**1. INTRODUCTION**

**1.1 Project Overview**

**Machine Learning**

**Machine learning** is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning** focuses on the development of computer programs that can access data and use it learn for themselves.

Machine learning algorithms are often categorized as supervised or unsupervised.

* **Supervised machine learning algorithms**can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.
* **Unsupervised machine learning algorithms**are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn’t figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

**Applications of Machine Learning:**

* Machine Learning is a key technology in the development of autonomous systems.
* In addition to driver less cars, Machine Learning is also used in collaborative robots.
* Automated diagnostic procedures.
* Credit card fraud detection.
* Analysis of the stock market.

**About Online News Content:**

In the recent years, online content has been playing a significant role in manipulating users’ decisions and opinions. In fact, fake news and misleading articles is one form of opinion spam. Some of the biggest sources of spreading fake news or rumors are social media websites such as Facebook, Twitter etc.,

**Fake News:**

Fake news is a type of news that consists of deliberate disinformation spread via broadcast news media or online social media. Internet and social media made the access to the news information much easier and comfortable. Often internet users can follow the events of their interest in online mode, and spread of the mobile devices makes this process even easier.

Mass media has a huge influence on the society, there is someone who wants to take advantage of this. Sometimes to achieve some goals mass-media may manipulate the information in different ways. This leads to producing of the news articles that are not completely true or even completely false. There even exists a lot of websites that produce fake news.

**1.2 Project Scope**

Fake news is a phenomenon which is having a significant impact on our social life, in particular in the political world. Research on fake news detection is still at an early stage, as this is a relatively recent phenomenon, at least regarding the interest raised by the society.

Many scientists believe that fake news issue may be addressed by means of machine learning and artificial intelligence. There is a reason for that: recently artificial intelligence algorithms have started to work much better on lots of application problems because hardware is cheaper and bigger datasets are available. Detecting fake news is believed to be a complex task and much harder than detecting fake product reviews given that they spread easily using social media and word of mouth.

Even though the problem of fake news is not a new issue, detecting fake news is

believed to be a complex task given that humans tend to believe misleading information

and the lack of control of the spread of fake content[1]. Fake news has been getting more attention in the last couple of years, especially since the US election in 2016. It is tough for humans to detect fake news. It can be argued that the only way for a person to manually identify fake news is to have a vast knowledge of the covered topic. Even with the knowledge, it is considerably hard to successfully identify if the information in the article is real or fake.

**2. LITERATURE SURVEY**

Research on fake news detection is still at an early stage. We review some of the published work in the following.

**Rubin** et al. [2] discuss three types of fake news. Each is a representation of inaccurate or deceptive reporting. Furthermore, the authors weigh the different kinds of fake news and the pros and cons of using different text analytics and predictive modeling methods in detecting them. In this paper, they separated the fake news types into three groups:

* Serious fabrications are news not published in mainstream or participant media, yellow press or tabloids, which as such, will be harder to collect.
* Large-Scale hoaxes are creative and unique and often appear on multiple platforms. The authors argued that it may require methods beyond text analytics to detect this type of fake news.
* Humorous fake news, are intended by their writers to be entertaining, mocking, and even absurd. According to the authors, the nature of the style of this type of fake news could have an adverse effect on the effectiveness of text classification techniques.

**Horne** et al.[3] illustrated how obvious is to distinguish between fake and honest articles. According to their observations, fake news titles have fewer stop-words and nouns, while having more nouns and verbs. They extracted different features grouped into three categories as follows :

* Complexity features calculate the complexity and readability of the text.
* Psychology features illustrate and measure the cognitive process and personal concerns underlying the writings, such as the number of emotion words and casual words.
* Stylistic features reflect the style of the writers and syntax of the text, such as the number of verbs and the number of nouns.

**Wang** et al.[4] introduced LIAR, a new dataset that can be used for automatic fake news detection. Thought LIAR is considerably bigger in size, unlike other datasets, this data set does not contain full articles, it contains 12800 manually labeled short statements from politicalFact.com.

**Rubin** et al.[5] proposed a model to identify satire and humor news articles. They examined and inspected 360 Satirical news articles in mainly four domains, namely, civics, science, business, and what they called “soft news” (‘entertainment/gossip articles’). They proposed an SVM classification model using mainly five features developed based on their analysis of the satirical news. The five features are Absurdity, Humor, Grammar, Negative Affect, and Punctuation. Their highest precision of 90% was achieved using only three combinations of features which are Absurdity, Grammar, and Punctuation.

**3. PROBLEM ANALYSIS**

**3.1 Existing System**

* Research on fake news detection is still at an early stage, as this is a relatively recent phenomenon, at least regarding the interest raised by society.
* The fake news is classified into 3 types:
  + Serious fabrications are news not published in mainstream or participant media, yellow press or tabloids, which as such, will be harder to collect.
  + Large-Scale hoaxes are creative and unique and often appear on multiple platforms.
  + Humorous fake news, are intended by their writers to be entertaining, mocking, and even absurd.

**3.2 Proposed System**

* In this proposed system, we have used machine learning algorithms to classify fake news articles.
* For feature selection we have used TF-IDF technique.
* Six different machine learning algorithms, namely, Stochastic Gradient Descent (SGD), Support Vector Machines (SVM), Linear Support Vector Machine (LSVM), Naive-Bayes, Random Forest and Logistic Regression are implemented.
* We have performed parameter tuning by implementing GridSearchCV methods on these candidate models and chose best performing parameters for these classifiers.
* Finally, selected model was used for calculation of probability of truth.

**Data Pre-Processing:**

Before representing the data using n-gram model, the data need to be subjected to certain improvement like tokenization, stop-word removal. This will help us to reduce the size of actual data by removing the irrelevant information that is present in that data.

**Stop Word Removal:**

Stop words are unimportant words in a language that will create noise when used as features in text classification. Articles, prepositions and conjunctions and some pronouns are considered stop words. Those words were removed from each document, and the processed documents were stored and passed on to the next step.

**Stemming:**

Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”.

**Feature Extraction:**

There is a large number of terms, words and phrases in documents that lead to computational burden for the learning process. Thus, it is best to perform feature reduction or reduce the text feature size and avoid large feature space dimension.

In this we are using Term Frequency-Inverted Document Frequency (TF-IDF) method for feature selection.

**TF-IDF:**

* **TF:** Term Frequency, which measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear in long documents than shorter ones. Thus, the term frequency is often divided by the document length.
* **TF(t) = (number of times term t appears in a document) / (total number of terms in the document).**
* **IDF:** Inverse Document Frequency, which measures how important a term is. While computing TF, all terms are considered equally important. However, it is known that certain terms, such as “is”, “of”, “that”, etc., may appear a lot of times but have little importance. Thus, we need to weigh down the frequent terms while scale up the rare ones, by computing the following:
* **IDF(t) =**

**log\_e(total number of documents / number of documents with term t in it)**.

**Logistic Regression**

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems (problems with two class values). Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It’s an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

where

e = the natural logarithm base (Euler’s number),

x0 = the x-value of the sigmoid's midpoint,

L = the curve's maximum value, and

k = the logistic growth rate or steepness of the curve.

**NAÏVE BAYES:**

It makes the assumption that the occurrence of a certain feature is independent of the occurrence of other features. It gives us a method to calculate the conditional probability i.e., the probability of an event based on previous knowledge available on the events.

**Bayes Theorem** is stated as probability of the event B given A is equal to the probability of the event A given B multiplied by the probability of A upon probability of B.

**P(A/B) = P(B/A)\*P(A)/P(B)**

* In a Machine Learning classification probability, there are multiple features and classes, say C1,C2,……..,Ck. The main aim in the NAÏVE BAYES algorithm is to calculate the conditional probability of an object with a feature vector X1,X2,…..,Xn belongs to a particular class Ci.

**P(Ci/X1,X2,….,Xn) = P(X1,X2,….,Xn/Ci)\*P(Ci)/P(X1,X2,….,Xn)**

**4. SYSTEM ANALYSIS**

**4.1 System requirement specifications**

A requirement is a feature that the system must have or a constraint that it must to be accepted by the client. Requirement engineering aims at defining the requirements of the system under construction. Requirement engineering include two main activities, requirement elicitation, which results in the specification of the system that the client understands, and analysis which in analysis model that the developer can unambiguously interpret. A requirement is a statement about what the proposed system will do. Requirements can be divided into two major categories: Functional requirements and Non-Functional requirements.

**4.1.1 Functional Requirements**

Functional requirements describe the interactions between the system and its environment independent of its implementation. The environment includes the user and any other external system with which the system interacts. Functional requirements capture the intended to behavior of the system, this behavior may be expressed as services, tasks or functions the system is required to perform.

* Data crawling is done from trusted sources.
* Pre-processing the data that is collected.
* Feature selection is done using n-grams technique.
* The extracted features are fed into different classifiers.

**4.1.2 Non-Functional Requirement**

Non-functional requirements describe the aspects of the system that are not directly related to the functional behavior of the system. Non-functional requirements include a broad variety of requirements that apply to many different aspects of the system, from usability to performance.

**Accuracy:** Which results how much faith is there in the particular article.

**Usability:** It is easy is to use and operate the system.

**Performance:** It performs fast using several algorithms and helps in detecting the unfaithful news.

**Response Time:** The response of the system is rapid.

**4.2 Feasibility Study**

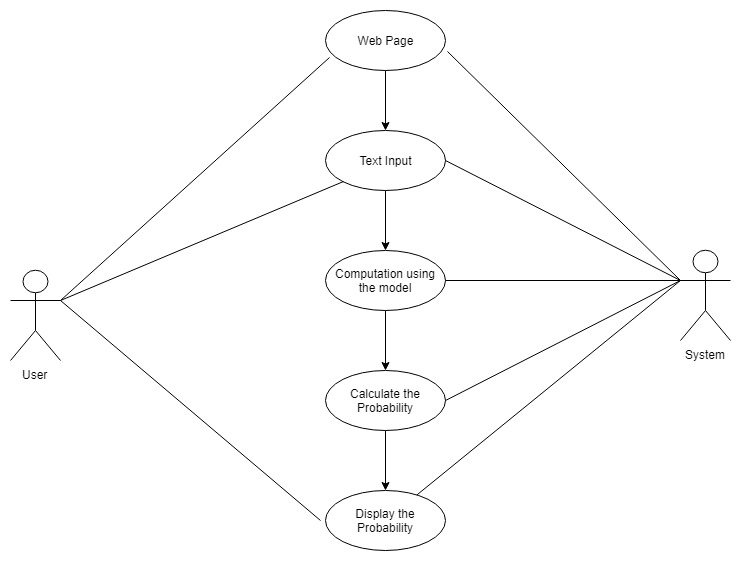
* 1. **Use Case Scenarios**

**4.3.1 Use-Case Diagrams**

An important part of the Unified Modeling Language (UML) is the facilities for drawing use case diagrams. Use cases are used during the analysis phase of a project to identify and partition system functionality. They separate the system into actors and use cases. Actors represent roles that can play by users of the system. Those users can be humans, other computers, pieces of hardware, or even other software systems. Use cases describe the behavior of the system.

**Table 4.1 Graphical Notations for Use Case Diagram**

|  |  |  |
| --- | --- | --- |
| Actor | An Actor as mentioned is a user of the system and is depcited using a stick figure. The role of the user is written beneath the icon. Actors are not limited to humans.If a system communicates with another application and expects input or delivers output then that application can also be considered as an actor. |  |
| Usecase | A Use Case is the functionality provided by the system typically described as verb+ object (eg: Register Car, Delete User). Use Cases are depicted with an ellipse. The name of the Use Case Is written within the ellipse. |  |
| Directed Association | Associations are used to link Actors with use cases and indicates that an actor participates in the Use Case in some form. Directed Association is same as association but difference is that it represented by a line having an arrow head. |  |
| System boundary boxes | You can draw a rectangle around the use cases, called the system boundary box, to indicate the scope of your system. Anything within the box represents functionality that is in scope and anything outside the box is not. |  |

****

***Fig 4.1 Use Case Diagram***

* 1. **System Requirements**

**4.4.1 Hardware Requirements**

* System : Dual Core or above
* Speed : 2.4GHz minimum
* Hard Disk : 250GB minimum
* RAM : 2GB or more

**4.4.2 Software Requirements**

* Windows 8/10
* Python – 3.XX
* Active Internet Connection

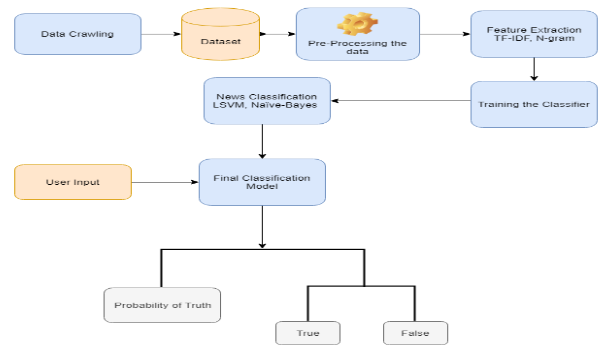
**5. SYSTEM DESIGN**

**5.1 Introduction**

Systems design is the process of defining the architecture components, modules, interfaces, and data for a system to satisfy specified requirements. Systems design could be seen as the application of systems theory to product development.

**5.1.1 Data Flow Diagram**

Data flow diagram is a way of representing a flow of data of a process or a system. A data-flow diagram has no control flow, there are no decision rules and no control flow.

****

***Fig 5.1 Data Flow Diagram***

**5.1.2 Sequence Diagram**

 A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

A sequence diagram shows, as parallel vertical lines, different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur.

***Table 5.1 Graphical Notations for Sequence Diagram***

|  |  |  |
| --- | --- | --- |
| Object | Objects are instances of classes and are arranged horizontally. The pictorial representation for an Object is class (a rectangle) with the name prefixed by the object name (optional). |  |
| Actor | Actor can also communicated with objects so they too can be listed as a column.An Actor is modeled using the stick figure. |  |
| Lifeline | The Lifeline identifies the existene of the object over time. The notation for a life time is a vertical dotted line etending from an object. |  |
| Activation | Activation modeled as rectangular boxes on the lifeline indicate when the object is performing an action. |  |
| Message | Messages modeled as horizontal arrows between Activations indicate the communication between the objects. |  |

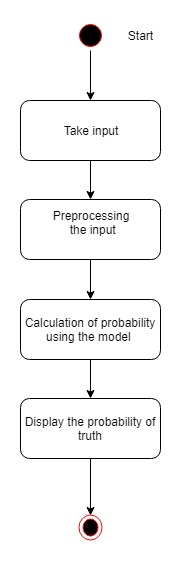
**5.1.3 Activity Diagram**

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 are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.

***Table 5.2 Graphical Notations for Activity Diagram***

|  |  |  |
| --- | --- | --- |
| Action | An action state represents a single step within an activity that is one not further decomposed of individual elements that are actions. Action states have sets of incoming and outgoing edges that specify control flow and data flow from and to other nodes. |  |
| Initial state | An initial node is a control node at which flow starts when the activity is invoked. An activity may have more than one initial state. Initial sate in activity is represented as filled circle. |  |
| Fork | For the branching of flows in two or more parallel flows we use a synchronization bar, which is depicted as a thick horizontal or vertical line. |  |
| Control Flow | A control flow is an edge that starts an activity node after the previous one is finished. |  |
| Final state | An activity may have more than one final node. The first one reached stops all flow in the activity. |  |

The following is the activity diagram for online fake news detection.



***Fig 5.2 Activity Diagram***

**5.2 Algorithm Description**

**Input:** News Snippet.

**Output:** Probability of truth.

**Step 1:** Data Crawling

Crawling of news articles from trusted sources.

**Step 2:** Data Pre-Processing

We created a generic processing function to remove punctuation and non-letter characters for each document; then we lowered the letter case in the document.

**Step 2.1**: Stop Word Removal – Stop words are insignificant words in language that will create noise when used as features in text classification. Those words were removed from each document, and the processed documents were stored and passed on to the next step.

**Step 2.2:** Stemming - After tokenizing the data, the next step is to transform the tokens into a standard form. Stemming simply is changing the words into their original form, and decreasing the number of word types or classes in the data. We use stemming to make classification faster and efficient. We used Porter stemmer, which is the most commonly used stemming algorithm due to its accuracy.

**Step 3:** Feature Extraction

There is a large number of terms, words, and phrases in documents that lead to a high computational burden for the learning process. Irrelevant and redundant features can damage the accuracy and performance of the classifiers. Thus, it is best to perform feature reduction to reduce the text feature size and avoid large feature space dimension. We have used Term Frequency-Inverted Document Frequency (TF-IDF) as feature selection method.

**TF-IDF -** The Term Frequency-Inverted Document Frequency (TF-IDF) is a weighting metric often used in information retrieval and natural language processing. It is a statistical metric used to measure how important a term is to a document in a dataset. A term importance increases with the number of times a word appears in the document, however this is counteracted by the frequency of the word in the corpus. One of the main characteristics of IDF is it weights down the term frequency while scaling up the rare ones.

**Step 4:** Classifier

The last step in the classification process is to train the classifier. We investigated different classifiers to predict the class of the documents. We investigated specifically six different machine learning algorithms, namely, Stochastic Gradient Descent (SGD), Support Vector Machines (SVM), Linear Support Vector Machine (LSVM), Naive-Bayes, Random Forest and Logistic Regression. We used implementations of these classifiers from the Python Natural Language Toolkit (NTLK) and scikit-learn. We have performed parameter tuning by implementing GridSearchCV methods on these candidate models and chose the best performing parameters for these classifiers.

**Step 5:** Prediction

Finally, selected model was used for calculation of probability of truth.

**6. IMPLEMENTATION**

**6.1 Technology Description**

**About SCIKIT-LEARN**

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy. The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from the French Institute for Research in Computer Science and Automation in Rocquencourt, France, took leadership of the project and made the first public release on February the 1st 2010. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

As of 2019, scikit-learn is under active development.

Scikit-learn is largely written in Python, with some core algorithms written in Cython to achieve performance. Support vector machines are implemented by a Cython wrapper around LIBSVM; logistic regression and linear support vector machines by a similar wrapper around LIBLINEAR.

Scikit-learn (formerly scikits.learn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

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.

The vision for the library is a level of robustness and support required for use in production systems. This means a deep focus on concerns such as easy of use, code quality, collaboration, documentation and performance.

Although the interface is Python, c-libraries are leverage for performance such as numpy for arrays and matrix operations, LAPACK, LibSVM and the careful use of cython.

**Key Features**

The library is focused on modeling data. It is not focused on loading, manipulating and summarizing data. Some popular groups of models provided by scikit-learn include:

* **Clustering:** for grouping unlabeled data such as KMeans.
* **Cross Validation:** for estimating the performance of supervised models on unseen data.
* **Datasets:** for test datasets and for generating datasets with specific properties for investigating model behavior.
* **Dimensionality Reduction:** for reducing the number of attributes in data for summarization, visualization and feature selection such as Principal component analysis.
* **Ensemble methods:** for combining the predictions of multiple supervised models.
* **Feature extraction:** for defining attributes in image and text data.
* **Feature selection:** for identifying meaningful attributes from which to create supervised models.
* **Parameter Tuning:** for getting the most out of supervised models.
* **Manifold Learning:** For summarizing and depicting complex multi-dimensional data.
* **Supervised Models:** a vast array not limited to generalized linear models, discriminate analysis, naive bayes, lazy methods, neural networks, support vector machines and decision trees.

**About Natural Language Toolkit (NLTK)**

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).

It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

The Natural Language Toolkit is an open source library for the Python programming language originally written by Steven Bird, Edward Loper and Ewan Klein for use in development and education.

It comes with a hands-on guide that introduces topics in computational linguistics as well as programming fundamentals for Python which makes it suitable for linguists who have no deep knowledge in programming, engineers and researchers that need to delve into computational linguistics, students and educators.

**6.2 Sample Source Code**

**Sample Code for Data Preprocessing**

test\_filename = 'test.csv'

train\_filename = 'train.csv'

valid\_filename = 'valid.csv'

# reading dataset files

train\_news = pd.read\_csv(train\_filename)

test\_news = pd.read\_csv(test\_filename)

valid\_news = pd.read\_csv(valid\_filename)

#Stemming

def stem\_tokens(tokens, stemmer):

stemmed = []

for token in tokens:

stemmed.append(stemmer.stem(token))

return stemmed

#process the data

def process\_data(data,exclude\_stopword=True,stem=True):

tokens = [w.lower() for w in data]

tokens\_stemmed = tokens

tokens\_stemmed = stem\_tokens(tokens, eng\_stemmer)

tokens\_stemmed = [w for w in tokens\_stemmed if w not in stopwords ]

return tokens\_stemmed

#creating ngrams

#unigram

def create\_unigram(words):

assert type(words) == list

return words

#bigram

def create\_bigrams(words):

assert type(words) == list

skip = 0

join\_str = " "

Len = len(words)

if Len > 1:

lst = []

for i in range(Len-1):

for k in range(1,skip+2):

if i+k < Len:

lst.append(join\_str.join([words[i],words[i+k]]))

else:

#set it as unigram

lst = create\_unigram(words)

return lst

# Porter stemming algorithm

porter = PorterStemmer()

**Sample Code for Feature Extraction**

#creating feature vector - document term matrix

countV = CountVectorizer()

train\_count = countV.fit\_transform(DataPrep.train\_news['Statement'].values)

#create tf-idf frequency features

tfidfV = TfidfTransformer()

train\_tfidf = tfidfV.fit\_transform(train\_count)

# Preprocessing

# Stop Word Removal

tfidf\_ngram = TfidfVectorizer(stop\_words='english',ngram\_range=(1,4),use\_idf=True,smooth\_idf=True)

#POS(Parts of Speech) Tagging

tagged\_sentences = nltk.corpus.treebank.tagged\_sents()

cutoff = int(.75 \* len(tagged\_sentences))

training\_sentences = DataPrep.train\_news['Statement']

**Sample Code for Classifier**

# Building classifier using logistic regression

logR\_pipeline\_ngram = Pipeline([('LogR\_tfidf',FeatureSelection.tfidf\_ngram), ('LogR\_clf',LogisticRegression(penalty="l2",C=1))])

logR\_pipeline\_ngram.fit(DataPrep.train\_news['Statement'],DataPrep.train\_news['Label'])

predicted\_LogR\_ngram = logR\_pipeline\_ngram.predict(DataPrep.test\_news['Statement'])

np.mean(predicted\_LogR\_ngram == DataPrep.test\_news['Label'])

#User defined functon for K-Fold cross validatoin

def build\_confusion\_matrix(classifier):

k\_fold = KFold(n=len(DataPrep.train\_news), n\_folds=5)

scores = []

confusion = np.array([[0,0],[0,0]])

for train\_ind, test\_ind in k\_fold:

train\_text = DataPrep.train\_news.iloc[train\_ind]['Statement']

train\_y = DataPrep.train\_news.iloc[train\_ind]['Label']

test\_text = DataPrep.train\_news.iloc[test\_ind]['Statement']

test\_y = DataPrep.train\_news.iloc[test\_ind]['Label']

classifier.fit(train\_text,train\_y)

predictions = classifier.predict(test\_text)

confusion += confusion\_matrix(test\_y,predictions)

score = f1\_score(test\_y,predictions)

scores.append(score)

return (print('Total statements classified:', len(DataPrep.train\_news)),

print('Score:', sum(scores)/len(scores)),

print('score length', len(scores)),

print('Confusion matrix:'),

print(confusion))

#logistic regression parameters

parameters = {'LogR\_tfidf\_\_ngram\_range': [(1, 1), (1, 2),(1,3),(1,4),(1,5)], 'LogR\_tfidf\_\_use\_idf': (True, False), 'LogR\_tfidf\_\_smooth\_idf': (True, False) }

gs\_clf = GridSearchCV(logR\_pipeline\_ngram, parameters, n\_jobs=-1)

gs\_clf = gs\_clf.fit(DataPrep.train\_news['Statement'][:10000],DataPrep.train\_news['Label'][:10000])

gs\_clf.best\_score\_

gs\_clf.best\_params\_

gs\_clf.cv\_results\_

#saving best model to the disk

model\_file = 'final\_model.sav'

pickle.dump(logR\_pipeline\_ngram,open(model\_file,'wb'))

**Sample Code for Predection**

def detecting\_fake\_news(ip):

load\_model = pickle.load(open('final\_model.sav', 'rb'))

prediction = load\_model.predict([ip])

prob = load\_model.predict\_proba([ip])

return jsonify( statement=ip, result=str(prediction[0]), probability = prob[0][1] )

**7.TESTING**

Testing is the major quality control measure employed for software development. Its basic function is to detect errors in the software. During requirement analysis and design, the output is a document which is usually textual and non-textual. After the coding phase, computer programs are available that can be executed for testing purpose. This implies that testing has to uncover errors introduced during coding phases. Thus, the goal of testing is to cover requirement, design, or coding errors in the program. The purpose is to exercise the different parts of the module code to detect coding errors. After this, the modules are gradually integrated into subsystems, which are then integrated themselves to eventually form the entire system. During the module integration, testing is performed. The goal is to detect designing errors, while focusing the interconnection between modules. After the system was put together, system testing is performed. Here the system is tested against the system requirements to see if all requirements were met and the system performs as specified by the requirements. Finally, testing is performed to demonstrate to the client for the operation of the system.

For the testing to be successful, proper selection of the test case is essential. There are two different approaches for selecting test case. The software or the module to be tested is treated as a black box, and the test cases are decided based on the specifications of the system or module. For this reason, this form of testing is also called “black box testing”.

**7.1 Testing approach**

Testing is a process, which reveals the errors in a program. It is the major quality measure employed during software development. During testing, the program is executed with a set of conditions known as test cases and the output is evaluated to determine whether the program is performing as expected. In order to make sure that the system does not have errors, the different levels of testing strategies are applied at differing phases of software development are as follows.

**7.1.1 Unit Testing**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**7.1.2 Integration Testing**

Integration testing ensures that the software and the subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly or not when integrated together.

**7.1.3 System Testing**

It involves in-house testing of the entire system before the delivery to the user. Its aim is to satisfy the user and the system that meets all the requirements of the client's specifications.

**7.1.4 Acceptance Testing**

It is a pre-delivery testing in which the entire system is tested at the client's site on the real-world data to find errors.

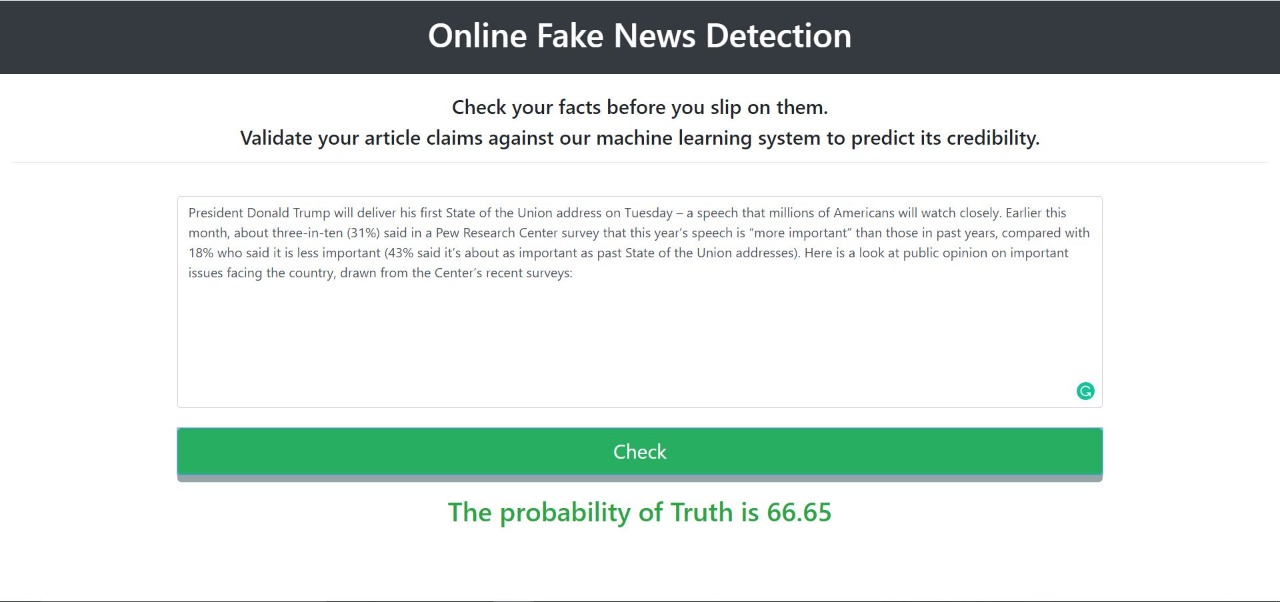
**7.1.5 Validation Testing**

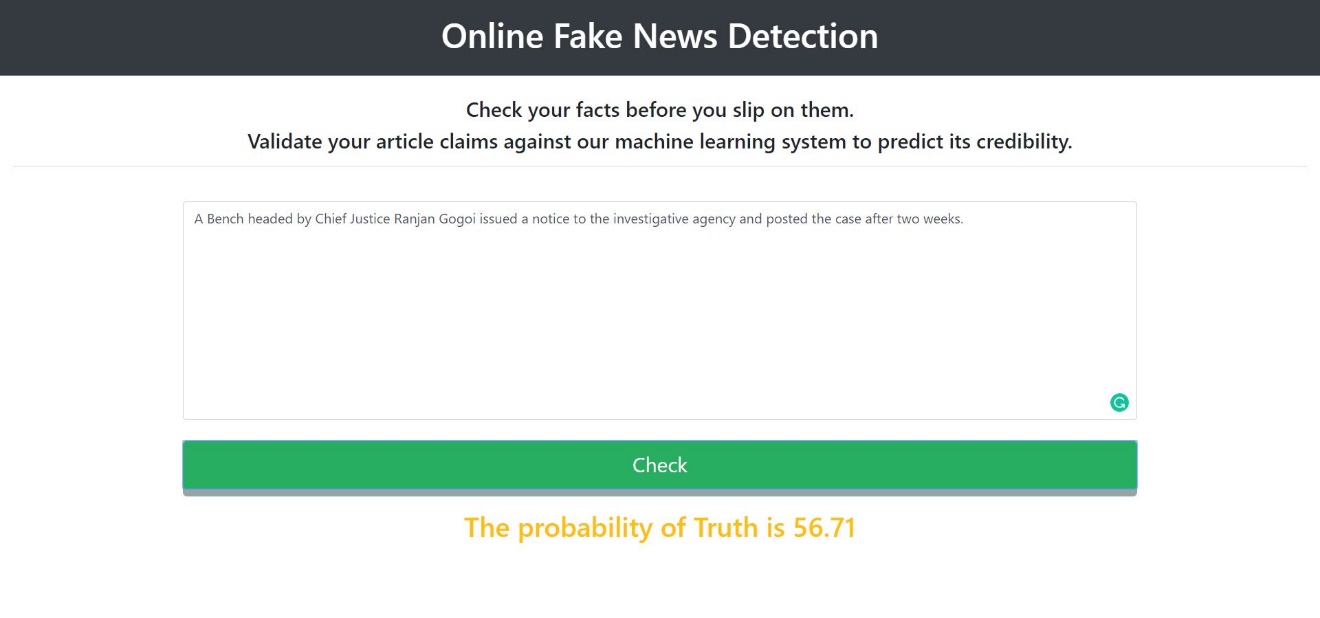
The system is tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

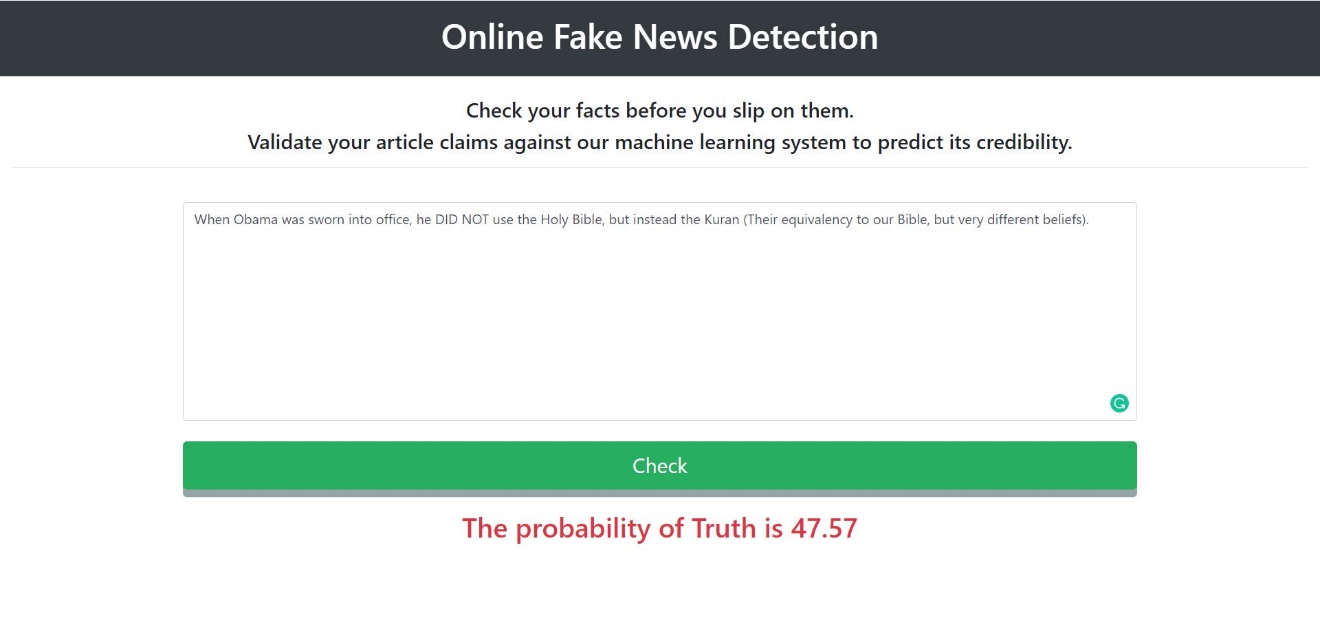
**7.2 Test cases**

**8. SCREEN SHOTS**

**8.1 Showing the probability of truth of a news article**

****

****

****

**9. CONCLUSION**

Most of the popular fake news stories were more widely shared on Social Media platforms. A sizable number of people who read fake news stories have reported that they believe them more than news from mainstream media.

Using the proposed system we can detect the fake news using different Machine Learning techniques. In this paper, we have presented a detection model for fake news using n-gram analysis through different features extraction techniques.

Fake news detection is an emerging research area with few public datasets. We run our model on an existing dataset, showing that our model outperforms the original approach published by the authors of the dataset.

**10. BIBLOGRAPHY**

**[1]** Lemann, N.: Solving the Problem of Fake News. The New Yorker (2017). http://www.newyorker.com/news/news-desk/solving-the-problem-of-fake-news

**[2]** Rubin, V.L., Chen, Y., Conroy, N.J.: Deception detection for news: three types of fakes. In: Proceedings of the 78th ASIS&T Annual Meeting: Information Science with Impact: Research in and for the Community (ASIST 2015). Article 83, p. 4, American Society for Information Science, Silver Springs (2015).

**[3]** Horne, B.D., Adali, S.: This just in: fake news packs a lot in title, uses simpler, repetitive content in text body, more similar to satire than real news. In: the 2nd International Workshop on News and Public Opinion at ICWSM (2017).

**[4]** Wang, W.Y.: Liar, Liar Pants on fire: a new Benchmark dataset for fake news detection. arXiv preprint (2017).

**[5]** Rubin., Victoria, L., et al.: Fake news or truth? Using satirical cues to detect potentially misleading news. In: Proceedings of NAACL-HLT (2016).