A Real Time Project Report

SMS SPAM MESSAGE DETECTION USING “**MACHINE LEARNING AND DEEP LEARNING TECHNIQUES”**

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

**BACHELOR OF TECHNOLOGY**

In

**CSE (DATA SCIENCE)**

By

**B.RAM CHARAN 22AG1A6772**

**HARINI 23AG5A6707**

**V.PRABHAS 22AG1A67C5**

Under the guidance of

ASHWANI ATTRI

ASST PROFFESOR



**DEPARTMENT OF CSE (DATA SCIENCE)**

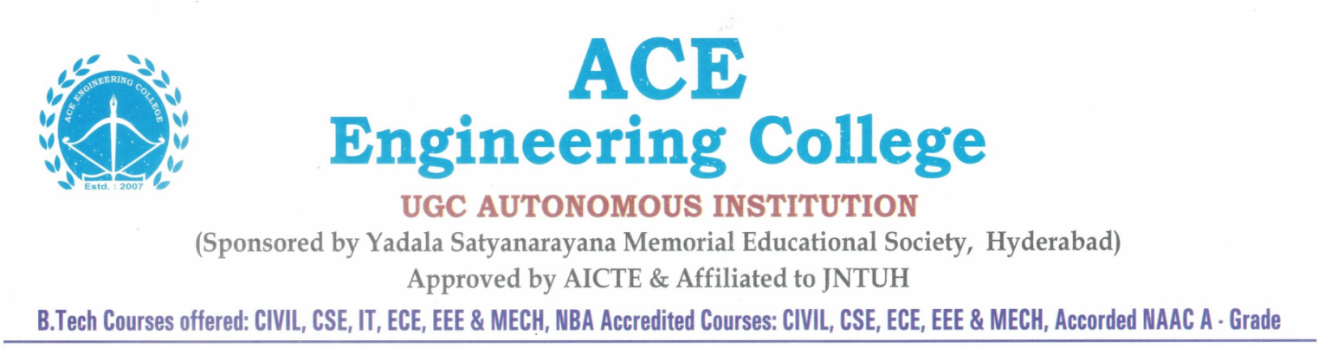
**ACE Engineering College**

**Ankushapur(V), Ghatkesar(M), Medchal Dist - 501 301**

***(An Autonomous Institution, Affiliated to JNTUH, Hyderabad)***

www.aceec.ac.in

**2024-2025**



**DEPARTMENT OF CSE (DATA SCIENCE)**

**CERTIFICATE**

This is to certify that the Societal Related (Or) Real Time project report entitled “project title” is a bonafide work done by **BACHU RAM CHARAN,HARINI,** **VUDGULA SAI PRABHAS** bearing **22AG1A6772,23AG5A6707,22AG1A67C5** in partial fulfillment for the award of Degree of BACHELOR OF TECHNOLOGY in CSE (Data Science) from JNTUH University, Hyderabad during the academic year 2024- 2025. This record of bonafide work carried out by them under our guidance and supervision.

The results embodied in this report have not been submitted by the student to any other University or Institution for the award of any degree or diploma*.*

**(Ashwani Attri) (Dr. P. Chiranjeevi)**

Assistant Professor Associate Professor

Supervisor H.O.D., CSE-DS.

**ACKNOWLEDGEMENT**

We would like to express my gratitude to all the people behind the screen who have helped me transform an idea into a real time application.

We would like to express my heart-felt gratitude to my parents without whom we would not have been privileged to achieve and fulfill my dreams.

A special thanks to our General Secretary, **Prof. Y. V. Gopala Krishna Murthy,** for having founded such an esteemed institution. Sincere thanks to our Joint Secretary **Mrs. M. Padmavathi,** for support in doing project work. I am also grateful to our beloved principal, **Dr. B. L. RAJU** for permitting us to carry out this project.

We profoundly thank **Dr. P. Chiranjeevi**, Associate Professor and Head of the Department of Computer Science and Engineering (Data Science), who has been an excellent guide and also a great source of inspiration to my work.

We extremely thank **Mrs. B. Saritha** and **Mr. M. Hari Krishna**, Assistant Professors, Project coordinators, who helped us in all the way in fulfilling of all aspects in completion of our Mini-Project.

We are very thankful to my internal guide (your guide name) who has been an excellent and also given continuous support for the Completion of my project work.

The satisfaction and euphoria that accompany the successful completion of the task would be great, but incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the efforts with success. In this context, I would like to thank all the other staff members, both teaching and non-teaching, who have extended their timely help and eased my tasks

**B.RAM CHARAN 22AG1A6772**

**HARINI 23AG5A6705**

**V.PRABHAS 22AG1A67C5**

SMS SPAM MESSAGE DETECTION USING **“MACHINE LEARNING AND DEEP LEARNING TECHNIQUES”**

**ABSTRACT**

The number of people using mobile devices increasing day by day. SMS (short message service) is a text message service available in smartphones as well as basic phones. So, the traffic of SMS increased drastically. The spam messages also increased. The hackers try to send spam messages for their financial or business benefits like market growth, lottery ticket information, credit card information, etc. So, spam classification has special attention. In this paper, we applied various machine learning and deep learning techniques for SMS spam detection. we used a dataset to train the machine learning and deep learning models like LSTM and NB. The SMS spam collection data set is used for testing the method. The dataset is split into two categories for training and testing the research. Our experimental results have shown that our NB model outperforms previous models in spam detection with an accuracy of good.

**CONTENT**

**S.NO CHAPTER NAME PAGE NO**

**1 INTRODUCTION 1**

**2 EXISTING SYSTEM & DRAWBACKS 2**

**3 LITERATURE SURVEY 3**

**4 PROPOSED MODEL / SYSTEM 4**

**5 REQUIREMENTS GATHERING 5**

**5.1 Software Requirements &**

**Hardware Requirements**

**5.2 Functional Requirements &**

**Non-Functional Requirements**

**5.3 Data Collection**

**6 SYSTEM ANALYSIS & DESIGN 6**

**6.1 Module Description**

**6.2 System Architecture**

**6.3 UML Diagrams**

**7 IMPLEMENTATION 7**

**7.1 Algorithm Used**

**7.2 Sample Code**

**8 TESTING 8**

**9 RESULTS 9**

**10 CONCLUSION 10**

**FUTURE WORK 11**

**REFERENCES 12**

**LIST OF FIGURES**

**F.NO FIGURE NAME PAGE NO**

**2.1** **Source code for Existing system 3**

**6.2 Architecture Diagram 10**

**6.3.1 Data Flow Diagram 10**

**6.3.2 Use case Diagram 11**

**6.3.3 Class Diagram 11**

**6.3.4 Sequence Diagram 12**

**6.3.5 Activity Diagram 12**

**6.3.6 State flow Diagram 12**

**1.INTRODUCTION**

The increasing mobile phones become one of the attached companions for many individuals. With the explosive penetration of mobile devices and millions of people sending messages every day, Short Message Service (SMS) has become a multi-million-dollar commercial industry with a value between 11.3 to 24.7 percent of the developing countries‘ Gross National Income (GNI) in the early year of 2013.

As the utilization of mobile phone devices has become commonplace, Short Message Service (SMS) has grown into a multi-billion dollars commercial industry [2]. SMS is a text communication platform that allows mobile phone users to exchange short text messages (usually less than 160 seven-bit characters). It is the most widely used data application with an estimated 3.5 billion active users, or about 80% of all mobile phone subscribers at the end of 2010 [3]. As the popularity of the platform has increased, we have seen a surge in the number of unsolicited commercial advertisements sent to mobile phones using text messaging. SMS spam is still not as common as email spam, where in 2010 around 90% of emails was spam, and in North America it is still not a major problem, contributing to less than 1% of text messages exchanged as of December 2012.

The spam increased in these days due more mobile devices deployed in environment for e-mail and message communication. Currently, 85% of mails and messages received by mobile users are spam. The cost of mails and messages are very low for senders but high for receipts of these messages. The cost paid some time by service providers and the cost of spam can be measured in the loss of human time and loss of important messages or mails. Due to these spam mails and messages, the values able e-mails and messages are affected because each user have limited Internet services, short time, and memory

We proposed a spam detection method using machine learning algorithms such as NB (naïve Bayes) and LSTM for classification of ham and spam messages. The SMS spam collection dataset was considered for testing of the current research. The dataset was divided into two categories: 30% for testing and 70% for training purpose for the predictive models. The evaluation metrics for performance such as specificity, accuracy, and sensitivity were considered evaluating the proposed study. The results obtained from experiments confirmed that the proposed research achieved high accuracy.

**2.EXISTING SYSTEM & DRAWBACKS**

Random Forest (RF) algorithm will used for classification of ham or spam during this phase. RF is averaging ensemble learning method that can be used for classification problem. This algorithm combines various decision tree models in order to eliminate the over fitting problem in decision trees. In RF algorithm, each tree is capable in providing its own prediction results, different from each other. As a result, each tree gives different performances, in which the average of their performances will be generalized and calculated. During the training phase, a set of decision trees will be constructed before they can operate on randomly selected features. Regardless, RF can work well with a large dataset with a variety of feature types, similar to binary, categorical and numerical. The algorithm works as follows diagram: for each tree in the forest, a bootstrap sample is selected from S where S (i) represents the ith bootstrap. A decision-tree is then learn using a modified decision-tree learning algorithm.

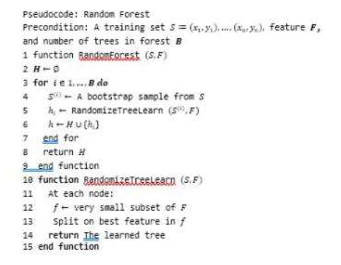


Fig 2.1 Source code for Existing system

The algorithm is modified as follows: at each node of the tree, instead of examining all possible feature-splits, some subset of the features text f ⊆ F is selected randomly. where F is the set of Spam features. The node then splits on the best feature in f rather than F. In practice f is much, much smaller than F. Deciding on which feature to split is oftentimes the most computationally expensive aspect of decision tree learning. By narrowing the set of features, the speed up the learning of the tree is increase drastically.

**DISADVANTAGES OF EXISTING SYSTEM**

* Accuracy is low
* Dataset selection is not correct
* Feature extraction is not accurate

**3.LITERATURE SURVEY**

**TITLE: SMS Spam Detection Based on Long Short-Term Memory and Gated Recurrent Unit**

Author: Pumrapee Poomka, Wattana Pongsena, Nittaya Kerdprasop, and Kittisak Kerdprasop

YEAR: - 2019

Abstract:

An SMS spam is the message that hackers develop and send to people via mobile devices targeting to get their important information. For people who are ignorant, if they follow the instruction in the message and fill their important information, such as internet 19 banking account in a faked website or application, the hacker may get the information. This may lead to loss their wealth. The efficient spam detection is an important tool in order to help people to classify whether it is a spam SMS or not. In this research, we propose a novel SMS spam detection based on the case study of the SMS spams in English language using Natural Language Process and Deep Learning techniques. To prepare the data for our model development process, we use word tokenization, padding data, truncating data and word embedding to make more dimension in data. Then, this data is used to develop the model based on Long Short-Term Memory and Gated Recurrent Unit algorithms. The performance of the proposed models is compared to the models based on machine learning algorithms including Support Vector Machine and Naïve Bayes. The experimental results show that the model built from the Long Short-Term Memory technique provides the best overall accuracy as high as 98.18%. On accurately screening spam messages, this model shows the ability that it can detect spam messages with the 90.96% accuracy rate, while the error percentage that it misclassifies a normal message as a spam message is only 0.74%.

**TITLE: SMS Spam Detection using Machine Learning and Deep Learning Techniques**

Author: Sridevi Gadde

YEAR: - 2021

Abstract:

The number of people using mobile devices increasing day by day.SMS (short message service) is a text message service available in smartphones as well as basic phones. So, the traffic of SMS increased drastically. The spam messages also increased. The spammers try to send spam messages for their financial or business benefits like market growth, lottery ticket information, credit card information, etc. So, spam classification has special attention. In this paper, we applied various machine learning and deep learning techniques for SMS spam detection. we used a dataset from UCI and 21 build a spam detection model. Our experimental results have shown that our LSTM model outperforms previous models in spam detection with an accuracy of 98.5%. We used python for all implementations.

**4.PROPOSED MODEL / SYSTEM**

Applying NB algorithm to the dataset using extracted features with different training set sizes. The performance in learning curve is evaluated by splitting the dataset into 70% training set and 30% test set. The NB algorithm shows good overall accuracy.

We notice that the length of the text message (number of characters used) is a very good feature for the classification of spams. Sorting features based on their mutual information (MI) criteria shows that this feature has the highest MI with target labels. Additionally, going through the misclassified samples, we notice that text messages with length below a certain threshold are usually hams, yet because of the tokens 22 corresponding to the alphabetic words or numeric strings in the message they might be classified as spams.

By looking at the learning curve, we see that once the NB is trained on features extracted, the training set error and test set error are close to each other. Therefore, we do not have a problem of high variance, and gathering more data may not result in much improvement in the performance of the learning algorithm. As the result, we should try reducing bias to improve this classifier. This means adding more meaningful features to the list of tokens can decrease the error rate, and is the option that is explored next.

**5.REQUIREMENTS GATHERING**

**5.1.SOFTWARE REQUIREMENTS AND HARDWARE REQUIREMENTS**

**1.HARDWARE REQUIREMENTS:**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It shows what the system does and not how it should be implemented

PROCESSOR: Intel I5

RAM: 4GB

HARD DISK: 40 GB

**2. SOFTWARE REQUIREMENTS:**

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a 28 basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the team‘s and tracking the team‘s progress throughout the development activity.

PYTHON IDE: Anaconda Jupyter Notebook

PROGRAMMING LANGUAGE: Python

**5.2Functional Requirements & Non-Functional Requirements:**

**Functional Requirements**

**1. Data Collection and Preprocessing**

* **Collect Data:** Gather SMS messages from various sources.
* **Clean Data:** Remove unwanted characters and duplicate messages.
* **Label Data:** Mark messages as spam or not spam.
* **Split Data:** Divide data into training, validation, and testing sets.

**2. Feature Engineering**

* **Process Text:** Break down messages into words, remove common words, and reduce words to their root form.
* **Extract Features:** Convert text into numerical data using methods like Bag of Words, TF-IDF, or embeddings.

**3. Model Development**

* **Choose Models:** Use traditional ML models (like Naive Bayes or SVM) or DL models (like RNNs or Transformers).
* **Train Models:** Use the data to train these models.
* **Evaluate Models:** Measure model performance using metrics like accuracy and F1-score.

**4. Model Deployment**

* **Integrate Model:** Add the trained model to an application for real-time or batch processing.
* **Handle Traffic:** Ensure the system can process many messages quickly.
* **User Interface:** Create a simple interface for users to submit messages and see results.

**5. Monitoring and Maintenance**

* **Monitor Performance:** Keep track of how well the model is performing over time.
* **Update Model:** Regularly retrain the model with new data.
* **Feedback Loop:** Allow users to provide feedback to improve the model.

**6. Security and Privacy**

* **Data Privacy:** Protect user data and comply with privacy laws.
* **System Security:** Ensure the system is secure from unauthorized access.

**7. Documentation and Support**

* **Documentation:** Provide clear instructions for developers and users.
* **Support:** Offer help and support for users experiencing issues.

**Non-Functional Requirements**

**1. Performance**

* **Speed:** The system should process messages quickly.
* **Scalability:** The system should handle increasing numbers of messages without slowing down.

**2. Reliability**

* **Uptime:** The system should be available and operational most of the time.
* **Accuracy:** The system should consistently provide accurate spam predictions.

**3. Usability**

* **User-Friendly:** The interface should be easy to use for non-technical users.
* **Accessibility:** The system should be accessible from various devices (e.g., mobile, desktop).

**4. Maintainability**

* **Ease of Update:** The system should be easy to update with new models and data.
* **Modularity:** The system components should be designed so they can be easily modified or replaced.

**5. Security**

* **Data Protection:** Ensure all data is securely stored and transmitted.
* **Access Control:** Only authorized users should have access to the system and data.

**6. Compliance**

* **Regulatory Compliance:** Ensure the system adheres to relevant laws and regulations (e.g., GDPR).

**5.3 Data Collection**

The public dataset of SMS labelled messages is obtained from UCI Machine Learning Repository. The dataset considered in the current research is available on kaggle, a machine learning repository. This study finds that there are only 5,574 labelled messages in the dataset, with 4827 of messages belong to ham messages while the other 747 messages belong to spam messages. Nonetheless, this dataset consists of two named columns starting with the message labels (ham or spam) followed by strings of text messages and three unnamed columns. It‘s time for a data analyst to pick up the baton and lead the way to machine learning implementation. The job of a data analyst is to find ways and sources of collecting relevant and comprehensive data, interpreting it, and analyzing results with the help of statistical techniques.

The type of data depends on what you want to predict.

There is no exact answer to the question “How much data is needed?” because each machine learning problem is unique. In turn, the number of attributes data scientists will use when building a predictive model depends on the attributes’ predictive value.

‘The more, the better ‘approach is reasonable for this phase. Some data scientists suggest considering that less than one-third of collected data may be useful. It’s difficult to estimate which part of the data will provide the most accurate results until the model training begins. That’s why it’s important to collect and store all data — internal and open, structured and unstructured.

The purpose of preprocessing is to convert raw data into a form that fits machine learning. Structured and clean data allows a data scientist to get more precise results from an applied machine learning model. The technique includes data formatting, cleaning, and sampling.

**6.SYSTEM ANALYSIS & DESIGN**

**6.1 Module Description**

**Data Collection and Preprocessing:**

The **Data Collection and Preprocessing** module gathers SMS data from various sources, cleans and normalizes it by removing duplicates and irrelevant characters, and labels messages as spam or not spam. This prepared data is then divided into training, validation, and testing sets. The **Feature Engineering** module processes the text by breaking it into tokens, removing stop words, and applying stemming and lemmatization. It converts the text into numerical features using techniques like Bag of Words, TF-IDF, or embeddings such as Word2Vec and BERT.

**Model Development:**

In the **Model Development** module, various machine learning models (e.g., Naive Bayes, SVM) and deep learning models (e.g., RNNs, LSTMs, Transformers) are trained using the preprocessed data. This module includes extensive hyperparameter tuning to optimize model performance. The **Model Evaluation** module measures the accuracy, precision, recall, and F1-score of each model to ensure robustness and reliability.

The **Model Deployment** module integrates the trained model into a user-facing application, enabling real-time or batch processing of incoming SMS messages. This module ensures scalability to handle large volumes of data and provides a user-friendly interface for submitting messages and displaying results. The **Monitoring and Maintenance** module continuously tracks model performance, implementing a feedback loop for users to report inaccuracies, and periodically updates the model with new data to maintain accuracy.

**Security and Privacy**

Security and privacy are paramount, with the **Security and Privacy** module ensuring that user data is protected and handled in compliance with regulations like GDPR. This module also includes mechanisms to safeguard the system from unauthorized access. Comprehensive **Documentation and Support** are provided, detailing setup instructions, API usage, and offering channels for user support.

**6.2 System Architecture**

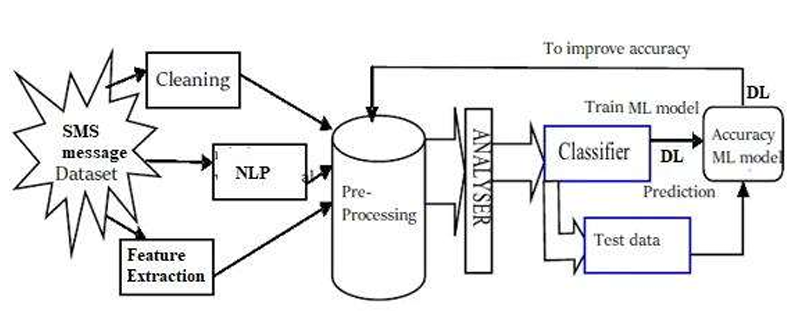


Fig. 6.2 Architecture Diagram

**6.3 UML Diagrams**

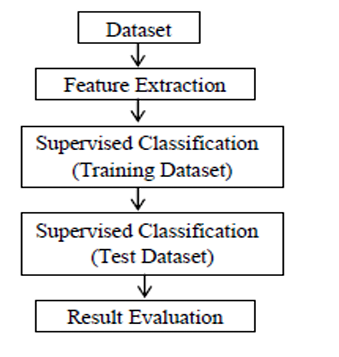


Fig 6.3.1 Data Flow Diagram

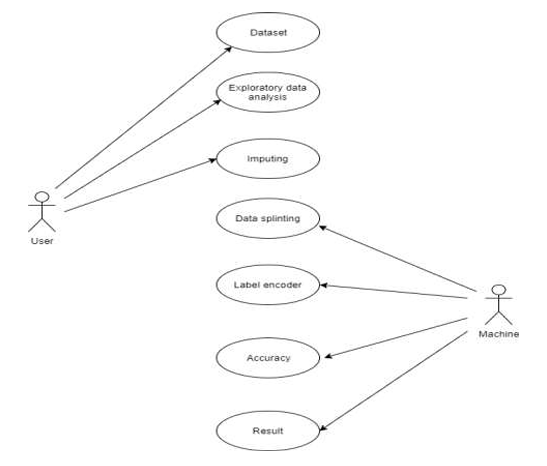


Fig 6.3.2 Usecase Diagram

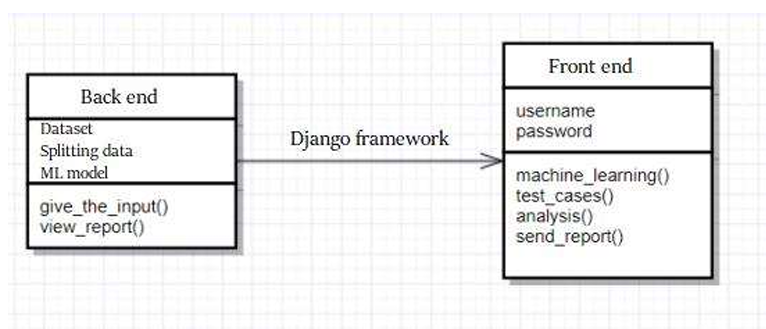


Fig 6.3.3 Class Diagram

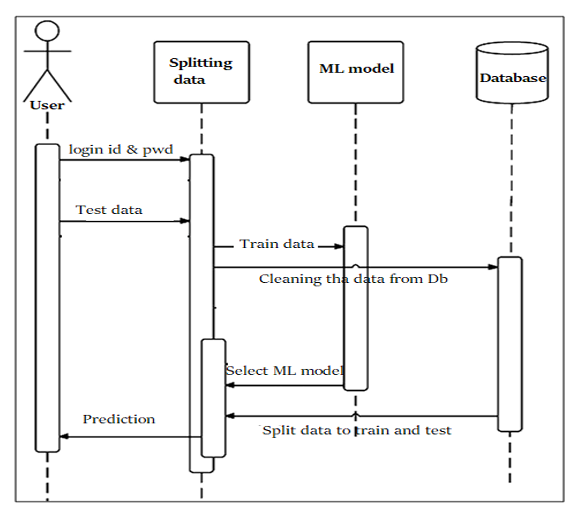


Fig 6.3.4 Sequence Diagram

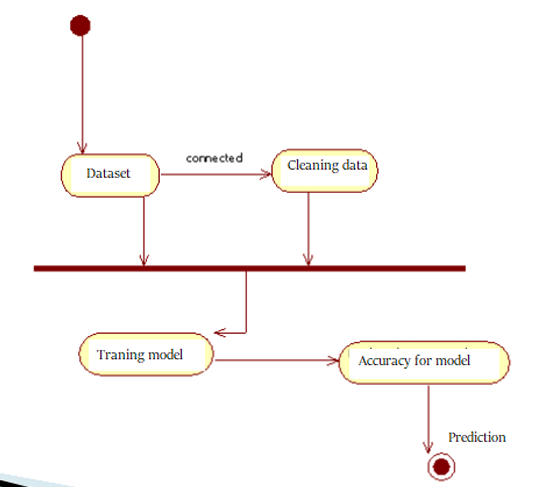


Fig 6.3.5 Activity Diagram

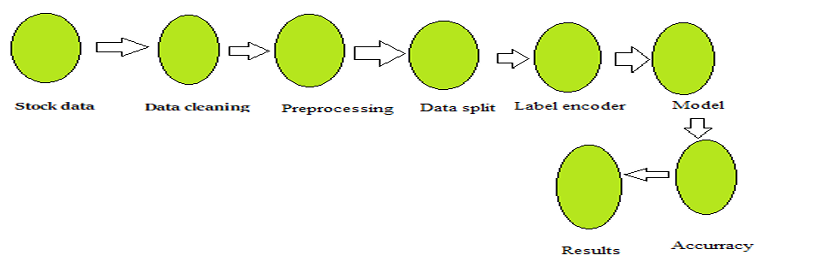


Fig 6.3.6 State Flow Diagram

**7.IMPLEMENTATION**

**7.1Algorithm Used**

Applying NB algorithm to the dataset using extracted features with different training set sizes. The performance in learning curve is evaluated by splitting the dataset into 70% training set and 30% test set. The NB algorithm shows good overall accuracy.

We notice that the length of the text message (number of characters used) is a very good feature for the classification of spams. Sorting features based on their mutual information (MI) criteria shows that this feature has the highest MI with target labels. Additionally, going through the misclassified samples, we notice that text messages with length below a certain threshold are usually hams, yet because of the tokens corresponding to the alphabetic words or numeric strings in the message they might be classified as spams.

By looking at the learning curve, we see that once the NB is trained on features extracted, the training set error and test set error are close to each other. Therefore, we do not have a problem of high variance, and gathering more data may not result in much improvement in the performance of the learning algorithm. As the result, we should try reducing bias to improve this classifier. This means adding more meaningful features to the list of tokens can decrease the error rate, and is the option that is explored next.

**7.2 Sample Code**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import keras**

**from sklearn.model\_selection import train\_test\_split**

**import seaborn as sns**

**from wordcloud import WordCloud**

**from sklearn.preprocessing import LabelEncoder**

**import nltk**

**import re**

**from nltk.corpus import stopwords**

**from nltk.stem import WordNetLemmatizer**

**from sklearn.feature\_extraction.text import TfidfVectorizer**

**from sklearn.metrics import accuracy\_score,confusion\_matrix**

**nltk.download('punkt')**

**d=pd.read\_csv(r'C:\Users\kbpra\Music\spam\_detection\Dataset\spam.csv',**

**encoding='latin-1')**

**d.head()**

**d**

**df = d.iloc[:,0:2].values**

**df = pd.DataFrame(df)**

**df.columns=['Class','Text']**

**df.head()**

**df.info()**

**df.isna().sum()**

**df.describe()**

**Visualization:**

**ns = df["Class"].isin(['ham']).sum(axis=0)**

**s = df["Class"].isin(['spam']).sum(axis=0)**

**label=['Spam','Not Spam']**

**a = [s,ns]**

**plt.pie(x=a,labels=label,autopct='%1.1f%%')**

**plt.legend()**

**plt.show()**

**plt.figure(figsize=(12,8))**

**ax =sns.barplot(x=label,y=a)**

**plt.title('Comparing number of spam messages to number of non spam**

**messages')**

**for p in ax.patches:**

**width, height = p.get\_width(), p.get\_height()**

**x, y = p.get\_xy()**

**ax.annotate('{}'.format(height), (x +0.25, y + height + 0.8))**

**plt.show**

**<function matplotlib.pyplot.show(close=None, block=None)>**

**Preprocessing:**

**def clean\_data(text):**

**out = re.sub('[^a-zA-Z]', ' ', text)**

**out = out.lower()**

**out = out.split()**

**out = ' '.join(out)**

**return out**

**def tokenize\_word(text):**

**return nltk.word\_tokenize(text)**

**def remove\_stopwords(text):**

**stop\_words = set(stopwords.words("english")+['u','ur','r','n'])**

**filtered\_text = [word for word in text if word not in stop\_words]**

**return filtered\_text**

**def lemmatize\_word(text):**

**lemmatizer = WordNetLemmatizer()**

**lemmas = [lemmatizer.lemmatize(word, pos ='v') for word in text]**

**return lemmas**

**def get\_processed\_tokens(text):**

**text = clean\_data(text)**

**text = tokenize\_word(text)**

**text = remove\_stopwords(text)**

**text = lemmatize\_word(text)**

**return text**

**nltk.download('stopwords')**

**nltk.download('wordnet')**

**df['processed\_text'] = df['Text'].apply(get\_processed\_tokens)**

**df.head()**

**corpus= []**

**for i in df["processed\_text"]:**

**msg = ' '.join([row for row in i])**

**corpus.append(msg)**

**tfidf = TfidfVectorizer()**

**X = tfidf.fit\_transform(corpus).toarray()**

**X.shape**

**Word Cloud:**

**text = ' '.join(corpus)**

**wc = WordCloud(background\_color='black').generate(text)**

**plt.figure(figsize=[15,20])**

**plt.title("WORD CLOUD")**

**plt.axis("off")**

**plt.imshow(wc,interpolation='bilinear')**

**spam\_corpus=[]**

**for i in df[df['Class']=='spam']["processed\_text"]:**

**msg = ' '.join([row for row in i])**

**spam\_corpus.append(msg)**

**text1 = ' '.join(spam\_corpus)**

**wc = WordCloud(background\_color='black').generate(text1)**

**plt.figure(figsize=[15,20])**

**plt.title("SPAM WORD CLOUD")**

**plt.axis("off")**

**plt.imshow(wc,interpolation='bilinear')**

**not\_spam\_corpus=[]**

**for i in df[df['Class']=='ham']["processed\_text"]:**

**msg = ' '.join([row for row in i])**

**not\_spam\_corpus.append(msg)**

**text2 = ' '.join(not\_spam\_corpus)**

**wc = WordCloud(background\_color='black').generate(text2)**

**plt.figure(figsize=[15,20])**

**plt.title("NOT SPAM WORD CLOUD")**

**plt.axis("off")**

**plt.imshow(wc,interpolation='bilinear')**

**X**

**X.shape**

**y = df.iloc[:,0:1]**

**print(y)**

**y = df.iloc[:,0:1]**

**print(y)**

**print(X.shape,y.shape)**

**Data Splitting**

**x\_train,x\_test,y\_train,y\_test= train\_test\_split(X,y,test\_size=0.2,random\_state=42)**

**x\_train.shape**

**x\_test.shape**

**y\_train.shape**

**y\_train**

**Multinomial Naive Bayes:**

**from sklearn.naive\_bayes import MultinomialNB**

**nb = MultinomialNB(fit\_prior=True)**

**nb.fit(x\_train,y\_train)**

**y\_nb = nb.predict(x\_test)**

**cm\_nb = confusion\_matrix(y\_test,y\_nb)**

**sns.heatmap(cm\_nb,annot=True,robust=True)**

**ac\_nb = accuracy\_score(y\_test,y\_nb) \* 100**

**print(ac\_nb)**

**K Nearest Neighbours**

**from sklearn.neighbors import KNeighborsClassifier**

**knn = KNeighborsClassifier()**

**knn.fit(x\_train,y\_train)**

**y\_knn=knn.predict(x\_test)**

**cm\_knn = confusion\_matrix(y\_test,y\_knn)**

**sns.heatmap(cm\_knn,annot=True,fmt='.2f')**

**ac\_knn = accuracy\_score(y\_test,y\_knn) \* 100**

**print(ac\_knn)**

**"""### Visualizing and Comparing the Accuracy Scores of the different Models"""**

**ac = [ac\_nb,ac\_knn]**

**label = ['Naive Bayes','K Nearest Neighbours']**

**plt.figure(figsize=(12,8))**

**ax = sns.barplot(x=label,y=ac)**

**plt.title('r2 score comparison among different regression**

**models',fontweight='bold')**

**for p in ax.patches:**

**width=p.get\_width()**

**height=p.get\_height()**

**x,y = p.get\_xy()**

**ax.annotate('{:.3f}%'.format(height), (x+0.25, y+height+0.8))**

**plt.show()**

**FRONT END CODE:**

**#!/usr/bin/env python**

**"""Django's command-line utility for administrative tasks."""**

**import os**

**import sys**

**def main():**

**os.environ.setdefault('DJANGO\_SETTINGS\_MODULE', 'new\_project.settings')**

**try:**

**from django.core.management import execute\_from\_command\_line**

**except ImportError as exc:**

**51**

**raise ImportError(**

**"Couldn't import Django. Are you sure it's installed and "**

**"available on your PYTHONPATH environment variable? Did you "**

**"forget to activate a virtual environment?"**

**) from exc**

**execute\_from\_command\_line(sys.argv)**

**if \_\_name\_\_ == '\_\_main\_\_':**

**main()**

**8.TESTING**

**1. Data Preparation for Testing**

**a. Test Dataset Collection:**

* Collect a diverse and representative test dataset containing SMS messages labeled as spam or not spam. Ensure the dataset is independent of the training data to avoid overfitting.

**b. Data Preprocessing:**

* Preprocess the test data similarly to the training data (e.g., tokenization, stop words removal, stemming/lemmatization, and feature extraction).

**2. Model Evaluation Metrics**

Evaluate the model using various metrics to measure its performance comprehensively:

* **Accuracy:** The proportion of correctly predicted messages (both spam and non-spam) out of the total messages.
* **Precision:** The proportion of correctly predicted spam messages out of all messages predicted as spam.
* **Recall (Sensitivity):** The proportion of correctly predicted spam messages out of all actual spam messages.
* **F1-Score:** The harmonic mean of precision and recall, providing a balance between the two.
* **Confusion Matrix:** A table showing the true positives, true negatives, false positives, and false negatives, offering insights into the model’s performance.
* **ROC-AUC:** The Area Under the Receiver Operating Characteristic Curve, which shows the trade-off between true positive rate and false positive rate.

**3. Model Validation Techniques**

**a. Cross-Validation:**

* Use k-fold cross-validation on the training data to validate the model performance during the training phase. This helps in assessing the model's ability to generalize to unseen data.

**b. Hold-out Validation:**

* Split the original dataset into separate training and testing datasets. Train the model on the training data and evaluate it on the testing data.

**4. Testing Process**

**a. Load the Trained Model:**

* Load the trained machine learning or deep learning model.

**b. Predict on Test Data:**

* Use the model to predict the class (spam or not spam) for each SMS message in the test dataset.

**c. Evaluate Predictions:**

* Compare the predicted labels with the true labels using the evaluation metrics mentioned above.

**5. Performance Analysis**

**a. Analyze Metrics:**

* Review the accuracy, precision, recall, F1-score, confusion matrix, and ROC-AUC to understand the model's performance.

**b. Error Analysis:**

* Analyze the false positives (non-spam messages predicted as spam) and false negatives (spam messages predicted as non-spam) to identify any patterns or reasons for misclassification.

**c. Compare Models:**

* If multiple models were trained (e.g., different algorithms or different hyperparameters), compare their performance to select the best one.

**6. Fine-Tuning**

**a. Hyperparameter Tuning:**

* Adjust the model’s hyperparameters to improve performance based on the evaluation results.

**b. Feature Engineering:**

* Experiment with different feature extraction methods or additional features to enhance model performance.

**c. Model Retraining:**

* Retrain the model with optimized hyperparameters and features if necessary.

**7. Real-World Testing**

**a. Deployment Testing:**

* Test the deployed model in a real-world environment to ensure it performs well on live data.

**b. User Feedback:**

* Collect feedback from users on the model's predictions to identify any issues and areas for improvement.

**8. Continuous Monitoring and Maintenance**

**a. Performance Monitoring:**

* Continuously monitor the model’s performance in production to detect any degradation over time.

**b. Model Update:**

* Regularly update the model with new data to maintain its accuracy and relevance.

**c. Logging and Alerting:**

* Implement logging and alerting mechanisms to track the model’s behavior and respond to any anomalies promptly.

**9.RESULTS**

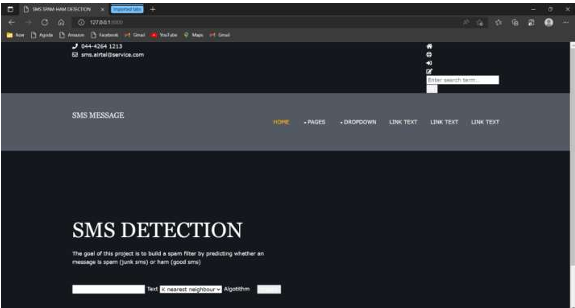


Fig 9.1 Home Screen

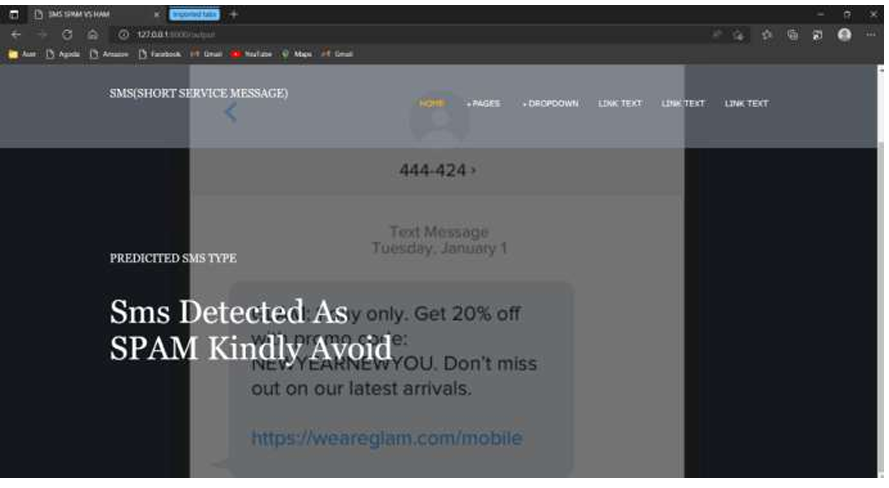


Fig 9.2 Outputs

**10.CONCLUSION**

The SMS spam message problem is plaguing almost every country and keeps increasing without a sign of slowing down as the number of mobile users increase in addition to cheap rates of SMS services. Therefore, this paper presents the spam filtering technique using various machine learning algorithms. Based on the experiment, TF-IDF with Nave bayes classification algorithm outperforms good compare to other algorithm like LSTM in terms of accuracy percentage. However, it is not enough to evaluate the performance based on the accuracy alone since the dataset is imbalanced. After some examinations, NB algorithm still manages to provide good precision and f measure with 0.98 of precision while 0.97 for f-measure. Different algorithms will provide different performances and results based on the features used. For future works, adding more features such as message lengths might help the classifiers to train data better and give better performance.

**FUTURE WORK**

Future scope of this project will involve adding more feature parameter. The more the parameters are taken into account more will be the accuracy. The algorithms can also be applied for analyzing the contents of public comments and thus determine patterns/relationships between the customer and the company. The use of traditional algorithms and data mining techniques can also help predict the corporation performance structure as a whole. In the future, we plan to integrate neural network with some other techniques such as genetic algorithm or fuzzy logic. Genetic algorithm can be used to identify optimal network architecture and training parameters. Fuzzy logic provides the ability to account for some uncertainty produced by the neural network predictions. Their uses in conjunction with neural network could provide an improvement for SMS spam prediction.

**APPLICATION:**

• It can used for company to prevent users using fake links.

• Hacking can be prevented.

**REFERENCES**

[1] Modupe, A., O. O. Olugbara, and S. O. Ojo. (2014) ―Filtering of Mobile Short

Messaging Communication Using Latent Dirichlet Allocation with Social Network

Analysis‖, in Transactions on Engineering Technologies: Special Volume of the World

Congress on Engineering 2013, G.-C. Yang, S.-I. Ao, and L. Gelman, Eds. Springer

Science & Business. pp. 671–686.

[2] Shirani-Mehr, H. (2013) ―SMS Spam Detection using Machine Learning Approach.‖

p. 4.

[3] Abdulhamid, S. M. et al., (2017) ―A Review on Mobile SMS Spam Filtering

Techniques.‖ IEEE Access 5: 15650–15666.

[4] Aski, A. S., and N. K. Sourati. (2016) ―Proposed Efficient Algorithm to Filter Spam

Using Machine Learning Techniques.‖ Pac. Sci. Rev. Nat. Sci. Eng. 18 (2):145–149.

[5] Narayan, A., and P. Saxena. (2013) ―The Curse of 140 Characters: Evaluating The

Efficacy of SMS Spam Detection on Android.‖ p. 33– 42.

[6] Almeida, T. A., J. M. Gómez, and A. Yamakami. (2011) ―Contributions to the Study

of SMS Spam Filtering: New Collection and Results.‖ p. 4.

[7] Mujtaba, D. G., and M. Yasin. (2014) ―SMS Spam Detection Using Simple Message

Content Features.‖ J. Basic Appl. Sci. Res. 4 (4): 5.

[8] Gudkova, D., M. Vergelis, T. Shcherbakova, and N. Demidova. (2017) ―Spam and

Phishing in Q3 2017.‖ Securelist - Kaspersky Lab‘s Cyberthreat Research and Reports.

Available from: https://securelist.com/spam-and-phishing-in-q3-2017/82901/. [Accessed:

10th April 2018].

[9] Choudhary, N., and A. K. Jain. (2017) ―Towards Filtering of SMS Spam Messages

Using Machine Learning Based Technique‖, in Advanced Informatics for Computing

Research 712: 18-30.

[10] Safie, W., N.N.A. Sjarif, N.F.M. Azmi, S.S. Yuhaniz, R.C. Mohd, and S.Y. Yusof.

(2018) ―SMS Spam Classification using Vector Space Model and Artificial Neural

Network.‖ International Journal of Advances in Soft Computing & Its Applications 10 (3):

129-141.

[11] Fawagreh, Khaled, Mohamed Medhat Gaber, and Eyad Elyan. (2014) ―Random

Forests: From Early Developments to Recent Advancements, Systems Science &

Control Engineering.‖ An Open Access Journal 2 (1): 602-609.

[12] Sajedi, H., G. Z. Parast, and F. Akbari. (2016) ―SMS Spam Filtering Using Machine

Learning Techniques: A Survey.‖ Machine Learning, 1 (1): 14.

[13] Q. Xu, E., W. Xiang, Q. Yang, J. Du, and J. Zhong. (2012) ―SMS Spam Detection

Using Noncontent Features.‖ IEEE Intell. Syst. 27(6): 44–51.

[14] Sethi, G., and V. Bhootna. (2014) SMS Spam Filtering Application Using Android.

[15] Nagwani, N. K. (2017) ―A Bi-Level Text Classification Approach for SMS Spam

Filtering and Identifying Priority Messages.‖ 14 (4): 8.

[16] Delany, S. J., M. Buckley, and D. Greene. (2012) ―SMS Spam Filtering: Methods

and Data,‖ Expert Syst. Appl. 39(10): 9899–9908.

[17] Chan, P. P. K., C. Yang, D. S. Yeung, and W. W. Y. Ng. (2015) ―Spam Filtering for

Short Messages in Adversarial Environment.‖ Neurocomputing 155: 167–176.

[18] Sethi, P., V. Bhandari, and B. Kohli. (2017) ―SMS Spam Detection and Comparison

of Various Machine Learning Algorithms‖, in 2017 International Conference on

Computing and Communication Technologies for Smart Nation (IC3TSN). pp. 28–31.

[19] Warade, S. J., P. A. Tijare, and S. N. Sawalkar. (2014) ―An Approach for SMS

Spam Detection.‖ Int. J. Res. Advent Technol. 2 (2): 4.

[20] Almeida. T. A., and J. M. G. Hidalgo. (2018) ―SMS Spam Collection.‖ Available

from: http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/. [Accessed: 11st April

2018].