Permanently Affiliated to JNTUH, Approved by AICTE, New Delhi)

Recognized Under Section 2(f) & 12(B) of the UGCAct.1956,

Kandlakoya (V), Medchal Road, Hyderabad-501401.

2017-21

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the project entitled "**FAKE OR REAL JOB POSTING PREDICTION**" being submitted by K**.LIKHITHA (177R1A0523), G.VINEETHA(177R1A0517) & C.PRATHIK REDDY (177R1A0518)** in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering to the Jawaharlal Nehru Technological University Hyderabad, is a record of bonafide work carried out by them under our guidance and supervision during the year 2020-21.

The results embodied in this thesis have not been submitted to any other University or Institute for the award of any degree or diploma.

Mr. J. Narasimha Rao
Assocaite Professor
INTERNAL GUIDE

Dr. A. Raji Reddy
DIRECTOR

Dr. K. Srujan Raju
HoD

EXTERNAL EXAMINER

Submitted for viva voice Examination held on ____________________

ACKNOWLEGDEMENT

Apart from the efforts of us, the success of any project depends largely on the encouragement and guidelines of many others. We take this opportunity to express our gratitude to the people who have been instrumental in the successful completion of this project. We take this opportunity to express my profound gratitude and deep regard to my guide

Mr. J. Narasimha Rao, Associate Professor for his exemplary guidance, monitoring and constant encouragement throughout the project work. The blessing, help and guidance given by him shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to Project Review Committee (PRC) Coordinators: **Mr. J. Narasimha Rao, Mr. B. P. Deepak Kumar, Mr. K. Murali, Dr. Suwarna Gothane and Mr. B. Ramji** for their cordial support, valuable information and guidance, which helped us in completing this task through various stages.

We are also thankful to **Dr. K. Srujan Raju,** Head, Department of Computer Science and Engineering for providing encouragement and support for completing this project successfully.

We are obliged to **Dr. A. Raji Reddy,** Director for being cooperative throughout the course of this project. We would like to express our sincere gratitude to Sri. **Ch. Gopal Reddy,** Chairman for providing excellent infrastructure and a nice atmosphere throughout the course of this project.

The guidance and support received from all the members of **CMR Technical Campus** who contributed to the completion of the project. We are grateful for their constant support and help.

Finally, we would like to take this opportunity to thank our family for their constant encouragement, without which this assignment would not be completed. We sincerely acknowledge and thank all those who gave support directly and indirectly in the completion of this project.

K.LIKHITHA (177R1A0523)

G.VINEETHA (177R1A0517)

C.PRATHIK REDDY (177R1A0518)

ABSTRACT

Fake job postings have deceived a number of jobseekers by wasting their time and effort in filling out the applications and waiting for response. Jobseekers have also had financial consequences due to these fake job postings as they ask you to travel considerable distances to interview for fake roles. It takes a lot of time for these job postings to be detected and removed from the websites. In order to detect these job postings, many people have worked on the employment scam dataset that has been published by The University of Aegean. In previous works, we notice that TF-IDF or Count vectorizer is used to convert the text data to word count vectors or word frequency vectors. Later, models like Logistic Regression, Naïve Bayes and some other classification algorithms were used to classify the data into fraudulent or non-fraudulent job posting. Recently, some people have also used regular artificial neural networks and LSTM along with Bert or 200 dimension Glove for word embedding. All these models have achieved very good accuracies.

We have implemented Logistic regression and Naïve Bayes as our baseline models to see how they perform on our unbalanced data. To balance the data we have Oversampled the data and implemented our baseline models on this data. We noticed that the model had started performing better. To improve the accuracy we have chosen to work with CNN models on our oversampled data and used Google's pre-trained model Glove 300d to perform word embedding. We have evaluated all our models and found out that CNN model works the best by achieving the highest accuracy.

LIST OF FIGURES

FIGURE NO	FIGURE NAME	PAGE NO
Figure 3.1	Project Architecture	6
Figure 3.2	Conceptual Diagram	7
Figure 3.3	Use Case diagram	8
Figure 3.4	Class diagram	9
Figure 3.5	Sequence diagram	10
Figure 3.6	Activity diagram	11

LIST OF SCREENSHOTS

SCREENSHOT NO.	SCREENSHOT NAME	PAGE NO.
Screenshot 5.1	Preprocessing of data	17
Screenshot 5.2	Naïve Bayes with all features	18
Screenshot 5.3	Logistic Regression with features	19
Screenshot 5.4	Over sampling(Naïve Bayes)	20
Screenshot 5.5	Over sampling	21
Screenshot 5.6	Final Accuracy	22

TABLE OF CONTENTS

ABSTRACT i

LIST OF FIGURES ii

LIST OF SCREENSHOTS iii

1. INTRODUCTION 1

1.1 PROJECT SCOPE 1

1.2 PROJECT PURPOSE 1

1.3 PROJECT FEATURES 1

2. SYSTEM ANALYSIS 2

2.1 PROBLEM DEFINITION 2

2.2 EXISTING SYSTEM 2

2.2.1 LIMITATIONS OF THE EXISTING SYSTEM 3

2.3 PROPOSED SYSTEM 3

2.3.1 ADVANTAGES OF PROPOSED SYSTEM 3

2.4 FEASIBILITY STUDY 4

2.4.1 ECONOMIC FESIBILITY 4

2.4.2 TECHNICAL FEASIBILITY 4

2.4.3 SOCIAL FEASIBILITY 4

2.5 HARDWARE & SOFTWARE REQUIREMENTS 5

2.5.1 HARDWARE REQUIREMENTS 5

2.5.2 SOFTWARE REQUIREMENTS 5

3. ARCHITECTURE 6

3.1 PROJECT ARCHITECTURE 6

3.2 CONCEPTUAL DIAGRAM 7

3.3 USECASE DIAGRAM 8

3.4 CLASS DIAGRAM 9

3.5 SEQUENCE DIAGRAM 10

3.6 ACTIVITY DIAGRAM 11

4. IMPLEMENTATION 12

4.1 SAMPLE CODE 12

5. SCREENSHOTS 24

6. TESTING 30

6.1 INTRODUCTION TO TESTING 30

6.2 TYPES OF TESTING 30

6.2.1 UNIT TESTING 30

6.2.2 INTEGRATION TESTING 30

6.2.3 FUNCTIONAL TESTING 31

6.3 TEST CASES 32

7. CONCLUSION & FUTURE SCOPE 33

7.1 PROJECT CONCLUSION 33

7.2 FUTURE SCOPE 33

8. REFERENCES 34

8.1 REFERENCES 34

8.2 WEBSITES 34

1. INTRODUCTION

1. INTRODUCTION

1.1 PROJECT SCOPE

It is important to know if one is applying for a genuine job posting or a fake job posting as he can apply for hundreds of companies mentioned in the job boards and hardly receive any interview calls. It not only wastes his time but also his energy and hope of getting a job. Companies often fake hiring ads in order to get market intel or use their job posting credits. It is necessary for us to know which companies are fake and genuine so that the genuine job posting companies don’t waste their time and resources and the ones seeking a job can achieve more by matching better.

1.2 PROJECT PURPOSE

In this project, we aim to classify a job posting as a fake or real one by taking the job posting’s title, profile of the company and other significant texts such as job description, requirements, benefits and many more. We chose Logistic Regression and Naïve Bayes as our baseline model. As our hypothesis is that only job description and requirements are the significant features for predicting fraudulent or non-fraudulent job postings, we implemented our models by considering only some features of the dataset as we wanted to check if our hypothesis was true. We realized that these features are not sufficient and decided to use the remaining features. Though we used all the features the model did not perform well. To address this issue we did analysis on weights and decided to oversample the data and implement the baseline models on the oversampled data. The model performance had shown great improvement.

1.3 PROJECT FEATURES

The main Feature of this is it analyzes whether the job posting is real or fake using the datasets. It also gives the accuracy of the taken datasets.

2. SYSTEM ANALYSIS

2. SYSTEM ANALYSIS

SYSTEM ANALYSIS

System Analysis is the important phase in the system development process. The System is studied to the minute details and analyzed. The system analyst plays an important role of an interrogator and dwells deep into the working of the present system. In analysis, a detailed study of these operations performed by the system and their relationships within and outside the system is done. A key question considered here is, “what must be done to solve the problem?” The system is viewed as a whole and the inputs to the system are identified. Once analysis is completed the analyst has a firm understanding of what is to be done.

2.1 PROBLEM DEFINITION

To predict if a job posting is a fraudulent posting or a genuine posting using our baseline models and score the models.To preprocess the data by cleaning it and then converting the text data into word frequency vector by using TFIDF. To see if job description and job requirements are the only significant features or the other features are also important. Do analysis of weights to know if oversampling can increase the model performance or not. Do oversampling and then split the data into train and test data. Choosing the right hyper tuning parameters for our CNN model. To implement our baseline models and CNN model on this oversampled data and score the models. Evaluate the models to see which model works the best for our dataset

2.2 EXISTING SYSTEM

It is important to know if one is applying for a genuine job posting or a fake job posting as he can apply for hundreds of companies mentioned in the job boards and hardly receive any interview calls. In previous works, people tried to implement this application with different models and also not oversampling the data which didn’t work well in terms of accuracy and compatibility. Choosing the right model for your application is very important to get better and efficient results.

2.2.1 LIMITATIONS OF EXISTING SYSTEM

1. Time inefficient
2. Gives inaccurate results which effects factors like:
   - Money
   - Effort
   - Time
3. Complexity issues

To avoid all these limitations and make the working more accurately the system needs tobe implemented efficiently.

2.3 PROPOSED SYSTEM

In Proposed System, we can predict whether the job posting is real or fake.To identify whether the posting is fake or real, We use the job posting’s title, profile of the company and other significant texts such as job description, requirements, benefits and many more. As our hypothesis is that only job description and requirements are the significant features for predicting fraudulent or non-fraudulent job postings, we implemented our models by considering only some features of the dataset as we wanted to check if our hypothesis was true.

2.3.1 ADVANTAGES OF THE PROPOSED SYSTEM

.

The advantages of the proposed system are:

1. Glove 300d which helps in increasing the time efficiency.
2. Give accurate results with 97% accuracy.
3. Complexity issue is solved.

2.4 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. Three key considerations involved in the feasibility analysis.

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

2.4.1 ECONOMIC FEASIBILITY

The developing system must be justified by cost and benefit. Criteria to ensure that effort is concentrated on project, which will give best, return at the earliest. One of the factors, which affect the development of a new system, is the cost it would require.

The following are some of the important financial questions asked during preliminary investigation:

- The costs conduct a full system investigation.
- The cost of the hardware and software.
- The benefits in the form of reduced costs or fewer costly errors.

Since the system is developed as part of project work, there is no manual cost to spend forthe proposed system. Also all the resources are already available, it give an indication of the system is economically possible for development.

2.4.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.4.3 SOCIAL FEASIBILITY

This includes the following questions:

- Is there sufficient support for the users?
- Will the proposed system cause harm?

The project would be beneficial because it satisfies the objectives when developed and installed. All behavioral aspects are considered carefully and conclude that the project is behaviorally feasible.

2.5 HARDWARE & SOFTWARE REQUIREMENTS

2.5.1 HARDWARE REQUIREMENTS:

Hardware interfaces specifies the logical characteristics of each interface between the software product and the hardware components of the system. The following are some hardware requirements.

- ■ Google Cloud to run our CNN model.
- ■ Instance specifications in Google Cloud:
  1. CPU 16 core
  2. 60 GB RAM
  3. GPU Nvidia Tesla V100

2.5.2 SOFTWARE REQUIREMENTS:

Software Requirements specifies the logical characteristics of each interface and software components of the system. The following are some software requirements,

1.Jupyter Notebook
2.Jupyter Lab
3.Machine Learning packages
4.Google Collab

3. ARCHITECTURE

3. ARCHITECTURE

3.1 PROJECT ARCITECTURE

This project architecture shows the procedure followed for finding the accuracy using naïve bayes and logistic regression.

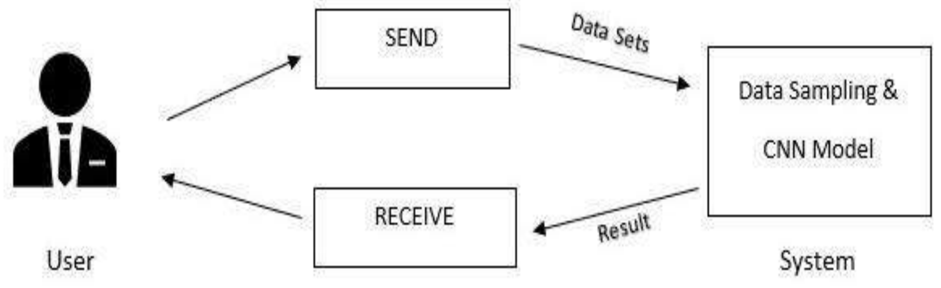


Figure 3.1: Project Architecture of Fake or real job posting prediction

DESCRIPTION

1.Data Sets: Data Sets with default parameters. The dataset was provided by the University of the Aegean, Laboratory of Information & Communication system security. The data set consists of 18k job descriptions out of which 800 are fake. This data set has 18 columns which consists of some significant variables with textual data.

2.Data sampling: Data sampling will be done using naïve bayes and logisctic regression algorithm.

3.Result: Result will show the accuracy of the data sets taken using CNN model. CNN model provides high accuracy.

3.2 CONCEPTUAL DIAGRAM

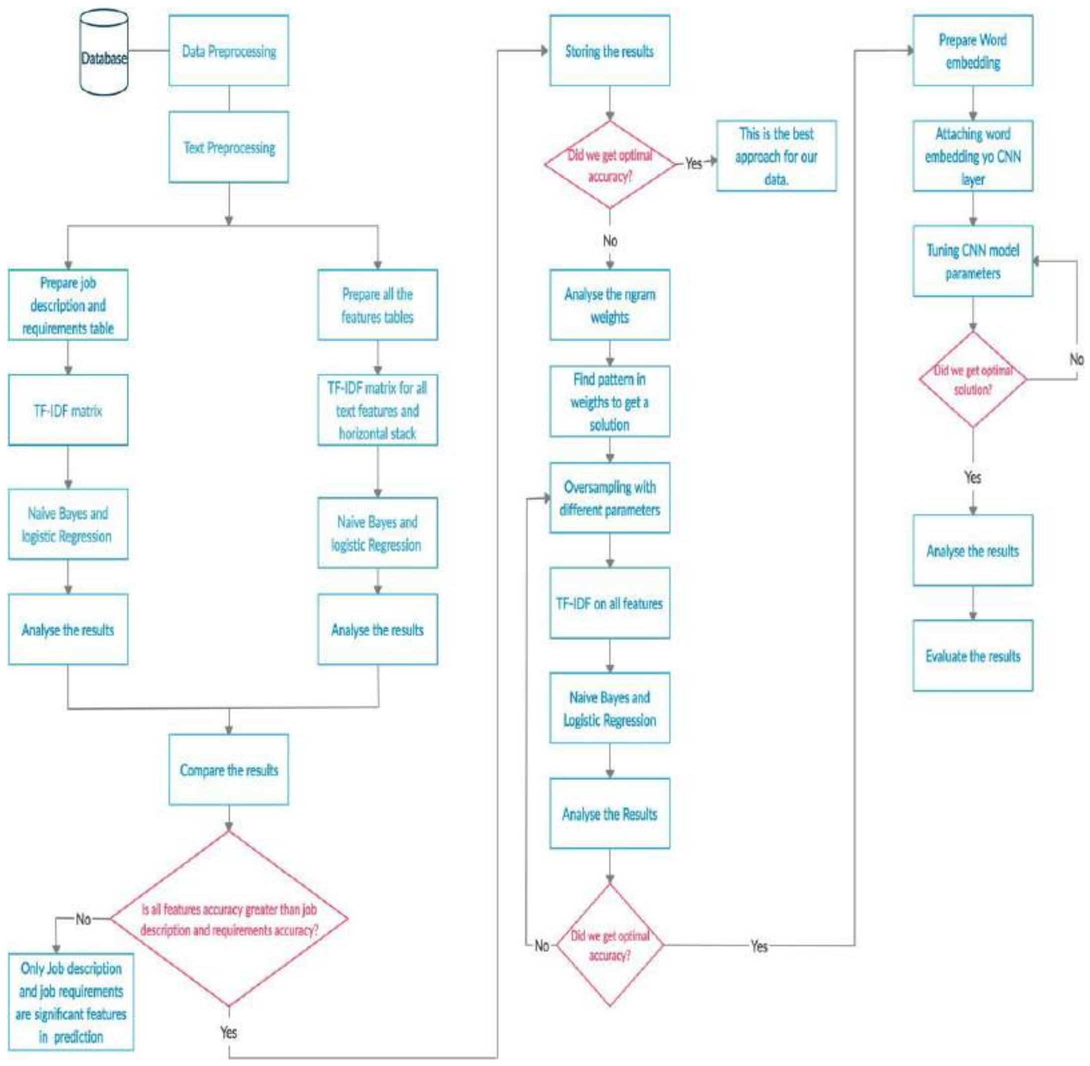

Database
Data Preprocessing
Text Preprocessing
Prepare job description and requirements table
Prepare all the features tables
TF-IDF matrix
TF-IDF matrix for all text features and horizontal stack
Naive Bayes and logistic Regression
Naive Bayes and logistic Regression
Analyse the results
Analyse the results
Compare the results
Is all features accuracy greater than job description and requirements accuracy?
No
Only Job description and job requirements are significant features in prediction
Yes
Storing the results
Did we get optimal accuracy?
Yes
This is the best approach for our data.
No
Analyse the ngram weights
Find pattern in weigths to get a solution
Oversampling with different parameters
TF-IDF on all features
Naive Bayes and Logistic Regression
Analyse the Results
Did we get optimal accuracy?
No
Yes
Prepare Word embedding
Attaching word embedding yo CNN layer
Tuning CNN model parameters
Did we get optimal solution?
No
Yes
Analyse the results
Evaluate the results

Figure 3.2: Conceptual Diagram of Fake or real job posting prediction

3.3 USE CASE DIAGRAM

In the use case diagram we have basically three actors who are the user A, user B and the application. The user A and User B have the rights to login, sign in, Forget Password and History. Whereas the application has the accuracy and the result.

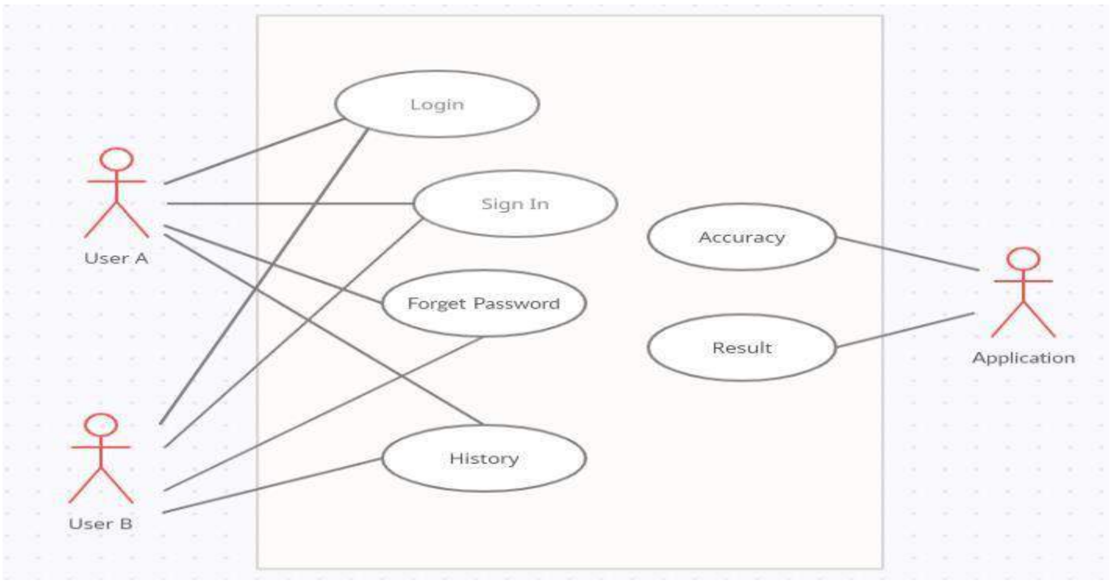


Figure 3.3: Use Case Diagram of user and application

3.4 CLASS DIAGRAM

Class Diagram is a collection of classes and objects.

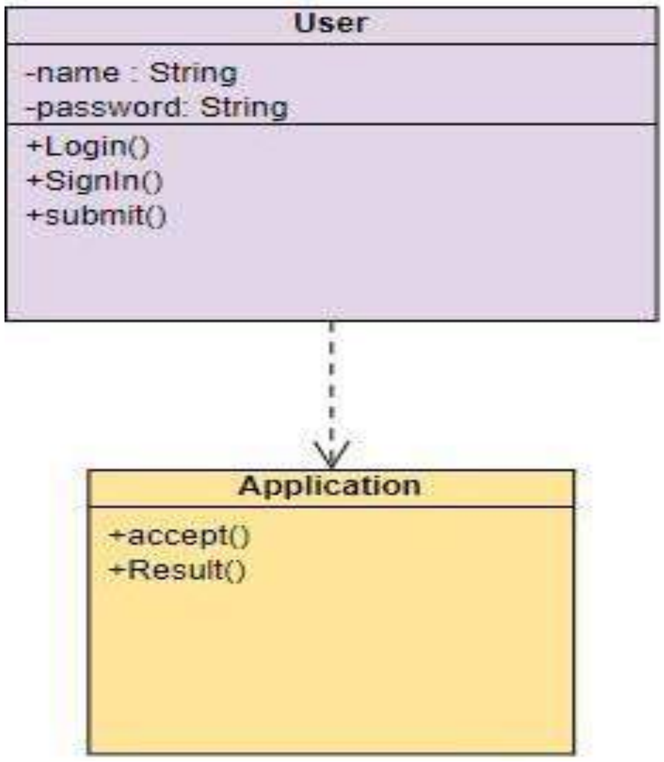


Figure 3.4: Class Diagram for Application and User

3.5 SEQUENCE DIAGRAM

Sequence Diagram are interaction diagrams that details how operations are carried out.

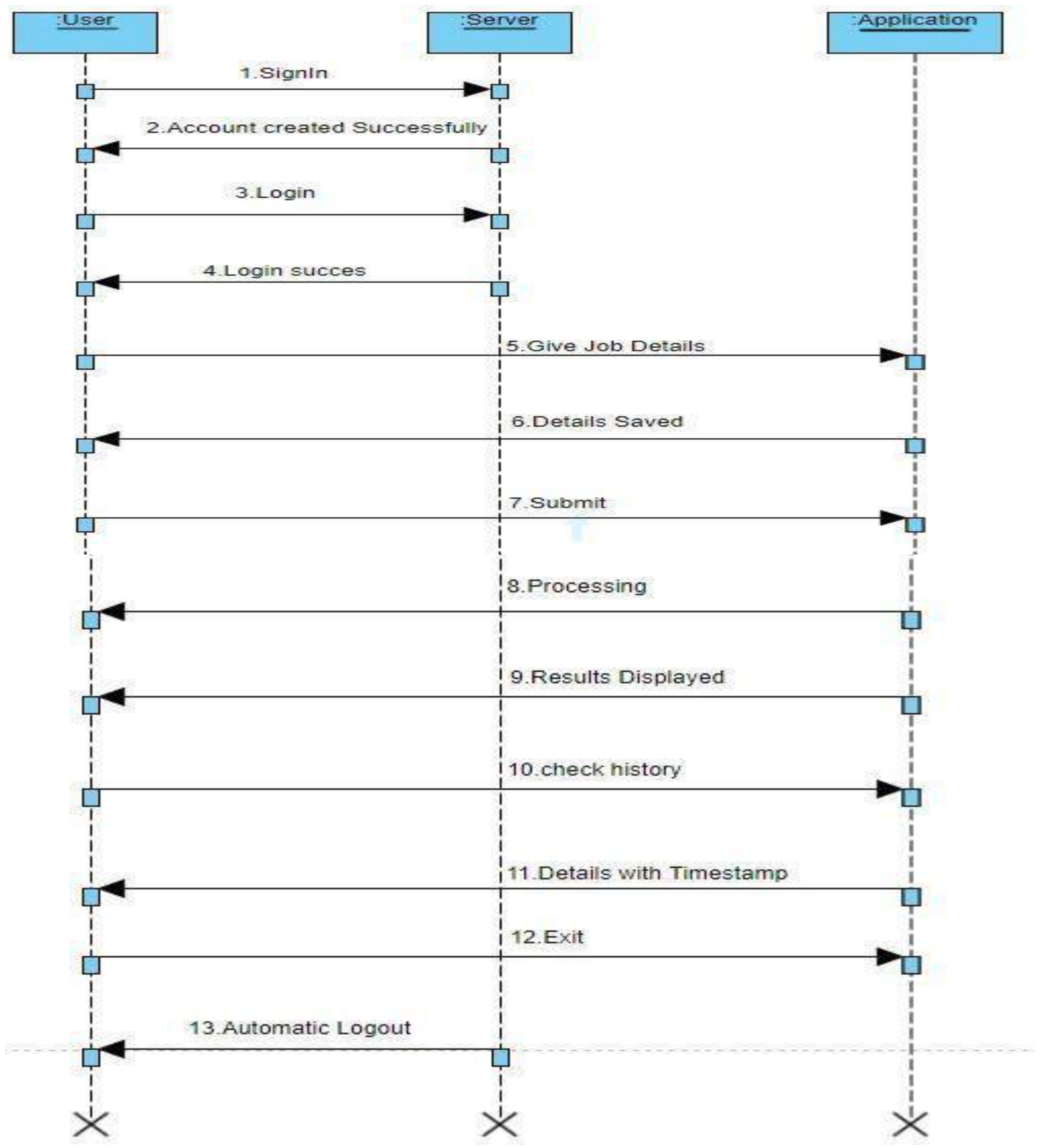


Figure 3.5: Sequence Diagram for Real or fake job posting prediction

3.6 ACTIVITY DIAGRAM

It describes about flow of activity states.

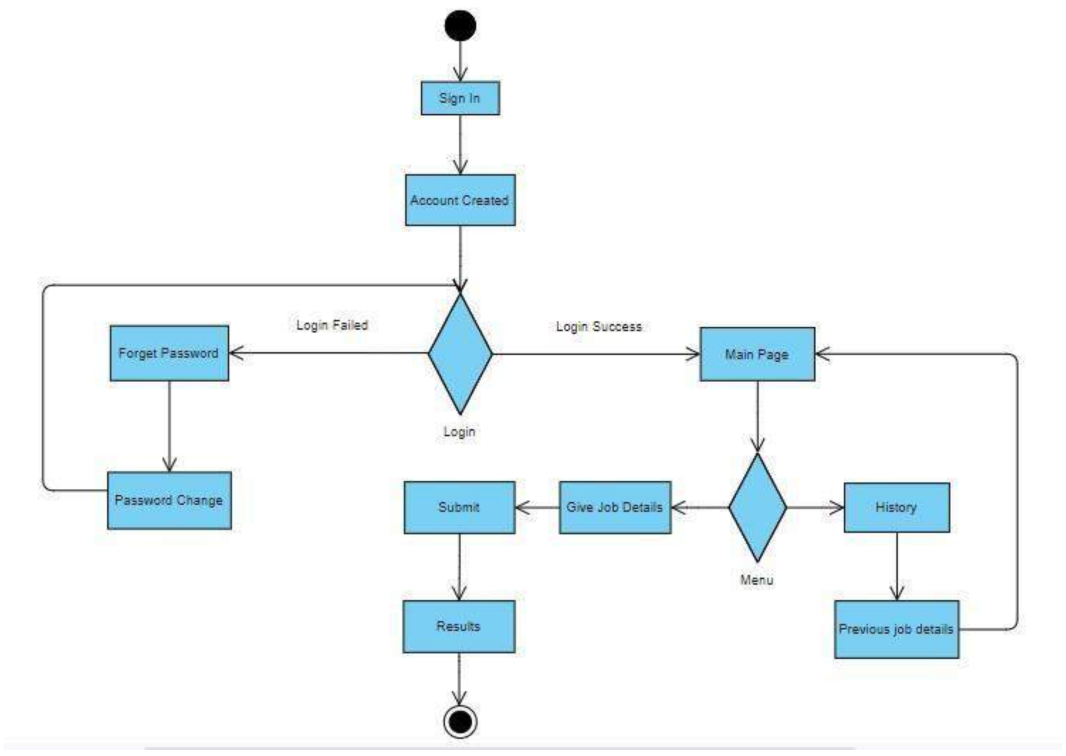


Figure 3.6: Activity Diagram of user for fake or real job posting prediction

4. IMPLEMENTATION

4. IMPLEMENTATION

4.1 SAMPLE CODE

Data Preprocessing code:

```
import numpy as np
import pandas as pd
import nltk as nltk
import matplotlib.pyplot as plt
from google.colab import drive
drive.mount('/content/drive')
df=pd.read_csv('job_postings.csv')
df.head()

#check the null values
df.isna().sum()
#shape of the dataset
df.shape

#dropping features as they are too many mising values
df.drop(columns=['location','department','salary_range','required_experience',
'required_education','industry'],axis=1,inplace=True)

# Replace missing values of these featues with no information
df['requirements'].fillna("Noinformation", inplace = True)
df['benefits'].fillna("Noinformation", inplace = True)
df['company_profile'].fillna("Noinformation", inplace = True)
#check the unique values of function column
df['function'].unique()

#categorize the 37 unique values into 5 categories
df['function'].replace(dict.fromkeys(['Marketing','Finance','Advertising','Accounting
/Auditing','Writing/Editing','Legal'], 'Marketing and Finance'),inplace=True)
df['function'].replace(dict.fromkeys(['Customer Service','Consulting','Health Care
Provider','Art/Creative','Science','Training','Education','Research'],'Other'),inplace=Tr
ue)
df['function'].replace(dict.fromkeys(['Production','Supply
Chain','Design','Manufacturing',
'Quality Assurance','Distribution','Purchasing'],'Sales'),inplace=True)
df['function'].replace(dict.fromkeys(['Administrative','Business Development','Project
Management'
,'Human Resources','Public Relations','Strategy/Planning','General Business','Product
Management']
,'Management'),inplace=True)
df['function'].replace(dict.fromkeys(['Information Technology','Engineering','Data
Analyst',
'Business Analyst','Financial Analyst'],'Information Technology'),inplace=True)

#now again check unique values
df['function'].unique()

#Replace the missing values of these columns with mode
for column in ['employment_type','function']:
    df[column].fillna(df[column].mode()[0], inplace=True)
#all missing data is now replcaed with mode
df['function'].unique()

# Drop the one record with description having missing data
df.dropna(axis=0,subset=['description'], inplace=True)
df.isna().sum()
# Now check the shape of the dataframe after cleaning the data
df.shape
df.head()

#Compare the target variable class
!pip install plotly

#Plot the target variable against count
import plotly.express as px
count = df["fraudulent"].value_counts()
fig = px.bar(df, x= count.index,y=count)
fig.update_layout(
    title="Count of target variable",
    xaxis_title="Count",
    yaxis_title="Target Class"
)
fig.show()

#plot for showing count of fraudulent and non-fraudulent job postings for each
function
#df.groupby(["function","fraudulent"])["job_id"].count()
Width = 0.30
nonf = [9259,1707, 1424, 2489, 2135]
f = [491,160, 59, 106,49 ]
g1 = np.arange(len(nonf))
g2 = [i + Width for i in g1]
plt.figure(figsize=(10,6))
plt.bar(g1, nonf, color='#F65656', width=Width, edgecolor='white', label='non
fraudulent')
plt.bar(g2, f, color='#0044E4', width=Width, edgecolor='white', label='fraudulent')
plt.title('Plot showing count of fraudulent and non-fraudulent job postings for each
function category',fontweight='bold')
plt.xlabel('Function categories', fontweight='bold')
plt.xticks([r + Width for r in range(len(nonf))], ['Information Technology',
'Management',
'Marketing and Finance', 'Other', 'Sales'])
plt.ylabel('Count', fontweight='bold')
plt.legend()
plt.grid()
plt.show()

#plot for showing count of fraudulent and non-fraudulent job postings for each
employment type
#df.groupby(["employment_type","fraudulent"])["job_id"].count()
Width = 0.30
non_f = [1480,14360, 212, 723, 239]
f_1 = [44,730, 15, 74,2 ]
b1 = np.arange(len(non_f))
b2 = [i + Width for i in b1]
plt.figure(figsize=(10,6))
plt.bar(b1, non_f, color='#F65656', width=Width, edgecolor='white', label='non
fraudulent')
plt.bar(b2, f_1, color='#0044E4', width=Width, edgecolor='white', label='fraudulent')
plt.title('Plot showing count of fraudulent and non-fraudulent job postings for each
employment type',fontweight='bold')
plt.xlabel('Employment type', fontweight='bold')
plt.xticks([r + Width for r in range(len(non_f))], ['Contract', 'Full-time', 'Other', 'Part-
time', 'Temporary'])
plt.ylabel('Count', fontweight='bold')
plt.legend()
plt.grid()
plt.show()

# Combine the description and requirements of job to one column called job_text
df['job_text'] = df['description'].str.cat(df['requirements'], sep =" ")


#import all packages for text preprocessing
import re
!pip install langdetect
from langdetect import detect
import nltk
from nltk.tokenize import RegexpTokenizer
from nltk.stem import PorterStemmer
!pip install emoji
import emoji
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.metrics import classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
import sklearn
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score


# Function of remove the emojis
def check_emoji(a):
   i=0
   vector=''
   array=[]
   for char in a:
      if char in emoji.UNICODE_EMOJI:
          array.append(char)
      else:
         vector= vector + a[i]
      i=i+1
   return vector

#function to tokenize the data, remove special characters and convefrt text to lower
case
def convert_token(a):
   txt=check_emoji(a)
   token = RegexpTokenizer('\w+|\!')
   toks=token.tokenize(txt.lower())
   return toks

#Function to do stemming using PorterStemmer()
def token_stemming(df):
   token_stemmed=[]
   sent=df['job_text']
   toks = convert_token(sent)
   stem = []
   s=PorterStemmer()
   for t in toks:
      stem.append(s.stem(t))
      token_stemmed.append(s.stem(t))
      ret=' '.join(stem)
   return(ret)


#send the job_text data for text preprocessing
df['job_text']=df[["job_text"]].apply(token_stemming,axis=1)

#create a new dataframe with processed text data and target variable
frame = { 'job_text': df['job_text'], 'fraudulent': df['fraudulent']}
des_req = pd.DataFrame(frame)
des_req.head(5)

# Copy this dataframe to a new csv file called job_text
#This csv will be used for TF-IDF vectorization later
des_req.to_csv('job_text.csv')
```

```
#Now combine all the text data features into one column called job_data
job_data=df[['title','company_profile','description','requirements','benefits']].agg('
'.join, axis=1)
df['job_data']=job_data

# Function to remove the emojis
def check_emoji(a):
   i=0
   vector=''
   array=[]
   for char in a:
      if char in emoji.UNICODE_EMOJI:
          array.append(char)
      else:
         vector= vector + a[i]
      i=i+1
   return vector

#function to tokenize the data, remove special characters and convefrt text to lower
case
def convert_token(a):
   txt=check_emoji(a)
   token = RegexpTokenizer('\w+|\!')
   toks=token.tokenize(txt.lower())
   return toks

#Function to do stemming using PorterStemmer()
def token_stemming(df):
   token_stemmed=[]
   sent=df['job_data']
   toks = convert_token(sent)
   stem = []
   s=PorterStemmer()
   for t in toks:
      stem.append(s.stem(t))
      token_stemmed.append(s.stem(t))
      ret=' '.join(stem)
   return(ret)

#send the job_data data for text preprocessing
df['job_data']=df[["job_data"]].apply(token_stemming,axis=1)
#Copy this preprocessed text data and other features into a new dataframe
frame1 = { 'job_data':
df['job_data'],'employment_type':df['employment_type'],'function':df['function'],
'fraudulent': df['fraudulent'],

'telecommuting':df['telecommuting'],'has_company_logo':df['has_company_logo'],
        'has_questions':df['has_questions']
        }
all_fea = pd.DataFrame(frame1)
```

```
#Do one hot encoding for categorical data
a=pd.get_dummies(all_fea['employment_type'])
b=pd.get_dummies(all_fea['function'])
all_fea=pd.concat([all_fea,a,b],axis=1)
all_fea.head(5)

#Drop the columns that were one hot encoded
all_fea.drop(['employment_type','function'],axis=1,inplace=True)
all_fea

# Copy this dataframe to a new csv file called job_data
#This csv will be used for TF-IDF vectorization later
all_fea.to_csv('job_data.csv')
```

Over Sampling Data:

```
#importing all the required packages
import numpy as np
import pandas as pd
import nltk as nltk
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
import sklearn
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import StratifiedKFold
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
import statistics as s
from sklearn.metrics import confusion_matrix
from scipy.sparse import hstack
from google.colab import drive
drive.mount('/content/drive')

#importing preprocessed data
df=pd.read_csv('/content/drive/My Drive/AIt/job_data.csv')
#Preparing the data for train test split
x=df.drop(['fraudulent'],axis=1)
y=df['fraudulent']
```

```
#separate the non-fraudulent data and fraudulent data to duplicate fraudulent data
nonfraudulent_train=X_train[X_train['fraudulent']==0]
fraudulent_train=X_train[X_train['fraudulent']==1]

#duplicate the fraudulent data by 17 times
#after conducting experiments duplicating it 17 times gives optimal accuracy
fraudulent_train=fraudulent_train.append([fraudulent_train]*17,ignore_index=True)

#join the duplicated fraudulent data to the non fraudulent data
final_train=fraudulent_train.append(nonfraudulent_train,ignore_index=True)
final_test=X_test
#save these train and test sets to use it for CNN model
final_train.to_csv("trainset.csv")
final_test.to_csv("testset.csv")
final_train.head()

# copy the data frame into a new variable to use it for horizontal stack
new_df=final_train.copy()
new_df.drop(['job_data','fraudulent'],axis=1,inplace=True)
#We see that train set increased from 17000 to 25000 after oversampling
final_train.shape

#arrange the data into final train and test sets


#Doing K fold Cross Validation on the training set. We use 5 folds so n_splits=5
folds = StratifiedKFold(n_splits=5, random_state=0)
i=0
#create a list for storing accuracy reports and scores
acc_report=[]
acc_score=[]
for train, test in folds.split(finalx_train, finaly_train):
   print(train)
   print(test)
   #taking only text data from training data into another variable to perform TF-idf
vectorization
   x_train1, x_test1 = finalx_train['job_data'][train], finalx_train['job_data'][test]
   #taking the test data into another variable
   y_train1, y_test1 = finaly_train[train], finaly_train[test]
   i=i+1
   #Using TF-IDF vectorizer to generate a TF-IDF matrix with ngram range (1,2)
   tfidf = TfidfVectorizer(stop_words="english",ngram_range=(1,2))
   x_train1 = tfidf.fit_transform(x_train1)
   x_test1 = tfidf.transform(x_test1)
   print("tfidf"+str(i))

   #Using the copy of the dataframe with no job_data to access the other features of
the dataset
   rx_train=new_df.iloc[train]
   rx_test=new_df.iloc[test]
   #Use hstack to horizontally stack the job_data Tf-idf matrix to other features of the
dataset
   xtrain_data=hstack( [x_train1,rx_train] )
   xtest_data=hstack( [x_test1,rx_test] )

   #Perform Naive Bayes classification to make predictions on the train data and
check how model works on our dataset.
   clf = MultinomialNB(alpha=1.0)
   clf.fit(xtrain_data, y_train1)
   y_pred = clf.predict(xtest_data)
   scores = accuracy_score(y_test1, y_pred)
   report=classification_report(y_test1,y_pred)
   #use append() to store all the 5 reports and scores in the created list
   acc_report.append(report)
   acc_score.append(scores)

print(s.mean(acc_score))

#Now using TF-IDF vectorizer to generate TF-IDF matrix for both X_train and x_test
with ngram range (1,2)
tfidf = TfidfVectorizer(stop_words="english",ngram_range=(1,2))
x_train=tfidf.fit_transform(finalx_train['job_data'])
x_test=tfidf.transform(finalx_test['job_data'])
#Drop job_data from train and test set for horizontal stacking of job_data's TF-IDF
matrix
finalx_train.drop('job_data',axis=1,inplace=True)
finalx_test.drop('job_data',axis=1,inplace=True)
#Use hstack to horizontal stack the TF-IDF matrix to the other features
train_data=hstack( [finalx_train,x_train] )
test_data=hstack( [finalx_test,x_test] )
#check the shape of train data after horizontal stack
train_data.shape

#using Naive Bayes with smoothing factor value as 1 and make predictionb on test
data
clf = MultinomialNB(alpha=1.0)
clf.fit(train_data,finaly_train)
pred = clf.predict(test_data)
scores = accuracy_score(finaly_test, pred)
report=classification_report(finaly_test, pred)
# Print the accuracy report, scores and confusion matrix
print("Accuracy:","\n",scores)
print("Accuracy report:","\n",report)
print("Confusion Matrix:","\n",confusion_matrix(y_test, pred))

#Using Logistic Regression with default parameters to make predictions on test data
classifier = LogisticRegression()
classifier.fit(train_data , finaly_train)
prediction=classifier.predict(test_data)
```

```
model_score = accuracy_score(finaly_test, prediction)
class_report=classification_report(finaly_test, prediction)
# Print the accuracy report, scores and confusion matrix
print("Accuracy:","\n", model_score)
print("Accuracy report:","\n", report)
print("Confusion Matrix:","\n", confusion_matrix(y_test, prediction))
```

Final Accuracy using CNN model:

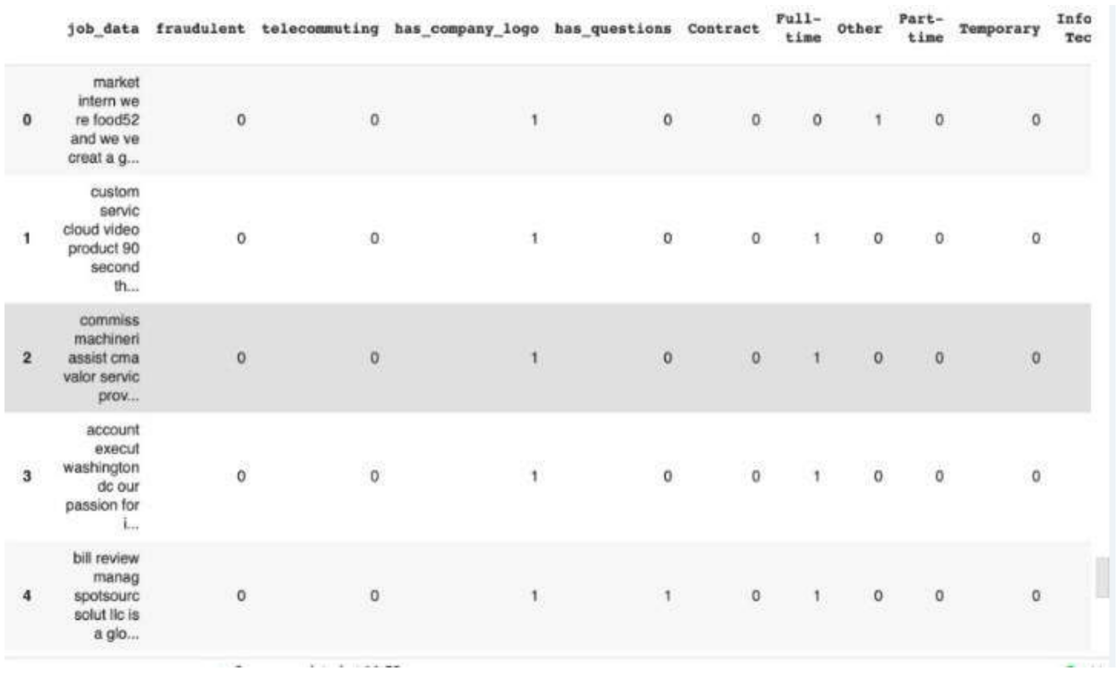
```
#import all the packages
import numpy as np
import pandas as pd
import re

from nltk.tokenize import RegexpTokenizer
from nltk.stem import PorterStemmer
!pip install emoji
import emoji
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.utils import shuffle
from sklearn.externals import joblib
from sklearn.preprocessing import Normalizer
from scipy.sparse import hstack
from sklearn.metrics import classification_report
from statistics import mean
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasClassifier
from keras.utils import np_utils
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import LabelEncoder
from sklearn.pipeline import Pipeline
from keras.layers import Embedding
from keras.layers.core import SpatialDropout1D
from keras.layers.core import Dropout
from keras.layers.recurrent import LSTM
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.layers import Conv1D, GlobalMaxPooling1D
from keras.layers import Dense, Activation, Flatten
from google.colab import drive
drive.mount('/content/drive')
from keras.layers import MaxPooling1D

import keras
from keras import backend as b
import tensorflow as tf

# Define our own loss function
def loss(y_true, y_pred):
    g = 2.0
    a = 0.25
    pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))
    pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))
    return -b.sum(a * b.pow(1. - pt_1, g) * b.log(pt_1))-b.sum((1-a) * b.pow( pt_0, g) *
b.log(1. - pt_0))

# Define a function to load embedding into memory
def embedding(filename):

    f = open(filename,'r')
    lines = f.readlines()
    f.close()

    emb =  dict()
    for line in lines:
        parts = line.split()

        emb[parts[0]] = np.asarray(parts[1:], dtype='float32')
    return emb
# define weight matrix dimensions
def weight_matrix(embedding, vocab):
    vocab_size = len(vocab) + 1
    weight_matrix = np.zeros((vocab_size, 300))

    for word, i in vocab.items():
        vector = embedding.get(word)
        if vector is not None:
            weight_matrix[i] = vector
    return weight_matrix
#load dataset train set
df_train=pd.read_csv("/content/drive/My Drive/AIt/trainset (1).csv")
#load test set
df_test=pd.read_csv("/content/drive/My Drive/AIt/testset (1).csv")
#drop unnamed column
df_train.drop(df_train.columns[df_train.columns.str.contains('unnamed',case =
False)],axis = 1, inplace = True)
#drop unnamed column
df_test.drop(df_test.columns[df_test.columns.str.contains('unnamed',case =
False)],axis = 1, inplace = True)
#Take job data in x and fraudulent in y and check the shape
X_train=df_train['job_data'].values
X_test=df_test['job_data'].values
Y_train=df_train['fraudulent']
Y_test=df_test['fraudulent']
print(X_train.shape,Y_train.shape)
print(X_test.shape,Y_test.shape)
#check unique values in y_train
Y_train.unique()

#do one hot encoding for y_train
Ytrain = pd.get_dummies(Y_train).values
Ytrain


# every word is assigned a unique number in the train data
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X_train)
#the unique values are replaced in the train set
encoding_layer = tokenizer.texts_to_sequences(X_train)
#identify longest sequence of words in document and do padding for remaining
sequences if needed
Xtrain = pad_sequences(encoding_layer, maxlen=3432, padding='post')
#finds length of tokens in a document
document_size = len(tokenizer.word_index) + 1
#load the golve 300 to a new variable using the function defined
glove_d = embedding('/content/drive/My Drive/AIt/glove.6B.300d.txt')
#load the weights of the words to a new variable
embed_vect = weight_matrix(glove_d, tokenizer.word_index)
#changing the words in the tokens to 300d vector
embed_layer = Embedding(document_size, 300, weights=[embed_vect],
input_length=3432, trainable=False)
#using sequential 2 layer model with certain parameters
cnn = Sequential()
cnn.add(embed_layer)

cnn.add(Conv1D(100, 2, activation='relu'))

cnn.add(MaxPooling1D(2))
cnn.add(Conv1D(200, 2, activation='relu'))
cnn.add(MaxPooling1D(2))

cnn.add(SpatialDropout1D(0.2))

cnn.add(Flatten())
cnn.add(Dense(2, activation='sigmoid'))
cnn.compile(loss=['binary_crossentropy'], optimizer='adam', metrics=['accuracy'])
#the final epochs value is 3 and batch size is 64
epochs = 3
```

```
batch_size = 64

fit = cnn.fit(Xtrain, Ytrain, epochs=epochs,
batch_size=batch_size,validation_split=0.1)


# every word is assigned a unique number in the test data
encod_layer_test = tokenizer.texts_to_sequences(X_test)
#the unique values are replaced in the test set
Xtest = pad_sequences(encod_layer_test, maxlen=3432, padding='post')
#evaluate model on test set and calculate accuracy
loss, accuracy = cnn.evaluate(Xtest, Ytest, verbose=0)
print('Accuracy of the model: %f' % (accuracy*100))
```

5. SCREENSHOTS

5. EXECUTION SCREENSHOTS

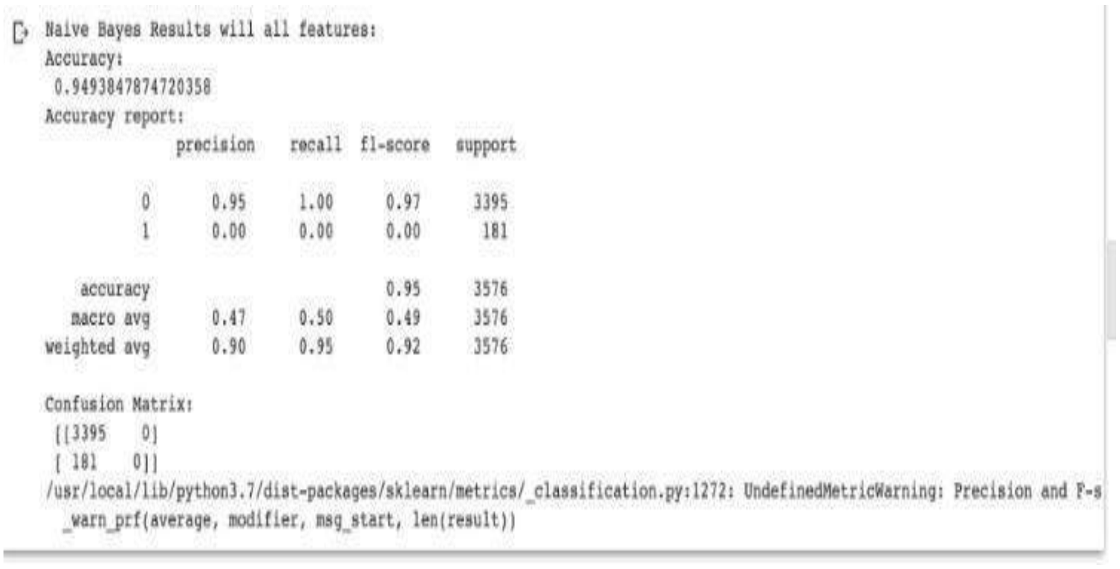
5.1 Preprocessing of Data



	job_data	fraudulent	telecommuting	has_company_logo	has_questions	Contract	Full-time	Other	Part-time	Temporary	Info Tec
0	market intern we re food52 and we ve creat a g...	0	0	1	0	0	0	1	0	0	
1	custom servic cloud video product 90 second th...	0	0	1	0	0	1	0	0	0	
2	commiss machineri assist cma valor servic prov...	0	0	1	0	0	1	0	0	0	
3	account execut washington dc our passion for i...	0	0	1	0	0	1	0	0	0	
4	bill review manag spotsourc solut llc is a glo...	0	0	1	1	0	1	0	0	0	

Screenshot 5.1: Preprocessing of data

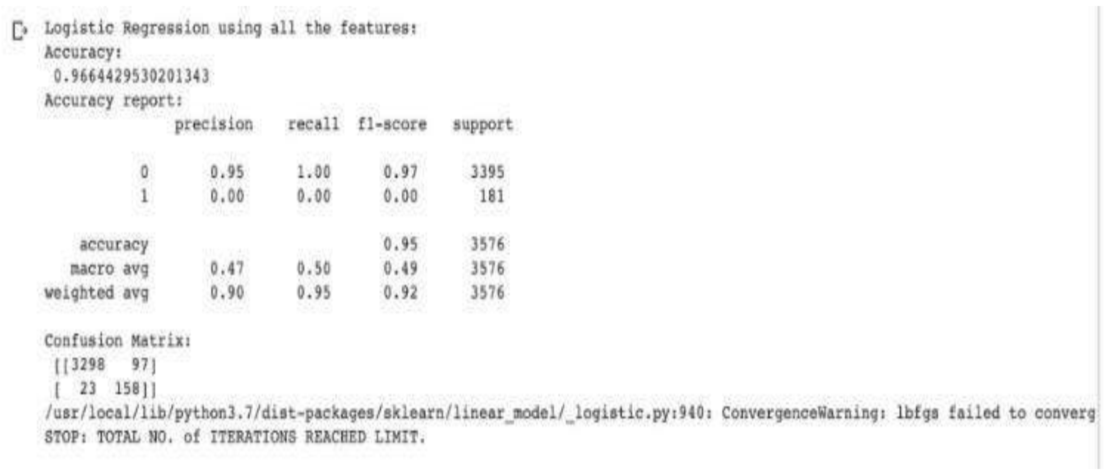
5.2 NAIVE BAYES WITH ALL FEATURES



```
Naive Bayes Results will all features:
Accuracy:
 0.9493847874720358
Accuracy report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      3395
           1       0.00      0.00      0.00       181

    accuracy                           0.95      3576
   macro avg       0.47      0.50      0.49      3576
weighted avg       0.90      0.95      0.92      3576

Confusion Matrix:
 [[3395    0]
 [ 181    0]]
/usr/local/lib/python3.7/dist-packages/sklearn/metrics/_classification.py:1272: UndefinedMetricWarning: Precision and F-s
  _warn_prf(average, modifier, msg_start, len(result))
```

Screenshot 5.2:Accuracy using Naïve bayes with all Features

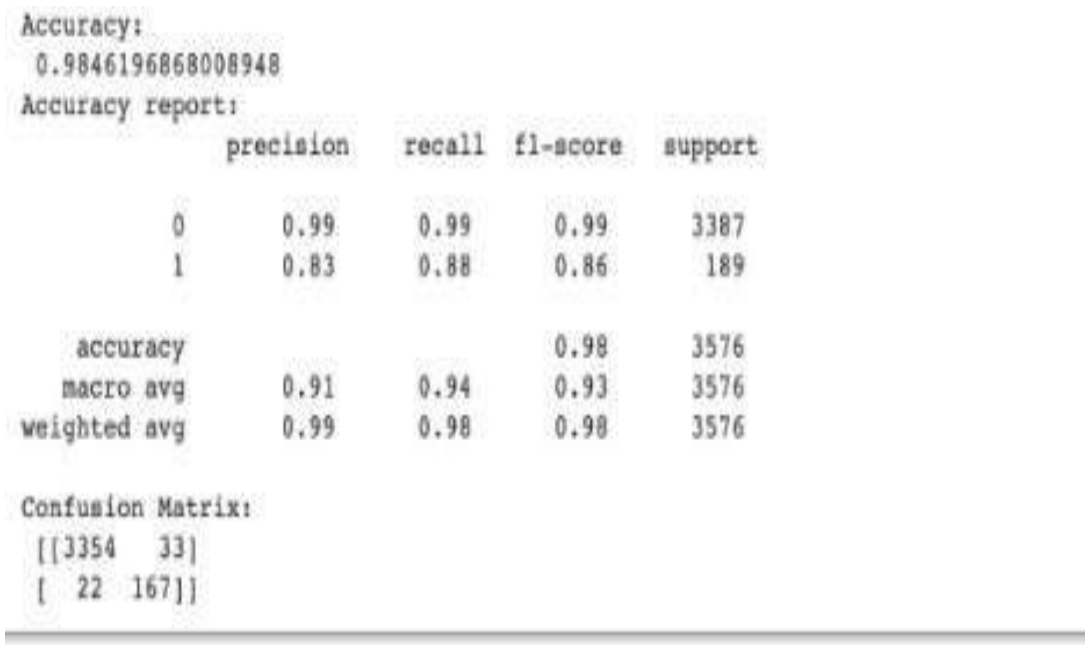
5.3 LOGISTIC REGRESSION WITH ALL FEATURES



```
Logistic Regression using all the features:
Accuracy:
 0.9664429530201343
Accuracy report:
              precision    recall  f1-score   support

           0       0.95      1.00      0.97      3395
           1       0.00      0.00      0.00       181

    accuracy                           0.95      3576
   macro avg       0.47      0.50      0.49      3576
weighted avg       0.90      0.95      0.92      3576

Confusion Matrix:
 [[3298   97]
 [  23  158]]
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.
```

Screenshot 5.3: Accuracy using all features with logistic regression

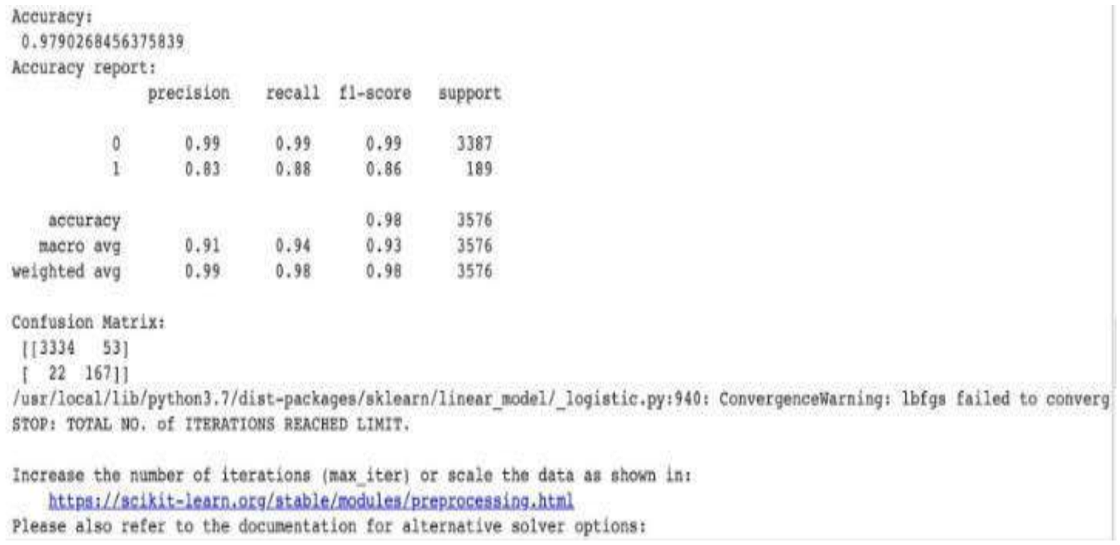
5.4 OVER SAMPLING DATA



```
Accuracy:
 0.9846196868008948
Accuracy report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      3387
           1       0.83      0.88      0.86       189

    accuracy                           0.98      3576
   macro avg       0.91      0.94      0.93      3576
weighted avg       0.99      0.98      0.98      3576

Confusion Matrix:
 [[3354   33]
 [  22  167]]
```

Screenshot 5.4:Over sampling data using naïve bayes

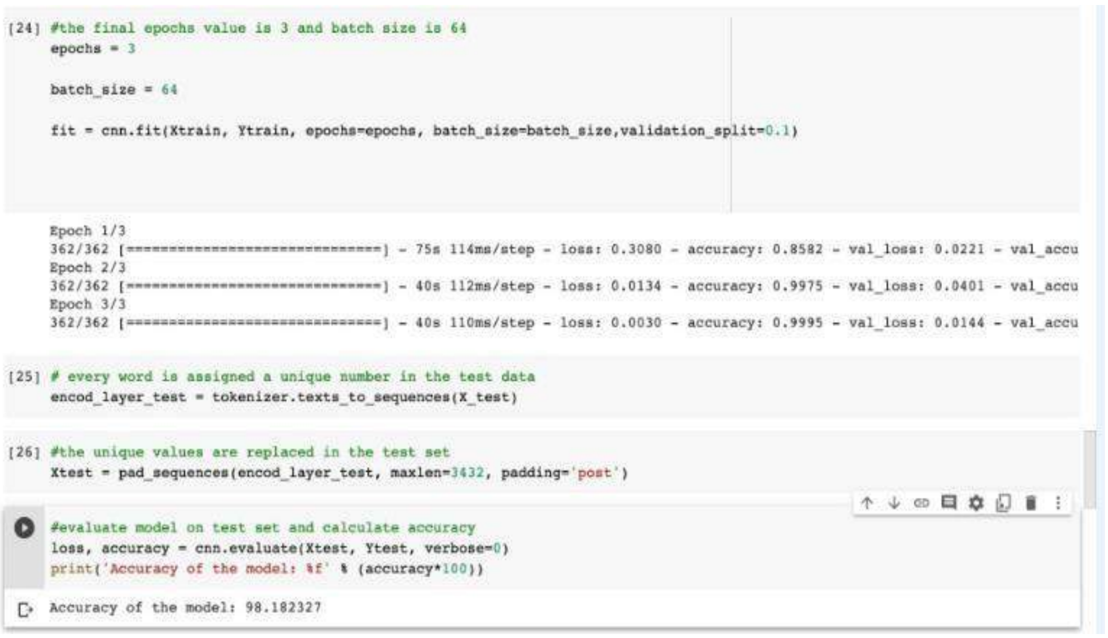
5.5 OVER SAMPLING DATA



```
Accuracy:
 0.9790268456375839
Accuracy report:
              precision    recall  f1-score   support

           0       0.99      0.99      0.99      3387
           1       0.83      0.88      0.86       189

    accuracy                           0.98      3576
   macro avg       0.91      0.94      0.93      3576
weighted avg       0.99      0.98      0.98      3576

Confusion Matrix:
 [[3334   53]
 [  22  167]]
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:940: ConvergenceWarning: lbfgs failed to converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
```

Screenshot 5.5: Over sampling data using Logistic Regression

5.6 FINAL ACCURACY



```
[24] #the final epochs value is 3 and batch size is 64
     epochs = 3

     batch_size = 64

     fit = cnn.fit(Xtrain, Ytrain, epochs=epochs, batch_size=batch_size,validation_split=0.1)

     Epoch 1/3
     362/362 [==============================] - 75s 114ms/step - loss: 0.3080 - accuracy: 0.8582 - val_loss: 0.0221 - val_accu
     Epoch 2/3
     362/362 [==============================] - 40s 112ms/step - loss: 0.0134 - accuracy: 0.9975 - val_loss: 0.0401 - val_accu
     Epoch 3/3
     362/362 [==============================] - 40s 110ms/step - loss: 0.0030 - accuracy: 0.9995 - val_loss: 0.0144 - val_accu

[25] # every word is assigned a unique number in the test data
     encod_layer_test = tokenizer.texts_to_sequences(X_test)

[26] #the unique values are replaced in the test set
     Xtest = pad_sequences(encod_layer_test, maxlen=3432, padding='post')

     #evaluate model on test set and calculate accuracy
     loss, accuracy = cnn.evaluate(Xtest, Ytest, verbose=0)
     print('Accuracy of the model: %f' % (accuracy*100))

     Accuracy of the model: 98.182327
```

Screenshot 5.6: Final Accuracy using CNN model

6. TESTING

6. TESTING

6.1 INTRODUCTION TO TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discoverevery conceivable fault or weakness in a work product. It provides a way to check the functionality of components, subassemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 UNIT TESTING

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.2.2 INTEGRATION TESTING

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent.

Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.2.3 FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input	: identified classes of valid input must be accepted.
Invalid Input	: identified classes of invalid input must be rejected.
Functions	: identified functions must be exercised.
Output	: identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes.

6.3 TEST CASES

Test case ID	Test case name	Purpose	Input	Output
1	Naïve Bayes with all Features	To check the accuracy	Datasets with default parameters	Accuracy
2	Logistic Regression with all features	To check the accuracy	Datasets with default parameters	Accuracy
3	Over sampling data using naïve bayes	To check accuracy	Datasets	Accuracy
4	Over sampling data using logisctic Regression	To check accuracy	Datasets	Accuracy
5	Final Accuracy	To check accuracy using CNN model	Over sampled data	Accuracy

7. CONCLUSION

7. CONCLUSION & FUTURE SCOPE

7.1 PROJECT CONCLUSION

From this project, we would like to conclude that: Not only job requirements and job description are the important features but all the text features such as title, benefits and other numeric data is also important in predicting the fraudulent job postings. Logistic Regression performs better than Naïve Bayes even when we consider all features or only two text features (description and requirements). Class weighted approach was done to see if giving priority to low bias class and decreasing priority of high bias class will improve lo bias data’s classification.

We concluded from this analysis that doing so will reduce the weights of high bias data and this will lead to wrong classification of even high bias data.To solve this issue oversampling was done by duplicating the low bias class data and the results proved that oversampling improved the classification of low bias class much better than before. The precision has also increased after oversampling. The ROC curve tells us that the Logistic Regression model is performing very well after oversampling the data.

We would also like to conclude that doing word embedding gives better results than doing TF-IDF vectorization as word embedding understands the context and meaning of the word rather than just using word frequency to find relevant words.Using word embedding on the CNN model has given a very good accuracy and is the best model we have used for classifying fraudulent job posting.

7.2 FUTURE SCOPE

We can focus on the class weight approach so that we can get an optimal accuracy even without performing oversampling on the training data and predict fraudulent data accurately. We should also focus on alteration of weights so that we need not do oversampling. We can also find patterns in n-gram weights when there is imbalance in the data to help the model give more priority to the low bias class.

8. BIBILOGRAPHY

8. BIBILOGRAPHY

8.1 REFERENCES

1. Albert, C. (n.d.). Inside the Shady New World of Fake Resumes, Professional Interviewees, and Other Job-Seeker Scams. Retrieved from Inc:

2. University of the Aegean. (2014). EMSCAD logoEmployment Scam Aegean Dataset. Retrieved from University of the Aegean Laboratory of Information & Communication Systems Security

3. Monsters, D. (2018, 08 15). Text Preprocessing in Python: Steps, Tools, and Examples. Retrieved from Medium

4. Peixeiro, M. (2019, 04 03). Introduction to Convolutional Neural Networks (CNN) with TensorFlow. Retrieved from Towards Data Science

5. Singh, V. (n.d.). Fake Job Post Prediction: Countvec, GloVe, Bert. Retrieved from Kaggle

8.2 WEBSITES

1. http://emscad.samos.aegean.gr/
2. https://www.kaggle.com/vikassingh1996/
3. https://towardsdatascience.com/

International Journal of All Research Education and Scientific Methods (IJARESM), ISSN: 2455-6211
Volume 9, Issue 4, April -2021, Impact Factor: 7.429, Available online at: www.ijaresm.com

Fake or Real Job Posting Prediction

Jonnadula Narasimha Rao[1], Likhitha Koganti[2], Vineetha Gangavarapu[3], C. Prathik Reddy[4]

[1]Associate Professor, CMR Technical Campus
[2,3,4]CMR Technical Campus

****

ABSTRACT

Every Job seeker who is looking for a job sometimes encounters a fake job posting which indeed will waste his time, effort and money.So we have implemented an application which will help in predicting whether the job posting is real or fake.It takes a lot of time for these job postings to be detected and removed from the websites.

Keywords: *CNN Model, Data Sampling, Data Pre-processing Etc*

INTRODUCTION

It is important to know if one is applying for a genuine job posting or a fake job posting as he can apply for hundreds of companies mentioned in the job boards and hardly receive any interview calls. It not only wastes his time but also his energy and hope of getting a job. Companies often fake hiring ads in order to get market intel or use their job posting credits. It is necessary for us to know which companies are fake and genuine so that the genuine job posting companies don't waste their time and resources and the ones seeking a job can achieve more by matching better.

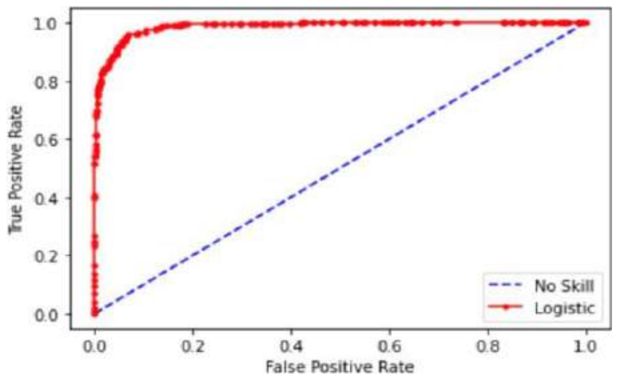
In this project, we aim to classify a job posting as a fake or real one by taking the job posting's title, profile of the company and other significant texts such as job description, requirements, benefits and many more. We chose Logistic Regression and Naïve Bayes as our baseline model. As our hypothesis is that only job description and requirements are the significant features for predicting fraudulent or non-fraudulent job postings, we implemented our models by considering only some features of the dataset as we wanted to check if our hypothesis was true. We realized that these features are not sufficient and decided to use the remaining features.

Overview

The first step is to process the text data and numeric data. The next step is to generate a TF-IDF matrix for only job description and requirements features and run our baseline models on it to analyze the results. For the next step, we will repeat the second step but instead of considering only two features we generate a TF-IDF matrix for all the text data columns. We will horizontally stack the one hot encoded features with the TF-IDF matrix. Now we implement our models on this new all features table and analyze the results. The next step is to do n-gram analysis and find a solution to get more accuracy. We will use oversampling to achieve more accuracy in this case. We implement our baseline models on this oversampled data and analyze the results. If we get optimal solution we move on to implement the CNN model and start this by doing word embedding. We will attach word embedding to our CNN layers and hyper tune the parameters. After running the models, we will analyze its results and then evaluate all our models.

ANALYSIS

We can see from the ROC curve generated for Logistic Regression on oversampled data, that AUC value is very close to 1 and is around 0.98 which means that the model is performing well in predicting the fraudulent classes.



CONCLUSION

We concluded from this analysis that doing so will reduce the weights of high bias data and this will lead to wrong classification of even high bias data. To solve this issue oversampling was done by duplicating the low bias class data and the results proved that oversampling improved the classification of low bias class much better than before. The precision has also increased after oversampling. The ROC curve tells us that the Logistic Regression model is performing very well after oversampling the data. We would also like to conclude that doing word embedding gives better results than doing TF-IDF vectorization as word embedding understands the context and meaning of the word rather than just using word frequency to find relevant words.

REFERENCES

[1]. Albert, C. (n.d.). *Inside the Shady New World of Fake Resumes, Professional Interviewees, and Other Job-Seeker Scams*. Retrieved from Inc:
[2]. University of the Aegean. (2014). *EMSCAD logoEmployment Scam Aegean Dataset*. Retrieved from University of the Aegean Laboratory of Information & Communication Systems Security:
[3]. Monsters, D. (2018, 08 15). *Text Preprocessing in Python: Steps, Tools, and Examples*. Retrieved from Medium:
[4]. Peixeiro, M. (2019, 04 03). *Introduction to Convolutional Neural Networks (CNN) with TensorFlow*. Retrieved from Towards Data Science:
[5]. Singh, V. (n.d.). *Fake Job Post Prediction: Countvec, GloVe, Bert*. Retrieved from Kaggle:

IJARESM

ISSN: 2455-6211, New Delhi, India
International Journal of All Research Education & Scientific Methods
An ISO & UGC Certified Peer-Reviewed Multi-disciplinary Journal

Certificate of Publication

Likhitha Koganti

CMR Technical Campus

TITLE OF PAPER

Fake or Real Job Posting Prediction

has been published in

IJARESM, Impact Factor: 7.429, Volume 9 Issue 4, April - 2021

Paper Id: IJARESM/Apr21

Date: 27-04-2021

CERTIFIED ISO 9001:2008 JOURNAL

Website: www.ijaresm.com
Email: editor.ijaresm@gmail.com

IJARESM Auth. Signatory INDIA

[illegible]

Authorized Signatory

IJARESM

ISSN: 2455-6211, New Delhi, India

International Journal of All Research Education & Scientific Methods

An ISO & UGC Certified Peer-Reviewed Multi-disciplinary Journal

Certificate of Publication

Vineetha Gangavarapu

CMR Technical Campus

TITLE OF PAPER

Fake or Real Job Posting Prediction

has been published in

IJARESM, Impact Factor: 7.429, Volume 9 Issue 4, April - 2021

Paper Id: IJARESM/Apr21

Date: 27-04-2021

Website: www.ijaresm.com
Email: editor.ijaresm@gmail.com

IJARESM Auth. Signatory INDIA

Authorized Signatory

IJARESM

ISSN: 2455-6211, New Delhi, India

International Journal of All Research Education & Scientific Methods

An ISO & UGC Certified Peer-Reviewed Multi-disciplinary Journal

Certificate of Publication

C. Prathik Reddy

CMR Technical Campus

TITLE OF PAPER

Fake or Real Job Posting Prediction

has been published in

IJARESM, Impact Factor: 7.429, Volume 9 Issue 4, April - 2021

Paper Id: IJARESM/Apr21

Date: 27-04-2021

CERTIFIED ISO 9001:2008 JOURNAL

Website: www.ijaresm.com
Email: editor.ijaresm@gmail.com

IJARESM Auth. Signatory INDIA

[illegible]

Authorized Signatory