A Project Report

on

Predicting Influencers in Social Network

Submitted in partial fulfillment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

By

D. Pravalika (16WH1A1214)

K. Sai Keerthi (16WH1A1238)

M. Sriya Sri (16WH1A1246)

Under the esteemed guidance of

Dr. K. Adi Narayana Reddy
Associate Professor

Department of Information Technology
BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN
(Accredited by NBA & NAAC '**A**' Grade, Approved by AICTE, and
Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

April 2020

DECLARATION

We hereby declare that the work presented in this project entitled **"PREDICTING INFLUENCERS IN SOCIAL NETWORK"** submitted towards completion of the major project in IV year of B.Tech IT at "BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN", Hyderabad is an authentic record of our original work carried out under the esteem guidance of Dr. K. Adi Narayana Reddy, Associate Professor, IT department.

D. Pravalika
16WH1A1214

K. Sai Keerthi
16WH1A1238

M. Sriya Sri
16WH1A1246

BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN

(Accredited by NBA & NAAC 'A' Grade, Approved by AICTE, and Affiliated to JNTUH, Hyderabad)

Bachupally, Hyderabad – 500090

Department of Information Technology

CERTIFICATE

This is to certify that the Project report on "**Predicting Influencers in Social Network**" is a bonafide work carried out by **D.Pravalika (16WH1A1214), K. Sai Keerthi (16WH1A1238), M. Sriya Sri (16WH1A1246)** in the partial fulfillment for the award of B.Tech degree in **Information Technology, BVRIT HYDERABAD COLLEGE OF ENGINEERING FOR WOMEN, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University, Hyderabad under my guidance and supervision.

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

Internal Guide
Mr.K. Adi Narayana Reddy
Associate Professor
Department of IT

Head of the Department
Dr. Aruna Rao S L
Professor and HOD
Department of IT

External Examiner

ACKNOWLEDGEMENTS

We would like to express our sincere thanks to **Dr. K. V. N. Sunitha, Principal**, BVRIT Hyderabad, for providing the working facilities in the college.

Our sincere thanks and gratitude to **Dr**. **Aruna Rao S L, Head**, Dept. of IT, BVRIT Hyderabad for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Dr. K. Adi Narayana Reddy, Associate Professor, Department of IT,** BVRIT Hyderabad for his constant guidance, encouragement and moral support throughout the project.

Finally, We would also like to thank our Project Co-coordinator, all the faculty and staff of IT Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

D. Pravalika
16WH1A1214

K. Sai Keerthi
16WH1A1238

M. Sriya Sri
16WH1A1246

ABSTRACT

Social Network plays an increasingly significant role in our daily lives. Modern social networking platforms provide users with tools for creating and sharing textual content, pointers to other web content, photographs or videos. From the millions of users that these platforms have, one can also acknowledge that the activities of a selected number of users are more rapidly perceived than those of others, and that the content produced by them flows swiftly through the network. We call these users the influencers. Influencers generate trends and shape opinions in social networks, being crucial in areas such as marketing, advertising or opinion mining. In this work, Given two persons and their social network features, our job is to predict which one is more influential. In our project, we collect training samples from Kaggle based on human judgement. We use several different models to make predictions, such as Logistic Regression, SVM, Naive Bayes, Random Forest and Neural Network. We also use some auxiliary techniques like cross validation, feature selection and data pre-processing.

LIST OF FIGURES

TABLE OF CONTENTS

1. INTRODUCTION

Social Network plays an increasingly significant role in our daily lives. We share our experiences and opinions with our friends on Face-book, Twitter, Instagram and so on. When you're browsing your friends' posts, you may find that some of them are more influential than others and we call them influencers. According to the research in Sociology, influencers have a great impact on other people's lives because people always tend to follow the opinions of these influencers in social networks. Therefore, it is important for us to figure out which persons are influencers and how they shape public opinions. The rise of social media platforms with their focus on user-generated content and social networks, has brought the study of authority and influence over social networks. For companies and other public entities, identifying and engaging with influential users in social networks is critical, since any opinions they express can rapidly spread far and wide. In our project, the goal is to find influencers in a specific social network—Twitter. Since we have two persons in each data sample, we have 22 features in total. What's more, there is a binary label representing a human judgement about which of the two individuals is more influential in each training sample. Label 1 means A is more influential than B. Label 0 means B is more influential than A. There are 3000 training samples and 2500 testing samples in our dataset. Given a test sample, our job is to predict which individual in this test sample is more influential.

1.1 Objective

In this, a model is built using Logistic Regression, SVM (Support Vector Machine), Naive Baeys, Random Forest and MLP(Neural Network) Classifier. All these algorithms generate model from train data-set and new data will be applied on train model to predict it class. Random Forest algorithm is giving better prediction accuracy compare to all other algorithms. The binary label represents a human judgement about which one of the two individuals is more influential. A label '1' means A is more influential than B. 0 means B is more influential than A. The goal of the challenge is to train a machine learning model which, for pairs of individuals, predicts the human judgement on who is more influential with high accuracy.

1.2 Problem in Existing System

The existing model studies the problem of identifying the most influential nodes in a social network with a quite straightforward and trivial method, based only on the number of followers, i.e. if A has more followers than B, then A is more inflfluential than B. Using this method, the test accuracy is about 70.2%. This result shows that the number of followers is a strong indicator of the inflfluencers. However, 70.2% is not good enough for our prediction and this result will be mainly considered as a benchmark. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression is estimating the parameters of a logistic model by which the prediction accuracy of test data has been observed very low and it was concluded that the best accuracy we can achieve is about 76% using linear models, which is not much better than our benchmark 70.2%. This suggests that the testing examples might not be linearly separable. Moreover, there might be data corruptions since sometimes human judgement can be highly biased. In order to achieve better performance, we can either try more nonlinear models or use decision trees to improve the accuracy of prediction.

1.3 Solution

In order to overcome the problem in the existing model, We apply cross validation and feature selection to the linear models and compare the performance among different models i.e compare their test accuracies and we built our model by using different linear and non linear models to improve accuracy. Cross Validation is a technique in which we train our model using the subset of the data-set and then evaluate using the complementary subset of the data-set. It is shown that cross validation can improve the test accuracy of SVM, while having no strong effect on Logistic Regression, the performance of SVM is better than Logistic Regression using cross validation. Furthermore, the accuracy of nonlinear models such as Neural Network is better than linear models, although it's not as good as we expected. Moreover, there might be data corruptions since sometimes human judgement can be highly biased. In order to achieve better performance, we can try using decision trees like Random Forest to improve the prediction accuracy.

1.4 Features

After familiarizing ourselves with our data set, we decided to generate some features that we thought would be useful in predicting how a given post performs. These features include those extracted raw text data such as captions in a more useful way, as well as more general features that could potentially contribute to a post's success i.e, reaching to more number of followers. After generating these features, we hoped to utilize them and determine their importance once we began training a final model, by comparing the performance of a base regression model both with and without these features.

After scraping data from Twitter, we had lots of meta data, including account followers and following, business/category information. Although some of the meta data is related to the account and not just the post, we felt including it would help give contextual information to the post, as well as help us derive more features specific to the post itself.

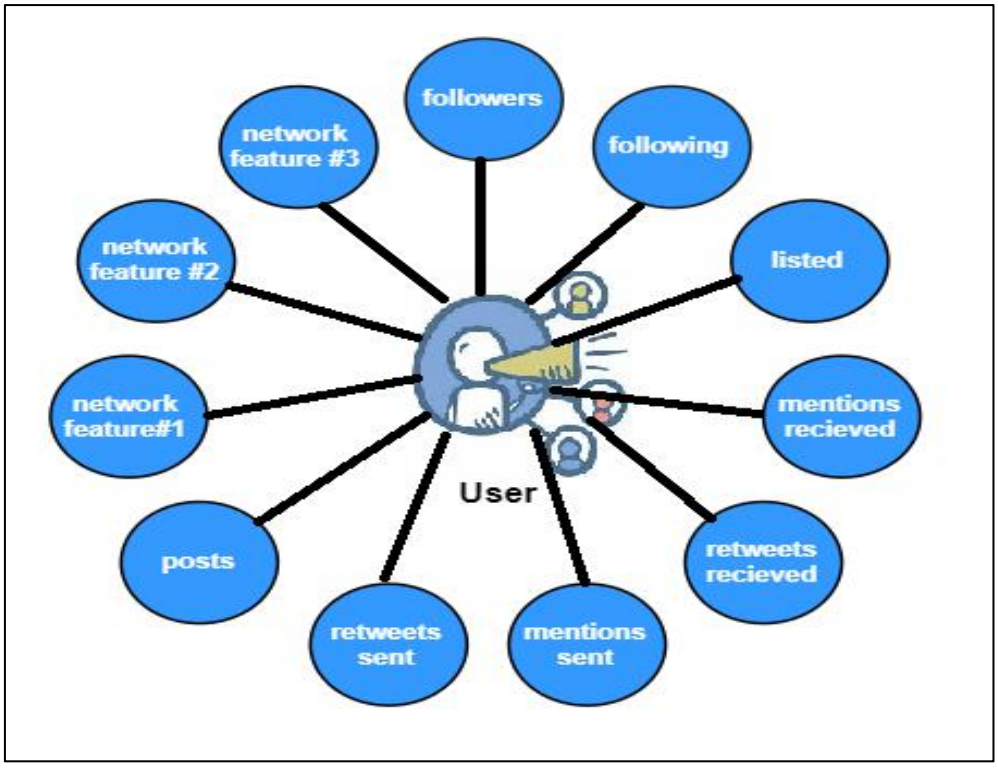


Figure 1: Features of User

2. LITERATURE SURVEY

Social media data analytics is the foundation of an advanced, effective digital marketing campaign. It removes the need for marketing based on instinct alone. With social media analytics, businesses can see exactly how customers perceive and interact with their brand, allowing them to make strategic changes that move the needle for the business. In order to gain insights from social data, first you'll need to establish a reliable method of social media data collection.

One of our targets is to achieve the highest level of engagement possible, and that means optimizing our social content according to our audience's preferences. When you collect social media data, you're able to see which platform is the most popular among your customer base. You can measure what type of social posts perform best and what time of day your customers are hanging out online. Using these findings, you can create and publish posts that have the maximum reach. Social proof marketing, when customers are influenced based on the actions of others, is proven to be most effective and trust worthy.

Word-of-Mouth is the action of informally sharing experiences and spreading information among consumers, whenever they are satisfied or dissatisfied with specific products . Customers are used to exchanging their opinions and experiences in and outside their personal social network. This marketing technique utilizes the consumers' networks in order to increase brand awareness, through self-replication and message diffusion, and limits the risk of customers' buying decision making process. Word-of-Mouth is considered the most valuable, influential and persuasive form of marketing in both business and academic communities.

Influencer Marketing is a hybrid of old and new marketing tools. It takes the idea of celebrity endorsement and places it into a modern-day content-driven marketing campaign. The main differentiator in the case of influencer marketing is that the results of the campaign are collaborations between brands and influencers. But influencer marketing doesn't just involve celebrities. Instead, it revolves around influencers, many of whom would never consider themselves famous in an offline setting.Influencers are the person who have the power to affect the purchasing decisions of others because of his or her authority, knowledge, position, or relationship with his or her audienceand a following in a distinct niche, with whom he or she actively engages. The size of the following depends on the size of his/her topic of the niche.

Figure 2: Benefits of Influencer Marketing

Engagement was considered a more important metric to measure the success of an influencer campaign than it with 90% of the respondents mentioning it. Working with influencers on a marketing campaign can drive 16X more engagement than paid or owned media. Consumers are continuing to see recommendations and online consumer reviews as the most credible sources of information, successful brand advertisers are "seeking better ways to connect with consumers and leverage their good-will in the form of consumer feedback and experiences."

Advantages :

1. Quickly Builds Trust

Influencers have built relationships, trust, and credibility with their followers. People respect their content and recommendations. By sharing an influencer's content, you'll soon gain their attention and they'll begin sharing yours, putting your message in front of an actively engaged audience and helps the brand to gain trust of his/her followers.

2. Improves Brand Awareness

Influencer marketing can greatly expand your reach and positioning online. Social users will begin to know more about your brand, who you are, and the solutions you offer. The key to maximizing influencer strategy is ensuring that you're providing valuable content that adds to their social media presence also, ensuring value on both sides.

3. Enriches Your Content Strategy

Sharing influencer content can helps fill in the gaps of your own content schedule. This works well in situations where one have run out of content ideas or simply need some quality content to publish on your social pages

4. Effectively Reaches Your Target Audience

Through relevant influencers, content is placed in front of social users that are already interested in brand niche. One don't have to spend additional funds on testing and finding your audience - the influencer has already fostered this audience on social media.

5. Provides Amazing Value to Your Audience

At the heart of inbound marketing is delivering content that solves problems, educates and inspires your intended audience. Influencer marketing embraces this concept, as influencers are already in tune with the needs of the people they serve.

6. Builds Winning Partnerships

Connecting and engaging with an influencer can be the start of a powerful relationship. When you're in it for the long-haul, you never know where these connections could end. Possible joint-ventures, live events, and other opportunities may be in the works.

2.1 Methodology

Our project, prediction of influencers in the social network comes under supervised machine learning model. Supervised learning uses classification algorithms and regression techniques to develop predictive models. The algorithms include linear regression, logistic regression, and neural networks, Support Vector Machine (SVM), random forest, naive Bayes.

Naive Bayes algorithm, we need to discretize our dataset first. For this purpose, we choose K-means algorithm to discretize each attribute into several clusters and each cluster corresponds to a class. Applying clustering algorithm like K-means can help us distinguish users from different levels. For instance, a film star may have a million followers while a normal user can only have hundreds of followers and in this case, their difference can't be ignored and we need to put them into different classes. In the original dataset, some attributes have very large ranges. Thus, we consider using logarithm function on the original dataset before applying K-means.

2.2 Information Gathering

Social data is information that's collected from social media platforms. It shows how users view, share and engage with your content. On Twitter, results include numbers of impressions, retweets, hashtag usage and engagement rates are included in the raw data. Once this information is mined, you can begin marking trends, measuring engagement and drawing insights that will help you accurately and efficiently keep up with these five marketing goals.

Twitter is a powerful social networking tool- a way of communicating through 140 character messages shared with a community of over 200 million users. The philosophy of the site is follower driven: anyone can follow anyone, celebrities and common people alike, the goal for many to simply collect followers and grow their own individual community. Users can follow those who interest them and choose not to follow those who do not interest them.

There are several ways to search Twitter for information, utilizing the #Discover tab and the Trending Topics through Twitter itself. Twitter.com has a search feature at search.twitter.com, where audiences can use hashtagged items or can simply search words that are tweeted. The simple search can be used much like a search engine such as Google- one simply inputs a key word or words and sees real time tweets using said words, as well as having

the option to scroll back and search tweets for up to two weeks. The social media site also offers advanced search options, which allows the researcher to search for only select words, while excluding others, with certain hashtags, from certain people, about certain people, from certain places, with a good, bad, or indifferent opinion, in any and all combinations. This way, tweets can be filtered very specifically, for a very specific purpose, for a very specific demographic. The filter is very helpful for a search for recent and ongoing events, like tracking breaking news and the public reactions to such events. This search will show the top results, but also all tweets mentioning the requested information. It has the capacity to display a lot of information and thoughts at one time.

In this data driven and instant gratification era, influencers are more valuable than cheesy celebrity endorsements. Major and lesser known brands are already tapping into the fan bases of online celebrities through paid endorsements. However, the difficulty for a brand lies in selecting the appropriate influencer from an ocean of choices. How do you know which influencer is a great brand ambassador, the one who can drive sales and create positive brand awareness? learn about how predictive analytics may help you. Predictive analytics looks at past patterns and tries to foresee future behavior, and this may prove to be an excellent tool for brands searching for the right influencer. We have studied many existing models of influencersnprediction and read different published research papers, collected information from models and have come up with the best features observed in different models to implement our prediction with higher accuracy.

3. REQUIREMENTS SPECIFICATION

3.1.1 Software Requirements

Programming Language : Python 3.6

Graphical User Interface : Tkinter

Dataset : Kaggle Dataset

Packages : Numpy, Pandas, Matplotlib, Scikit-learn

Tool : Spyder (IDE)

3.1.2 Hardware Requirements

Operating System: Windows 10

Processor : Intel Core i3-2348M

CPU Speed : 2.30 GHz

Memory : Minimum 2 GB (RAM)

3.2 Components

● **Keras** - It is an open-source neural network library that provides support for Python. It is popular for its modularity, speed, and ease of use.

● **Tkinter -** It is the Python interface to the Tk GUI toolkit shipped with Python.

Technologies Description

Spyder

It is an open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. Spyder integrates with a number of prominent packages in the Python stack including NumPy, SciPy, Matplotlib, pandas, IPython, SymPy and Cython, as well as other open source software. It is released under the MIT license.

Initially created and developed by Pierre Raybaut in 2009, since 2012 Spyder has been maintained and continuously improved by a team of scientific Python developers and the community.

Spyder is extensible with first- and third-party plugins, includes support for interactive tools for data inspection and embeds Python-specific code quality assurance and introspection instruments, such as Pyflakes, Pylint and Rope. It is available cross-platform through Anaconda, on Windows, on macOS through MacPorts, and on major Linux distributions.

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts,

scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis
- Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

4. DESIGN

4.1 Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

Architecture Diagram :-

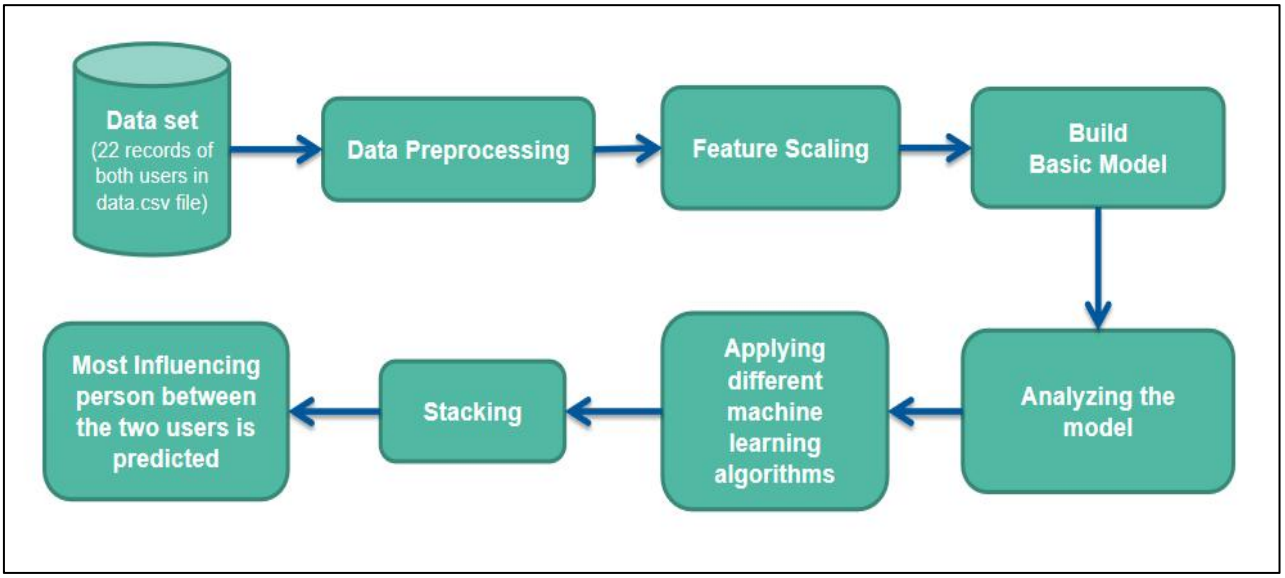


Figure 3: Architecture Diagram

4.2 UML Diagrams

4.2.1 Use Case Diagram

To model a system, the most important aspect is to capture the dynamic behavior. Dynamic behavior means the behavior of the system when it is running/operating.These internal and external agents are known as actors. Use case diagrams consist of actors, use cases and their relationships. The diagram is used to model the system/subsystem of an application. A single use case diagram captures a particular functionality of a system.

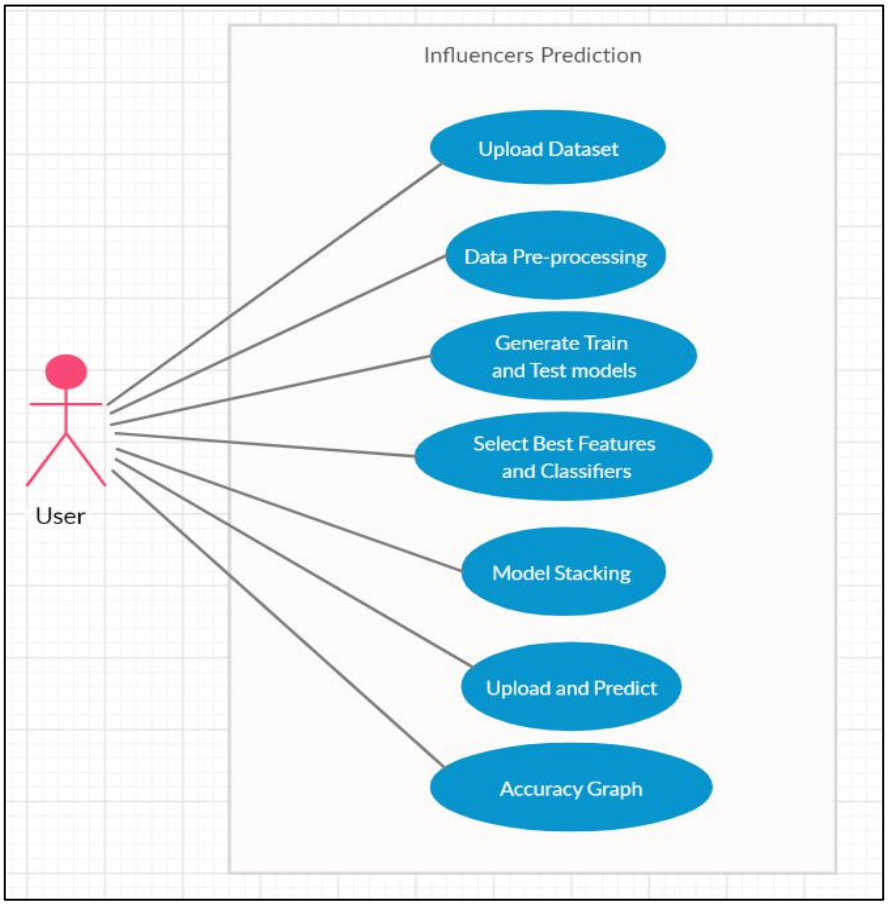


Figure 4: Use Case Diagram

4.2.2 Sequence Diagram

Sequence Diagrams represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment. A use case can include possible variations of its basic behavior, including exceptional behavior and error handling.

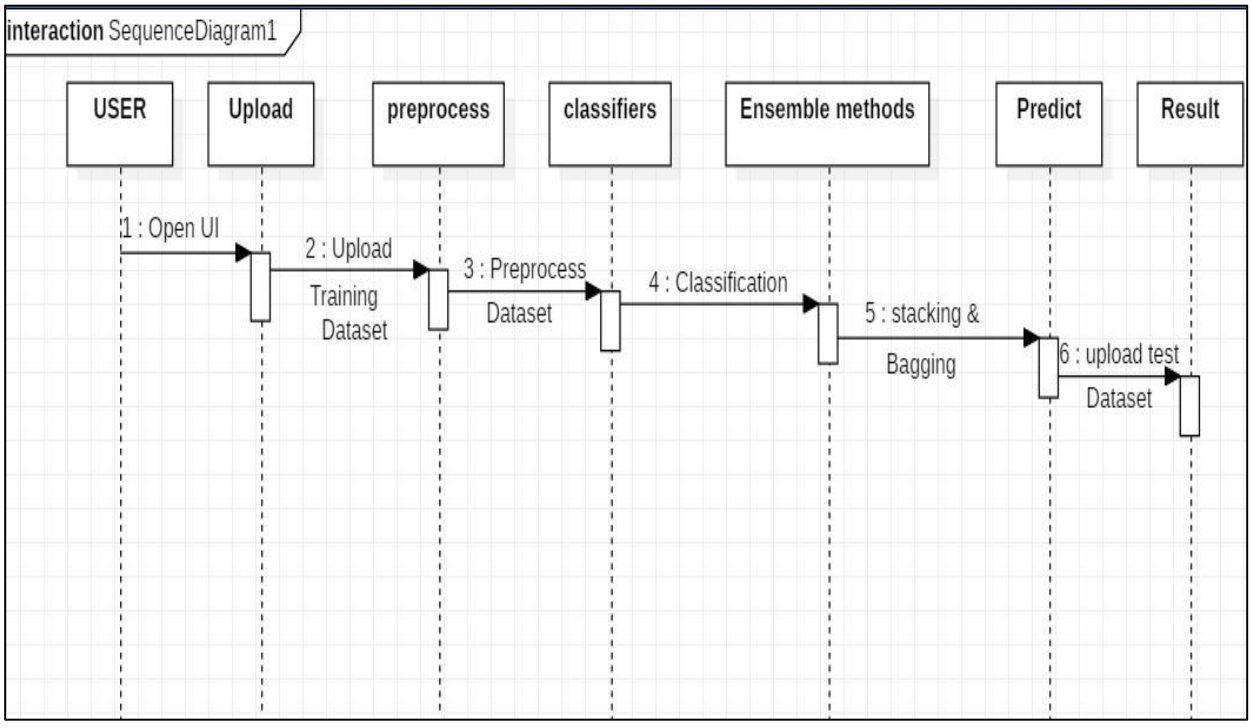


Figure 5: Sequence Diagram

4.2.3 Activity Diagram

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

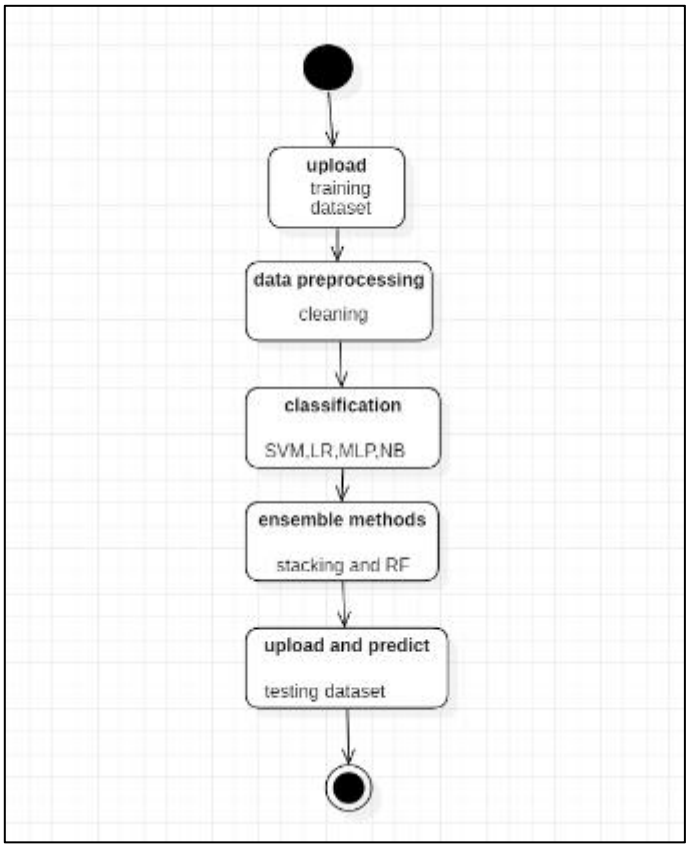


Figure 6: Activity Diagram

4.2.4 Component Diagram

Component diagrams are used to visualize the physical components in a system. These components are libraries, packages, files, etc. Component diagrams can also be described as a static implementation view of a system.

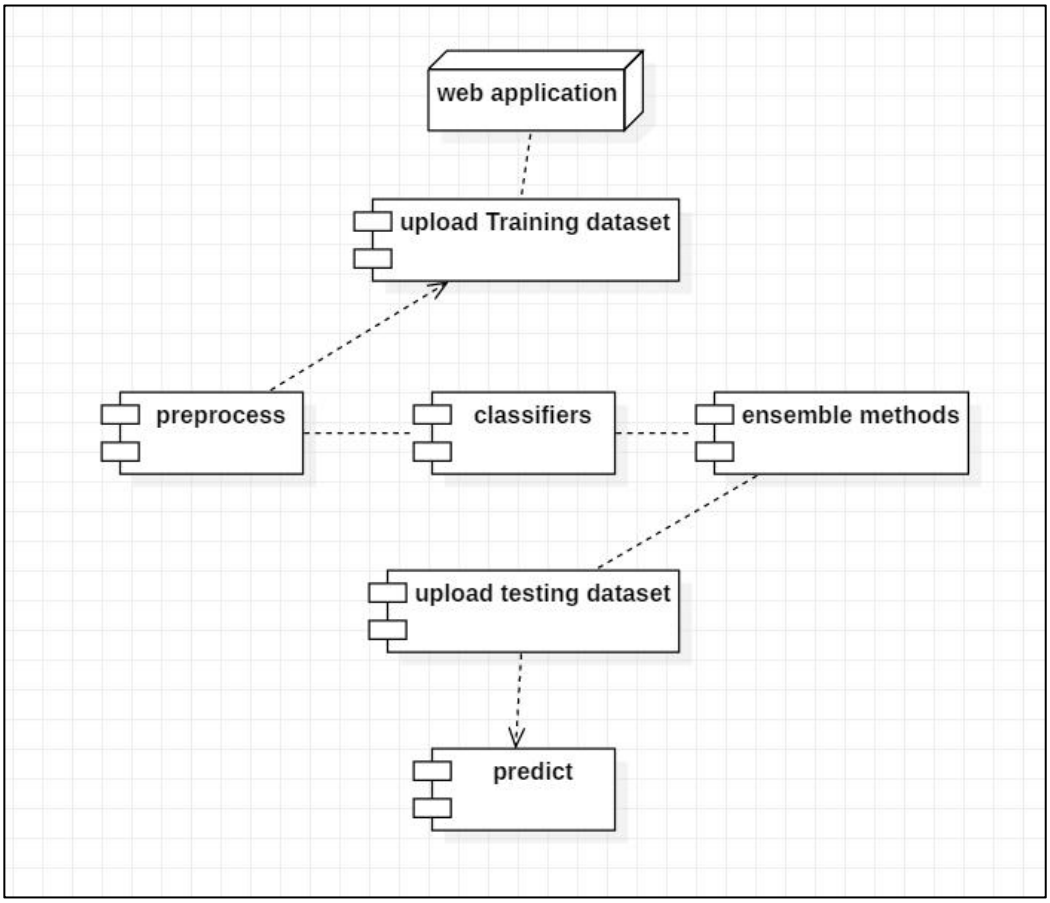


Figure 7: Component Diagram

4.2.5 Class Diagram

The class diagram is the main building block of object-oriented modeling. It is used for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

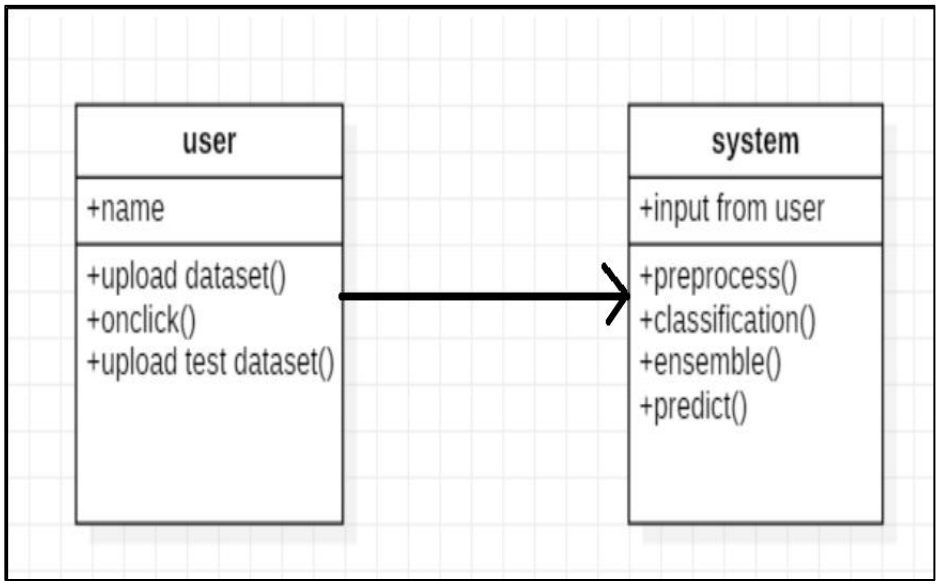


Figure 8: Class Diagram

5. MODULES

5.1 Upload Dataset and Data Pre-processing

► Upload the data from the Dataset Folder.

- The dataset taken from kaggle repository is downloaded.
- The dataset is uploaded to build the model.

► Preprocess the uploaded data.

- Creating new features i.e, handicrafted from the existing data.
- Check for duplicates and removing if exist.
- Scaling the data.
- Spliting the data into train and test samples.

5.2 Build Model

- Build the model for SVM, Logistic Regression, Naive Bayes, Neural Network using MLP and Random Forest for all the significant features.
- Analysing the accuracies of every model with help of Area Under Curve.

5.3 Model Stacking

- Stacking is an ensemble learning technique that uses predictions from multiple models to build a new model.
- This model is used for making predictions on the test set.
- We can build multiple different learners and you use them to build an intermediate prediction,one prediction for each learned model, then you add a new model which learns from the intermediate predictions for the same target.

● This final model is said to be stacked on the top of the others.

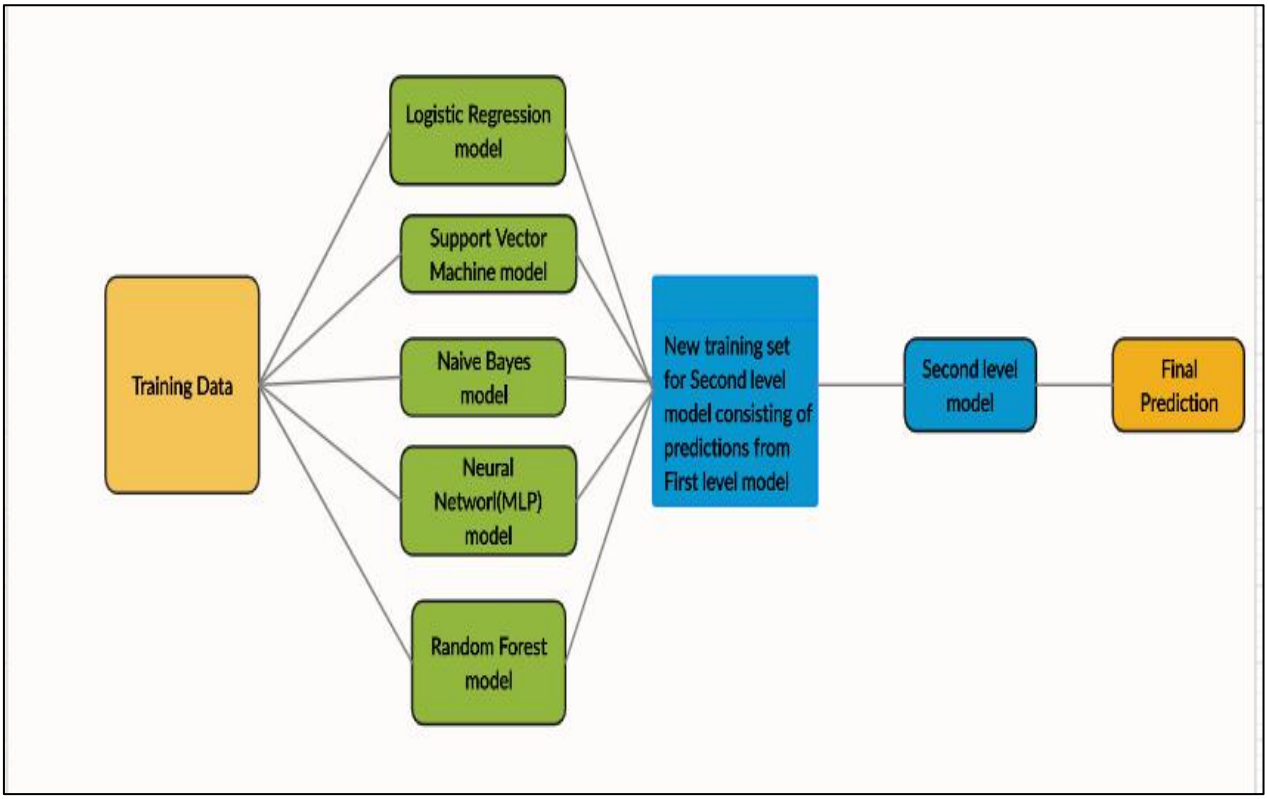


Figure 9: Stacking Model

5.4 Influencer Prediction

► Predict New Data

- A new model is built by stacking the logistic regression, svm, naive bayes, mlp and random forest.
- Now, we provide new data as input and predict the output for the new data.
- Based on prediction, we get the output of our model i.e, the most influencing person present in our dataset.

► **Graph Accuracy**

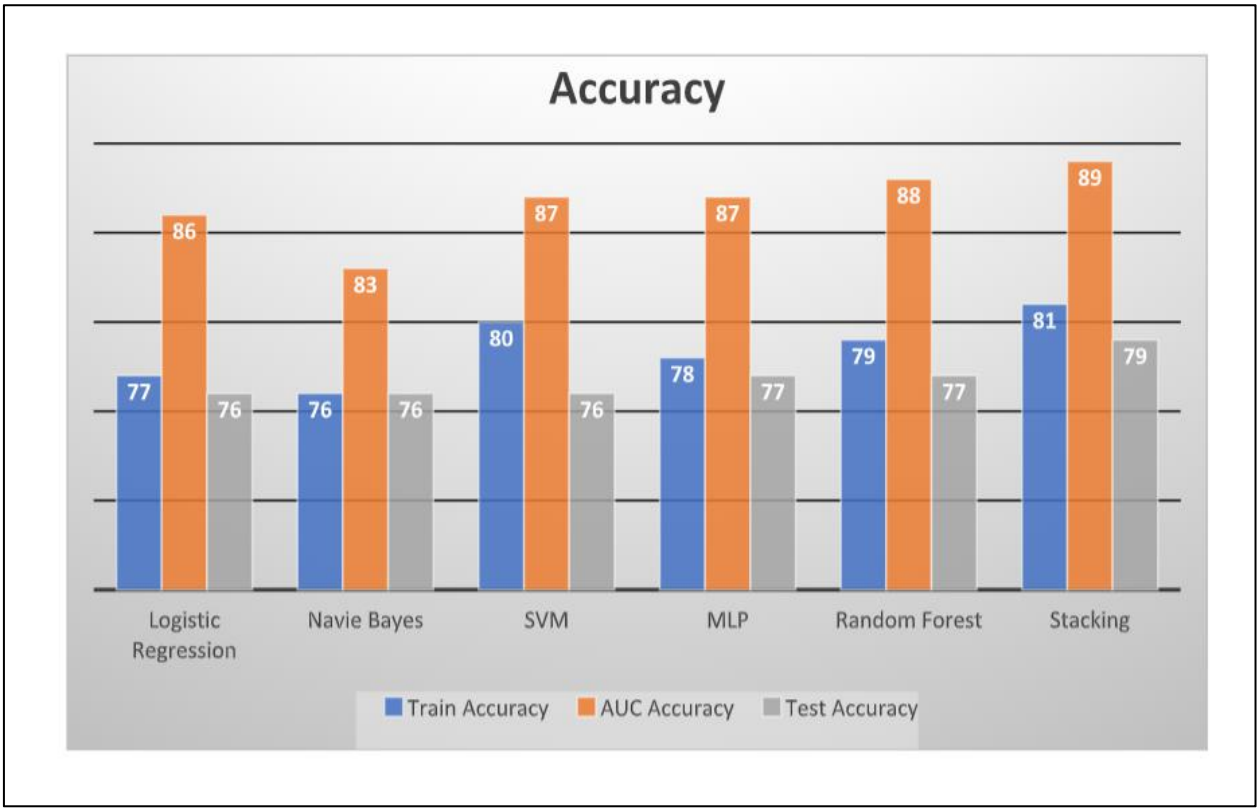


Figure 10: Accuracy Graph of Models

6. IMPLEMENTATION

Source Code :

```
from tkinter import messagebox

from tkinter import *

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

from imutils import paths

from tkinter.filedialog import askopenfilename

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings(action='ignore')

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.neural_network import MLPClassifier
```

```
from sklearn.model_selection import KFold

from sklearn.model_selection import cross_val_score

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import MinMaxScaler

from sklearn.preprocessing import FunctionTransformer

from sklearn.preprocessing import Normalizer

from sklearn.feature_selection import SelectKBest, chi2

from sklearn.model_selection import GridSearchCV

from sklearn import metrics

main = tkinter.Tk()

main.title("Predict Influencers in the Social Network")

main.geometry("1300x1200")

global filename

global testfile

global X_train_log,X_test_log,y_test,y_train

global data

global cols

global model_SVM_pred

def upload():

    global filename

    text.delete('1.0', END)
```

```
    filename = askopenfilename(initialdir = "Dataset")
    pathlabel.config(text=filename)
    text.insert(END,"Dataset loaded\n\n")
def preprocess():
    #Loading data
    global filename
    global X_train_log,X_test_log,y_test,y_train
    global data
    text.delete('1.0', END)
    data = pd.read_csv(filename)
    data.head()
    #data.info()
    #Checking for duplicate entries
    data.duplicated().sum()
    #Droping duplicate entries
    data=data.drop_duplicates()
    data.info()
    #Seperating dependent and independent variables from data
    Y = np.asarray(data['Choice'])
    X = data.drop(['Choice'],axis=1)
    # Setting seed value for reproducibility
```

```
    #Creating a new binary feature based on follower count
    ##assigning 1 if a person has more than a million follower
    data['A_is_popular'] =  [1 if x >= 1000000 else 0 for x in data['A_follower_count']]
    data['B_is_popular'] =  [1 if x >= 1000000 else 0 for x in data['B_follower_count']]
    #Creating new feature by dividing a follower count from following count
    data['A_popularity_score'] = data['A_following_count'].divide(data['A_follower_count'])
    data['B_popularity_score'] = data['B_following_count'].divide(data['B_follower_count'])
    X = data.drop(['Choice'],axis=1
    # Splitting data into 90% training set and 10% test set
    X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.1, random_state=4
    text.insert(END,"No of Training Samples: "+str(X_train.shape[0])+"\n")
    text.insert(END,"No of Testing Samples: "+str(X_test.shape[0])+"\n")
    text.insert(END,"No of Features: "+str(X_test.shape[1])+"\n")
    X_train_log = feature_transform('log',X_train
    # Applying log transform to test data
    X_test_log = feature_transform('log',X_test)
def mlModels():
    text.delete('1.0', END)
    global X_train_log,X_test_log,y_test,y_train
    build(X_train_log,X_test_log,y_train,y_test
def mlModelsBestFeatures():
```

```
build(Xtrainlog,Xtestlog,y_train,y_test)
    # Function for feature scaling and transformation
    def feature_transform(fn_name, data):
    #Create scaling or transformation object based on user input
    if fn_name == 'standard':
        tran_fn = StandardScaler()
    elif fn_name =='minmax':
        tran_fn = MinMaxScaler()
    elif fn_name =='log':
        tran_fn = FunctionTransformer(np.log1p, validate=True)
    elif fn_name =='normalize':
        tran_fn = Normalizer()
    #Applying transformation
    transfx_data = tran_fn.fit_transform(data.astype(float))
    #Converting back to dataframe
    transfx_data = pd.DataFrame(transfx_data, columns = data.columns)
            return transfx_data
def build(X_train_log,X_test_log,y_train,y_test):
    global model_SVM_pred
    model_LR = LogisticRegression(C = 0.055)
    model_LR_pred = model_LR.fit(X_train_log,y_train)
```

```
global X_train_log,X_test_log,y_test,y_train

global data

global cols

#Checking 6 Least important features using Univariate Selection

test = SelectKBest(chi2, k=20)

test.fit_transform(X_train_log, y_train)

index=sorted(range(len(test.scores_)), key=lambda k: test.scores_[k])[0:6]

print(index)

print(X_train_log.columns[index])

#Removing least important features

cols = ['A_following_count','B_following_count', 'B_network_feature_2', 'A_retweets_sent',
      'A_network_feature_2', 'A_is_popular']

data = data.drop(cols,axis=1)

Y = np.asarray(data['Choice'])

X = data.drop(['Choice'],axis=1)

# Spliting data into 90% training set and 10% test set

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.1, random_state=4

Xtrainlog = feature_transform('log',X_train)

# Applying log transform to test data

Xtestlog = feature_transform('log',X_test)

text.insert(END,"Total No of Features: "+str(Xtrainlog.shape[1])+"\n")
```

```
lrpred = model_LR_pred.predict_proba(X_train_log)

text.insert(END,'LR Accuracy on whole training data:
        '+str(model_LR_pred.score(X_train_log,y_train))+"\n")

fpr, tpr, _ = metrics.roc_curve(y_train, lrpred[:,1:2], pos_label=1)

auc = metrics.auc(fpr,tpr)

text.insert(END,'LR AUC: '+str(auc)+"\n")

lrpred = model_LR_pred.predict_proba(X_test_log)

text.insert(END,'LR Accuracy on whole testing data:
    '+str(model_LR_pred.score(X_test_log,y_test))+"\n")

model_NB = GaussianNB()

model_NB_pred = model_NB.fit(X_train_log,y_train)

nbpred = model_NB_pred.predict_proba(X_train_log)

text.insert(END,'NB Accuracy on whole training data:
      '+str(model_NB_pred.score(X_train_log,y_train))+"\n")

fpr, tpr, _ = metrics.roc_curve(y_train, nbpred[:,1:2], pos_label=1)

auc = metrics.auc(fpr,tpr)

text.insert(END,'NB AUC: '+str(auc)+"\n")

nbpred = model_NB_pred.predict_proba(X_test_log)

  text.insert(END,'NB Accuracy on whole testing data :
'+str(model_NB_pred.score(X_test_log,y_test))+"\n")

model_SVM = SVC(probability=True, gamma='auto')

model_SVM_pred = model_SVM.fit(X_train_log,y_train)
```

```
svmpred = model_SVM_pred.predict_proba(X_train_log)

text.insert(END,'SVM Accuracy on whole training data:
  '+str(model_SVM_pred.score(X_train_log,y_train))+"\n")

fpr, tpr, _ = metrics.roc_curve(y_train, svmpred[:,1:2], pos_label=1)

auc = metrics.auc(fpr,tpr)

text.insert(END,'SVM AUC: '+str(auc)+"\n")

svmpred = model_SVM_pred.predict_proba(X_test_log)

text.insert(END,'SVM Accuracy on whole testing data:
  '+str(model_SVM_pred.score(X_test_log,y_test))+"\n")

model_MLP = MLPClassifier(activation= 'tanh', learning_rate = 'adaptive', solver= 'sgd')

model_MLP_pred = model_MLP.fit(X_train_log,y_train)

mlppred = model_MLP_pred.predict_proba(X_train_log)

text.insert(END,' MLP Accuracy on whole training data:
    '+str(model_MLP_pred.score(X_train_log,y_train))+"\n")

fpr, tpr, _ = metrics.roc_curve(y_train, mlppred[:,1:2], pos_label=1)

auc = metrics.auc(fpr,tpr)

text.insert(END,'MLP AUC: '+str(auc)+"\n")

lrpred = model_MLP_pred.predict_proba(X_test_log)

text.insert(END,'MLP Accuracy on whole testing data:
    '+str(model_MLP_pred.score(X_test_log,y_test))+"\n")

model_RF = RandomForestClassifier(n_estimators=1000,min_samples_leaf=1,max_depth=5)

model_RF_pred = model_RF.fit(X_train_log,y_train)
```

```
    rfpred = model_RF_pred.predict_proba(X_train_log)

    text.insert(END,'RF Accuracy on whole training data:
       '+str(model_RF_pred.score(X_train_log,y_train))+"\n")

    frp,trp, _ = metrics.roc_curve(y_train,rfpred[:,1:2],pos_label=1)

    auc = metrics.auc(frp,trp)

    text.insert(END,'RF AUC: '+str(auc)+"\n")

    rfpred = model_RF_pred.predict_proba(X_test_log)

    text.insert(END,'RF Accuracy on whole testing data:
       '+str(model_RF_pred.score(X_test_log,y_test))+"\n")

def uploadPred():

    # Loading given test data

    global testfile

    global cols

    global model_SVM_pred

    text.delete('1.0', END)

    testfile = askopenfilename(initialdir = "Dataset")

    pathlabel.config(text=testfile)

    X_Test = pd.read_csv(testfile)

    #Adding hand crafted features to given test data

    X_Test['A_is_popular'] =  [1 if x >= 1000000 else 0 for x in X_Test['A_follower_count']]

    X_Test['B_is_popular'] =  [1 if x >= 1000000 else 0 for x in X_Test['B_follower_count']]
```

```
    #Adding hand crafted features to given test data

    X_Test['A_popularity_score'] =
    X_Test['A_following_count'].divide(X_Test['A_follower_count'])

    X_Test['B_popularity_score'] =
    X_Test['B_following_count'].divide(X_Test['B_follower_count'])

    #Removing selected features from given test data

    X_Test=X_Test.drop(cols,axis=1)

    #Applying log transform to given test data

    X_Test_log = feature_transform('log',X_Test)

    pred_test = model_SVM_pred.predict_proba(X_Test_log)

    pred_test=pred_test[:,1:2]

    text.insert(END,"Predicted Values: \n"+str(pred_test)+"\n")
font = ('times', 16, 'bold')

title = Label(main, text='Predict Influencers in the Social Network')

title.config(bg='sky blue', fg='black')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 14, 'bold')

upload = Button(main, text="Upload Dataset", command=upload)

upload.place(x=700,y=100)

upload.config(font=font1)
```

```
pathlabel = Label(main)

pathlabel.config(bg='dark orchid', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=700,y=150)

df = Button(main, text="Data Pre-Processing", command=preprocess)

df.place(x=700,y=200)

df.config(font=font1)

split = Button(main, text="Classifier", command=mlModels)

split.place(x=700,y=250)

split.config(font=font1)

ml= Button(main, text="Select Best Features and Classifiers", command=mlModelsBestFeatures)

ml.place(x=700,y=300)

ml.config(font=font1)

graph= Button(main, text="Upload and Predict", command=uploadPred)

graph.place(x=700,y=350)

graph.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=30,width=80)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=100)
```

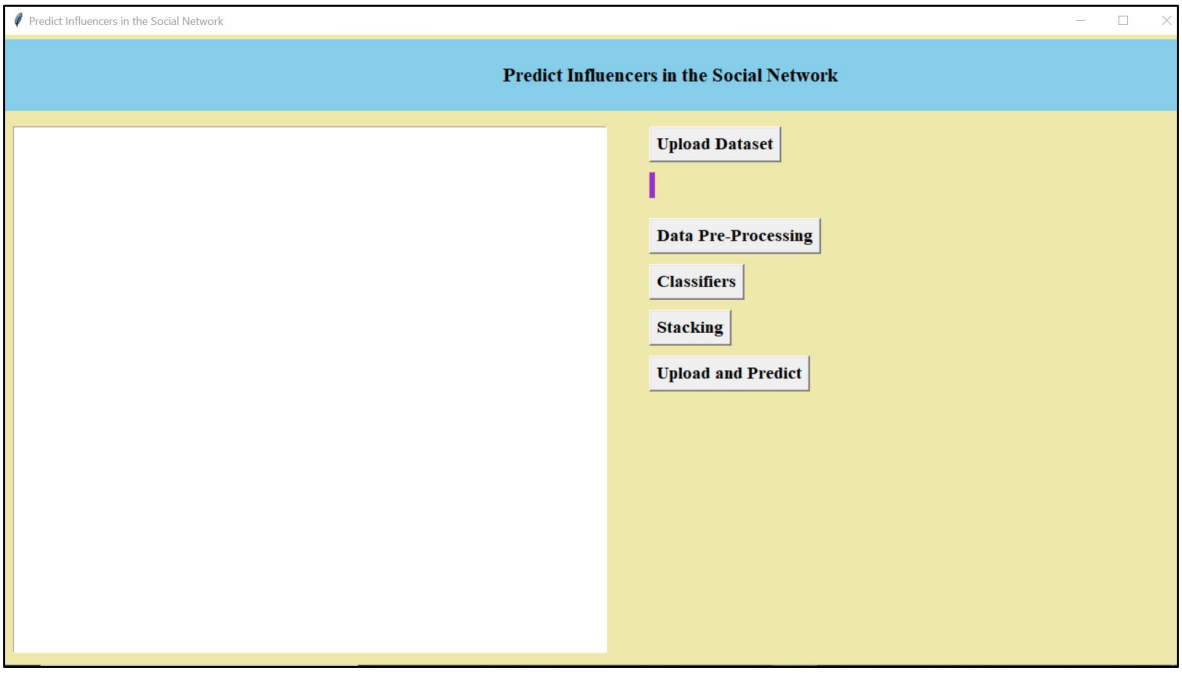
```
text.config(font=font1)

main.config(bg='pale goldenrod')

main.mainloop()

# Load Training Data

train_data = pd.read_csv("./train.csv")

train_data.head()

#Features

X = train_data.iloc[:,1:].as_matrix()

m,_ = X.shape

print("X: ", X.shape)

# Labels

Y = np.reshape(train_data["Choice"].as_matrix(), (-1, 1))

print("Y: ",Y.shape)

#Split Data into training and validation set

train_n = int(.80 * m)

train_x = X[:train_n, :]

train_y = Y[:train_n, :]

val_x = X[train_n:, :]

val_y = Y[train_n:, :]

print("Training set size: ", train_x.shape)

print("Validation set size: ", val_x.shape)
```

```
# Build the Model

def rec_model():

    model = Sequential()

    model.add(Dense(50, activation='relu', input_dim=22, kernel_initializer='uniform'))

    model.add(Dropout(0.5))

    model.add(Dense(10, kernel_initializer='uniform'))

    model.add(BatchNormalization())

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

    model.compile(loss='binary_crossentropy',

            optimizer='adam',

            metrics=['accuracy'])

    return model

roc_curve_area = callback.roc_curve_area()

model = rec_model()

model.summary()

#Start Training

model = rec_model()

model.fit(train_x, train_y,

        epochs=20,

        batch_size=64,

        validation_data=(val_x, val_y),
```

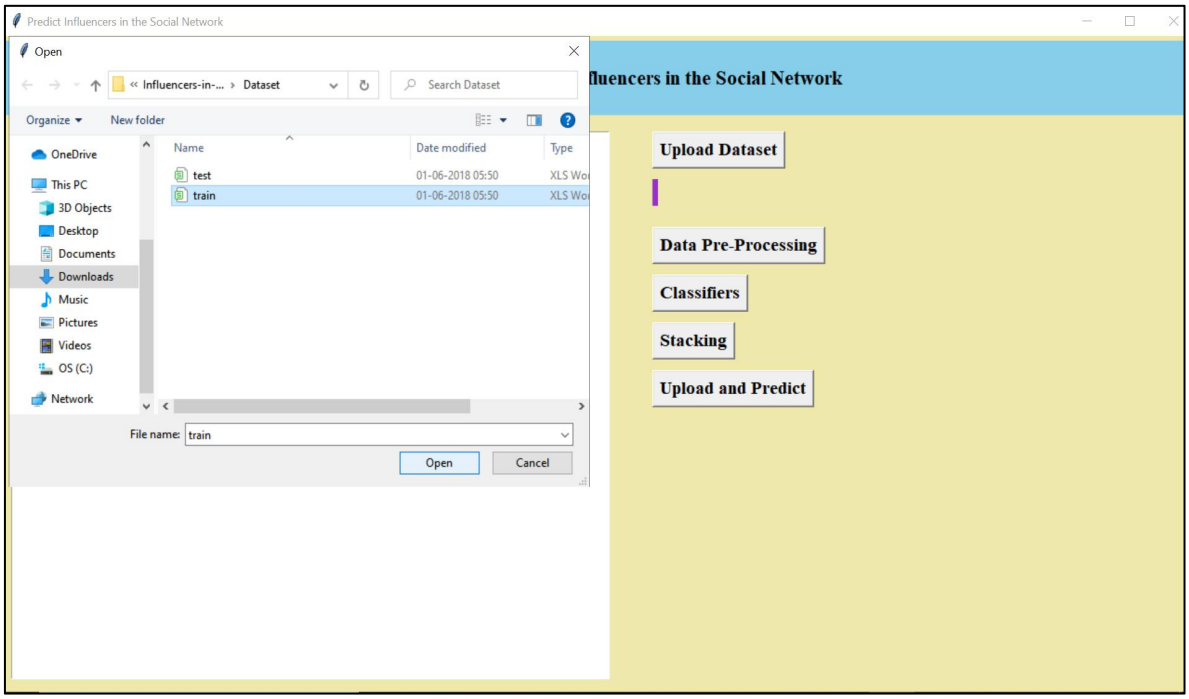
```
        callbacks=[roc_curve_area])

#Create Submission file

test_data = pd.read_csv("./test.csv")

test_data = test_data.as_matrix()

test_data.shape

y_pred = model.predict(test_data)

sample_submission = pd.read_csv('./sample_predictions.csv')

sample_submission.shape

sample_submission["Choice"] = y_pred

sample_submission.to_csv('./submission.csv', index=False)
```

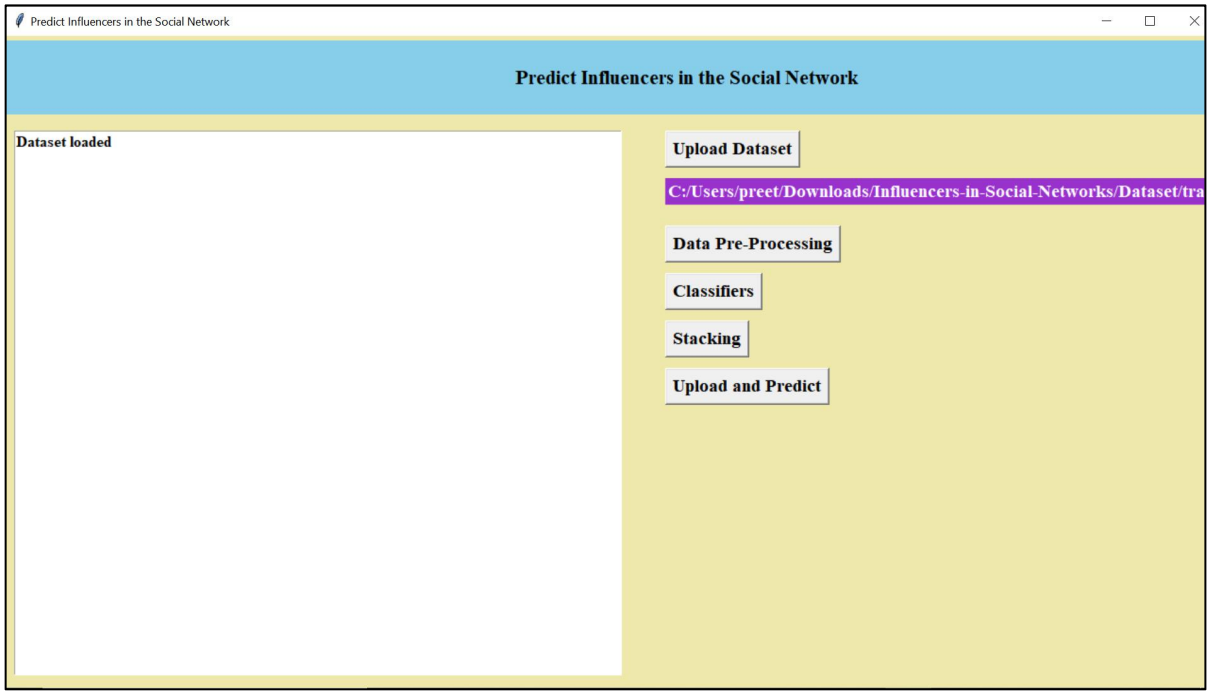
7. TESTING

7.1 Output Screenshots

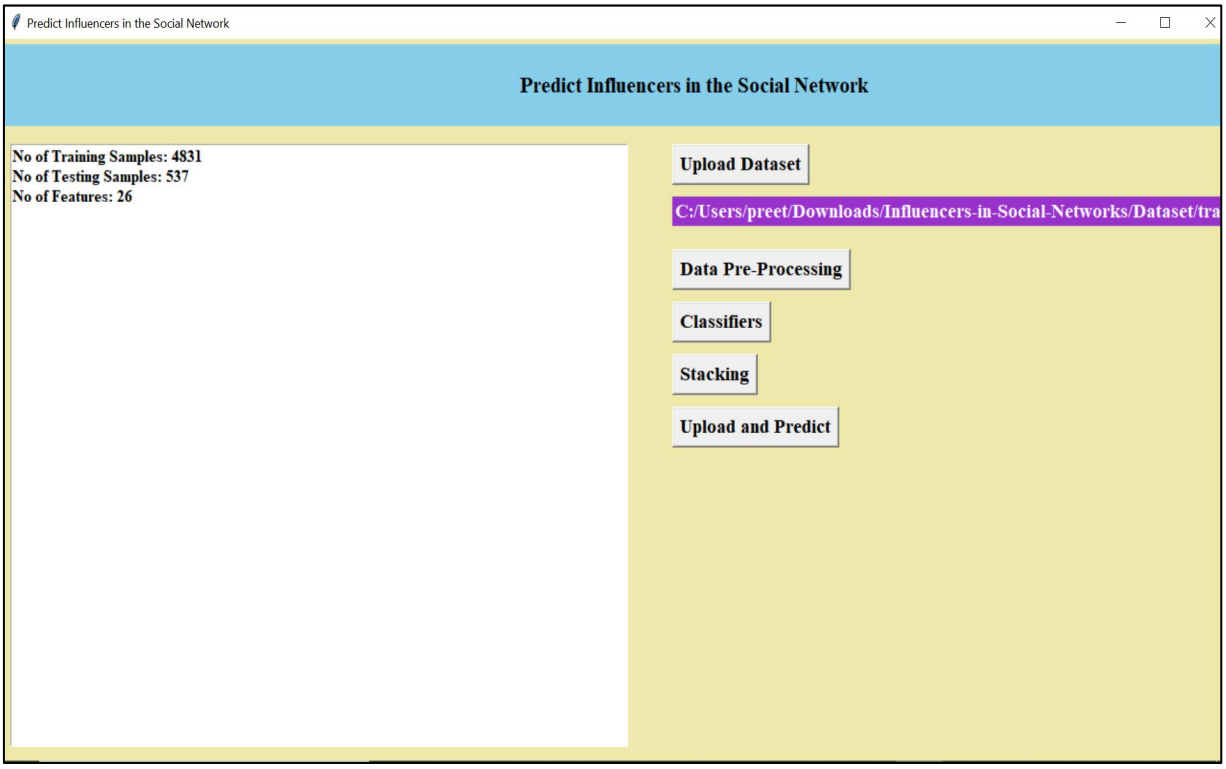


► Now click on 'Upload Dataset' to upload the data

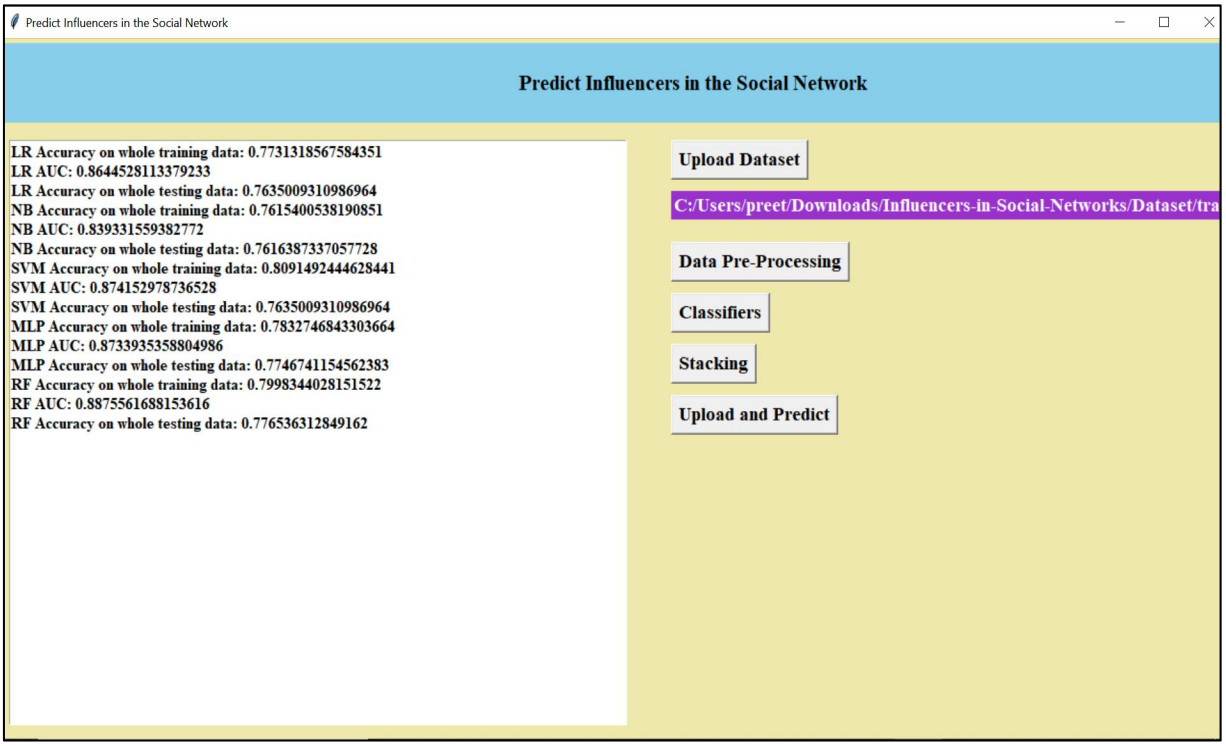




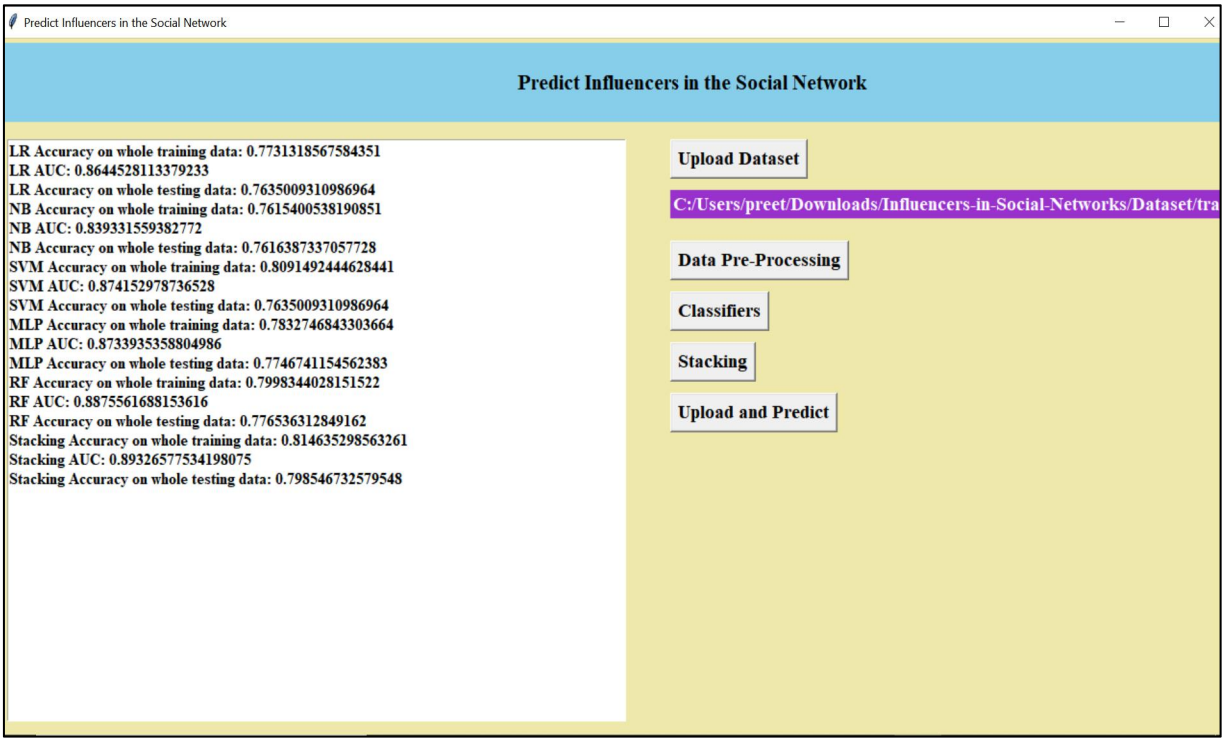
► Now click on ‘Pre-Precossing’ it preprocess the data.



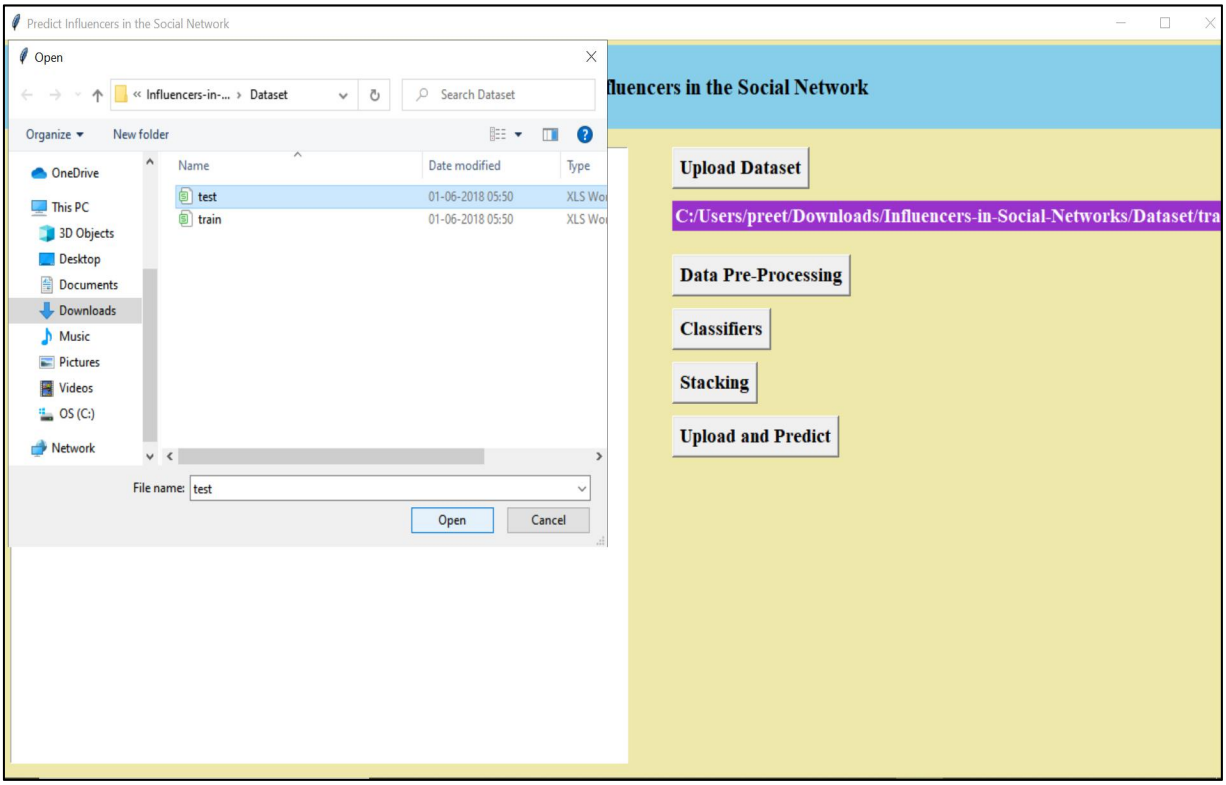
► Now click on ‘Classifiers’ to classify the models with all the significant features.

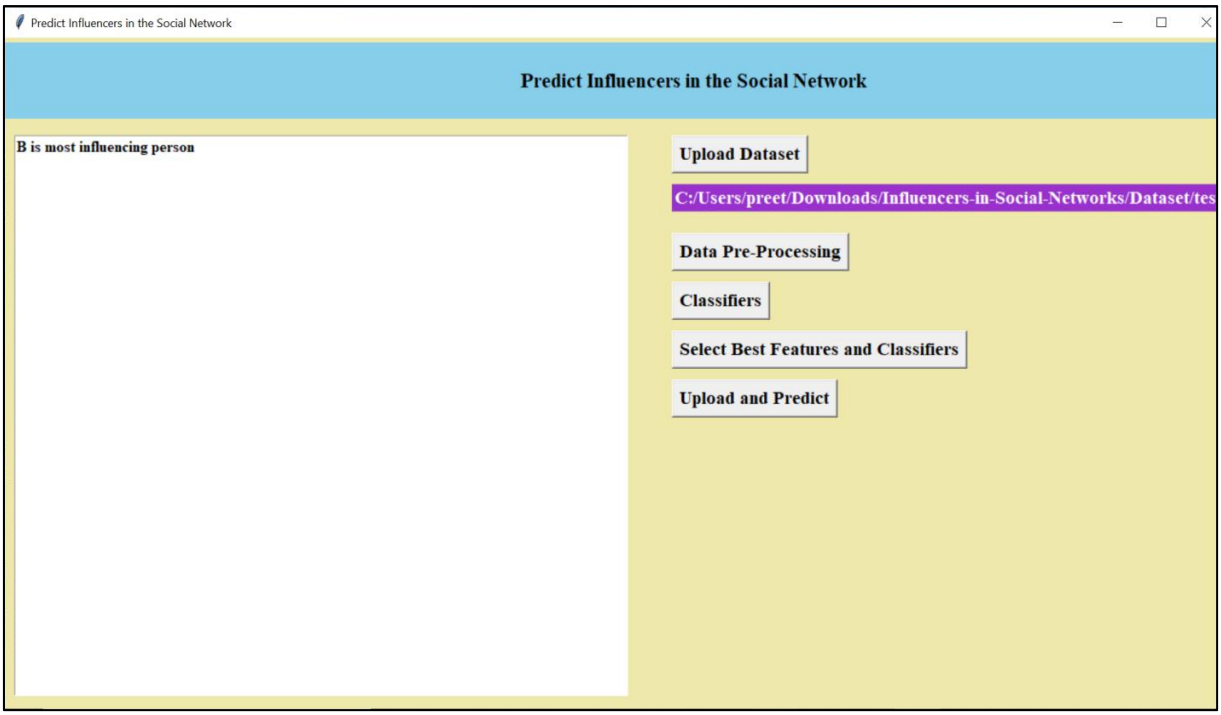


► Now click on ‘Stacking’ to apply model stacking to all the classifiers.



► Now click on ‘Upload and Predict’ to Predict on new data.





► Thus our model has predicted that “B is most influencing person” than A.

8. CONCLUSION AND FUTURE SCOPE

8.1 Conclusion

From the above results, we can conclude that the best accuracy we can achieve is about 88% using Random Forest among other implemented models. In this model, we have done Normalization, Feature selection and Cross Validation in order to improve the prediction accuracy when compared with the existing model. We also implemented Model Stacking so as a meta-learner generalizes better than a single model and it makes better predictions on unseen data, than just a single model. Wolpert in his paper, Stacked Generalization by David H. Wolpert (1992) argues that model stacking deduces the bias in a model on a particular dataset by which prediction of model becomes unbiased and accurate. This model has a best speculation, and it can be trusted that it is used to identify most influencing person among different influencers. Precise influencer predtion is essential for accurate classification for the purpose of influencer marketing of a company, which involves in improving the companies brand and gaining trust among the customers.

8.2 Future Scope

In future, we will try to consider the nature of the content being shared by influencers on the networking platform into account, which also plays a major role in shaping the opinion of the users and we also need to consider the individuals who have been influential in the past and who have creditability to have many followers in the future are more likely to be influential. The overall performance could also be improved and customized with the help of considering more features in the dataset. Moreover, there might be data corruptions since sometimes human judgement can be highly biased. In order to achieve better performance, we can either try more nonlinear models or use boosting techniques inorder to improve prediction accuracy.

9. REFERENCES

- [1] J. Furnkranz and E. Hullermeier. Preference Learning: A Tutorial Introducton, DS 2011, Espoo, Finland, Oct 2011.
- [2] Hsu C W, Chang C C, Lin C J. A practical guide to support vector classification[J]. 2003.
- [3] Chang C C, Lin C J. LIBSVM: a library for support vector machines[J]. ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(3): 27.
- [4] Swingler K. Applying neural networks: a practical guide[M].Morgan Kaufmann, 1996.\
- [5] M. Cha, H. Haddadi, F. Benevenuto, and K. P. Gummadi. Measuring user influence in twitter: The million follower fallacy. In Proceedings of the 2010 International AAAI Conference on Weblogs and Social Media, 2010.
- [6] P. Chen, H. Xie, S. Maslov, and S. Redner. Finding scientific gems with google's pagerank algorithm. Journal of Informetrics, 1(1), 2007.
- [7] D. J. Cook and L. B. Holder. Mining Graph Data. John Wiley & Sons, 2006.
- [8] B. S. Demoll and D. Mcfarland. The Art and Science of Dynamic Network Visualization. Journal of Social Structure, Volume 7, 2005.
- [9] Y. Ding, E. Yan, A. Frazho, and J. Caverlee. Pagerank for ranking authors in co-citation networks. Journal of the American Society for Information Science and Technology, 60(11), 2009.
- [10] G. Dutton. Improving locational specificity of map data - a multi-resolution, metadata-driven approach and notation. International Journal of Geographical Information Science, 10(3), 1996.
- [11] D. Easley and J. Kleinberg. Networks, Crowds, and Markets: Reasoning About a Highly Connected World. Cambridge University Press, 2010.
- [12] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In Proceedings of the 7th International Conference on World Wide Web, 1998.