YouTube Video Classification using Supervised Learning and comparison of Classification Algorithms

Introduction

YouTube is the principal platform for video-sharing [1] that allows users to upload, view and share videos and the second-most visited site after Google [2]. Due to the vast variety of video content available, it has garnered over 2 billion active users world-wide. Approximately, 720k hours' worth of video content is being added every single day and this figure is growing exponentially, making it the popular choice for the users of all age-groups.

With volume comes the plethora of choices and it makes the user overwhelmed and confused on which video content is best suitable for their specific purpose. YouTube recommender system is designed to navigate this issue and involves the algorithm to classify and pick up the best recommendations for the search query. These algorithms are built on different machine learning and data mining techniques and are capable to manage big data. The method goal of these algorithms varies which can be prediction, estimations, classification or clustering. With the huge amount of content being present on YouTube site, it becomes necessary that the videos are categorised properly to reduce the search time and present only relevant content to the user. Many times it happens that the user is not able to retrieve desired videos as per their search queries and the result being displayed is irrelevant.

In this report we will use classification method to differentiate YouTube videos based on their associated data and evaluate the performance of different classifiers being used in this supervised learning model. YouTube data can be mainly divided into broad categories such as textual data, graph data, time-series data, video and image data etc. They all serve for different analysis purposes like time-series analysis, clustering, and deep learning using image data.

Textual data is very useful as they offer many different analysis techniques using natural language processing (NLP), such as sentiment analysis, keyword analysis and text classification. Therefore, for our classification purpose, we will only use text data such as video titles, description, and tags associated with each video.

There are number of classifiers available to choose from, depending on the classification problem and dataset being used. We will be using the popular algorithms, Naïve -Bayes, Support Vector machine and Random Forest algorithms as they are effective on handling high-dimensional data and non-linearity of feature space. At the end of the report, we will compare the accuracy of each classifier and understand their performance.

Process

The diagram above presents the flow of the processes adopted in this study. Each process is further explained in this report, followed by the conclusion.

1) Data collection

In this classification problem, we are classifying videos into five broad categories viz. Machine Learning, Python, Statistics, Cloud Computing and Visualization. For this study, we are using only text data for YouTube videos. The data is collected using YouTube open API v3. This API allows user to fetch various data associated with YouTube channel, playlist, and videos. The process is carried out by initial search for relevant playlists in the search bar for each of the categories above. After picking up few playlists for each class, we have made an API call to YouTube API using the key via Python script. Then, using the in-built methods for fetching playlist data, we have initially collected channel ids, channel name and video titles for the first category i.e. machine learning. Using the same method again on the response from first method, we can collect video ids for each video titles.

Now, using methods built for videos, we passed the video ids to collect remaining parameters like description, tags, published date etc. The data retrieved is in JSON format which is then stored in Pandas DataFrame and a new column named ‘class’ is added to explicitly label each data record with their class.

The same method is repeated for remaining classes and final dataset is constructed by appending the output data frames for each class into final dataset as shown below. The dataset contains 1051 records with nearly 200 records for each class. We have removed the columns channel-id, channel name and published date as they don’t serve the purpose of our study.

	video_title	description	tags	class
0	KNN (K Nearest Neighbors) in Python - Machine Learning From Scratch 01 - Python Tutorial	Get my Free NumPy Handbook:\nhttps://www.python-engineer.com/numpybook\n\nIn this Machine Learning from Scratch Tutorial, we are going to implement the K Nearest Neighbors (KNN) algorithm, using only built-in Python modules and numpy. We will also learn about the concept and the math behind this popular ML algorithm.\n\n~~~~~~~~~~~~~~~ GREAT PLUGINS FOR YOUR CODE EDITOR ~~~~~~~~~~~~~~~\n✅ Write cleaner code with Sourcery: https://sourcery.ai/?utm_source=youtube&utm_campaign=pythonengineer *\n\n📓 Notebooks available on Patreon:\nhttps://www.patreon.com/patrickloeber\n\n⭐ Join Our Discord : https://discord.gg/FHMg9tKFSN\n\nIf you enjoyed this video, please subscribe to the channel!\n\nThe code can be found here:\nhttps://github.com/patrickloeber/MLfromscratch\n\nYou can find me here:\nWebsite: https://www.python-engineer.com\nTwitter: https://twitter.com/patloeber\nGitHub: https://github.com/patrickloeber\n\n#Python #MachineLearning\n\n---...	['Python', 'Machine Learning', 'ML', 'numpy', 'KNN', 'Tutorial']	machine learning
1	Linear Regression in Python - Machine Learning From Scratch 02 - Python Tutorial	Get my Free NumPy Handbook:\nhttps://www.python-engineer.com/numpybook\n\nIn this Machine Learning from Scratch Tutorial, we are going to implement the Linear Regression algorithm, using only built-in Python modules and numpy. We will also learn about the concept and the math behind this popular ML algorithm.\n\n~~~~~~~~~~~~~~~ GREAT PLUGINS FOR YOUR CODE EDITOR ~~~~~~~~~~~~~~~\n✅ Write cleaner code with Sourcery: https://sourcery.ai/?utm_source=youtube&utm_campaign=pythonengineer *\n\n📓 Notebooks available on Patreon:\nhttps://www.patreon.com/patrickloeber\n\n⭐ Join Our Discord : https://discord.gg/FHMg9tKFSN\n\nIf you enjoyed this video, please subscribe to the channel!\n\nThe code can be found here:\nhttps://github.com/patrickloeber/MLfromscratch\n\nFurther readings:\nhttps://ml-cheatsheet.readthedocs.io/en/latest/linear_regression.html\nhttps://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html\n\n\nYou can find me here:\nWebsite: https://www.python-engineer.com\nTwitt...	['Python', 'Machine Learning', 'Regression', 'numpy', 'Tutorial', 'Gradient descent']	machine learning
2	Logistic Regression in Python - Machine Learning From Scratch 03 - Python Tutorial	Get my Free NumPy Handbook:\nhttps://www.python-engineer.com/numpybook\n\nIn this Machine Learning from Scratch Tutorial, we are going to implement the Logistic Regression algorithm, using only built-in Python modules and numpy. We will also learn about the concept and the math behind this popular ML algorithm.\n\n~~~~~~~~~~~~~~~ GREAT PLUGINS FOR YOUR CODE EDITOR ~~~~~~~~~~~~~~~\n✅ Write cleaner code with Sourcery: https://sourcery.ai/?utm_source=youtube&utm_campaign=pythonengineer *\n\n📓 Notebooks available on Patreon:\nhttps://www.patreon.com/patrickloeber\n\n⭐ Join Our Discord : https://discord.gg/FHMg9tKFSN\n\nIf you enjoyed this video, please subscribe to the channel!\n\nThe code can be found here:\nhttps://github.com/patrickloeber/MLfromscratch\n\nFurther readings:\nhttps://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html\nhttps://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc\n\n\nYou can find me here:\nWebsite: https://www.python-en...	['Python', 'Machine Learning', 'numpy', 'Tutorial', 'Logistic Regression']	machine learning

2) Data Pre-processing

As seen from the image, the data we have gathered is in raw form and we cannot use it for modelling as it can have significant effect on the performance of our classification algorithms. Hence, we perform the data cleaning process using various methods from NLTK (Natural Language Toolkit) library. Text data is first converted into lower-case and removing any punctuation, white space and numeric values as they don't contribute any value. Then the text is tokenized and checked again to remove any non-alphabetic symbols like emoticons.

Stop-words are removed from the text as they are not relevant and finally the text is lemmatized to bring each word into its original root form which is necessary to improve the accuracy of text based machine-learning models. This tokenized text is again converted to string format. At last, all the rows are checked for the missing values against each data column and those with the missing values have been removed. The processed dataset reduces to 882 records as seen below.

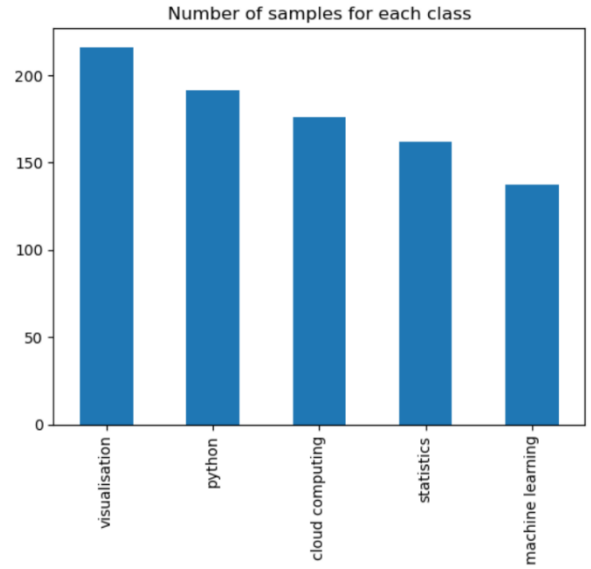
	video_title	description	tags	class
0	knn k nearest neighbor python machine learning scratch python tutorial	get free numpy handbook httpswwwpythonengineercomnumpybook machine learning scratch tutorial going implement k nearest neighbor knn algorithm using builtin python module numpy also learn concept math behind popular ml algorithm great plugins code editor write cleaner code sourcery httpssourceryaiutmsourceyoutubeutmcampaignpythonengineer notebook available patreon httpswwwpatreoncompatrickloeber join discord httpsdiscordggfhmgtkfsn enjoyed video please subscribe channel code found httpsgithubcompatrickloebermlfromscratch find website httpswwwpythonengineercom twitter httpstwittercompatloeber github httpsgithubcompatrickloeber python machinelearning sponsored link clicking additional cost instead support project thank much support	python machine learning ml numpy knn tutorial	1
1	linear regression python machine learning scratch python tutorial	get free numpy handbook httpswwwpythonengineercomnumpybook machine learning scratch tutorial going implement linear regression algorithm using builtin python module numpy also learn concept math behind popular ml algorithm great plugins code editor write cleaner code sourcery httpssourceryaiutmsourceyoutubeutmcampaignpythonengineer notebook available patreon httpswwwpatreoncompatrickloeber join discord httpsdiscordggfhmgtkfsn enjoyed video please subscribe channel code found httpsgithubcompatrickloebermlfromscratch reading httpsmlcheatsheetreadthedocsioenlatestlinearregressionhtml httpsmlcheatsheetreadthedocsioenlatestgradientdescenthtml find website httpswwwpythonengineercom twitter httpstwittercompatloeber github httpsgithubcompatrickloeber python machinelearning sponsored link clicking additional cost instead support project thank much support	python machine learning regression numpy tutorial gradient descent	1
2	logistic regression python machine learning scratch python tutorial	get free numpy handbook httpswwwpythonengineercomnumpybook machine learning scratch tutorial going implement logistic regression algorithm using builtin python module numpy also learn concept math behind popular ml algorithm great plugins code editor write cleaner code sourcery httpssourceryaiutmsourceyoutubeutmcampaignpythonengineer notebook available patreon httpswwwpatreoncompatrickloeber join discord httpsdiscordggfhmgtkfsn enjoyed video please subscribe channel code found httpsgithubcompatrickloebermlfromscratch reading httpsmlcheatsheetreadthedocsioenlatestlogisticregressionhtml httpstowardsdatasciencecomlogisticregressiondetailedoverviewcdabc find website httpswwwpythonengineercom twitter httpstwittercompatloeber github httpsgithubcompatrickloeber python machinelearning sponsored link clicking additional cost instead support project thank much support	python machine learning numpy tutorial logistic regression	1

3) Feature Exploration

In this step, we first plot the number of sample present in final dataset and as seen from the figure below, the classes are imbalanced which may have an impact on our classifiers and may introduce bias. However, the number of each class is closely similar hence, we will process with the same dataset.

Also, for data exploration purpose, we have tried to extract best 5 keywords in terms of unigrams and bigrams for each of the classes using text data each from video-title, description and tags. Figures below give an idea of the retrieved keywords. It may be noted that the text of Video-tile is contains some noise and contain keywords and terms which are non-relevant to the class.

```
In [25]: data['class'].value_counts().sort_values(ascending=False).plot(kind='bar', y='Number of Samples',
                                                                        title='Number of samples for each class')
Out[25]: <Axes: title={'center': 'Number of samples for each class'}>
```




```
# 'machine learning':
Most correlated unigrams:
------------------------------
. algorithm
. neural
. deep
. autumn
. stanford
. lecture
. tensorflow
. practical
. machine
. learning
Most correlated bigrams:
------------------------------
. nearest neighbor
. python machine
. learning autumn
. neural network
```

```
# 'cloud computing':
Most correlated unigrams:
------------------------------
. gcve
. azure
. lec
. priya
. bhanu
. service
. linode
. aws
. computing
. cloud
Most correlated bigrams:
------------------------------
. srm gcve
. beginner aws
. aws azure
. computing simplilearn
```

4) Proposed Models

In this section, we will discuss about the selection of our classifiers and how they work providing the justification of our selection. Given the popularity and resourcefulness of YouTube, there have been much research is done by using its data. We will refer to some of these previous works to draw out our selection model.

In one such study, the research is done to improve the classification accuracy of YouTube videos using user-provided tags [3]. The study compares the well-known classifiers KNN, Naïve-Bayes, Decision -tree and logistic regression algorithms. The result is then evaluated against baseline approach which uses Bags of Words technique for feature reduction.

However, this study only uses tags as the feature space and limits the generalizability of the approach on more structured texts like sentences and review.

A similar study where the YouTube videos are classified into two broad categories using video-titles and description [4] and tested for the accuracy of Naïve-Bayes, Support Vector Machine, and Random Forest decision-tree. The highest accuracy is yielded by Naïve-Bayes for their dataset. The study also evaluates the effect of label encoding on the class and concludes that label encoding has no effect on the accuracy.

In terms of text-based emotion detection, the most popular choice is SVM algorithm [5] which classifies the text into categories of emotions. Other similar work which classifies the YouTube data-based opinion mining uses decision-tree algorithms to analyse the sentiment of text features and classify into positive, negative or neutral class.

Referring to these previous works, in our study we will test the classification algorithm performance on dataset that contains both the tags as well as structured text like video titles and description. For this study, we will implement Naïve-Bayes, Support Vector Machine (SVM) and Random-Forest classifiers.

I. Naïve-Bayes

It is a classification algorithm that is based on the Bayes' theorem which works on conditional probability. Conditional probability calculates the probability of occurrence of event using its prior knowledge. It is given as,

$$P(Y|X) = P(X|Y) * P(Y) / P(X)$$

where $P(Y|X)$ is the probability of class Y on the posteriori condition x, $P(X|Y)$ is the probability of the input features X given class Y, $P(Y)$ is the prior probability of class y and $P(X)$ is the prior probability of the input features X.

It assumes that all the features are independent from each other and learns from the probability of each feature from each class using the labelled dataset. It is a simple and fast algorithm and works well with high-dimensional data hence an ideal choice to test our classification problem as our text features involves high dimensionality as we will see in the feature exploration.

There are different types of Naïve-Bayes algorithm -Gaussian Naïve-Bayes, Multinomial Naïve Bayes and Bernoulli Naïve Bayes. For the text classification problem, Multinomial Naïve -Bayes is more preferred as it is efficiently designed to handle multinomially of independent and high dimensional text features. It works on frequency of a word in given document for a class divided by the total number of words in that class. It also considers the inverse document frequency which is the logarithm of ratio of total number of documents present to the number of documents that contain the word. This method is called TF-IDF (Term frequency-Inverse Document Frequency). TF-IDF is implemented using NLP and Scikit learn

libraries in Python. Because it is simple and easy to implement, MultiNomial Naïve-bayes is often used as baseline model in text-classification.[6]

For our comparison, we will use this classifier as a baseline model to evaluate against.

II. Support Vector Machine (SVM)

SVM is a popular algorithm for high-dimensional data and their effectiveness to handle non-linear feature space. SVM can effectively overcome the problems of traditional over-fitting methods and neural network learning problems that are commonly seen at a local minimum so that they have strong generalization capabilities [7]

The aim of SVM is to find a hyperplane in feature space that can separate two classes. The idea is to represent datapoints as vectors in high-dimensional feature space. Then, the next step is to find a hyperplane that separates these datapoints in two classes. For high dimensional space, SVM find the hyperplane such that it maximizes the margin between classes, which is the distance between hyperplane and closest data points of each class being present.

The linear function is given by,

$$F(x) = sign(wT\ xi + b = 0)$$

With the value W is the weight representing the hyperplane position in the normal plane, X is the input data vector. B is the bias representing the position of the plane relative to the centre of the coordinates.[4]

However, not all data can be separated linearly in two dimensions. Therefore, the linear limiting function is then transformed into hyperplanes by using the kernel function so that the hyperplanes can separate data in higher dimensional spaces.[8]

For our text-based data, linear SVM is effective as classes are well separable hence, we will test for the linear kernel.
Linear SVM is widely used for text classification because it is computationally faster and less expensive than other non-linear classifiers used to handle high-dimensional data.

III. Random Forest

Random forest is an ensemble learning algorithm used for regression and classification. It constitutes of large consists of large number of individual decision

trees that operate as an ensemble.[9]. Basically, n number of decision trees are built on random sample datasets and trained using them. Then, their learning is combined to get the best prediction. These decision trees are set to be relatively uncorrelated so that they each can be protected from error produced by other trees. The number of trees present is the hyperparameter that affects the accuracy of the model. Each decision tree is generated via inserting random vector by selecting random value F which is given by formula, where M is the total number of features.[4]

$$F = Log2(M + 1)$$

The uncorrelated forest is created by bagging and feature randomness to ensure higher level of accuracy.

5) Model Training and Evaluation

Before training the classifiers, we have used the variant of Bag of Words, called TF-IDF to transform our text data into vectorized form. In this method we first individually fit the data of each column named as video_title, description and tags to transform into vectorized format and then constructed a single array by concatenating these three vectors horizontally into single array. The dimension of the same is as below.

```
df_tfidf = hstack([title_tfidf, desc_tfidf, tags_tfidf ])
print(df_tfidf.shape)
```

```
(882, 9633)
```

Training

The dataset is first split into train-test with 80% data used as training and 20% unseen data for testing model performance. At this point we have not reduced the feature space and training the model on complete feature space.

I. Naïve-Bayes

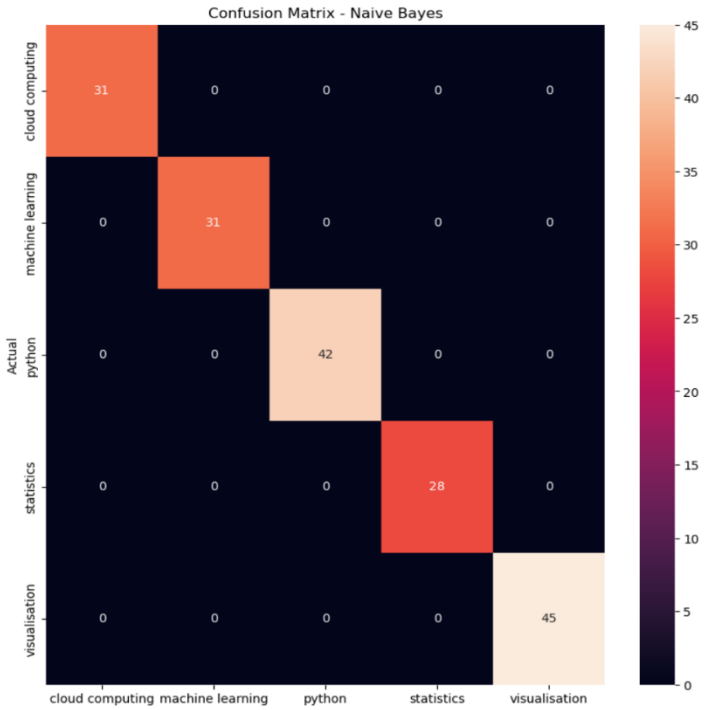
We have used multinomial Naïve Bayes on high-dimensional feature to test its accuracy. As seen below, the accuracy show is 1.0 which means our classifier covers high variance and is overfitting the data due to the high dimensional feature space.

```
from sklearn.metrics import accuracy_score

y_pred = nb.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

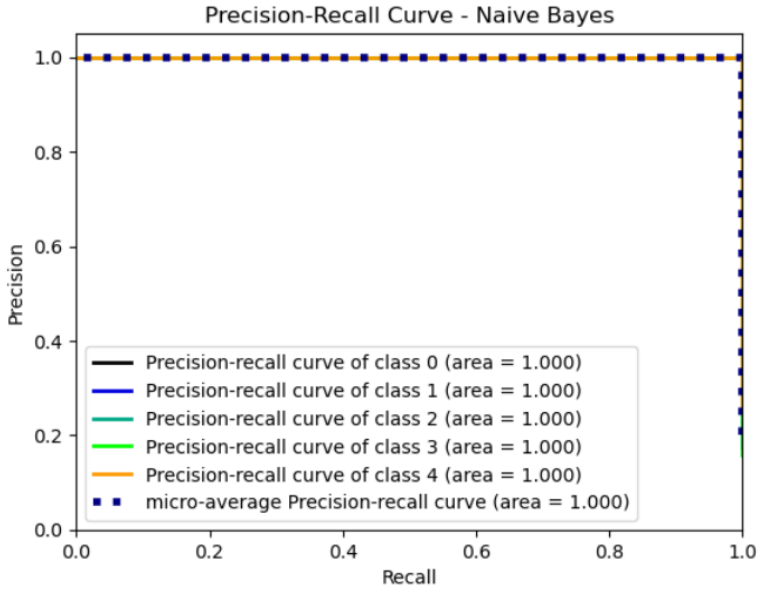
```
Accuracy: 1.0
```

Using the predicted values on test data, we have constructed the confusion matrix, classification report and precision-recall curve as shown below.



Classification report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	31
1	1.00	1.00	1.00	31
2	1.00	1.00	1.00	42
3	1.00	1.00	1.00	28
4	1.00	1.00	1.00	45
accuracy			1.00	177
macro avg	1.00	1.00	1.00	177
weighted avg	1.00	1.00	1.00	177



We have then performed cross validation to with 5-folds to get average accuracy which comes as 98%

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(nb, df_tfidf, labels, cv=5)
print("Accuracy: %0.2f (+/- %0.2f)" % (scores.mean(), scores.std() * 2))
```

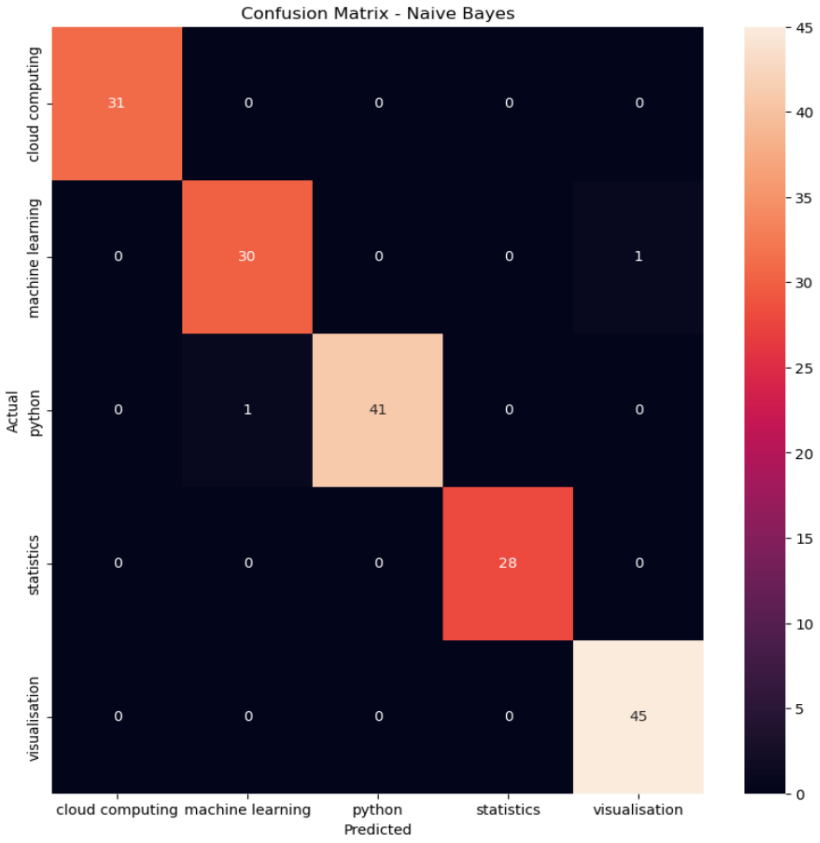
```
Accuracy: 0.98 (+/- 0.05)
```

Using this knowledge, we have tried to reduce the feature space by selecting top 1000 features by performing chi-squared test. We have again trained our classifier on only selected features. As seen, the accuracy is dropped which is close to the value received after cross-validation test.

```
# Train and evaluate the model on the selected features
nb= MultinomialNB()
nb.fit(X_train_selected, y_train)
accuracy = nb.score(X_test_selected, y_test)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.9887005649717514
```

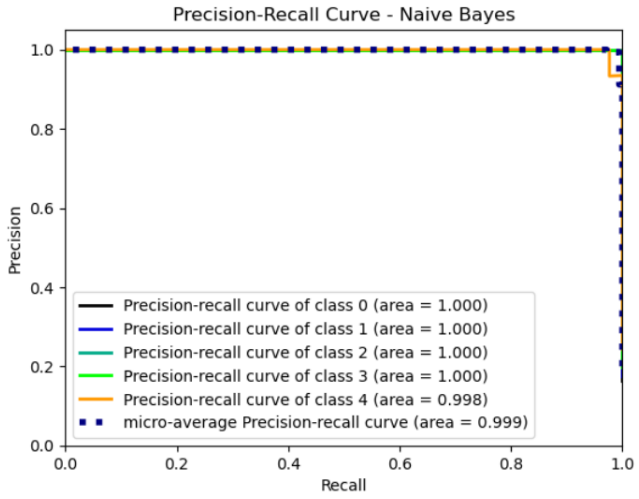
Confusion matrix, classification report and precision curve are constructed on test data of selected features as show in below diagrams.



```
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        31
           1       0.97      0.97      0.97        31
           2       1.00      0.98      0.99        42
           3       1.00      1.00      1.00        28
           4       0.98      1.00      0.99        45

    accuracy                           0.99       177
   macro avg       0.99      0.99      0.99       177
weighted avg       0.99      0.99      0.99       177
```



II. Support Vector Machine (SVM)

Similarly, we tarin SVM model using high-dimensional feature space to test its accuracy which as shown below, is 99.43% which is slightly lesser than Naïve-Bayes

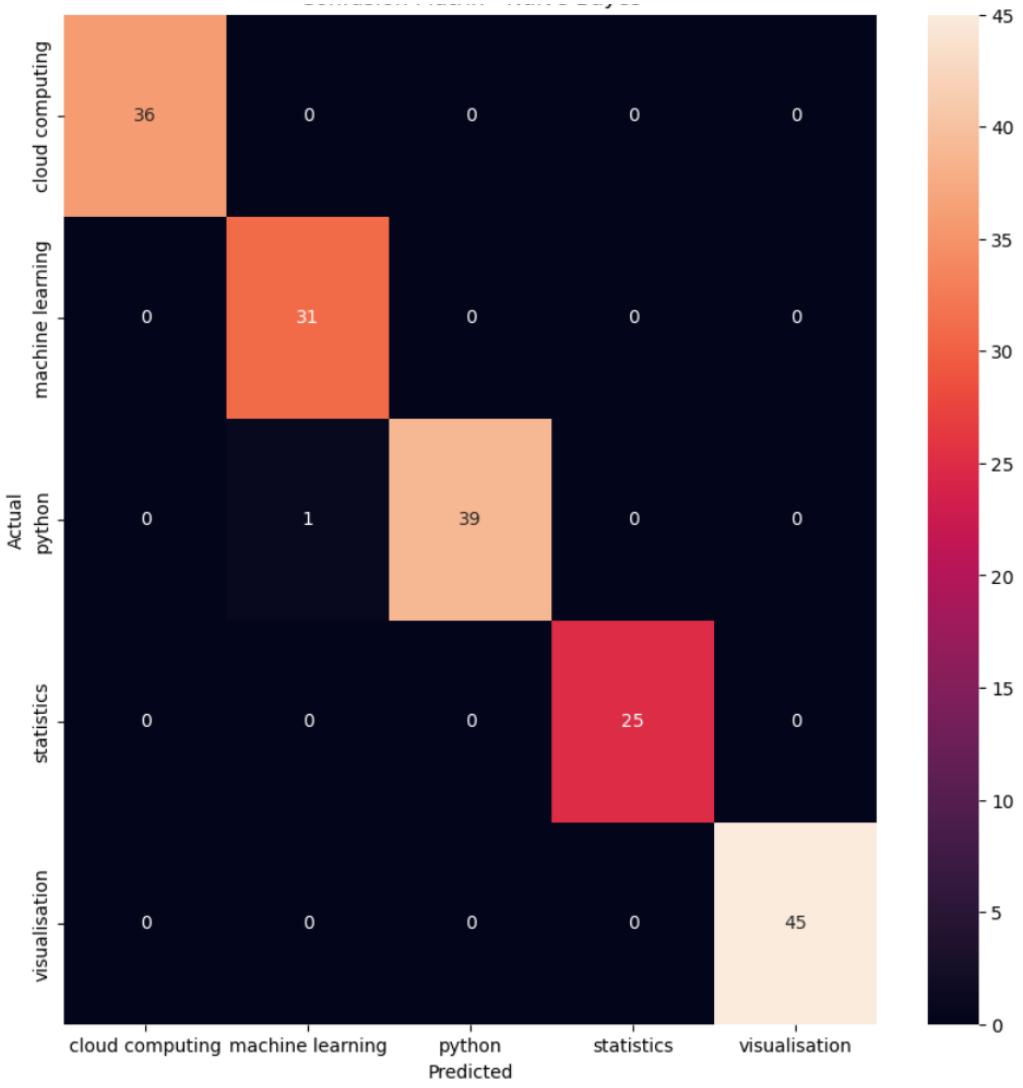
```
Accuracy: 0.9943502824858758
```

Also, it is tested using cross validation to check mean accuracy for 5-fold cross validation. The average accuracy is 96% hence our model may be overfitting and we try to improve using feature selection.

To improve model performance, we further evaluate it for top 1000 features drawn using chi-squared method. However, the resulting accuracy is again 99.43%.

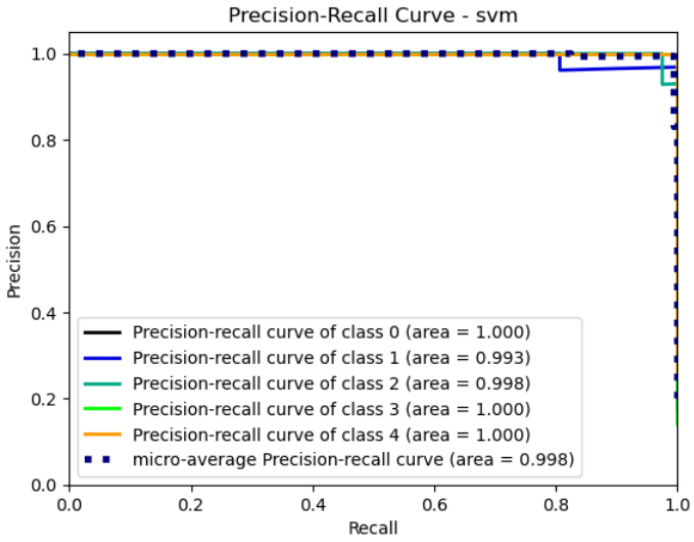
Hence, we will try to optimize the hyperparameters by tuning it. We use Grid Search method of SVM to find the best set of hyperparameters C which controls the error term and Gamma which decides the curvature of decision boundary. For this model training we use RBF kernel

instead of linear kernel.



Classification report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	36
1	0.97	1.00	0.98	31
2	1.00	0.97	0.99	40
3	1.00	1.00	1.00	25
4	1.00	1.00	1.00	45
accuracy			0.99	177
macro avg	0.99	0.99	0.99	177
weighted avg	0.99	0.99	0.99	177



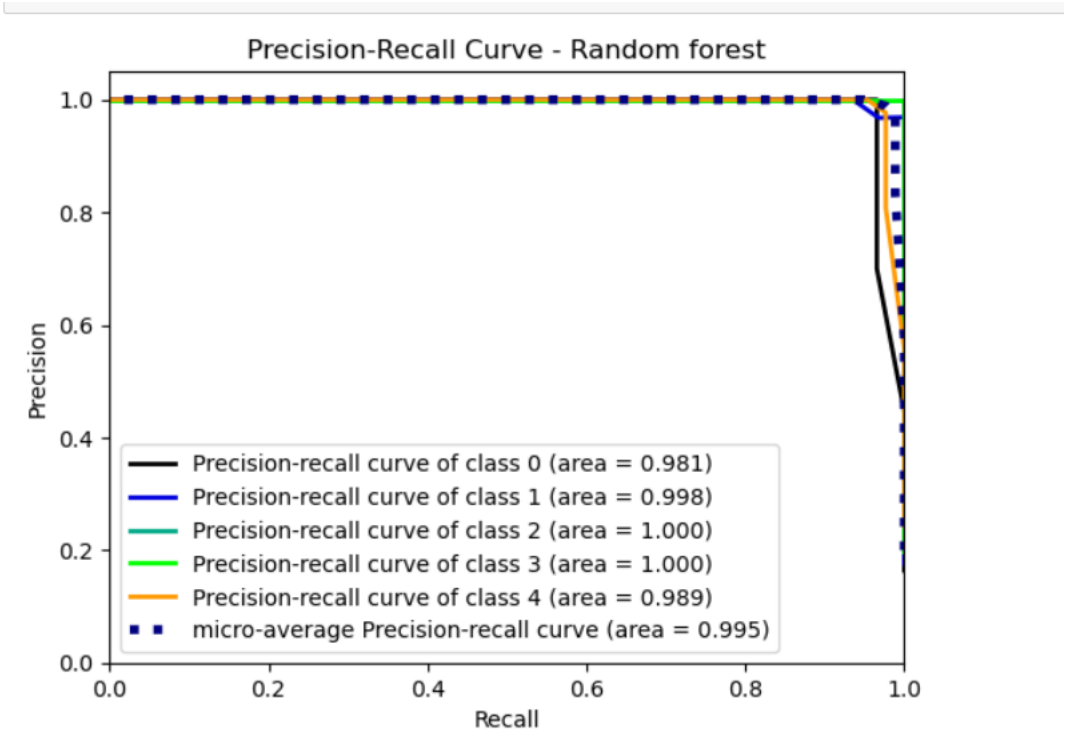
III. Random Forest

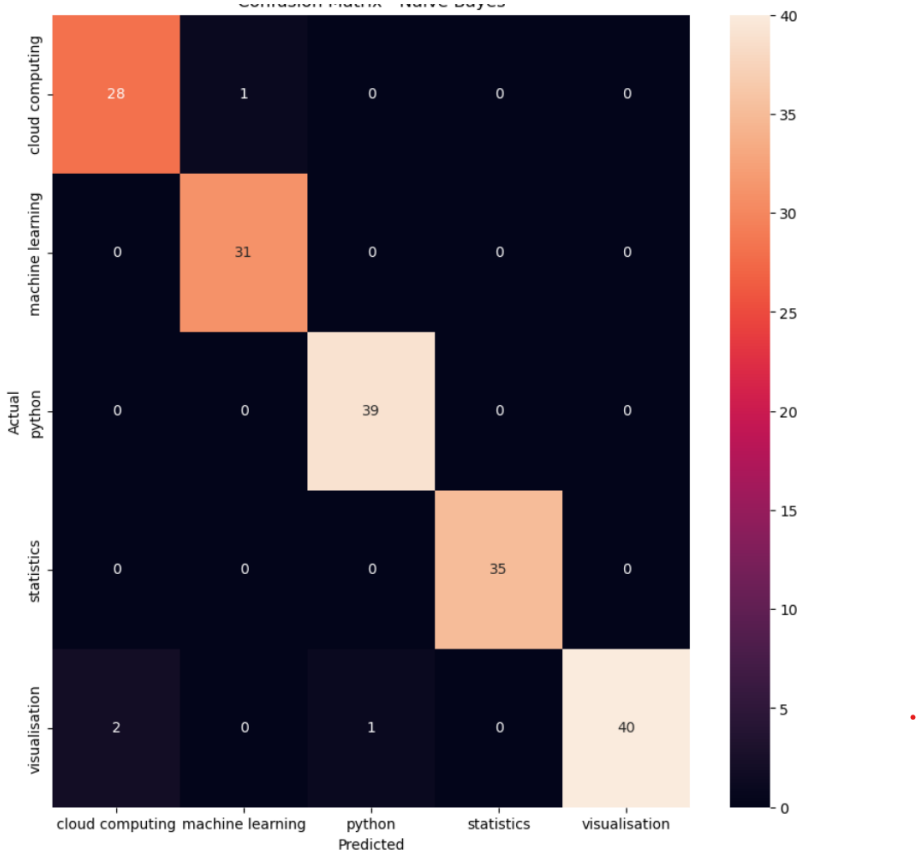
For this classifier we have used complete feature space and tried to evaluate the performance by setting different values for ensemble. We begin by choosing an arbitrarily small value , in this case n =6 and the accuracy received is 97.74% which is marginally smaller than previous two classifiers.

The resulting confusion matrix, classification report and precision recall curve are as shown.

Classification report:

	precision	recall	f1-score	support
0	0.93	0.97	0.95	29
1	0.97	1.00	0.98	31
2	0.97	1.00	0.99	39
3	1.00	1.00	1.00	35
4	1.00	0.93	0.96	43
accuracy			0.98	177
macro avg	0.98	0.98	0.98	177
weighted avg	0.98	0.98	0.98	177





Now we increase the ensemble value to arbitrary high value to cover high variance, in this case the accuracy of 100% is achieved with n estimator set to 500. The idea is to find an ideal value of ensemble parameter such that the optimal results are achieved.

We again sue cross validation to find an average accuracy using 10 folds. The result obtained is 99.57 accuracy.

```
Cross-validation scores: [1.         0.98591549 1.         1.         1.         0.97142857
 1.         1.         1.         1.        ]
Mean CV accuracy: 0.9957344064386318
```

With n_estimator= 90 we get the average accuracy of 90.43 as show in below confusion matrix and classification report

```
Confusion matrix:
[[29  0  0  0  0]
 [ 0 31  0  0  0]
 [ 0  0 39  0  0]
 [ 0  0  0 35  0]
 [ 0  1  0  0 42]]
Classification report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        29
           1       0.97      1.00      0.98        31
           2       1.00      1.00      1.00        39
           3       1.00      1.00      1.00        35
           4       1.00      0.98      0.99        43

    accuracy                           0.99       177
   macro avg       0.99      1.00      0.99       177
weighted avg       0.99      0.99      0.99       177
```

6) Result and Conclusions

	Accuracy	AUC
NAÏVE BAYES	0.99	0.999
SVM	0.99	0.998
RANDOM FORREST	0.99	1.00

It can be noted that all the classifiers work well for our classification and yields nearly the same accuracy which is rounded to 99%. The support column in each report shows the number of test samples in test for each class and even with imbalanced classed our classifier gives good results on test data as well. From, AUC it is apparent that Random Forest shows the best precision and recall. The high performance can be attributed to high dimension and variance present in our dataset. It can be further improved by normalising the dataset and regularization techniques, also Dimensionality reduction such as PCA can be applied before training the model

[1] Susarla, A., Oh, J.-H., & Tan, Y. (2012). Social networks and the diffusion of user-generated content: Evidence from youtube. Information Systems Research, 23(1), 23–41.

[2] Alexa. (2021). Alexa - top sites. Retrieved April 01, 2021 from https:// www. alexa. com/ topsi tes/

[3] Mahapatra, Amogh & Kapoor, Komal & Kasturi, Ravindra & Srivastava, Jaideep. (2013). Improving classification accuracy of youtube videos by exploiting focal points in social tags. Electronic Proceedings of the 2013 IEEE International Conference on Multimedia and Expo Workshops, ICMEW 2013. 1-6. 10.1109/ICMEW.2013.6618382.

[4] Amanda, Riyan, and Edi Surya Negara. "Analysis and implementation machine learning for youtube data classification by comparing the performance of classification algorithms." Jurnal Online Informatika 5.1 (2020): 61-72.

[5] Chaffar S, Inkpen D. Using a Heterogeneous Dataset for Emotion Analysis in Text. ACM Trans Asian Lang Inf Process. 2006;5(2):165-183.

[6] Xu, Shuo & Li, Yan & Zheng, Wang. (2017). Bayesian Multinomial Naïve Bayes Classifier to Text Classification. 347-352. 10.1007/978-981-10-5041-1_57.#

[7] L. Cunhe and W. Chenggang, “A new semi-supervised support vector machine learning algorithm based on active learning,” in 2010 2nd International Conference on Future Computer and Communication, Wuhan, China, 2010, pp. V3-638-V3-641, doi: 10.1109/ICFCC.2010.5497471.

[8] M. Hofmann, “Support Vector Machines — Kernels and the Kernel Trick,” Notes, vol. 26, no. 3, Art. no. 3, Jun. 2006.

[9] https://towardsdatascience.com/understanding-random-forest-58381e0602d2