INTERNSHIP REPORT

A report submitted in partial fulfilment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

ARTIFICIAL

INTELLIGENCE AND MACHINE LEARNING

by

Y.ANURAG CHOWDARY

Regd. No.: 21671A7365

Under Supervision of

Mr. Gautam kumar

VERZEO EDU TECH Pvt. Ltd,

Hyderabad.

(Duration: 20th October 2022 to 19th November 2022)

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

J.B. INSTITUTE OF ENGINEERING AND TECHNOLOGY

(UGC Autonomous)

Approved by AICTE, Accredited by NBA & NAAC Permanently affiliated to JNTUH, Hyderabad, Telangana

2021 – 2025

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

J.B. INSTITUTE OF ENGINEERING AND TECHNOLOGY

(UGC Autonomous)

CERTIFICATE

This is to certify that the "Internship report" submitted by Y.Anurag chowdary (Regd. No.:21671A7365) is work done by her and submitted during academic year 2022 – 2023, in partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING, at VERZEO EDU TECH Pvt. Ltd, Hyderabad.

Mr. S. Sathish Kumar
Assistant professor & Internship Coordinator
Department of AI&ML

Dr. G. Arun Sampaul Thomas
Associate Professor & HOD
Department of AI&ML

ACKNOWLEDGEMENT

First I would like to thank Mr. Gautam kumar VERZEO EDU TECH Pvt. Ltd, Hyderabad for giving me the opportunity to do an internship within the organization.

I also would like all the people that worked along with me VERZEO EDU TECH Pvt. Ltd with their patience and openness they created an enjoyable working environment.

It is indeed with a great sense of pleasure and immense sense of gratitude that I acknowledge the help of these individuals.

I would like to thank **Dr.Sankar sarma,** internship coordinator, Department of **ARTIFICIAL INTELLIGENCE & MACHINE LEARNING** for his support and advices to get and complete internship in above said organization. I am extremely great full to my department staff members and friends who helped me in successful completion of this internship.

I would like to thank my Head of the Department **Dr.G.ARUN SAMPUL THOMAS** for his constructive criticism throughout my internship. I am highly indebted to **Principal Dr. P.C. Krishnamachary**, for the facilities provided to accomplish this internship.

Y. ANURAG CHOWDARY
21671A7365

TABLE OF CONTENTS:

1.ABSTRACT

Sentiment analysis is a classification problem where the main focus is to predict the polarity of words and then classify them into positive or negative sentiment. Classifiers used are of mainly two types, namely lexicon-based and machine learning based. The former include SentiWordNet and Word Sense Disambiguation while the latter include Multinomial Naive Bayes(MNB), Logistic Regression(LR), Support Vector Machine(SVM) and RNN Classifier. In this paper, existing datasets have been used, the first one from "Sentiment140" from Stanford University, consisting of 1.6 million tweets and the other one originally came from "Crowdflower's Data for Everyone library",consisting of 13870 entries, and both datasets are already categorised as per the sentiments expressed in them. Textblob, Sentiwordnet, MNB, LR, SVM and RNN Classifier are applied on the above dataset and a comparison is drawn between the results obtained from above mentioned sentiment classifiers, classifying tweets according to the sentiment expressed in them, i.e. positive or negative. Also, along with the machine learning approaches, an ensemble form of MNB, LR and SVM has been performed on the datasets and compared with the above results. Further the above trained models can be used for sentiment prediction of a new data.

Organization Information:

Address:Address: VERZEO EDU TECH Pvt. Ltd.,No.675,3RD FLOOR, 9TH MAIN SECTOR 7, HSR LATOUT,BANGLORE -Karnataka-India 560102
Email: support@verzeo.com

Verzeo help thousands of students decide their careers. We guarantee that Verzeo is 'the' one-stop online learning platform that provides students with all the resources needed to make them industry-ready. We aim to deliver our mantra "Everything Learning" to every household that is brewing an aspiring careerist.

At Verzeo, they bridge the gap between classrooms and workplaces with our flagship Mentorship Programs. Our course is for everyone ranging from Kids and Professionals to even Home-Makers! We aim to provide quality education, thus preparing them for better career prospects!

2.INTERNSHIP OBJECTIVES

One of the main objectives of an internship is to expose you to a particular job and a profession or industry. While you might have an idea about what a job is like, you won't know until you actually perform it if it's what you thought it was, if you have the training and skills to do it and if it's something you like. For example, you might think that advertising is a creative process that involves coming up with slogans and fun campaigns. Taking an internship at an advertising agency would help you find that advertising includes consumer demographic research, focus groups, knowledge of a client's pricing and distribution strategies, and media research and buying. When you apply for jobs, the more experience and accomplishments you have, the more attractive you'll look to a potential employer. Just because you have an internship with a specific title or well-known company doesn't mean your internship will help you land a nice gig. Make an impact where you work by asking for responsibility and looking for ways to achieve accomplishments. Be willing to work more hours than you're required and ask to work in different departments to expand your skill set. Don't just fetch coffee, make copies and sit in on meetings, even if that's all it will take to finish your internship.

Another benefit of an internship is developing business contacts. These people can help you find a job later, act as references or help you with projects after you're hired somewhere else. Meet the people who have jobs you would like some day and ask them if you can take them to lunch. Ask them how they started their careers, how they got to where they are now and if they have any suggestions for you to improve your skills.

3.WEEKLY REPORT OF INTERNSHIP ACTIVITIES

WEEK	PROGRESS
WEEK-I (20/10/22) to (27/10/22)	○ Python Fundamentals. ○ Data types, list, dictionary, array, string operations. ○ Condition and loops.
WEEK-II (27/10/22) to (03/11/22)	○ Inbuilt and user defined functions IO, Numpy, Error Handling. ○ Pandas and Visualisation. ○ Introduction to probability & distribution central limit theorem, hypothesis testing.
WEEK-III (03/11/22) to (10/11/22)	○ What is Machine Learning, Difference between a rule based algorithm and a machine learning algorithm. Supervised vs Unsupervised learning. Classification vs Regression. ○ Training, Testing and Cross Validation Data Features and labels pickling and scaling and Techniques, Error Metrics. ○ Linear Regression, Forecasting and prediction using regression, logistic regression, knn classification.
WEEK-IV (10/11/22) to (19/11/22)	○ Support vector machines, k-means clustering random forestlinear + minor project discussion. ○ Introduction to NLTK, stopwords stemming lemmetization named entity recognition text classification sentiment analysis using naïve bayes. ○ Implemetation of all the algorithms using sklearn and explanation on major project.

4.INTRODUCTION

Sentiment Analysis which means to analyse the underlying emotions of a given text using Natural Language Processing (NLP) and other techniques to extract a significant pattern of information and features from a given large corpus of text. It analyses the sentiment and attitude of the author towards the topic of the subject mentioned in the text. This text can be a part of any document, post on social media or from any database source. Sentiments can be classified as objective or subjective, positive or negative or neutral.This classification can be either lexicon-based or machine learning based. Lexicon based classification makes use of already existing dictionary which has pre-assigned scores to each word and those scores are used to calculate the overall sentiment expressed in the sentence whereas in machine learning based classification, a model is trained using some ML algorithm using some labelled data and then use that model to predict a class for a new text. Twitter is nowdays easily one of the most popular microblogging platforms and millions of users express their views publicly on Twitter making it a rich source of information on public opinions and thus, helpful in sentiment analysis on any topic. In this paper, lexicon and machinelearning based approaches have been employed to reveal the prevailing sentiments of tweets. Textblob, SentiWordNet and Word Sense Disambiguation are giving the correct sense of a word in a given context for Lexicon-based Sentiment analysis while among the machine learning based algorithms MNB, LR, SVM and RNN Classifier have been used. In this paper, a comparison has been presented in terms of accuracy in predicting the sentiment of a given tweet. An ensemble approach has also been implemented on the datasets which involve majority voting of MNB, LR and SVM and the resultsare being compared with the rest of the approaches.

5.METHODOLOGIES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, 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.

TYPES OF TESTS

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.

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.

Functional test

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, and successive processes must be considered for testing. Before

functional testing is complete, additional tests are identified and the effective value of current tests is determined.

System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

8. Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

6.SYSTEM ANALYSIS

EXISTING SYSTEM:

Sentiment analysis is a classification problem where the main focus is to predict the polarity of words and then classify them into positive or negative sentiment.Classifiers used are of mainly two types, namely lexicon-based and machine learning based. The former include SentiWordNet and Word Sense Disambiguation while the latter include Multinomial Naive Bayes(MNB), Logistic Regression(LR), Support Vector Machine(SVM) and RNN Classifier

DISADVANTAGES OF EXISTING SYSTEM:

- Sentiment Analysis which means to analyse the underlying emotions of a given text using Natural Language Processing
- ML algorithm using some labelled data and then use that model to predict a class for a new text.

PROPOSED SYSTEM:

lexicon and machinelearning based approaches have been employed to reveal the prevailing sentiments of tweets. Textblob, SentiWordNet and Word Sense Disambiguation are giving the correct sense of a word in a given context for Lexicon-based Sentiment analysis while among the machine learning based algorithms MNB, LR, SVM and RNN Classifier have been used

ADVANTAGES OF PROPOSED SYSTEM:

- The tweets directly, they have to be preprocessed using NLTK in order to convert it to a form that can be easily utilised for further analysis
- It provide better result.
- To improve the results a little more, majority voting has been used as an ensemble approach on the MNB, LR and SVM and it is also used for the classification and comparing the results
- this will help the farmers which crop to be selected for their land or the region

7.SYSTEM REQUIREMENTS SPECIFICATIONS

HARDWARE REQUIREMENTS:

- **System** : Intel i3
- **Hard Disk** : 1 TB.
- **Monitor** : 14’ Colour Monitor.
- **Mouse** : Optical Mouse.
- **Ram** : 4GB.

SOFTWARE REQUIREMENTS:

- **Operating system** : Windows 10.
- **Coding Language** : Python.
- **Front-End** : Html. CSS
- **Designing** : Html,css,javascript.
- **Data Base** : SQLite.

8.TECHNOLOGIES

PYTHON

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. 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.

Interactive Mode Programming

Invoking the interpreter without passing a script file as a parameter brings up the following prompt −

```
$ python
Python 2.4.3 (#1, Nov 11 2010, 13:34:43)
[GCC 4.1.2 20080704 (Red Hat 4.1.2-48)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Type the following text at the Python prompt and press the Enter −

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in print ("Hello, Python!");. However in Python version 2.4.3, this produces the following result −

Hello, Python!

Script Mode Programming

Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.

Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file −

Live Demo

```
print "Hello, Python!"
```

We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows −

```
$ python test.py
```

This produces the following result −

Hello, Python!

Let us try another way to execute a Python script. Here is the modified test.py file −

Live Demo

```
#!/usr/bin/python

print "Hello, Python!"
```

We assume that you have Python interpreter available in /usr/bin directory. Now, try to run this program as follows −

$ chmod +x test.py # This is to make file executable

$./test.py

This produces the following result −

Hello, Python!

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).

Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language. Thus, Manpower and manpower are two different identifiers in Python.

Here are naming conventions for Python identifiers −

Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

Starting an identifier with a single leading underscore indicates that the identifier is private.

Starting an identifier with two leading underscores indicates a strongly private identifier.

If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Reserved Words

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

and	exec	not	
assert	finally	or	
break	for	pass	
class	from	print	
continue	global	raise	
def	if	return	
del	import	try	
elif	in	while	
else	is	with	
except	lambda	yield	

Lines and Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example −

```
if True:
   print "True"
else:
   print "False"
```

However, the following block generates an error −

```
if True:
print "Answer"
print "True"
```

```
else:
print "Answer"
print "False"
```

Thus, in Python all the continuous lines indented with same number of spaces would form a block. The following example has various statement blocks −

Note − Do not try to understand the logic at this point of time. Just make sure you understood various blocks even if they are without braces.

```
#!/usr/bin/python

import sys

try:
   # open file stream
   file = open(file_name, "w")
except IOError:
   print "There was an error writing to", file_name
   sys.exit()
print "Enter '", file_finish,
print "' When finished"
while file_text != file_finish:
   file_text = raw_input("Enter text: ")
   if file_text == file_finish:
      # close the file
      file.close
      break
   file.write(file_text)
   file.write("\n")
file.close()
file_name = raw_input("Enter filename: ")
```

```
if len(file_name) == 0:
   print "Next time please enter something"
   sys.exit()
try:
   file = open(file_name, "r")
except IOError:
   print "There was an error reading file"
   sys.exit()
file_text = file.read()
file.close()
print file_text
```

Multi-Line Statements

Statements in Python typically end with a new line. Python does, however, allow the use of the line continuation character (\) to denote that the line should continue. For example −

```
total = item_one + \
        item_two + \
        item_three
```

Statements contained within the [], {}, or () brackets do not need to use the line continuation character. For example −

```
days = ['Monday', 'Tuesday', 'Wednesday',
        'Thursday', 'Friday']
```

Quotation in Python

Python accepts single ('), double (") and triple ('" or """) quotes to denote string literals, as long as the same type of quote starts and ends the string.

The triple quotes are used to span the string across multiple lines. For example, all the following are legal −

```
word = 'word'
sentence = "This is a sentence."
paragraph = """This is a paragraph. It is
made up of multiple lines and sentences."""
```

Comments in Python

A hash sign (#) that is not inside a string literal begins a comment. All characters after the # and up to the end of the physical line are part of the comment and the Python interpreter ignores them.

Live Demo

```
#!/usr/bin/python

# First comment
print "Hello, Python!" # second comment
```

This produces the following result −

```
Hello, Python!
```

You can type a comment on the same line after a statement or expression −

```
name = "Madisetti" # This is again comment
```

You can comment multiple lines as follows −

```
# This is a comment.
# This is a comment, too.
# This is a comment, too.
# I said that already.
```

Following triple-quoted string is also ignored by Python interpreter and can be used as a multiline comments:

```
'''
This is a multiline
comment.
```

'''

Using Blank Lines

A line containing only whitespace, possibly with a comment, is known as a blank line and Python totally ignores it.

In an interactive interpreter session, you must enter an empty physical line to terminate a multiline statement.

Waiting for the User

The following line of the program displays the prompt, the statement saying “Press the enter key to exit”, and waits for the user to take action −

```
#!/usr/bin/python

raw_input("\n\nPress the enter key to exit.")
```

Here, "\n\n" is used to create two new lines before displaying the actual line. Once the user presses the key, the program ends. This is a nice trick to keep a console window open until the user is done with an application.

Multiple Statements on a Single Line

The semicolon (;) allows multiple statements on the single line given that neither statement starts a new code block. Here is a sample snip using the semicolon.

```
import sys; x = 'foo'; sys.stdout.write(x + '\n')
```

Multiple Statement Groups as Suites

A group of individual statements, which make a single code block are called suites in Python. Compound or complex statements, such as if, while, def, and class require a header line and a suite.

Header lines begin the statement (with the keyword) and terminate with a colon (:) and are followed by one or more lines which make up the suite. For example −

```
if expression :
   suite
```

```
elif expression :
   suite
else :
   suite
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python -h
usage: python [option] ... [-c cmd | -m mod | file | -] [arg] ...
Options and arguments (and corresponding environment variables):
-c cmd : program passed in as string (terminates option list)
-d     : debug output from parser (also PYTHONDEBUG=x)
-E     : ignore environment variables (such as PYTHONPATH)
-h     : print this help message and exit
```

You can also program your script in such a way that it should accept various options. Command Line Arguments is an advanced topic and should be studied a bit later once you have gone through rest of the Python concepts.

Python Lists

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example −

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5 ];
list3 = ["a", "b", "c", "d"]
```

Similar to string indices, list indices start at 0, and lists can be sliced, concatenated and so on.

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally you can put these comma-separated values between parentheses also. For example −

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing −

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value −

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

Accessing Values in Tuples

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example −

Live Demo

```
#!/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0]: ", tup1[0];
print "tup2[1:5]: ", tup2[1:5];
```

When the above code is executed, it produces the following result −

tup1[0]: physics
tup2[1:5]: [2, 3, 4, 5]
Updating Tuples

Accessing Values in Dictionary
To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example −

Live Demo
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
When the above code is executed, it produces the following result −

dict['Name']: Zara
dict['Age']: 7
If we attempt to access a data item with a key, which is not part of the dictionary, we get an error as follows −

Live Demo
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print "dict['Alice']: ", dict['Alice']
When the above code is executed, it produces the following result −

dict['Alice']:

```
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    print "dict['Alice']: ", dict['Alice'];
KeyError: 'Alice'
```

Updating Dictionary

You can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry

print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

When the above code is executed, it produces the following result −

```
dict['Age']:  8
dict['School']:  DPS School
```

Delete Dictionary Elements

You can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the del statement. Following is a simple example −

Live Demo

```
#!/usr/bin/python
```

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name']; # remove entry with key 'Name'
dict.clear();     # remove all entries in dict
del dict ;        # delete entire dictionary

print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

This produces the following result. Note that an exception is raised because after del dict dictionary does not exist any more −

```
dict['Age']:
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    print "dict['Age']: ", dict['Age'];
TypeError: 'type' object is unsubscriptable
```

Note − del() method is discussed in subsequent section.

Properties of Dictionary Keys

Dictionary values have no restrictions. They can be any arbitrary Python object, either standard objects or user-defined objects. However, same is not true for the keys.

There are two important points to remember about dictionary keys −

(a) More than one entry per key not allowed. Which means no duplicate key is allowed. When duplicate keys encountered during assignment, the last assignment wins. For example −

Live Demo

```
#!/usr/bin/python

dict = {'Name': 'Zara', 'Age': 7, 'Name': 'Manni'}
```

```
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
dict['Name']:  Manni
```

(b) Keys must be immutable. Which means you can use strings, numbers or tuples as dictionary keys but something like ['key'] is not allowed. Following is a simple example −

Live Demo

```
#!/usr/bin/python

dict = {['Name']: 'Zara', 'Age': 7}
print "dict['Name']: ", dict['Name']
```

When the above code is executed, it produces the following result −

```
Traceback (most recent call last):
  File "test.py", line 3, in <module>
    dict = {['Name']: 'Zara', 'Age': 7};
TypeError: unhashable type: 'list'
```

Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates −

Live Demo

```
#!/usr/bin/python

tup1 = (12, 34.56);
tup2 = ('abc', 'xyz');

# Following action is not valid for tuples
# tup1[0] = 100;
```

```
# So let's create a new tuple as follows
tup3 = tup1 + tup2;
print tup3;
```

When the above code is executed, it produces the following result −

```
(12, 34.56, 'abc', 'xyz')
```

Delete Tuple Elements

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the del statement. For example −

Live Demo

```
#!/usr/bin/python

tup = ('physics', 'chemistry', 1997, 2000);
print tup;
del tup;
print "After deleting tup : ";
print tup;
```

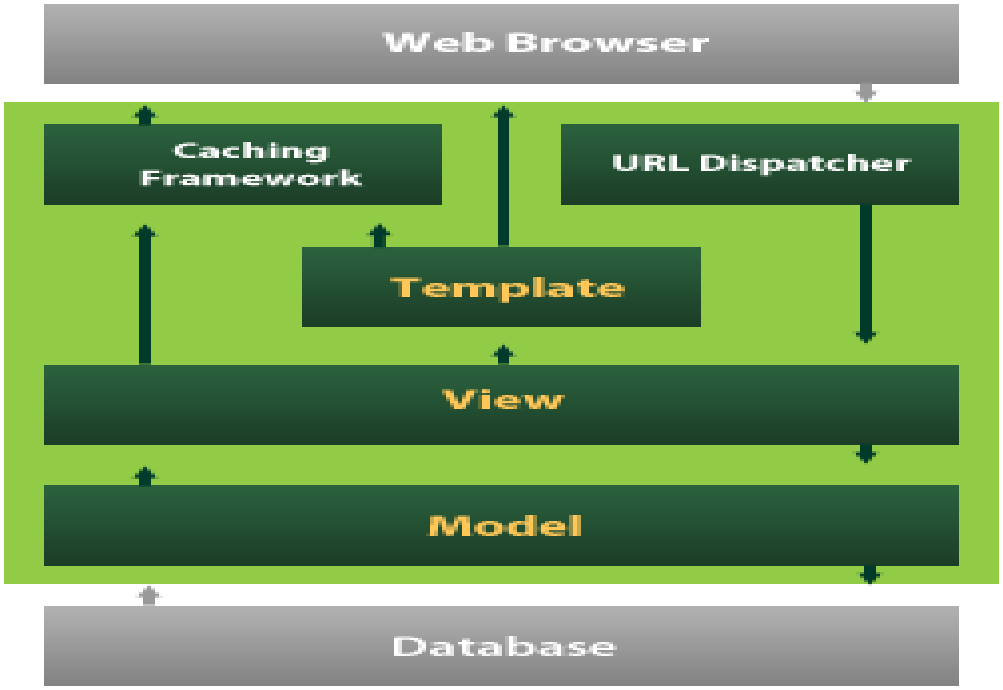
This produces the following result. Note an exception raised, this is because after del tup tuple does not exist any more −

```
('physics', 'chemistry', 1997, 2000)
After deleting tup :
Traceback (most recent call last):
  File "test.py", line 9, in <module>
    print tup;
NameError: name 'tup' is not defined
```

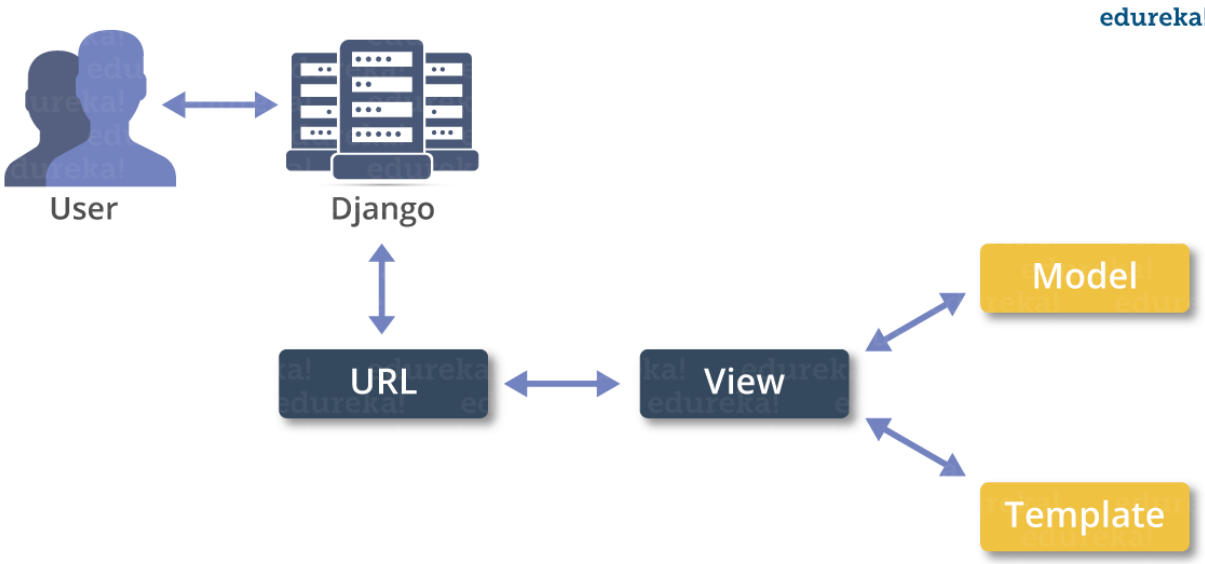
DJANGO:

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusabilityand "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models.



Django also provides an optional administrative create, read, update and delete interface that is generated dynamically through introspection and configured via admin models



Create a Project

Whether you are on Windows or Linux, just get a terminal or a cmd prompt and navigate to the place you want your project to be created, then use this code −

$ django-admin startproject myproject

This will create a "myproject" folder with the following structure −

```
myproject/
   manage.py
   myproject/
      __init__.py
      settings.py
      urls.py
      wsgi.py
```

The Project Structure

The “myproject” folder is just your project container, it actually contains two elements −

manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

$ python manage.py help

The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −

__init__.py − Just for python, treat this folder as package.

settings.py − As the name indicates, your project settings.

urls.py − All links of your project and the function to call. A kind of ToC of your project.

wsgi.py − If you need to deploy your project over WSGI.

Setting Up Your Project

Your project is set up in the subfolder myproject/settings.py. Following are some important options you might need to set −

DEBUG = True

This option lets you set if your project is in debug mode or not. Debug mode lets you get more information about your project's error. Never set it to ‘True’ for a live project. However, this has to be set to ‘True’ if you want the Django light server to serve static files. Do it only in the development mode.

```
DATABASES = {
  'default': {
    'ENGINE': 'django.db.backends.sqlite3',
    'NAME': 'database.sql',
    'USER': '',
    'PASSWORD': '',
    'HOST': '',
    'PORT': '',
```

```
  }
}
```

Database is set in the ‘Database’ dictionary. The example above is for SQLite engine. As stated earlier, Django also supports −

MySQL (django.db.backends.mysql)
PostGreSQL (django.db.backends.postgresql_psycopg2)
Oracle (django.db.backends.oracle) and NoSQL DB
MongoDB (django_mongodb_engine)
Before setting any new engine, make sure you have the correct db driver installed.

You can also set others options like: TIME_ZONE, LANGUAGE_CODE, TEMPLATE…

Now that your project is created and configured make sure it's working −

```
$ python manage.py runserver
```

You will get something like the following on running the above code −

```
Validating models...

0 errors found
September 03, 2015 - 11:41:50
Django version 1.6.11, using settings 'myproject.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

A project is a sum of many applications. Every application has an objective and can be reused into another project, like the contact form on a website can be an application, and can be reused for others. See it as a module of your project.

Create an Application

We assume you are in your project folder. In our main “myproject” folder, the same folder then manage.py −

```
$ python manage.py startapp myapp
```

You just created myapp application and like project, Django create a “myapp” folder with the application structure −

```
myapp/
  __init__.py
  admin.py
  models.py
  tests.py
  views.py
```

__init__.py − Just to make sure python handles this folder as a package.

admin.py − This file helps you make the app modifiable in the admin interface.

models.py − This is where all the application models are stored.

tests.py − This is where your unit tests are.

views.py − This is where your application views are.

Get the Project to Know About Your Application

At this stage we have our "myapp" application, now we need to register it with our Django project "myproject". To do so, update INSTALLED_APPS tuple in the settings.py file of your project (add your app name) −

```
INSTALLED_APPS = (
  'django.contrib.admin',
```

```
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'myapp',
)
```

Creating forms in Django, is really similar to creating a model. Here again, we just need to inherit from Django class and the class attributes will be the form fields. Let's add a forms.py file in myapp folder to contain our app forms. We will create a login form.

myapp/forms.py

```
#-*- coding: utf-8 -*-
from django import forms

class LoginForm(forms.Form):
    user = forms.CharField(max_length = 100)
    password = forms.CharField(widget = forms.PasswordInput())
```

As seen above, the field type can take "widget" argument for html rendering; in our case, we want the password to be hidden, not displayed. Many others widget are present in Django: DateInput for dates, CheckboxInput for checkboxes, etc.

Using Form in a View

There are two kinds of HTTP requests, GET and POST. In Django, the request object passed as parameter to your view has an attribute called "method" where the type of the request is set, and all data passed via POST can be accessed via the request.POST dictionary.

Let's create a login view in our myapp/views.py −

```
#-*- coding: utf-8 -*-
from myapp.forms import LoginForm

def login(request):
  username = "not logged in"

  if request.method == "POST":
    #Get the posted form
    MyLoginForm = LoginForm(request.POST)

    if MyLoginForm.is_valid():
      username = MyLoginForm.cleaned_data['username']
  else:
    MyLoginForm = Loginform()

  return render(request, 'loggedin.html', {"username" : username})
```

The view will display the result of the login form posted through the loggedin.html. To test it, we will first need the login form template. Let's call it login.html.

```
<html>
  <body>

    <form name = "form" action = "{% url "myapp.views.login" %}"
      method = "POST" >{% csrf_token %}

      <div style = "max-width:470px;">
        <center>
          <input type = "text" style = "margin-left:20%;"
            placeholder = "Identifiant" name = "username" />
        </center>
```

```
        </div>

        <br>

        <div style = "max-width:470px;">
          <center>
            <input type = "password" style = "margin-left:20%;"
              placeholder = "password" name = "password" />
          </center>
        </div>

        <br>

        <div style = "max-width:470px;">
          <center>

            <button style = "border:0px; background-color:#4285F4; margin-top:8%;
              height:35px; width:80%;margin-left:19%;" type = "submit"
              value = "Login" >
              <strong>Login</strong>
            </button>

          </center>
        </div>

      </form>

    </body>
</html>
```

The template will display a login form and post the result to our login view above. You have probably noticed the tag in the template, which is just to prevent Cross-site Request Forgery (CSRF) attack on your site.

{% csrf_token %}

Once we have the login template, we need the loggedin.html template that will be rendered after form treatment.

```
<html>

   <body>
      You are : <strong>{{username}}</strong>
   </body>

</html>
```

Now, we just need our pair of URLs to get started: myapp/urls.py

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',
   url(r'^connection/',TemplateView.as_view(template_name = 'login.html')),
   url(r'^login/', 'login', name = 'login'))
```

When accessing "/myapp/connection", we will get the following login.html template rendered −

Setting Up Sessions

In Django, enabling session is done in your project settings.py, by adding some lines to the MIDDLEWARE_CLASSES and the INSTALLED_APPS options. This should be done while creating the project, but it's always good to know, so MIDDLEWARE_CLASSES should have −

'django.contrib.sessions.middleware.SessionMiddleware'
And INSTALLED_APPS should have −

'django.contrib.sessions'
By default, Django saves session information in database (django_session table or collection), but you can configure the engine to store information using other ways like: in file or in cache.

When session is enabled, every request (first argument of any view in Django) has a session (dict) attribute.

Let's create a simple sample to see how to create and save sessions. We have built a simple login system before (see Django form processing chapter and Django Cookies Handling chapter). Let us save the username in a cookie so, if not signed out, when accessing our login page you won't see the login form. Basically, let's make our login system we used in Django Cookies handling more secure, by saving cookies server side.

For this, first lets change our login view to save our username cookie server side −

```
def login(request):
   username = 'not logged in'

   if request.method == 'POST':
      MyLoginForm = LoginForm(request.POST)

      if MyLoginForm.is_valid():
         username = MyLoginForm.cleaned_data['username']
         request.session['username'] = username
      else:
         MyLoginForm = LoginForm()
```

```
   return render(request, 'loggedin.html', {"username" : username}
```

Then let us create formView view for the login form, where we won't display the form if cookie is set −

```
def formView(request):
   if request.session.has_key('username'):
      username = request.session['username']
      return render(request, 'loggedin.html', {"username" : username})
   else:
      return render(request, 'login.html', {})
```

Now let us change the url.py file to change the url so it pairs with our new view −

```
from django.conf.urls import patterns, url
from django.views.generic import TemplateView

urlpatterns = patterns('myapp.views',
   url(r'^connection/','formView', name = 'loginform'),
   url(r'^login/', 'login', name = 'login'))
```

When ac

9.CODING
SAMPLE CODE

url.py:

```
"""TweetData URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/2.2/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.contrib import admin
from django.urls import path
from admins import views as admins
from TweetData import views as TweetData
from django.conf import settings
from django.conf.urls.static import static

urlpatterns = [
    #path('admin/', admin.site.urls),
    path('admin/', admin.site.urls),
    path('index/', admins.index, name='index'),
    path('adminlogin/', admins.adminlogin, name='adminlogin'),
    path('adminloginentered/', admins.adminloginentered, name='adminloginentered'),
    path('storecsvdata/', admins.storecsvdata, name='storecsvdata'),
    path('svm/', admins.svm, name='svm'),
    path('NaiveBayes/', admins.NaiveBayes, name='NaiveBayes'),
    path('logout/', admins.logout, name='logout'),



    path('searchahashtags/', TweetData.searchahashtags, name='searchahashtags'),
    path('searchresults/', TweetData.searchresults, name='searchresults'),




]
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL,document_root=settings.MEDIA_ROOT)
```

models.py

```
from django.db import models

# Create your models here.
class storedatamodel(models.Model):

    #id = models.CharField(max_length=500)
    label = models.CharField(max_length=300)
    tweet = models.CharField(max_length=300)



    def __str__(self):
        return self.label,self.tweet
```

Views.py

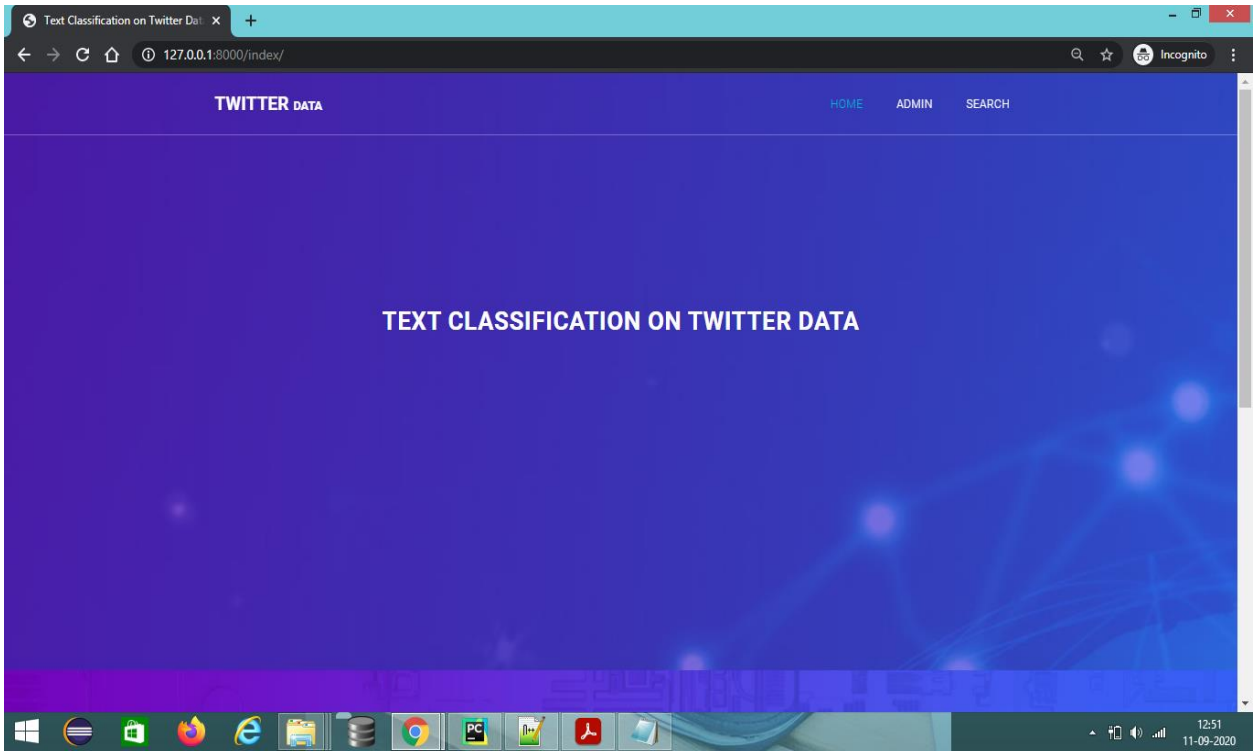
```
from django.http import HttpResponse
from django.shortcuts import render
from io import TextIOWrapper
import csv
from collections import defaultdict
# Create your views here.
from admins.models import storedatamodel
from django_pandas.io import read_frame
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix


def index(request):
    return render(request,'index.html')

def adminlogin(request):
    return render(request, "admins/adminlogin.html")

def logout(request):
    return render(request,'index.html')



def adminloginentered(request):
    if request.method == 'POST':
        uname=request.POST['uname']
        passwd=request.POST['upasswd']
        if uname == 'admin' and passwd == 'admin@2020':
            return render(request,"admins/adminloginentered.html")
```

```
        else:
            return HttpResponse("invalied credentials")

def storecsvdata(request):
    if request.method == 'POST':
        name = request.POST.get('name')
        csvfile =TextIOWrapper( request.FILES['file'],encoding='utf-8')
        columns = defaultdict(list)

        storecsvdata = csv.DictReader(csvfile)

        for row1 in storecsvdata:
                #id = row1["id"]
                label = row1["label"]
                tweet = row1["tweet"]


                storedatamodel.objects.create( label=label, tweet=tweet
                                        )

        print("Name is ",csvfile)
        return HttpResponse('CSV file successful uploaded')
    else:

        return render(request, 'admins/storedata.html', {})
def NaiveBayes(request):
    qs = storedatamodel.objects.all()
    data = read_frame(qs)
    data = data.fillna(data.mean())
    # data[0:label]
    data.info()
    print(data.head())
    print(data.describe())
    #print(data.shape)
    # print("data-label:",data.label)
    dataset = data.iloc[:,[0,1]].values
    print("x", dataset)
    dataset1 = data.iloc[:,1].values
    print("y", dataset1)
    print("shape", dataset.shape)
    X = dataset
    y = dataset1
    print(dataset.shape)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1 / 3, random_state=0)
    st_X = StandardScaler()
    X_train = st_X.fit_transform(X_train)
    X_test = st_X.transform(X_test)
    classifier = KNeighborsClassifier()
    classifier.fit(X_train, y_train)

    y_pred = classifier.predict(X_test)
    print("y_pred", y_pred)
    cm = confusion_matrix(y_test, y_pred)
```

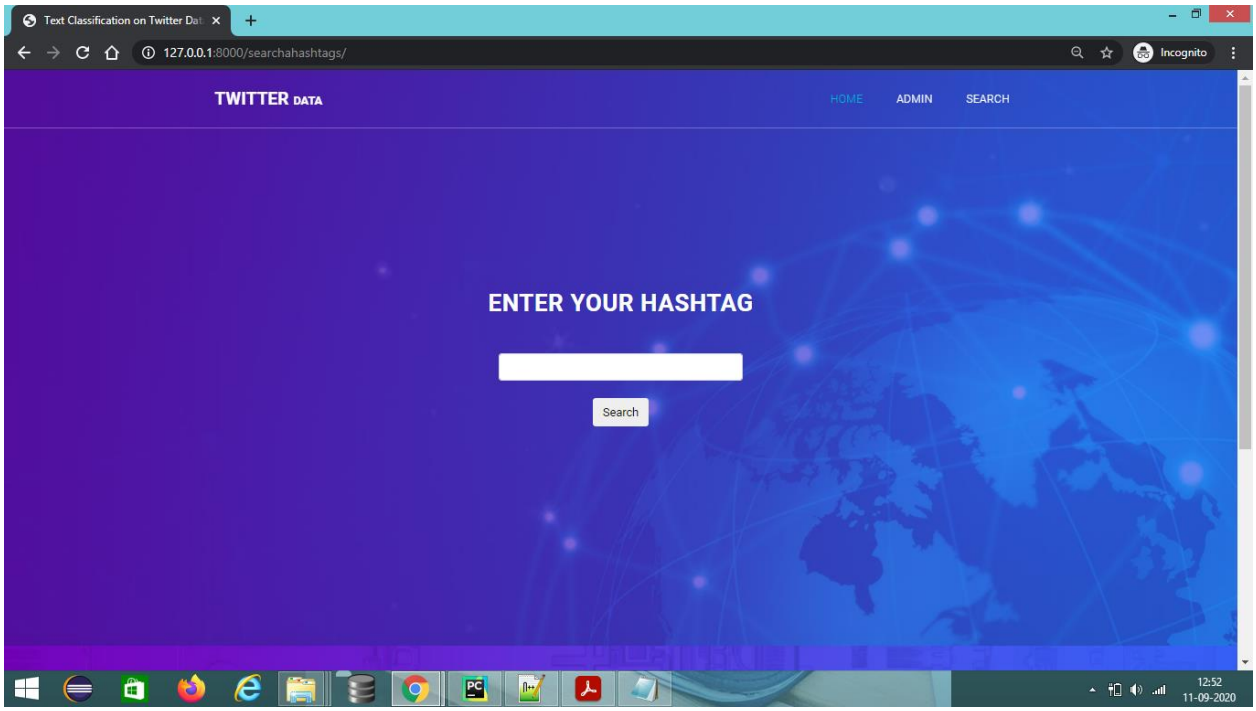
```
    print("cm", cm)
    accurancy = classifier.score(X_train, y_train)
    print("accurancy", accurancy)
    predicition =classification_report(y_test, y_pred)
    print("predicition",predicition)
    x = predicition.split()
    print("Toctal splits ", len(x))
    dict = {
        "accurancy": accurancy,
        'len0': x[0],
        'len1': x[1],
        'len2': x[2],
        'len3': x[3],
        'len4': x[4],
        'len5': x[5],
        'len6': x[6],
        'len7': x[7],
        'len8': x[8],
        'len9': x[9],
        'len10': x[10],
        'len11': x[11],
        'len12': x[12],
        'len13': x[13],
        'len14': x[14],
        'len15': x[15],
        'len16': x[16],
        'len17': x[17],
        'len18': x[18],
        'len19': x[19],
        'len20': x[20],
        'len21': x[21],
        'len22': x[22],
        'len23': x[23],
        'len24': x[24],
        'len25': x[25],
        'len26': x[26],
        'len27': x[27],
        'len28': x[28],
}

    return render(request, 'admins/NaiveBayes.html',dict)

def svm(request):
    qs = storedatamodel.objects.all()
    data = read_frame(qs)
    data = data.fillna(data.mean())
    data.info()
    print(data.head())
    print(data.describe())
    # print("data-label:",data.label)
    dataset = data.iloc[:, [0, 1]].values
    print("x", dataset)
    dataset1 = data.iloc[:, 1].values
```

```
print("y", dataset1)
print("shape", dataset.shape)
X = dataset
y = dataset1
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=1 / 3, random_state=0)
svclassifier = SVC(kernel='linear')
svclassifier.fit(X_train, y_train)
# print(svclassifier.predict([0.58, 0.76]))
y_pred = svclassifier.predict(X_test)
m = confusion_matrix(y_test, y_pred)
accurancy = classification_report(y_test, y_pred)
print(m)
print(accurancy)
x = accurancy.split()
print("Toctal splits ", len(x))
dict = {

    #"m": m,
    "accurancy": accurancy,
    'len0': x[0],
    'len1': x[1],
    'len2': x[2],
    'len3': x[3],
    'len4': x[4],
    'len5': x[5],
    'len6': x[6],
    'len7': x[7],
    'len8': x[8],
    'len9': x[9],
    'len10': x[10],
    'len11': x[11],
    'len12': x[12],
    'len13': x[13],
    'len14': x[14],
    'len15': x[15],
    'len16': x[16],
    'len17': x[17],
    'len18': x[18],
    'len19': x[19],
    'len20': x[20],
    'len21': x[21],
    'len22': x[22],
    'len23': x[23],
    'len24': x[24],
    'len25': x[25],
    'len26': x[26],
    'len27': x[27],
    'len28': x[28],


}

return render(request, 'admins/svm.html',dict)
```

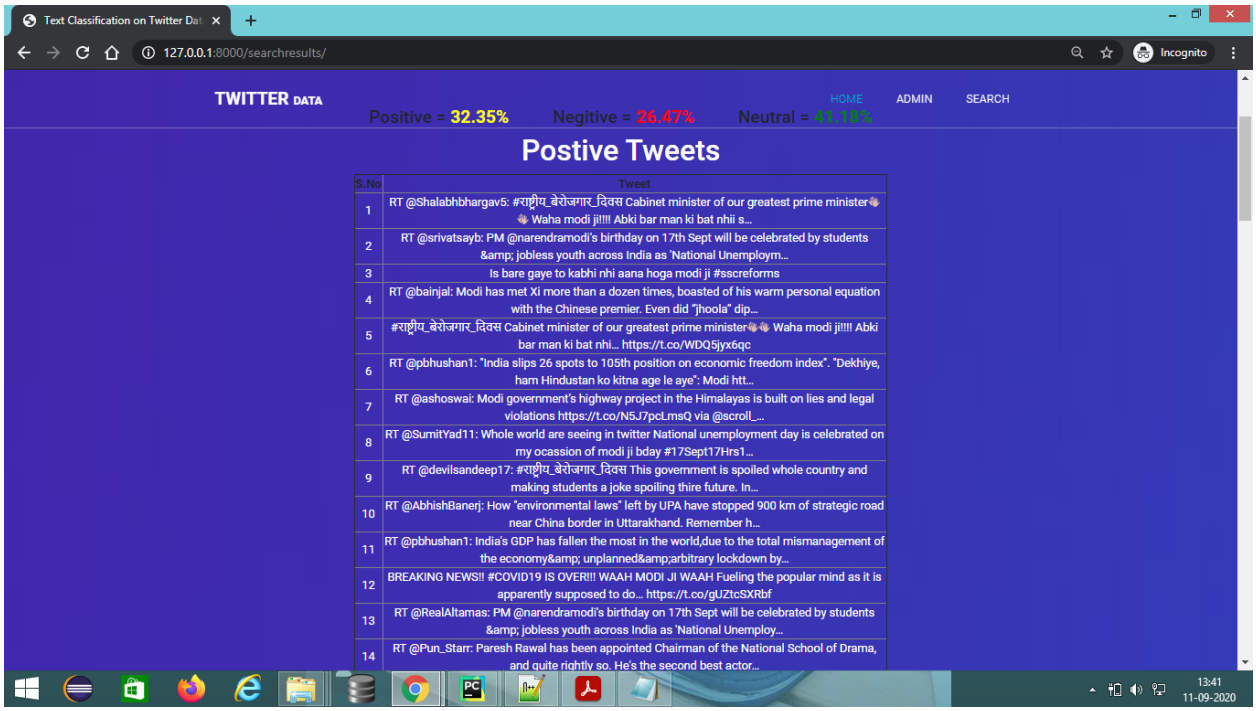
10.SCREEN SHOTS



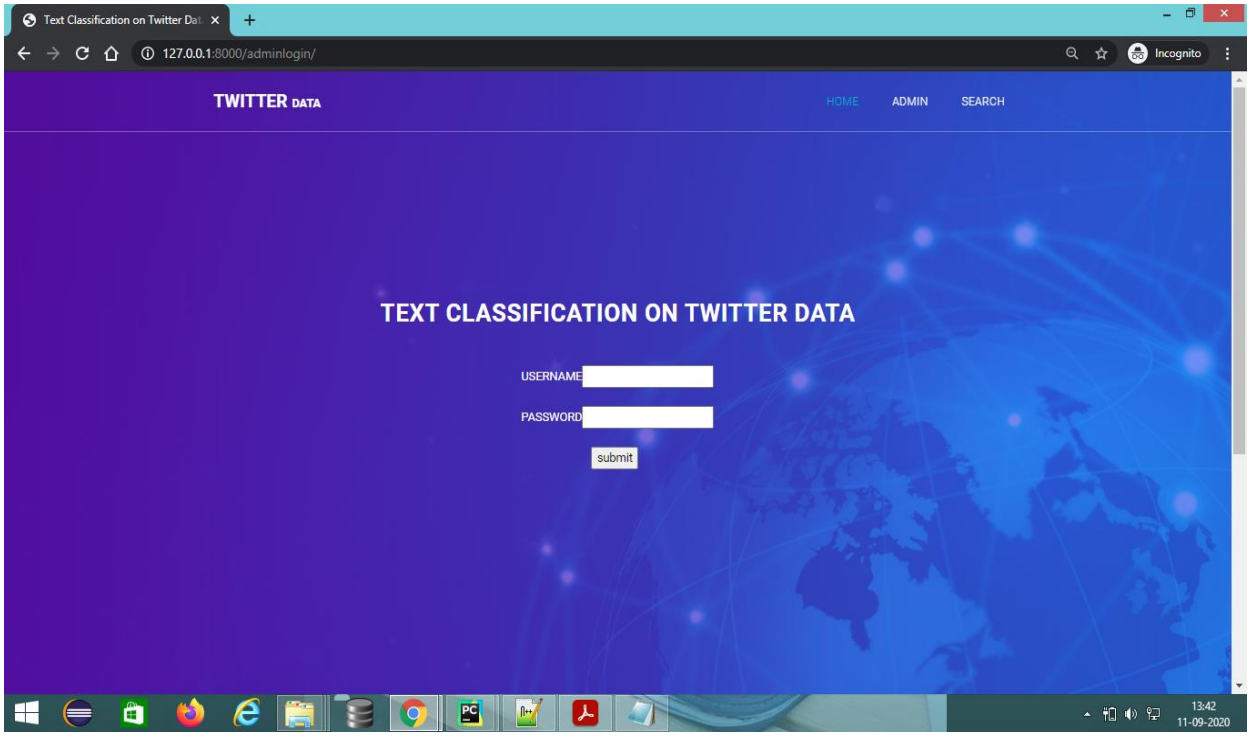
1:fig_Home page



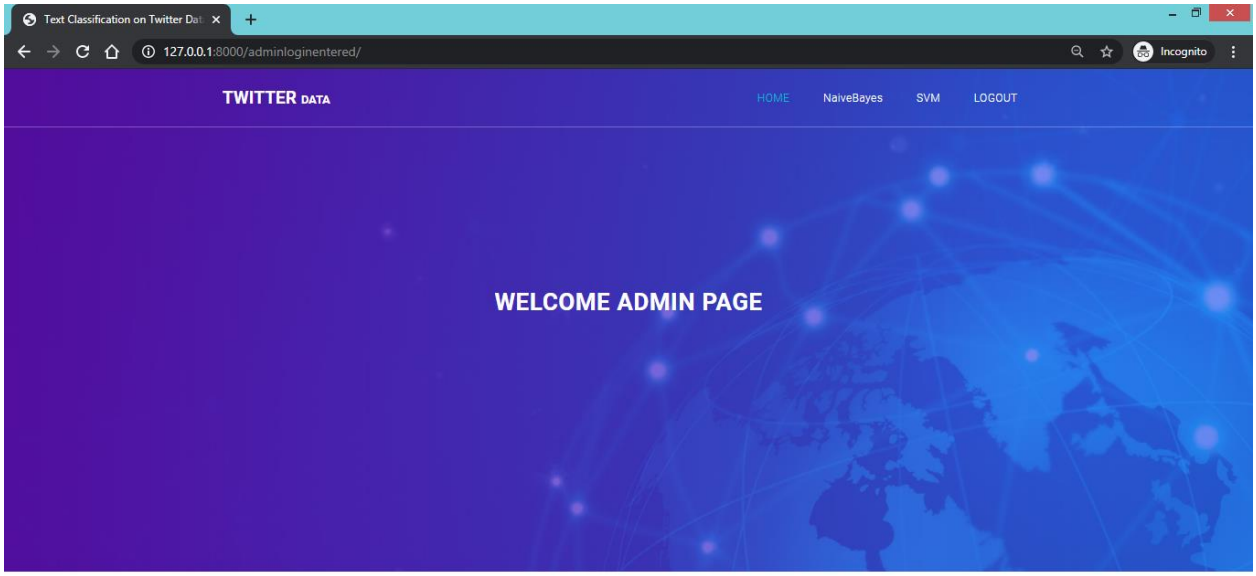
2:fig_Search:



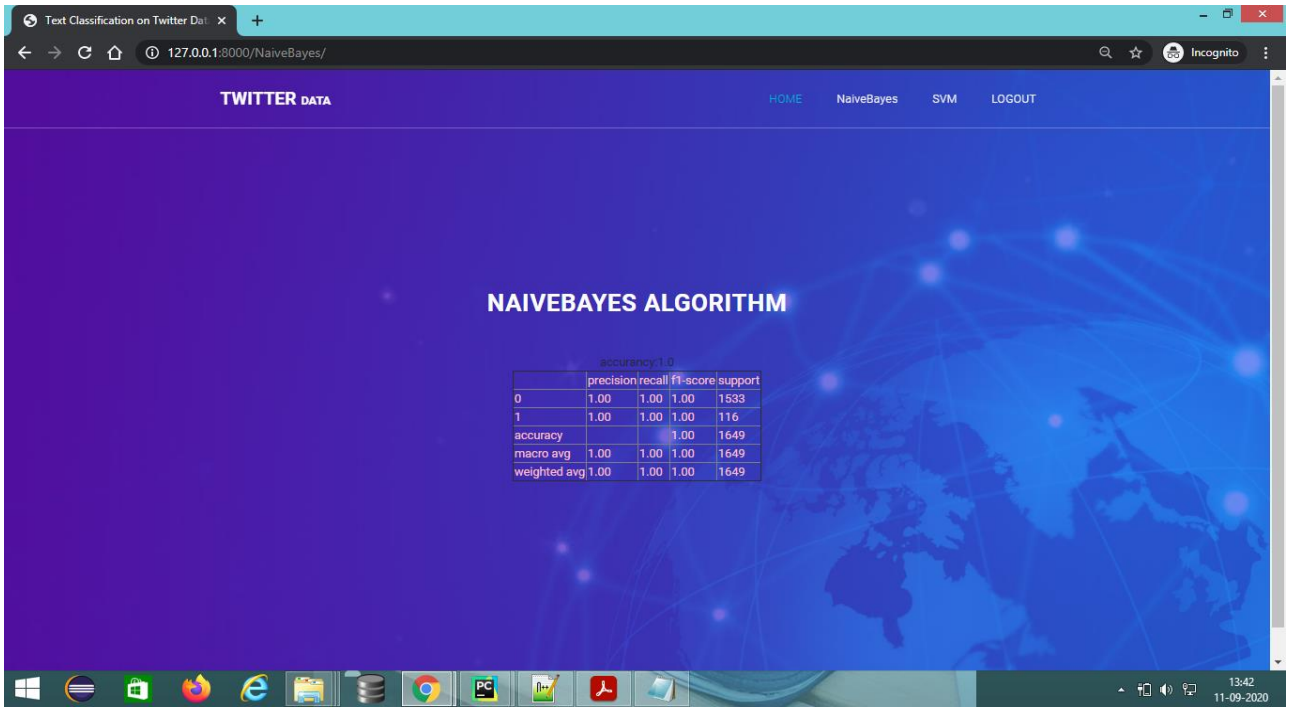
3:fig_Admin



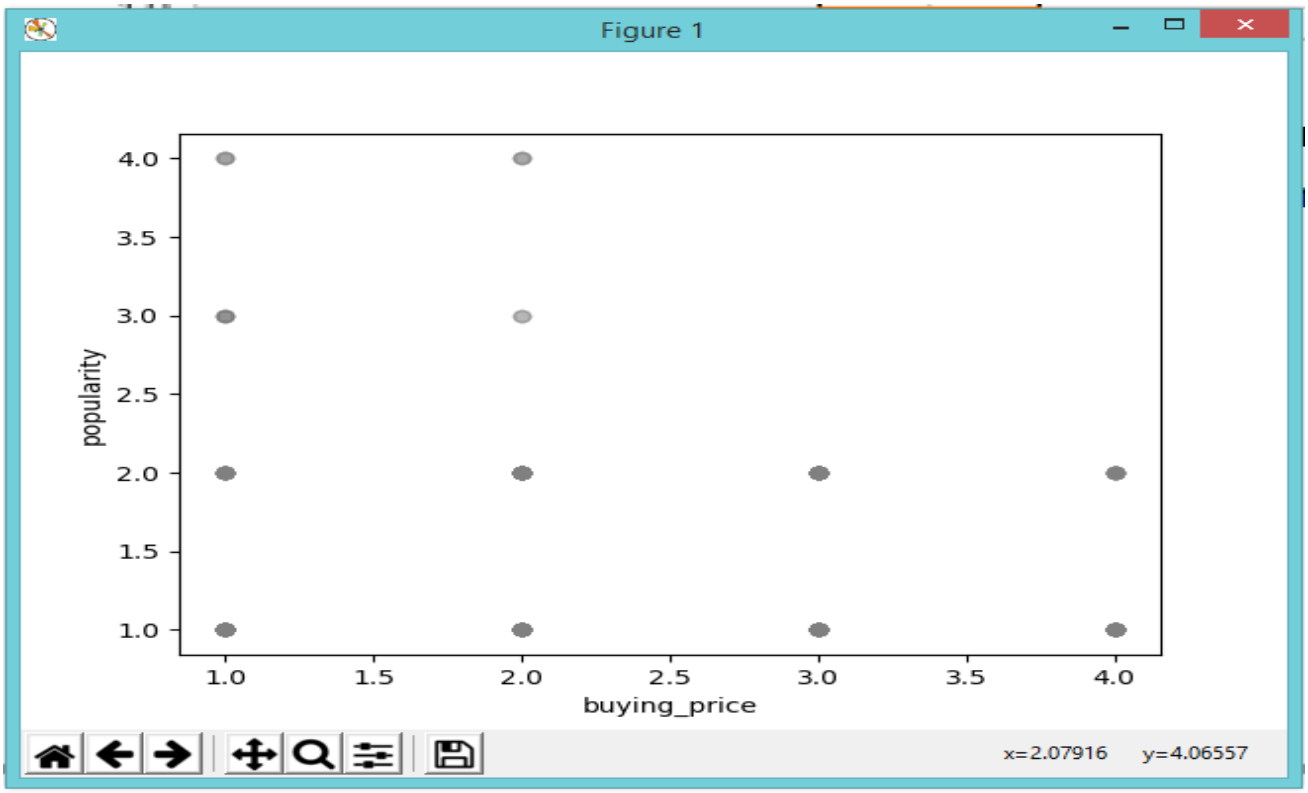
4:fig_Admin home



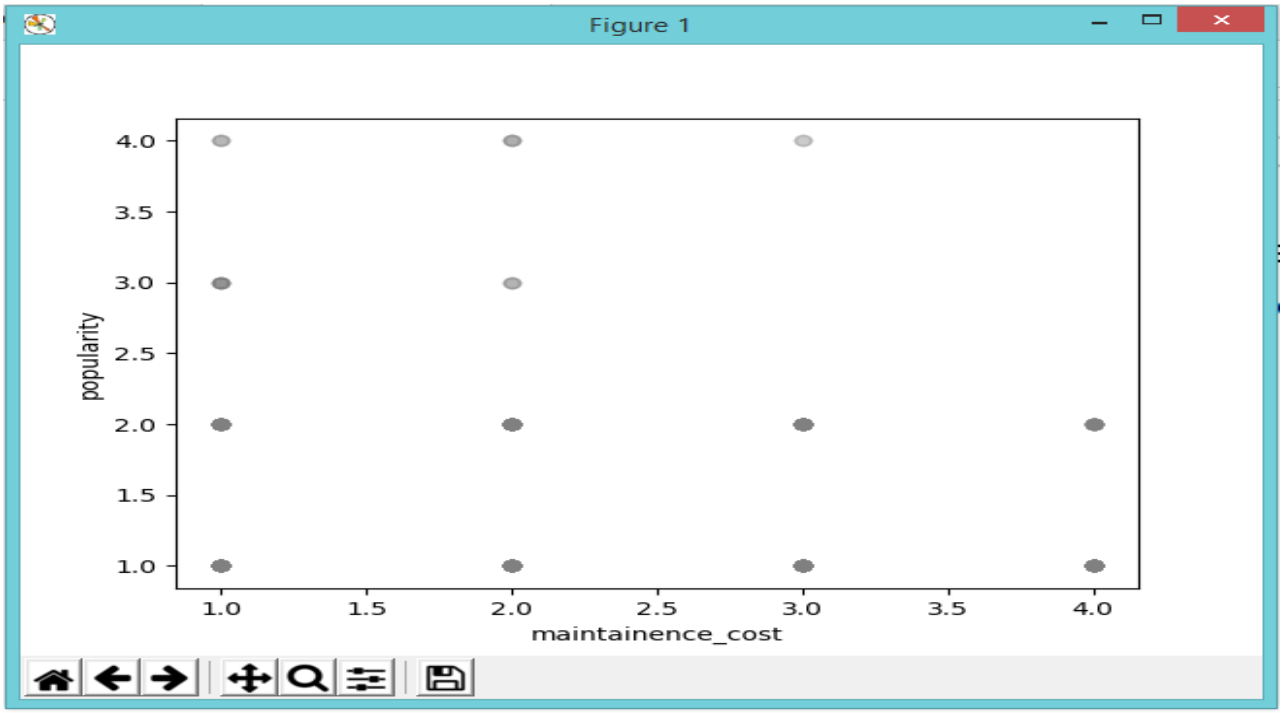
5:fig_Naive Bayes



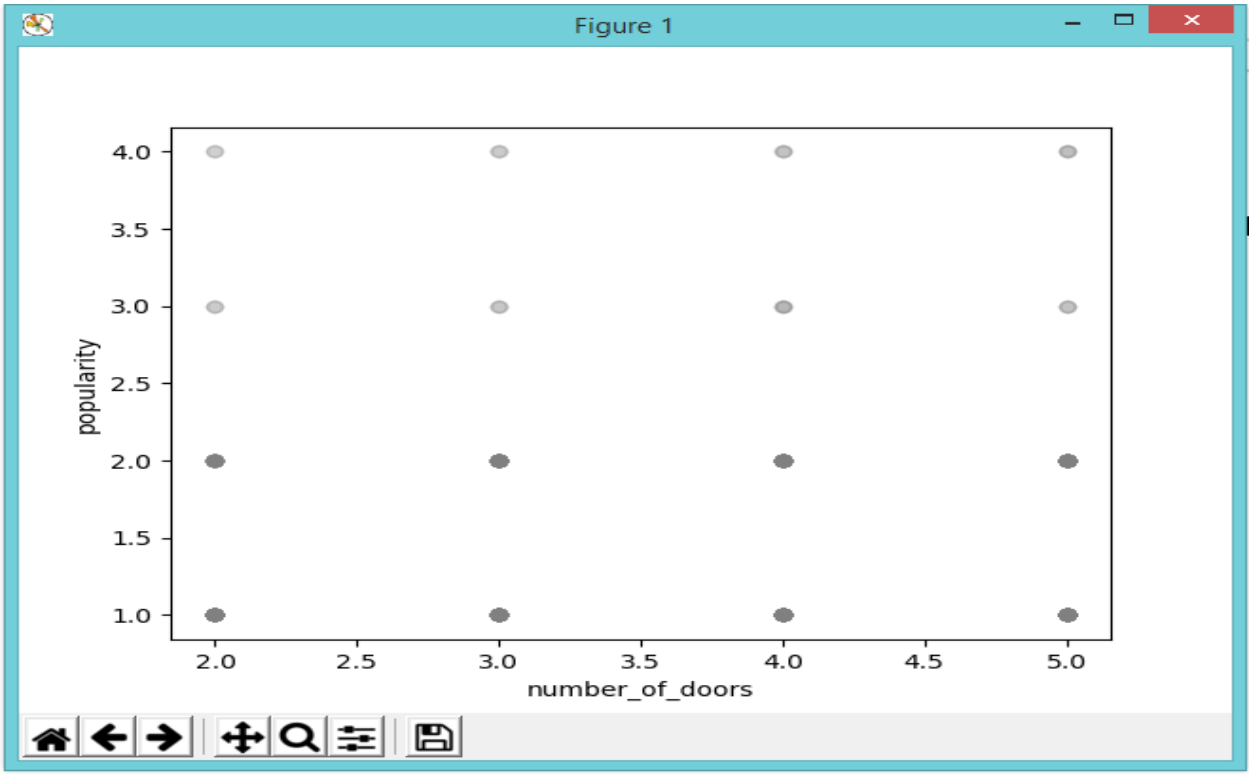
6:fig_svm



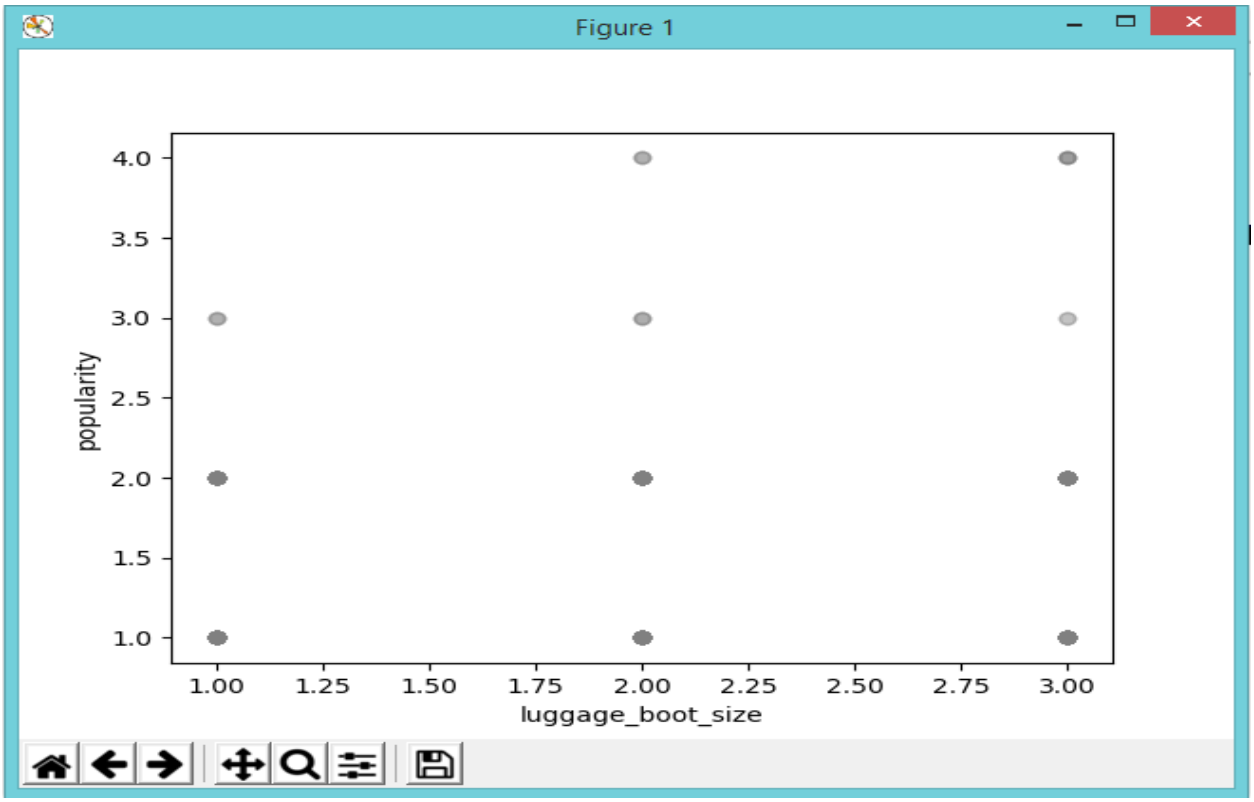
7:fig_Maintaince_cost



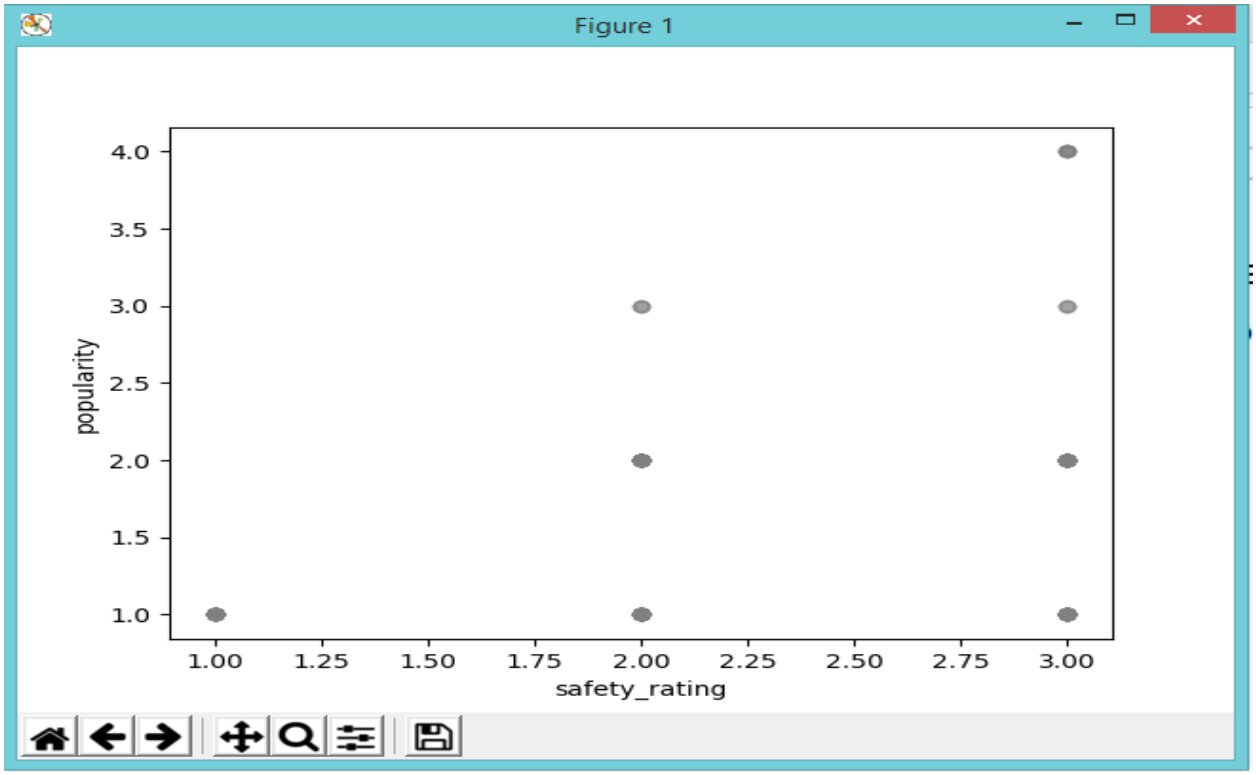
8:fig_Number_of_doors:



9:fig_Number_of_seats;



10:fig_Safety_rating:



11:fig_Luggage_boot_size:

11.CONCLUSION

Various techniques for both lexicon-based and machinelearning based, have been applied in this project and the results are compared. It has been observed that for a totally new data/text machine learning based models trained over a related data are much more accurate than the classification based on standard dictionaries. This is because of the fact that the text that's being observed i.e the tweets are highly informal and do not use the standard grammar rules or the spelling and thus the data here is highly unstructured. The comparison results can be clearly observed among different machine learning algorithms also. As of now, among the algorithms used, RNN is observed to have the highest accuracy.

12.BIBILOGRAPHY

[1] Harpreet Kaur, Veenu Mangat, Nidhi, "A survey of sentiment analysis techniques", February 2017,2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC).

[2] Ali Hasan, Sana Moin, Ahmad Karim And Shahaboddin Shamshirband, "Machine Learning-Based Sentiment Analysis For Twitter Accounts", February 2018.

[3] Biswarup Nandi, Mousumi Ghanti, Souvik Paul, "Text Based Sentiment Analysis", November 2017, 2017 International Conference on Inventive Computing and Informatics (ICICI).

[4] Bhagyashri Wagh, Prof. J. V. Shinde, Prof. P. A. Kale, "A Twitter Sentiment Analysis Using NLTK and Machine Learning Techniques", December 2017,International Journal of Emerging Research in Management & Technology.

[5] Hamid Bagheri, Md Johirul Islam, "Sentiment analysis of twitter data", Iowa State University .

[6] Apoorv Agarwal, Vivek Sharma, Geeta Sikka, Renu Dhir, " Opinion mining of news headlines using SentiWordNet", March 2016, 2016 Symposium on Colossal Data Analysis and Networking (CDAN).

[7] Rincy Jose, Varghese S Chooralil, "Prediction of election result by enhanced sentiment analysis on Twitter data using Word Sense Disambiguation", November 2015, 2015 International Conference on Control Communication Computing India (ICCC).

[8] Umar Farooq, Tej Prasad Dhamala, Antoine Nongaillard, Yacine Ouzrout, Muhammad Abdul Qadir, "A word sense disambiguation method for feature level sentiment analysis", December 2015, 2015 9th International Conference on Software, Knowledge, Information Management and Applications (SKIMA).

[9] Lesk Algorithm-Wikipedia

[10] Bhumika Gupta, Monika Negi, Kanika Vishwakarma, Goldi Rawat, Priyanka Badhani, "Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python", May 2017,International Journal of Computer Applications.

[11] Support Vector Machines(SVM) — An Overview

[12] Understanding Logistic Regression, (https://www.geeksforgeeks.org/understandinglogistic- regression/)

[13] Shadi Diab, Al-Quds Open University, Ramallah, Palestine, "Optimizing Stochastic Gradient Descent in Text ClassificationBased on Fine-Tuning Hyper-Parameters Approach.", December 2018,International Journal of Computer Science and Information Security (IJCSIS).

[14] Fenna Miedema, Prof. dr. Sandjai Bhulai, "Sentiment Analysis with Long Short-Term Memory networks" , 2018, Vrije Universiteit Amsterdam.

[15] Mochamad Ibrahim, Omar Abdillah, Alfan F. Wicaksono and Mirna Adriani, "Buzzer Detection and Sentiment Analysis for Predicting Presidential Election Results in a Twitter Nation", November 2015, IEEE International Conference on data mining workshop(ICDMW).

[16] Ajay Deshwal and Sudhir Kumar Sharma, "Twitter Sentiment Analysis using Various Classification Algorithms", September 2016, International Conference on

Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO).
[17] Dan Li, Jiang Qian, ”Text sentiment analysis based on long short-term memory”, December 2016, 2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI).
[18] Khin Zezawar Aung, Nyein Nyein Myo, ”Sentiment analysis of students’ comment using lexicon based approach”, June 2017, 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS).
[19] Diego Tumitan, Karin Becker, ”Sentiment-based Features for Predicting Election Polls: a Case Study on the Brazilian Scenario”, August 2014, 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT).