PROJECT REPORT

Detection of Spam Comments in youtube videos

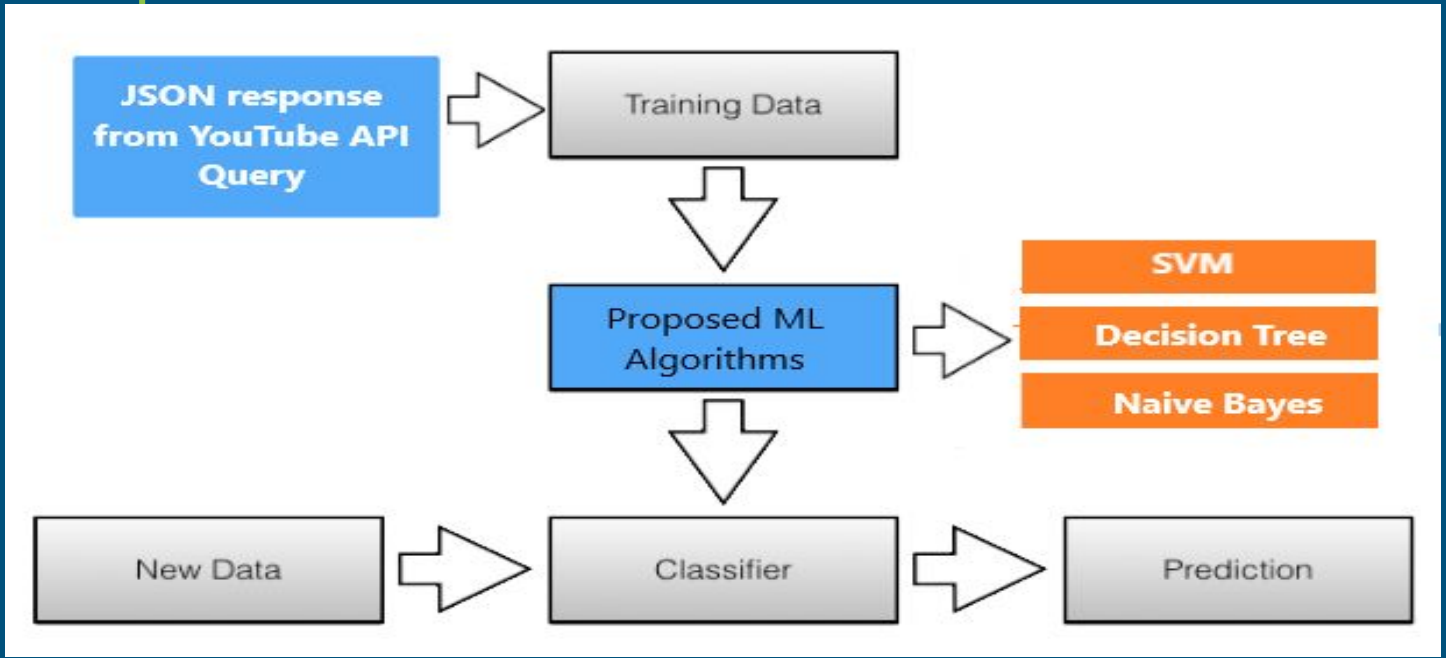
SPARSH AHUJA 2017258
SIDDHANT LOHIA 2017254
SAURABH KUMAR 2017232

Preface

1. We describe spam comments as those which have a promotional intent or those who deem to be contextually irrelevant for a given video. The prospects of monetisation through advertising on popular social media channels over the years has attracted an increasingly larger number of users.
2. This has in turn led to the growth of malicious users who have begun to develop automated bots, capable of large-scale orchestrated deployment of spam messages across multiple channels simultaneously. The presence of these comments significantly hurts the reputation of a channel and also the experience of normal users.
3. In this work, we attempt to detect such comments by applying conventional machine learning algorithms.

OverView Diagram

The steps described in the previous diagram have been summarised in this diagram



Brief Implementation

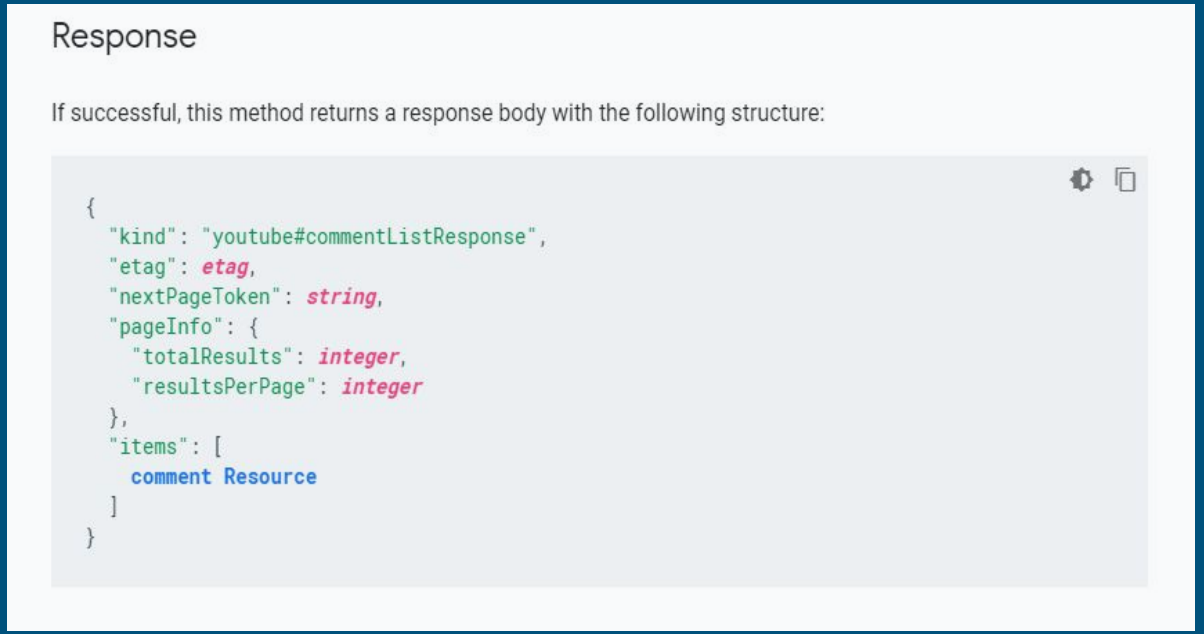
1. **PROCURING DATASET**
   A comment dataset from music videos of some popular singers was used, Using the standard youtube APIs .
   This response form the APIs are used to populate the training as well as testing dataSet

2. **PRE - PREPROCESSING**
   For the Pre-processing phase, the raw dataset will be executed and the data cleaning methods which are **tokenization, stopwords removal** and **stemming** are performed. The clean dataset will be used for the next process of feature selection and extraction

3. **FEATURE EXTRACTION AND CLASSIFICATION PHASE**
   Results in Feature Extraction and Feature Selection As features identified from the literature review, various features may be extracted from YouTube classification purposes. Besides, the data already consists of two (2) classes where the classes are **spam** and **ham** .

Procuring Dataset

1. A comment dataset from music videos of some popular singers was used, Using the standard youtube APIs ,
2. if the API request is successful it returns a JSON response body with the Following structure which is used to populate the training as well as testing dataSet.
3. We have used the standard Base URL to query for the Data and populate the Dataset which has been Attached in the drive.

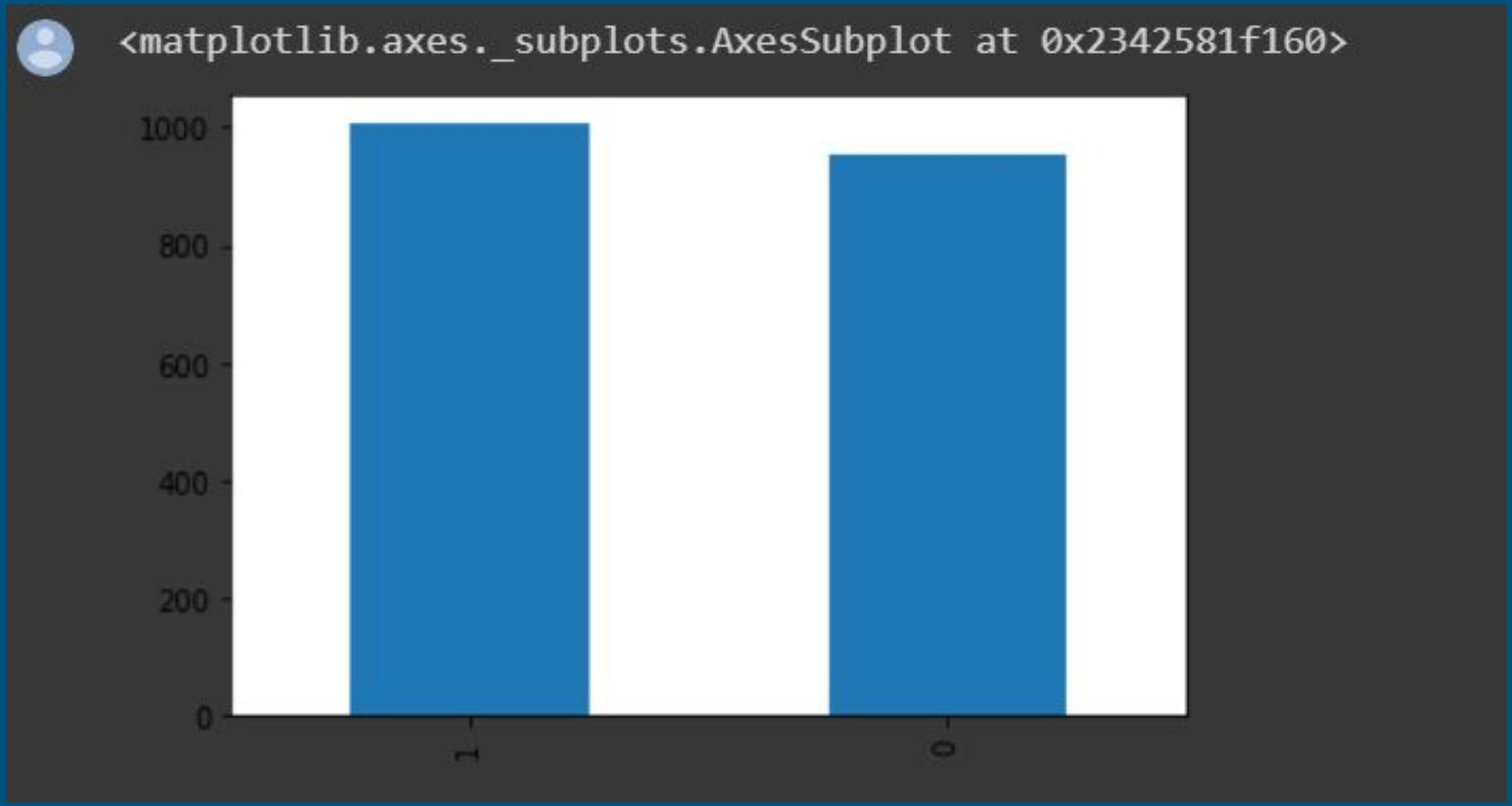


Response

If successful, this method returns a response body with the following structure:

```
{
  "kind": "youtube#commentListResponse",
  "etag": etag,
  "nextPageToken": string,
  "pageInfo": {
    "totalResults": integer,
    "resultsPerPage": integer
  },
  "items": [
    comment Resource
  ]
}
```

BALANCED DATASET



Pre - Processing

Before we could start actual preprocessing of data we had to comeup with something to homogenise our data so we took help of regex and processes our dataset to make it ready for pre-processing.

1. We used regex to have A cumulate a whole lot of Entities

    1. Emails
    2. web Addresses
    3. Money symbols
    4. Phone Numbers
    5. Real Numbers

```
#REGEX TO ENCODE useless data as in emails, numbers etc into useful text features

# Replace email addresses with 'email'
processed = text_messages.str.replace(r'^.+@[^\.].*\.[a-z]{2,}$',
                                      'emailaddress')
```

```
# Replace money symbols with 'moneysymb'
processed = processed.str.replace(r'£|\$', 'moneysymb')
```

Current State of Implementation

We have used seven different classifiers and compared their accuracies uptill now in the project Implementation.

1. K Nearest Neighbors
2. Naive Bayes
3. Decision Tree
4. Random Forest
5. Logistic Regression
6. SGD Classifier
7. SVM Linear

K- Nearest Neighbours

1. K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). O
2. **KNN Accuracy Obtained : *89.366605***

Naive Bayes

Naive bayes is a well-known classifier relying on Bayes theorem. Naive Bayes classifier, assumes that the features do not correlate with each other. Therefore, a probability score is calculated with multiplication of some conditional probabilities

1. **NB Accuracy Obtained : *89.16155***

Decision Trees

Decision Tree classifier aims to reach a classification decision with the help of a decision tree structure it constructed. Nodes in the decision tree structure generally represent feature values and leaves in the decision tree structure represent specific class labels

DT Accuracy Obtained : *95.910020*

Logistic Regression

1. Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, **logistic regression** is estimating the parameters of a logistic model (a form of binary regression).
2. **Logistic Regression Accuracy Obtained : *95.910020***

SVM Linear

A Support Vector Machine (SVM) i s a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning) , the algorithm outputs an optimal hyperplane which categorizes new examples.

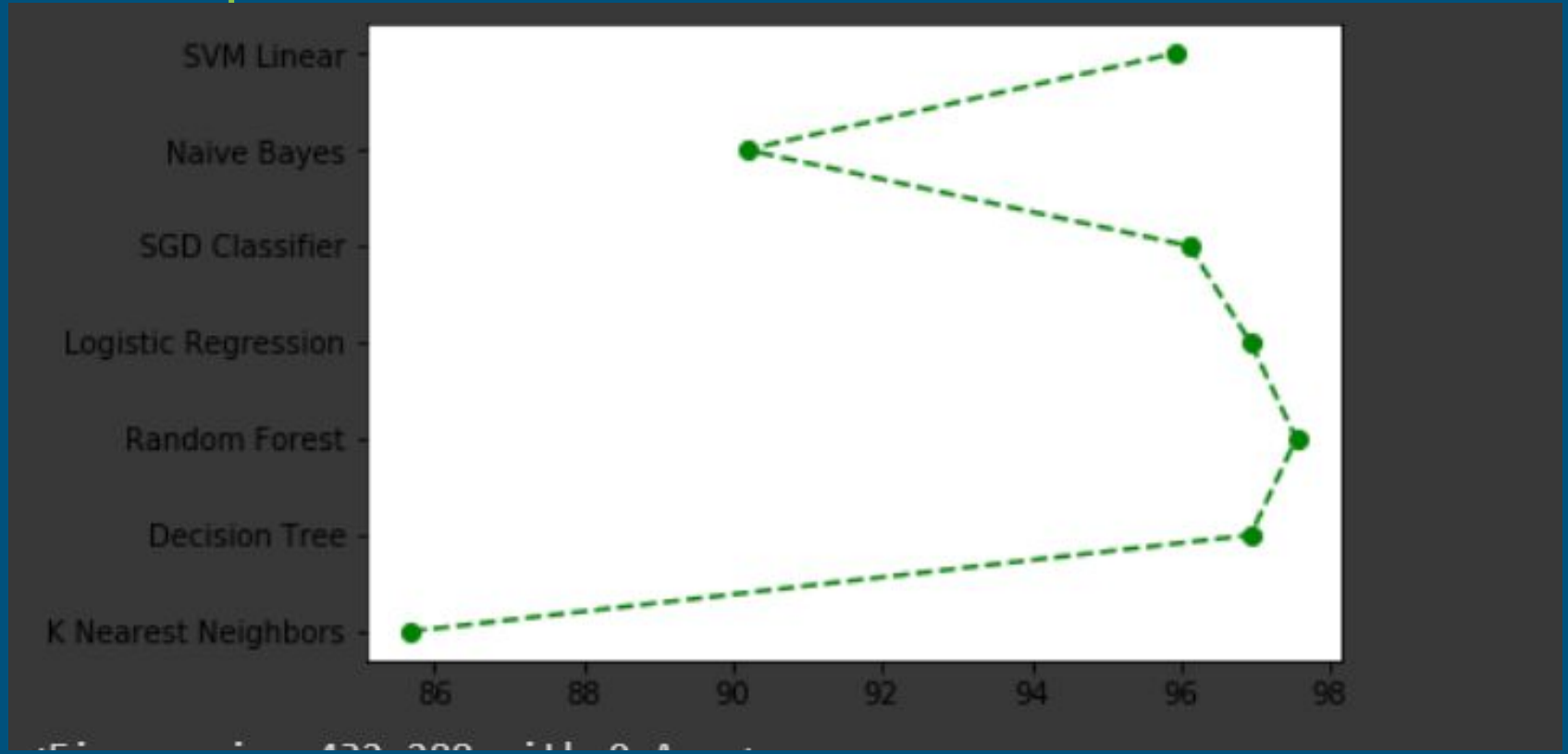
SVM Accuracy Obtained : *94.478527*

Stochastic Gradient Descent Classifier

1. Stochastic Gradient Descent is a solver. It is a simple and efficient approach for discriminative learning of linear classifiers under convex loss functions such as Support Vector Machines and Logistic Regression

SGD Accuracy Obtained : *95.910020*

ALL CLASSIFIERS ACCURACY COMPARED (MAX Random Forest)



Random Forest

1. Random forests or random decision forests are an ensemble learning method for classification,
regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

2. Random decision forests correct for decision trees' habit of overfitting to their training set.

3. ***RF Accuracy Obtained : 97.315102(Best uptil)***
(confusion matrix on next page)

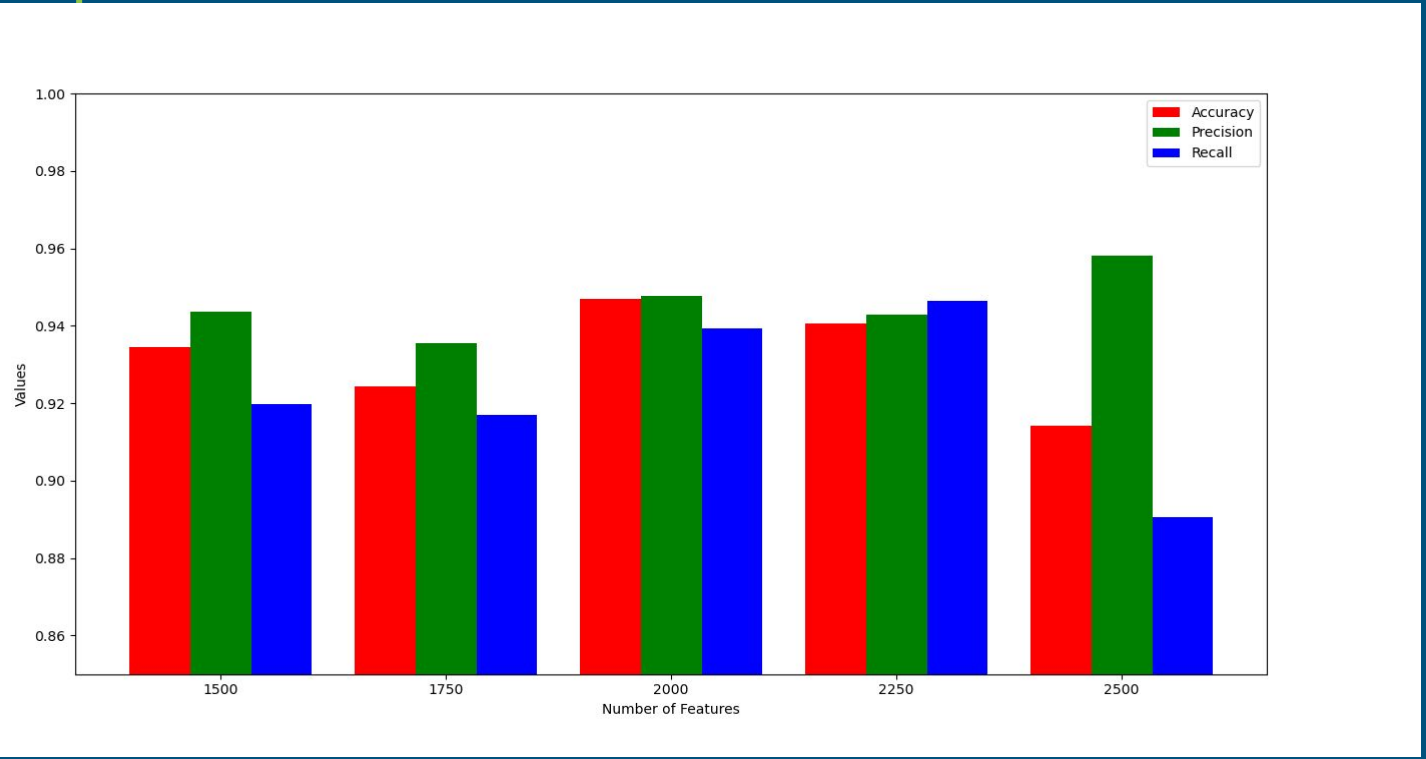
CONFUSION MATRIX ON RANDOM FOREST RESULTS

	precision	recall	f1-score	support
0	0.95	0.99	0.97	247
1	0.99	0.94	0.97	242
accuracy			0.97	489
macro avg	0.97	0.97	0.97	489
weighted avg	0.97	0.97	0.97	489

		predicted	
		ham	spam
actual	ham	245	2
	spam	14	228

Hyper Parameter tuning Graph

Hyper parameter tuning was performed based on the results of Random Forest Classifier



Additional Improvements

1. **Improving Accuracy using Deep learning**
   Using keras and Tensorflow2.0 to train a model and output the two class using sigmoid activation. The building block of the **deep neural networks** is called the **sigmoid** neuron. **Sigmoid** neurons are similar to perceptrons, but they are slightly modified such that the output from the **sigmoid** neuron is much smoother than the step functional output from perceptron.

2. **Check fitting of our model using validation dataset**
   We are currently using test dataset to check the accuracy of our model on various classifiers we can check the accuracy of our models using validation dataset to see how the Model reacts to the new set of points.

DETAILS ABOUT IMPROVEMENTS

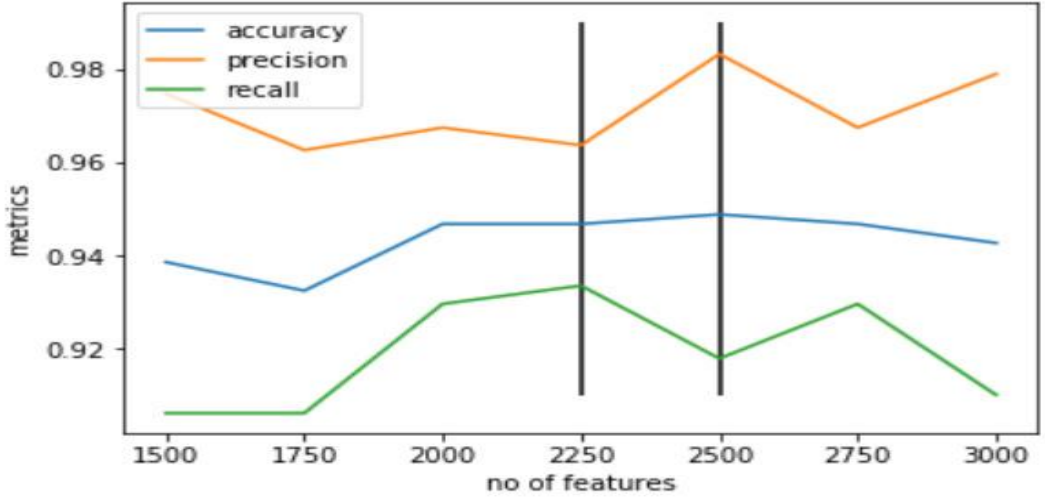
Deep Learning model was implemented as to improve the precision and recall metrics. The deep learning model comprised of 2 hidden Dense layers of *16 and 8 units* respectively. *Adam optimiser used to escape local minima in non convex fucntions. 'Relu' activation was used in both the hidden layer.*

The output layer had 1 unit and was activated with the '*sigmoid' function for classification*

```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_29 (Dense)             (None, 16)                36016
_________________________________________________________________
dense_30 (Dense)             (None, 8)                 136
_________________________________________________________________
dense_31 (Dense)             (None, 1)                 9
=================================================================
Total params: 36,161
Trainable params: 36,161
Non-trainable params: 0
```

HyperParameter Tuning

The input dimension of the first layer was varied as per the number of features from 1500 words to 3000 words at a difference of 250 .

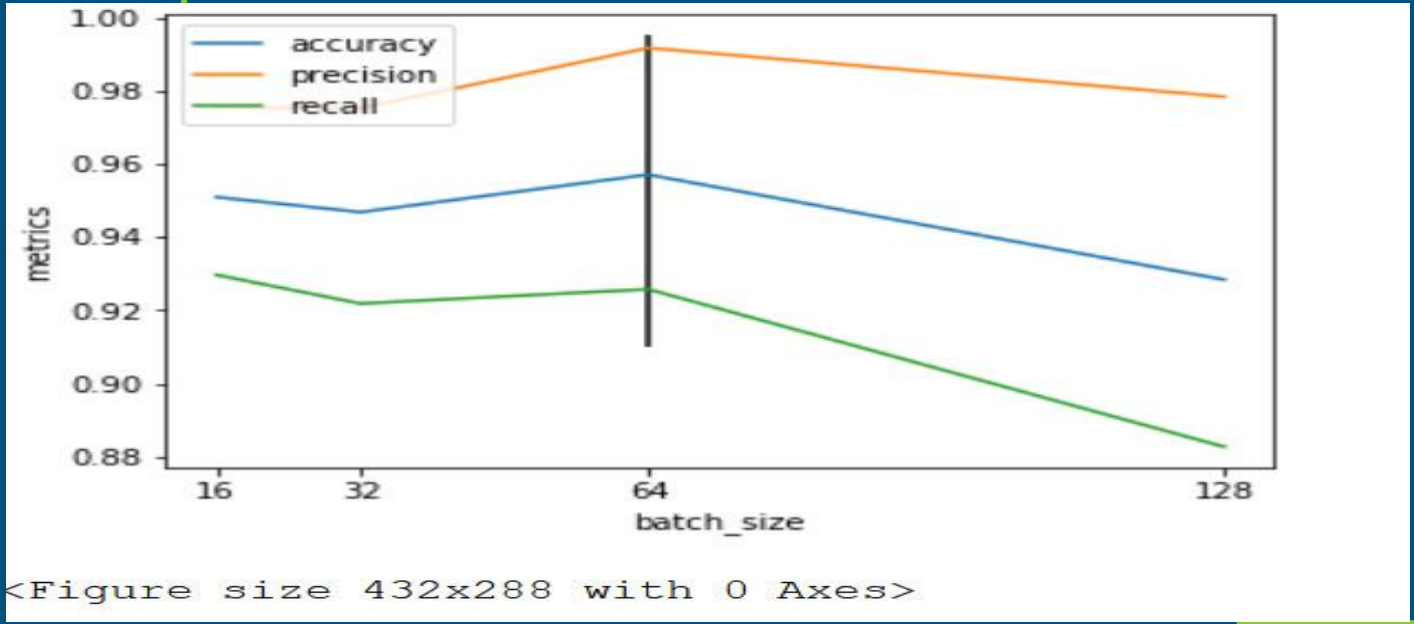


<Figure size 432x288 with 0 Axes>

Optimum feature value determined : 2250

HyperParameter Tuning

Model was trained on different batch sizes varying in ***8, 16, 32, 128*** with the following results



Optimum Batch Size determined : 64

MODEL EVALATION USING VALIDATION SET

Validation Accuracy and Accuracy Curve vs EPOCHS after implementing the Second Improvement