Machine Learning Engineer Nanodegree

DETECT A YOUTUBE COMMENT AS SPAM

Hitesh Gupta
September, 2018

I. Definition

YouTube is one of the largest video sharing platforms in the world with more than 1.5 billion users in the world which is more than number of households that owns television. As of 2017 over 400 hours of videos were being uploaded on YouTube every single minute. The videos range from many topics like educational, funny, controversial, entertainment, and etc.

Users can also make advertising revenue through their videos content (you will have to satisfy YouTube policies and conditions before a user can make advertising revenue. YouTube policies and conditions are out of scope for this project). This is an important aspect for companies that produces original content or publishing companies who have enough number of subscribers and enough content (videos) that they can leverage it to monetize from YouTube.

Obviously, advertisers would want to showcase their brand via advertisement (with ads shown while watching YouTube videos) on YouTube videos which are popular, where the advertisers think they can reach a large group of people who are engaged with the video content, i.e. the user engagement can be quantified by the video completion rate, number of views, number of comments, comments written by the users, etc.

Users engagement can also directly be linked to the amount of money that a publishing company can ask an advertiser. Since YouTube has over a billion users and many of them leaving comments often on videos it becomes necessary to have a solution in hand that can help both the publishing and the advertising company to analyze whether these comments are spam or not. Basically, understanding whether users are really engaged with the content.

This project will help the publishing organization (can also be used by any advertising organization or agency) understand whether the comment written by the users on the video content, how reliable are they to design a metrics that can be shared with the advertiser to make advertisement pricing decision.

I wasn't able to find any thorough research similar to this topic. However, an interesting paper "Detecting Spammer in YouTube: A study to find spam content in a video platform" by P. Sai Kiran discusses how the author uses SVM method to identify if a given user could be a potential spammer. The author was able to perform the classification with 82% accuracy. I tried implementing SVM in my project as well but it fails to classify with a good level of accuracy.

Project Overview

The project tries to identify whether a comment on YouTube is spam or not using Bag of Words and Decision Tree classifier. I work for a publishing company and we produce YouTube videos and allows pre-roll and mid- roll advertising i.e. advertisement seen before the beginning of the video is called pre-roll advertising and advertisement that is shown in the middle of the video is called mid-roll advertising. It's important from the business perspective to know if the users engagement (using comment) is relevant or not hence detecting spam can help the business generate a valid metrics to share with the advertiser for better pricing decision.

The dataset being used is obtained from the following site:

https://archive.ica.uci.edu/ml/datasets/YouTube+Spam+Collection

The dataset contains comments on some of the popular videos on YouTube. It is formatted in such way that each row has a comment followed by a 'class' that defines if the comment is a spam (defined as class = 1) or not spam (defined as class = 0)

Problem Statement

The project tries to classify whether comment written on any YouTube video is spam or not. This will generally help any publishing organization (or any advertising organization or agency) for their business pricing decision.

I began the project by cleaning the data and preprocessing the data by using `CountVectoriser` method to format the data into numbers so the machine learning algorithm can learn from the formatted data well. I have then worked through a number of classifiers for e.g. Naïve Bayes, Random Forest, Decision Tree, Neural Network, SVC, AdaBoost to identify which classifier will work best for the given problem.

Based on the result from each classifier, I had chosen the best one to build the model. I have further validated the model to analyze if it's underfitting / a good fit / overfitting the data. I have used stratified cross validation to estimate if our model is learning well from our training set and accurately applies the training on the test set for a better result. The model is judged by accuracy score. I am expecting the accuracy score to be more than 90% and have picked the one that has the maximum accuracy.

Metrics

I have used accuracy score to determine how well the model detects whether a comment is a spam or not. Since the benchmark model's accuracy varied from 70%. – 80%, I tried to beat the accuracy generated from the benchmark model. The training set is divided into training set and cross validation (CV) set. The cross-validation set will be used to evaluate the model before it's tested on the actual test set.

I started the model by importing the dataset which has well distribution (please see the fig.1 below) of data with spam (class = 1) and not spam (class = 0). Since the target class is well distributed, using accuracy score makes it appropriate to evaluate the model.

Fig. 1

```
Total number of rows in the dataset:  1956
No of content which are classified as spam:  1005
No of content which are classified as not spam:  951
```

Accuracy score is very intuitive and a simple measure. A balanced dataset makes accuracy score a better measure to decide if a model's classification power is good enough. Given this characteristic, the YouTube dataset being used for the project is well balanced (please see Fig. 1) And thus make accuracy score a better evaluation metric for the project.

The dataset has 1956 rows in total. First step would be divide the dataset into training and test. The training set will be divided into training set and cross validation set. The CV set will be used to evaluate the model before it's tested on the actual test set.

II. Analysis

Data Exploration

The dataset consists of 1956 data points, with each data point having 5 features please see Fig. 2 for a sample of dataset). The dataset is obtained from
https://archive.ica.uci.edu/ml/datasets/YouTube+Spam+Collection

Features:
- `*COMMENT_ID*`: ID of the person who wrote the comment.
- `*AUTHOR*`: Person's name who wrote the comment.
- `*DATE*`: Date when the comment was written.
- `*CONTENT*`: Actual comment (message) written by an author.

Target Variable:
- `*CLASS*`: 1 for Spam and 0 for not Spam

Fig. 2

	COMMENT_ID	AUTHOR	DATE	CONTENT	CLASS
345	z13th1q4yzihf1bll23qxzpjeujterydj	Carmen Racasanu	2014-11-14T13:27:52	How can this have 2 billion views when there's...	0
346	z13fcn1wfpb5e51xe04chdxakpzgchyaxzo0k	diego mogrovejo	2014-11-14T13:28:08	I don't now why I'm watching this in 2014	0
347	z130zd5b3titudkoe04ccbeohojxuzppvbg	BlueYetiPlayz -Call Of Duty and More	2015-05-23T13:04:32	subscribe to me for call of duty vids and give...	1
348	z12he50arvrkivl5u04cctawgxzkjfsjcc4	Photo Editor	2015-06-05T14:14:48	hi guys please my android photo editor downloa...	1
349	z13vhvu54u3ewpp5h04ccb4zuoardrmjlyk0k	Ray Benich	2015-06-05T18:05:16	The first billion viewed this because they tho...	0

The content length varies from 100 letters to 1900 (i.e. from 2 words to 213 words) which makes content quite dispersed. The dataset consists of around 1,005 comments with spam class and 951 comments with non-spam class.

Since we have analyzed whether comments (which is the *CONTENT* column in Fig.2) is spam or not, I have converted the comments into a number format using Bag of Words concept so that the machine learning algorithm can function well. Basically, each word in the comment becomes a column in the new dataset and each row represent a data point (one of each 1956 row we have). The intersection of each row and col represent the frequency of the word count (please see fig 3. For more details).

Exploratory Visualization

The dataset being analyzed are words, I have converted the data into number format by using Bag of Words. I began the data preprocessing by implementing CountVectorizer() method that entailed cleaning our data first. This cleaning involved converting all of our data to lower case and removing all punctuation marks. CountVectorizer() has certain parameter that takes care of processing for e.g. converting to lowercase, ignoring punctuations, etc.

- ***lowercase = True***
  The lowercase parameter has a default value of True which converts all of our text to its lower case form.

- ***token_pattern = (?u)\\b\\w\\w+\\b***
  The token_pattern parameter has a default regular expression value of (?u)\\b\\w\\w+\\b which ignores all punctuation marks and treats them as delimiters, while accepting alphanumeric strings of length greater than or equal to 2, as individual tokens or words.

- ***stop_words***
  The stop_words parameter is set to english which will remove all words from our document set that match a list of English stop words which is defined in scikit-learn.

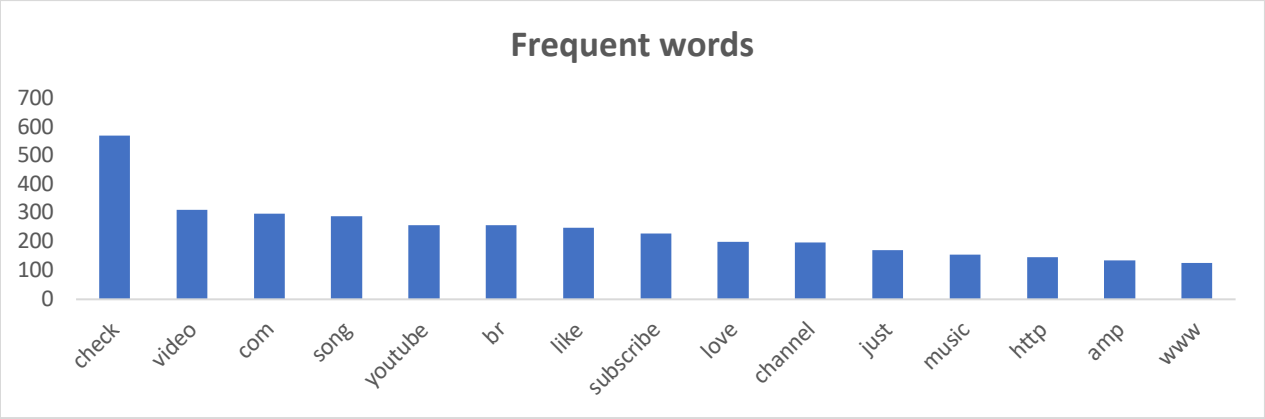
The dataset after processing is saved in a format where each word in the comment becomes a column in the new dataset and each row represent a data point (one of each 1956 row we have).

The intersection of each row and col represent the frequency of the word count. For e.g. in Fig.3 below the word `Walmart` appears in the row 1909 once.

Fig. 3

	walmart	wan	wanderfol	wank	wanna	want	wanted	wants	war	waratel	...
1900	0	0	0	0	0	0	0	0	0	0	...
1901	0	0	0	0	0	0	0	0	0	0	...
1902	0	0	0	0	0	0	0	0	0	0	...
1903	0	0	0	0	0	1	0	0	0	0	...
1904	0	0	0	0	0	0	0	0	0	0	...
1905	0	0	0	0	0	0	0	0	0	0	...
1906	0	0	0	0	0	0	0	0	0	0	...
1907	0	0	0	0	0	0	0	0	0	0	...
1908	0	0	0	0	0	0	0	0	0	0	...
1909	1	0	0	0	1	1	0	0	0	0	...
1910	0	0	0	0	0	0	0	0	0	0	...
1911	0	0	0	0	0	0	0	0	0	0	...
1912	0	0	0	0	0	0	0	0	0	0	...
1913	0	0	0	0	0	1	0	0	0	0	...
1914	0	0	0	0	0	0	0	0	0	0	...
1915	0	0	0	0	0	0	0	0	0	0	...
1916	0	0	0	0	0	0	0	0	0	0	...

It's also interesting to see what kind of words were frequently used in the dataset (please see fig below). For e.g. the word "check" was used more than 500 times in the whole dataset. Most the comments that had the word "check" were spam. For e.g. here is a sample message with word "check" in it "*Check out this playlist on YouTube*". Similarly, the word "subscribe" appeared more than 200 times. Here is a sample message "*subscribe to my channel*".



Algorithms and Techniques

I have Split the dataset into a training and testing set by using the train_test_split method in sklearn.

X_train: is the training data for the 'CONTENT' column.
y_train: is the training data for the 'CLASS' column
X_test: is the testing data for the 'CONTENT' column.
y_test: is the testing data for the 'CLASS' column

Then, I have performed the following 2 steps:

- Firstly, we will fit and transform our training set (X_train) using CountVectorizer() to a matrix form.
- Secondly, we will transform our test set (X_test) that will return a matrix.

Since, train_test_split just splits the model only once, we need to be sure that the mean accuracy is high across multiple splits, so I have used cross_val_score with 10 different splits.

Since, cross_val_score takes the entire dataset i.e. X (where X is 'CONTENT' column) and y (where y is the `CLASS` column) values directly (not after splitting, as cross_val_score takes care of splitting the data), I have performed fit_tranform on the entire dataset `X` using CountVectorizer. Once the data was transformed, I then ran a Decision Tree classifier to train the model.

Decision Tree uses the concept of information gain to identify what feature / attribute can be used to further divide the dataset. This is performed using entropy which measures how closely related or homogenous a subset of the dataset is. A higher entropy will have a measure of '0'. Information gain tries to maximize the difference between the parent node (or the attribute) entropy and the average of child node entropy. The node with the maximum information will then be used to further divide the dataset. I also performed Grid Search technique for hyper parameterization to identify the best parameters values to be used to avoid overfitting. Parameters used are 'max_depth' and 'min_sample_leaf'. Without grid search the model might run into a problem of learning the whole training set as is i.e. leaving no room for generalization therefore causing a problem for overfitting. Based on Grid Search technique the model will get the best result with max_depth of 50 and a default min_sample_leaf. Overall the accuracy of the model using the technique above is approximately 94%.

III. Methodology

Data Preprocessing

The dataset being analyzed are words, I have converted the data into number format by using Bag of Words. I will begin data preprocessing by implementing CountVectorizer() method that entailed cleaning our data first. This cleaning involved converting all of our data to lower case and

removing all punctuation marks. CountVectorizer() has certain parameter that takes care of processing for e.g. converting to lowercase, ignoring punctuations, etc.

- ***lowercase = True***
  The lowercase parameter has a default value of True which converts all of our text to its lower-case form.

- ***token_pattern = (?u)\\b\\w\\w+\\b***
  The token_pattern parameter has a default regular expression value of (?u)\\b\\w\\w+\\b which ignores all punctuation marks and treats them as delimiters, while accepting alphanumeric strings of length greater than or equal to 2, as individual tokens or words.

- ***stop_words***
  The stop_words parameter is set to English which will remove all words from our document set that match a list of English stop words which is defined in scikit-learn.

I have Split the dataset into a training and testing set by using the train_test_split method in sklearn.

X_train: is the training data for the 'CONTENT' column.
y_train: is the training data for the 'CLASS' column
X_test: is the testing data for the 'CONTENT' column.
y_test: is the testing data for the 'CLASS' column

I will perform the following 2 steps:

- Firstly, we will fit and transform our training set (X_train) using CountVectorizer() to a matrix form.
- Secondly, we will transform our test set (X_test) that will return a matrix.

Train_test_split just splits the model only once, we need to be sure that the mean accuracy is high across multiple splits, so we will use cross_val_score with 10 different splits.

Since cross_val_score takes the entire dataset i.e. X and y values directly (not after splitting as cross_val_score takes care of splitting the data), I have performed fit_tranform on the entire dataset `X` using CountVectorizer.

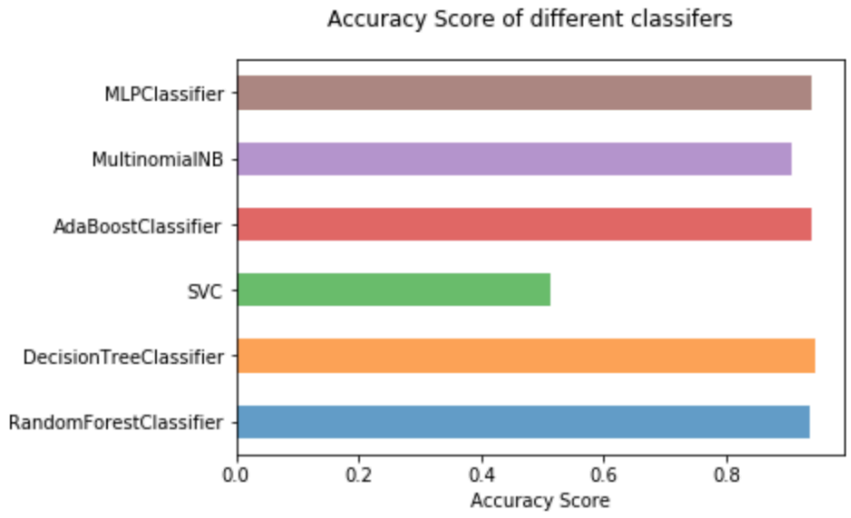
Implementation

As discussed in the previous section the data was processed first using CountVectorizer() method. The dataset was passed through multiple classifier (to identify which classifier will have a better accuracy score) like Decision Tree, Random Forest, SVM, AdaBoost, Multinomial Naïve Bayes, and Neural Network. The accuracy score from each classifier was measured using cross_val_score. Fig.4 below shows the accuracy score for each classifier.

Fig. 4

```
Accuracy Scores are as follow:

RandomForestClassifier : 93.66 %
DecisionTreeClassifier : 94.37 %
SVC : 51.38 %
AdaBoostClassifier : 93.81 %
MultinomialNB : 90.59 %
MLPClassifier : 93.96 %
```



Based on the accuracy score, I have picked Decision Tree classifier for further analysis.

The dataset contained 'CONTENT' which was converted to bag of words using CountVectorizer() technique and 'CLASS' consisted of the labels used for classifying spam or not spam.

Based on the accuracy score, the obvious choice was Decision Tree Classifier. Before I moved to perform Grid Search, I did some validation on the model using scikit-learn validation_curve method (please see fig. 5) to see if the model is underfitting / just right / over fitting the data. The interesting and also challenging part about validation_curve method was to choose what parameter to pick and the range of the parameter to use for the analysis. Since, I have used Decision Tree, max_depth became a clear choice as one wouldn't want the tree to be a lot deeper as the model then might run into overfitting scenario.

Second test I performed before Grid Search was to see the impact of varying training size on the accuracy score. The expectation was that the cross-validation score along with training score should ideally be increasing with increase in training size. It's an interesting plot (please see fig. 6) to look at as it can save computation time if one is using training size which isn't providing any benefit as the scores aren't improving.

Finally, I performed Grid Search technique for hyper parameterization to identify the best parameters values to be used to avoid overfitting. Parameters used are 'max_depth' and 'min_sample_leaf'. Without grid search the model might run into a problem of learning the whole training set as is i.e. leaving no room for generalization therefore causing a problem for overfitting. Based on Grid Search technique the model will get the best result with max_depth of 50 and a default value for min_sample_leaf. Overall the accuracy of the model using the technique above is approximately 94%.

Refinement

Once the classifier was picked i.e. Decision Tree classifier, I have implemented grid search technique for hyper parameterization to identify the best parameters that can be used to further improve the result.

The model was further refined using pipeline technique. Let say that if we start getting new comments we will have to run the comment through CountVectorizer followed by the classifier (in the case above Decision Tree Classifier). Therefore, I have combined both the steps to avoid missing any information while building the model. This is done by using a feature provided by scikit-learn called a Pipeline. Some of the benefits for using Pipeline are:

Convenience and encapsulation:
We only have to call fit and predict once on the data to fit a whole sequence of estimators.

Joint parameter selection:
We can grid search over parameters of all estimators in the pipeline at once.

Safety:
Pipelines help avoid leaking statistics from the test data into the trained model in cross-validation, by ensuring that the same samples are used to train the transformers and predictors.

Pipeline technique helped to combine both CountVectorizer() and Decision Tree Classifier. Since both the techniques were combined it became easy to apply grid search. The following parameters were tested to get the best estimator.

```
parameter_pipe ={
    'count_vector__ngram_range': ((1,1), (1,2), (1,3)), # unigram, bigram, or trigram
    'count_vector__stop_words': ('english', None),
    'count_vector__max_features': (None, 1000, 2000, 3000),
    'clf__max_depth': (10, 20, 30, 50, 70, None),
    'clf__min_samples_leaf': (1, 5, 10, 15)
}

grid_search_pipe = GridSearchCV(estimator = pipe, param_grid = parameter_pipe, scoring = 'accuracy')

grid_search_pipe.fit(X, y)
```

As can be seen from the code above, I tried testing if I should use

- ***Count_vector_ngram_range:*** unigram, bigram, or trigram words
- ***Stop_words:*** will using 'English' as stop-word or not using any stop-word will produce a better result
- ***Max_features:*** Considers top max_features ordered by term frequency across the corpus
- ***Max_depth:*** depth of the tree for the Decision Tree Classifier
- ***Min_sample_leaf:*** The number of min sample required to further break the nodes into leaf.

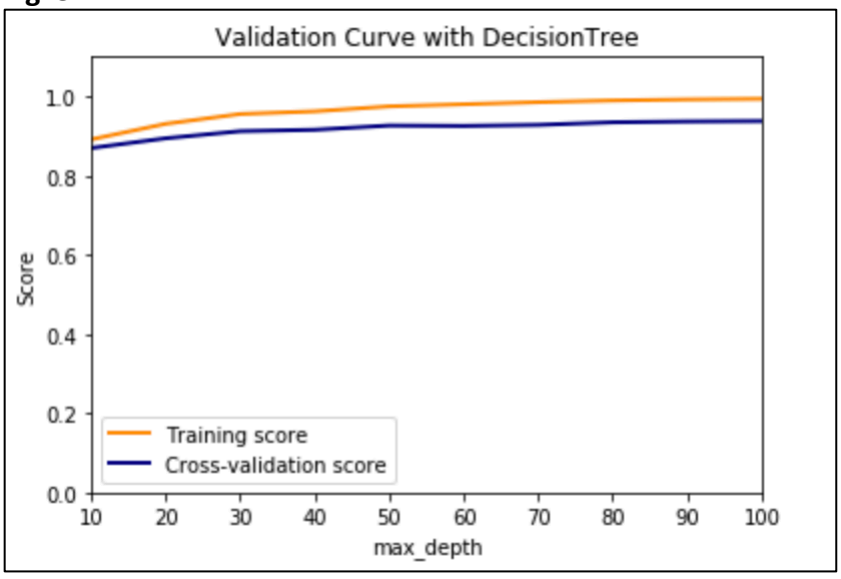
The best estimator chosen by grid search that provided that the best results were unigram, no stop-words, max_feature to be used as 1000, max_depth of the tree to be 50 and the default value for the min_sample_leaf. Based on this value the overall model accuracy comes out to be approximately 95% which is one percent point more than the model used before refinement.

IV. Results

Model Evaluation and Validation

The model was validated using validation curve. Since Decision Tree Classifier gives the best accuracy we will use Decision Tree Classifier for validation. To validate whether the model is underfitting / just right / overfitting the data, I have used validation curve for Decision Tree Classifier by varying `max_depth` parameter. We can see that both the training score as well validation score keeps increasing with the increase in our parameter suggesting that the model is neither underfit nor overfit. Please see fig. 5 below.

Fig. 5

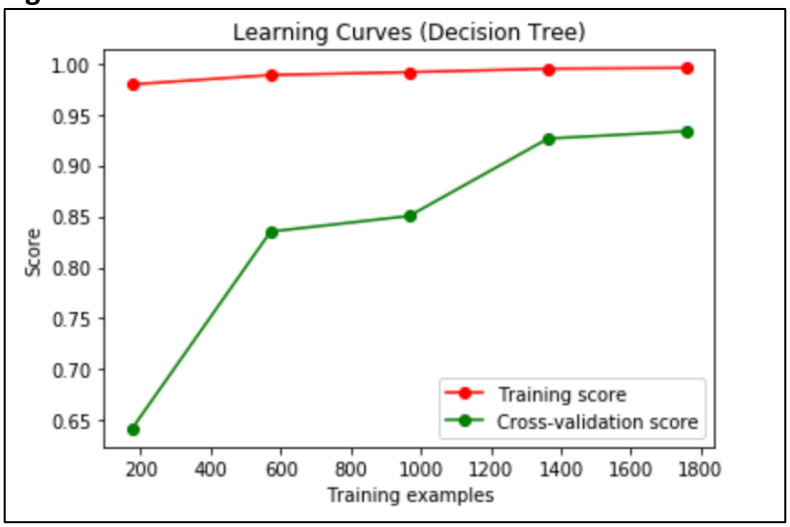


The final classifier as discussed in the previous section was chosen based on accuracy score.

Also, I tested that what would be the impact on of changing the number of training dataset on the training score and cross validation score. We can see that with the increase in training data

size both training score and validation score keeps increasing. This suggests that the model would have benefitted more by increasing the dataset. Please see fig. 6 below.

Fig. 6



V. Conclusion

Reflection

I began by cleaning the data and preprocessing the data by using CountVectoriser method to format the data into numbers so the machine learning algorithm can learn well from the formatted data.

I have worked through a number of classifiers for e.g. Naive Bayes, Random Forest, Decision Tree, Neural Network, SVC, AdaBoost to identify which classifier works the best given the problem. Clearly, the accuracy score by Decision Tree proved to be greater than other classifier and hence making Decision Tree Classifier as our choice of model.

I have used Grid Search for hyper parameterization to understand what best combination of various parameters provided by Decision Tree can be used to increase our accuracy.

Later, I plotted the validation curve to understand if our model is a good fit / under fitting / over fitting our data. We saw that the training and cross validation scores were appropriately high throughout the different parameter values.

Finally, I have improved (or upgraded) the model by introducing pipeline. Without pipeline for any new dataset we will have to run it through CountVectorizer and then run through our classifier. During this process we might have missed critical steps. Therefore, with the usage

of pipeline I made sure that the dataset is passed through all the sequences (i.e. CountVectorizer, Classifier, Grid Search) while maintaining the same training and testing dataset.

The project is really interesting in a way that it has a unique property that the data was well distributed making accuracy score as a good measurement to evaluate the model. Some of the key challenges while working on the model were:

- To come up with a process that can be used to evaluate and validate the model. Clearly, using validation curve and learning curve technique came handy with scikit-learn. It was helpful to understand that model built was not overfitting or underfitting the data.
- Since there were multiple steps involved in the project i.e. transforming the data into bag of words using CountVectorizer() then splitting the data then finally applying the classifier. Using each step separately would have proved to be inefficient and there might be a chance of data leakage i.e. we wouldn't be able to monitor if the sample set of training data or testing data were used across the process. Hence to avoid this using `pipeline` technique was incorporated. It became easy to apply grid search at one time and prevent the data statistics from leakage (for benefits of using pipeline please see 'Refinement' section).

After overcoming all the challenges and by following all the above steps, I was able to create a model that identified whether a comment on a YouTube video is a spam or not with approximately an excellent score of 95% accuracy.

Improvement

The model discussed above can further be improved by following the steps below.

- Increase the number of training dataset (the overall dataset). As we saw from the learning curve that the training and validation score kept increasing with the increase in the dataset. Hence, the model would benefit by increasing the number of data points.
- Multiple classifiers were used before deciding which classifier to pick for further analysis using Grid Search. All the classifiers were used with their default values. If we could have done grid search on each we might have received different accuracy score estimate (or probably better score).

The above model also holds a great property to be an input to another model. For example, let's say once the comments are classified as spam and not spam, then the non-spam comments can be fed into a model that can analyze the sentiment of the comments i.e. positive / negative comments. Overall the model did a pretty good job in classifying the comments.

References

https://www.youtube.com/watch?v=6dbrR-WymjI

https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6

Book: Python for Data Analysis by Wes McKinney

Book: Python Artificial Intelligence by Joshua Eckroth

https://www.iosrjen.org/Papers/vol5_issue7%20(part-4)/E05742630.pdf

Book: Introduction to Algorithmic Marketing: Artificial Intelligence for Marketing Operations by Katsov, Ilya.

Book: Hand on Machine Learning using Scitkit-Learn and Tensor Flow by Aurélien Géron