A Comparative Study of Classification Algorithm for Spam Data Analysis

Sakshi[1], Mohd. Bilal[2], Prince Kumar[3], Priyanka Bhardwaj[4]

[1,2,3,4] *Department of Computer science and Engineering Meerut Institute of Engineering and Technology, Meerut, India*

***Abstract* - Spams are more than just annoying texts; they pose a severe threat to the online community as they often have malware or ransom ware embedded in them. Other than this direct cost, there is also the cost of reputation and loss of productivity. For these and many other reasons, spam filtering solutions are very much needed. In this paper, we conducted experiment in the chat box environment by using three algorithms namely Naïve Bayes, Ridge Regression and Linear Regression on the spam filter on chat box and late the three algorithms were compared in terms of classification accuracy. According to our simulation results the Naïve Bayes classifier outperforms the Ridge and Linear Regression in terms of classification accuracy.**

***Index Terms* – classification accuracy, Multinomial Naive Bayes, Ridge Regression, Linear Regression, CountVectorizer, sklearn, Pandas.**

I.INTRODUCTION

In today's modern era of digitalization, communication plays a vital role in it may it be a formal or informal communication. This growth of digital communication leads to an unprecedented increase in the number of illegitimate messages (also known as spam).

Nowadays, chatbot has become one of the quickest and most inexpensive means of communication. However, popularity of email has further increased spam mails during the past years. Data mining classify cation algorithms are used to categorize spam or non-spam. In this paper, we conducted experiment using three algorithms namely Linear Regression, Ridge Regression and Naïve Bayes. Using Naïve Bayes, Ridge Regression, Linear Regression we find out which algorithm is accurate for spam filtering.

A. Ridge Regression

Ridge Regression is a model tuning method that is used to analyse any data that suffers from multicollinearity which performs L2 regularization. When multicollinearity issue occurs, least-squares are unbiased, and variances are quite

large, which results in predicted values to be far different from the actual values.

The cost function for ridge regression:

Min (||Y – X(theta)||^2 + λ||theta||^2)

Lambda given here is the penalty term. λ given here is denoted by an alpha parameter in the ridge function. So, by changing the values of alpha, we are controlling the penalty term. Higher the values alpha, bigger is the penalty and therefore the magnitude of coefficients is reduced.

- It shrinks the parameters. Therefore, it is used to prevent multicollinearity
- It reduces the model complexity by coefficient shrinkage.

B. LINEAR REGRESSION

Linear Regression Algorithm is a machine learning algorithm based on supervised learning (the learning of machine learning in which the input and output already given to the machine on the basis of previous learning machine give output). Linear regression is a part of regression analysis which is a technique of predictive modelling that help us to find out the relationship between Input and target variable. Linear regression is one of the very basic form of machine learning in which we train we train a model to predict the behaviour of your data based on some variables. The name Linear regression suggests itself the linear that means the two variables which are on the x- axis and y- axis should be linearly correlated.

C. Multinomial Naive Bayes:

Multinomial Naïve Bayes uses term frequency i.e. the number of times a given term appears in a document. Term frequency is often normalized by dividing the raw term frequency by the document length. After normalization, term frequency can be used to compute maximum likelihood estimates based on the training data to estimate the conditional probability.

Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). The algorithm is based on the Bayes theorem and predicts the tag of a text such as a piece of email or newspaper article. It calculates the probability of each tag for a given sample and then gives the tag with the highest probability as output.

Naive Bayes classifier is a collection of many algorithms where all the algorithms share one common principle, and that is each feature being classified is not related to any other feature. The presence or absence of a feature does not affect the presence or absence of the other feature. Naive Bayes is a powerful algorithm that is used for text data analysis and with problems with multiple classes. To understand Naive Bayes theorem's working, it is important to understand the Bayes theorem concept first as it is based on the latter.

Bayes theorem, formulated by Thomas Bayes, calculates the probability of an event occurring based on the prior knowledge of conditions related to an event. It is based on the following formula:

$P(A|B) = P(A) * P(B|A)/P(B)$

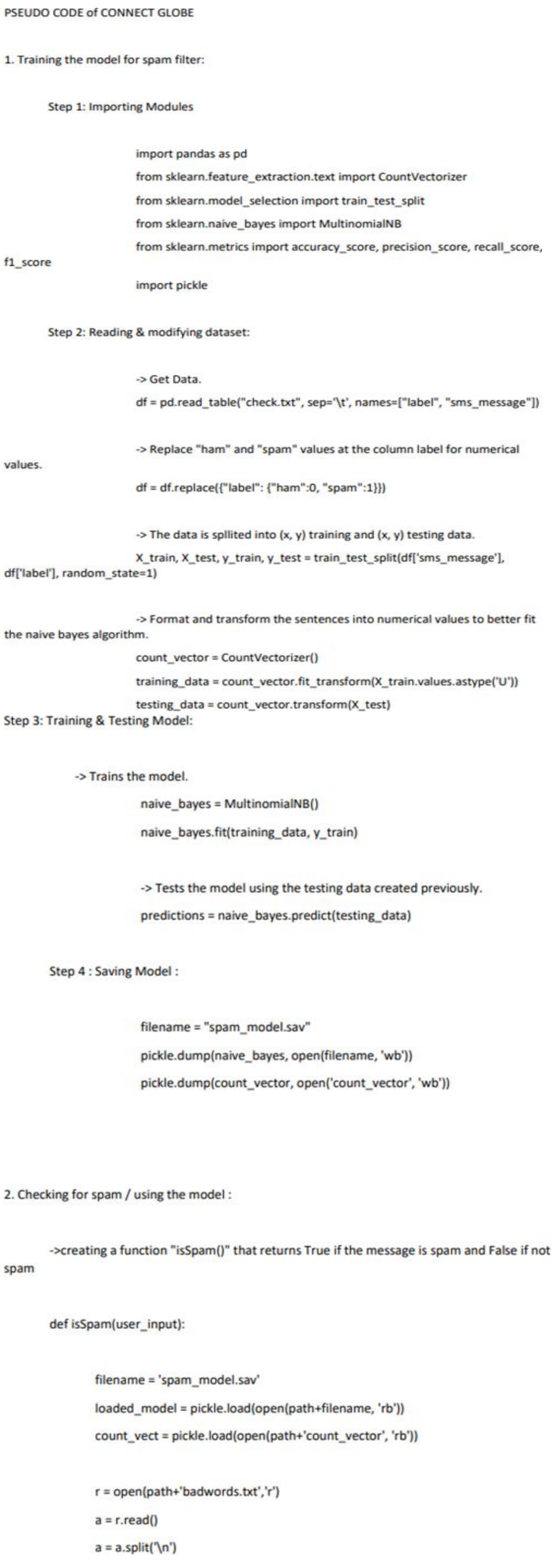
Where we are calculating the probability of class A when predictor B is already provided. P(B) = prior probability of B

P(A) = prior probability of class A

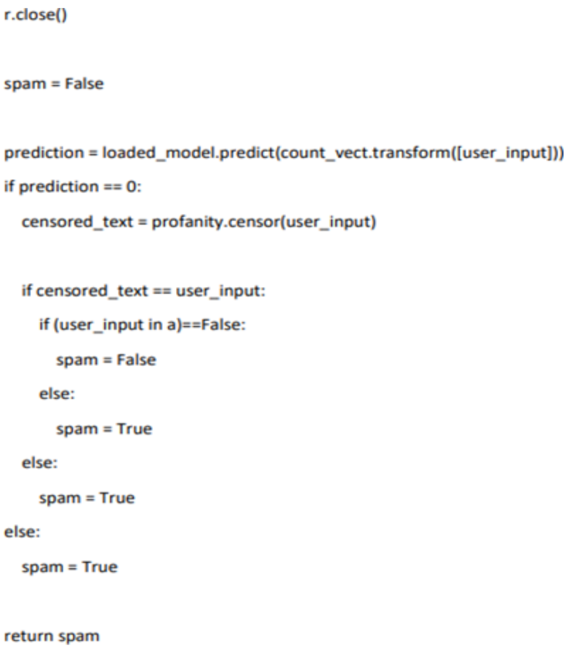
P(B|A) = occurrence of predictor B given class A probability

II.METHODOLGY

In this section we compare the classification accuracy results of the three algorithms namely Naive Bayes, Ridge Regression, Linear Regression. This is phase in which spam filter plays the main role as soon as someone posts any message the message first goes through the Spam Filter and then it checks the message whether it is spam or ham, if its spam the user gets a message that it is spam otherwise the message is posted in the group. We list below the steps taken to achieve desired results:

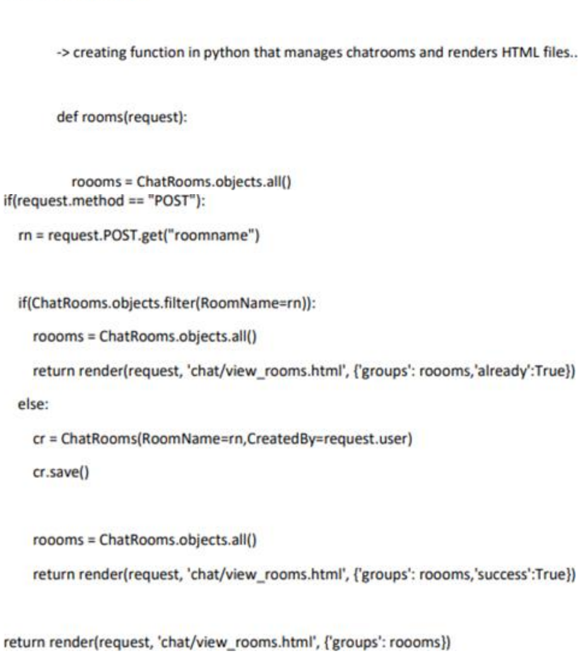


```
PSEUDO CODE of CONNECT GLOBE

1. Training the model for spam filter:

        Step 1: Importing Modules

                        import pandas as pd
                        from sklearn.feature_extraction.text import CountVectorizer
                        from sklearn.model_selection import train_test_split
                        from sklearn.naive_bayes import MultinomialNB
                        from sklearn.metrics import accuracy_score, precision_score, recall_score,
f1_score
                        import pickle

        Step 2: Reading & modifying dataset:

                        -> Get Data.
                        df = pd.read_table("check.txt", sep='\t', names=["label", "sms_message"])

                        -> Replace "ham" and "spam" values at the column label for numerical
values.
                        df = df.replace({"label": {"ham":0, "spam":1}})

                        -> The data is spllited into (x, y) training and (x, y) testing data.
                        X_train, X_test, y_train, y_test = train_test_split(df['sms_message'],
df['label'], random_state=1)

                        -> Format and transform the sentences into numerical values to better fit
the naive bayes algorithm.
                        count_vector = CountVectorizer()
                        training_data = count_vector.fit_transform(X_train.values.astype('U'))
                        testing_data = count_vector.transform(X_test)
Step 3: Training & Testing Model:

              -> Trains the model.
                        naive_bayes = MultinomialNB()
                        naive_bayes.fit(training_data, y_train)

                        -> Tests the model using the testing data created previously.
                        predictions = naive_bayes.predict(testing_data)

        Step 4 : Saving Model :

                        filename = "spam_model.sav"
                        pickle.dump(naive_bayes, open(filename, 'wb'))
                        pickle.dump(count_vector, open('count_vector', 'wb'))


2. Checking for spam / using the model :

        ->creating a function "isSpam()" that returns True if the message is spam and False if not
spam

        def isSpam(user_input):

                filename = 'spam_model.sav'
                loaded_model = pickle.load(open(path+filename, 'rb'))
                count_vect = pickle.load(open(path+'count_vector', 'rb'))

                r = open(path+'badwords.txt','r')
                a = r.read()
                a = a.split('\n')
```



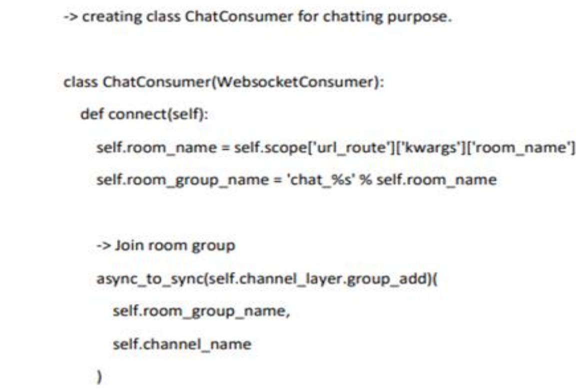
```
r.close()


spam = False


prediction = loaded_model.predict(count_vect.transform([user_input]))
if prediction == 0:
  censored_text = profanity.censor(user_input)


  if censored_text == user_input:
    if (user_input in a)==False:
      spam = False
    else:
      spam = True
  else:
    spam = True
else:
  spam = True


return spam
```

3. Creating ChatRoom :

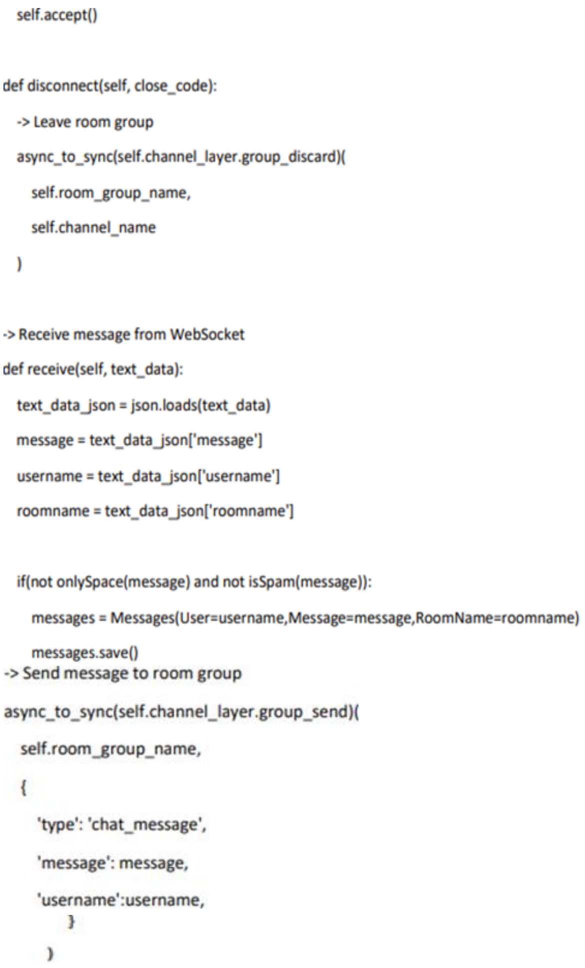


```
        -> creating function in python that manages chatrooms and renders HTML files..


        def rooms(request):


          roooms = ChatRooms.objects.all()
if(request.method == "POST"):
  rn = request.POST.get("roomname")


  if(ChatRooms.objects.filter(RoomName=rn)):
    roooms = ChatRooms.objects.all()
    return render(request, 'chat/view_rooms.html', {'groups': roooms,'already':True})
  else:
    cr = ChatRooms(RoomName=rn,CreatedBy=request.user)
    cr.save()


    roooms = ChatRooms.objects.all()
    return render(request, 'chat/view_rooms.html', {'groups': roooms,'success':True})


return render(request, 'chat/view_rooms.html', {'groups': roooms})
```

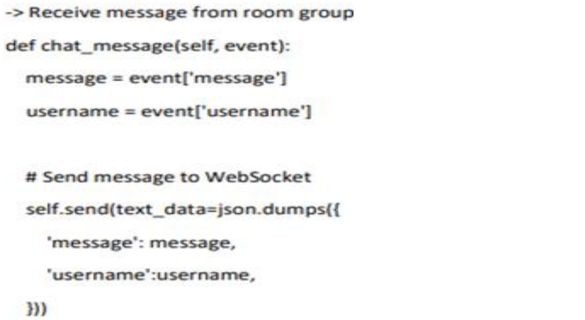
4. Managing Chats



```
        -> creating class ChatConsumer for chatting purpose.


        class ChatConsumer(WebsocketConsumer):
          def connect(self):
            self.room_name = self.scope['url_route']['kwargs']['room_name']
            self.room_group_name = 'chat_%s' % self.room_name


            -> Join room group
            async_to_sync(self.channel_layer.group_add)(
              self.room_group_name,
              self.channel_name
            )
```

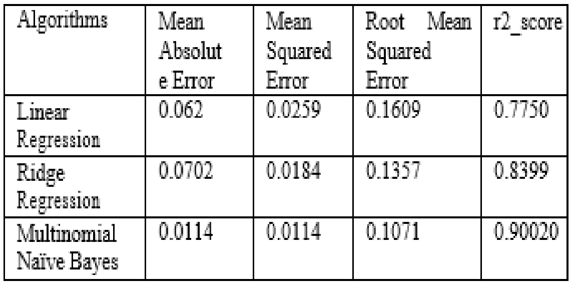


```
  self.accept()


def disconnect(self, close_code):
  -> Leave room group
  async_to_sync(self.channel_layer.group_discard)(
    self.room_group_name,
    self.channel_name
  )


-> Receive message from WebSocket
def receive(self, text_data):
  text_data_json = json.loads(text_data)
  message = text_data_json['message']
  username = text_data_json['username']
  roomname = text_data_json['roomname']


  if(not onlySpace(message) and not isSpam(message)):
    messages = Messages(User=username,Message=message,RoomName=roomname)
    messages.save()
-> Send message to room group
async_to_sync(self.channel_layer.group_send)(
  self.room_group_name,
  {
    'type': 'chat_message',
    'message': message,
    'username':username,
      }
    )
```



```
-> Receive message from room group
def chat_message(self, event):
  message = event['message']
  username = event['username']


  # Send message to WebSocket
  self.send(text_data=json.dumps({
    'message': message,
    'username':username,
  }))
```

TABLE: CLASSIFICATION ACCURACY TESTS RESULTS



Algorithms	Mean Absolute Error	Mean Squared Error	Root Mean Squared Error	r2_score
Linear Regression	0.062	0.0259	0.1609	0.7750
Ridge Regression	0.0702	0.0184	0.1357	0.8399
Multinomial Naïve Bayes	0.0114	0.0114	0.1071	0.90020

Table demonstrate the classification accuracy results of four classification decision tree algorithms. In above table there clearly shown that Multinomial has the minimum mean absolute error, mean squared error, root mean square error and it has highest r2_score whereas, linear regression has the lowest r2_score. As multinomial Naïve bayes has the highest r2_score that's why we used Multinomial Naïve Bayes. since we know that, R-squared is a statistical measure of how close the data are to the fitted regression line. we can see from the above table that, MultinomialNB is having the greatest R2_score hence, it is building best "best fit" line compared to other algorithms and it directly denotes that, it gives better accuracy than others.

III.CONCLUSION AND FUTURE RESEARCH

In this research we have performed the experiments in order to determine the classification accuracy of three algorithms in terms of which algorithm better determine whether a content words is spam or not with the help of some algorithms. Three algorithms namely Naive Bayes, Ridge Regression, Linear Regression were compared on the basis of different percentage of correctly classified instances. Naive Bayes classifier used in this has a very important role in this process of filtering spam messages. The quality of performance of Naive Bayes classifier is also based on datasets that were used. As shown, datasets that have fewer instances of messages and attributes can give good results for Naive Bayes classifiers. This classifier can also get the highest precision that gives the highest percentage spam message and hence manages to block if the dataset is collected from a single sender. Moreover, other factors can also be taken for instance the time requirement to compare the accuracy of the proposed algorithms with respect to the decrease in the number of attributes while keeping the number of instances constant which we believe shall surely bring out certain important aspects about the different algorithm which can prove useful in the research field.

REFERENCES

[1] Sharma, Aman Kumar, and Suruchi Sahni. "A comparative study of classification algorithms for spam data analysis." International Journal on Computer Science and Engineering 3.5 (2011): 1890-1895.

[2] Abdulhamid, Shafi'I. Muhammad, et al. "Comparative Analysis of Classification Algorithms for Spam Detection." International Journal of Computer Network & Information Security 10.1 (2018).

[3] Aher, Sunita B., and L. M. R. J. Lobo. "Comparative study of classification algorithms". International Journal of Information Technology 5.2 (2012): 239-243.

[4] Ragab, Abdul Hamid M., et al. "A comparative analysis of classification algorithms". Proceedings of the 2014 Workshop on Interaction Design in Educational Environments. 2014.

[5] Abdullah, Abdullah O., et al. "A comparative analysis of common YouTube comment spam filtering techniques". 2018 6th International Symposium on Digital Forensic and Security (ISDFS). IEEE, 2018.

[6] Nikam, Sagar S. "A comparative study of classification techniques in data mining algorithms." Oriental journal of computer science & technology 8, no. 1 (2015): 13-19.

[7] Gupte, Amit, et al. "Comparative study of classification algorithms used in sentiment analysis." International Journal of Computer Science and Information Technologies 5.5 (2014): 6261-6264.

[8] Yu, B., & Xu, Z. B. (2008). A comparative study for content-based dynamic spam classification using four machine learning algorithms. Knowledge-Based Systems, 21(4), 355-362.

[9] Gupta, Ajay Kumar. "Spam mail filtering using data mining approach: A comparative performance analysis." Handling priority inversion in time constrained distributed databases. IGI Global, 2020.253-282.

[10] Trivedi, Shrawan Kumar, and Prabin Kumar Panigrahi. "Spam classification: a comparative analysis of different boosted decision tree approaches." Journal of Systems and Information Technology (2018).