Automatic Classification of Informative YouTube Comments on Coding Tutorial Videos: A Replication Study

Alikhan Makhamadaliev
CSC 3400, Fall 2026
Belmont University
Nashville, TN, USA

***Abstract*—User comments on YouTube coding tutorials can contain useful feedback for content creators, but most comments do not provide technical information about the video. This report replicates the comment classification experiment of Poché et al. to determine whether informative, content-relevant comments can be identified automatically. Using their dataset of 6,000 manually labeled comments from 12 coding tutorial videos, four classifiers were trained on TF-IDF features: Naïve Bayes, a linear Support Vector Machine (SVM), a Decision Tree, and a Random Forest. Each classifier was evaluated with stratified 10-fold cross-validation using accuracy, precision, recall, and F1 score, and the area under the ROC curve (AUC) was computed for each. The SVM achieved the highest F1 score (0.774) and AUC (0.927) for the content concern class, which closely matches the F1 score of 0.77 reported in the original study. Naïve Bayes achieved the highest precision but lower recall, Random Forest achieved an AUC close to that of the SVM but a lower F1 score at the default decision threshold, and the Decision Tree performed worst on all metrics. The results support the original study's finding that informative comments can be identified automatically, although a substantial share of them is still missed.**

***Index Terms*—text classification, YouTube comments, support vector machines, naïve Bayes, random forests, replication**

I. Introduction

YouTube has become a common resource for learning programming [1, p. 196], and viewers often use the comment section to ask questions about a tutorial, point out errors, and request new content [1, p. 198]. These comments can help content creators improve their videos, but they are mixed with a much larger number of comments that contain praise, unrelated discussion, or spam. Poché et al. found that only about 30% of a sample of 6,000 comments from coding tutorial videos contained information relevant to the video content [1, p. 198]. For popular videos, reviewing every comment manually is impractical [1, p. 197], which motivates automatic classification.

This report replicates the classification portion of that study. The research question is the same as in the original work: *Can informative content-relevant comments be automatically identified and classified?* To address this question, four machine learning classifiers (Naïve Bayes, SVM, Decision Tree, and Random Forest) were implemented in Python with scikit-learn [2] and evaluated using 10-fold cross-validation with accuracy, precision, recall, and F1 score. As additional analyses, the AUC and ROC curve were computed for each classifier, and the results were compared with those reported in the original paper.

The remainder of this report is organized as follows. Section II describes the dataset, and Section III describes the preprocessing, classifiers, and evaluation procedure. Section IV presents the results for each classifier, and Section V presents the ROC curves and AUC values. Section VI analyzes and compares the classifiers, Section VII compares the results with the original study, Section VIII discusses limitations, and Section IX concludes.

II. Dataset

The dataset was released by Poché et al. and consists of 6,000 comments, with 500 sampled at random from each of 12 coding tutorial videos covering HTML, CSS, C++, C#, Java, Python, MySQL, and Android. The comments were collected on September 6, 2016. Each comment was labeled by five annotators, and the final label was determined by majority vote [1, p. 198]. Each comment belongs to one of two classes:

- *Content concerns*: questions or concerns about the video content, reports of errors in the video, requests for future content, or suggestions for improving the tutorial.
- *Miscellaneous*: all other comments, such as praise, insults, or spam, which do not provide technical information about the video.

In this report, content concerns are treated as the positive class.

The dataset is distributed as one folder per video. Within each folder, content concerns and miscellaneous comments are stored in separate text files with one comment per line, so the label of each comment is determined by the file that contains it. The file names are not fully consistent across folders (two folders use `conerns.txt`, and one uses `other.txt` instead of `others.txt`), so the loading code identifies each file by the beginning of its name.

Based on the per-video counts reported in the original paper, the dataset should contain 1,827 content concerns and 4,173 miscellaneous comments [1, p. 198]. The loaded data matched the 1,827 content concerns exactly for every video. The miscellaneous files contained 4,160 comments and 13 blank lines. The cause of the blank lines is not documented;

one possibility is that these comments became empty during the original authors' text cleaning, for example if they contained only emojis. The blank lines were excluded, leaving 5,987 comments, of which 30.5% are content concerns and 69.5% are miscellaneous. No comment text appears with both labels, although 85 comments are exact duplicates of another comment.

III. Methodology

A. Preprocessing and Feature Extraction

Each comment was converted to lowercase and represented as a TF-IDF vector using scikit-learn's `TfidfVectorizer` [2]. The following settings were chosen to follow the preprocessing decisions described by Poché et al.:

- *Sublinear term frequency* (`sublinear_tf`): each term frequency is replaced with $1 + \log(tf)$, which is similar to the $\ln(1 + tf)$ weighting used in the original study [1, p. 199].
- *Minimum document frequency of 2* (`min_df = 2`): words that appear in only one comment are removed, as in the original study, since such words are unlikely to carry generalizable information [1, p. 199].
- *No stemming or stop-word removal*: the original study reported that removing stop words reduced SVM performance, which the authors attributed to content concerns often being phrased as questions that rely on words such as "why" and "what" [1, p. 200]. Its best SVM configuration also did not use stemming.

The vectorizer and classifier were combined in a single scikit-learn `Pipeline`, so the vocabulary and IDF weights were fit only on the training folds within each cross-validation split. This prevents information from the test fold from influencing the features, which could otherwise inflate the reported scores.

B. Classifiers

Four classifiers were evaluated, all using scikit-learn [2]:

- *Naïve Bayes*: `MultinomialNB` with default Laplace smoothing ($\alpha = 1$), the same multinomial variant used by Poché et al. [1, p. 199].
- *SVM*: `LinearSVC`, a linear SVM with regularization parameter $C = 1$.
- *Decision Tree*: `DecisionTreeClassifier` using Gini impurity and no maximum depth.
- *Random Forest*: `RandomForestClassifier` with 200 trees. Each tree is trained on a bootstrap sample of the training data and considers a random subset of features at each split, and the forest averages the predicted probabilities of its trees.

Default hyperparameters were used apart from the settings listed above, and no hyperparameter tuning was performed. A fixed random seed (`random_state = 42`) was used for the SVM, Decision Tree, and Random Forest so that the results are reproducible.

C. Evaluation Procedure

All classifiers were evaluated with stratified 10-fold cross-validation (`StratifiedKFold`, shuffled, random seed 42). Stratification preserves the class proportions in each fold, which is important because content concerns make up only about 30% of the data. In each fold, the model was trained on the remaining nine folds and evaluated on the held-out fold. Results are reported as the mean and standard deviation across the 10 folds.

Four metrics were used, with content concerns as the positive class. Let TP, FP, TN, and FN denote the numbers of true positives, false positives, true negatives, and false negatives. Then

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP}, \tag{2}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{3}$$

$$F_1 = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{4}$$

Precision measures the fraction of comments predicted as content concerns that actually are content concerns, and recall measures the fraction of actual content concerns that the classifier identifies. These definitions follow those used in the original study [1, pp. 199–200].

To compute AUC and plot ROC curves, `cross_val_predict` was used with the same folds to obtain an out-of-fold score for every comment. Naïve Bayes, Decision Tree, and Random Forest provide predicted probabilities for the positive class. Since `LinearSVC` does not output probabilities, its decision function (the signed distance from the separating hyperplane) was used as the SVM score. The AUC was then computed from these scores across all 5,987 comments.

IV. Results

Table I summarizes the 10-fold cross-validation results for all four classifiers. The following subsections describe the results of each classifier.

A. Naïve Bayes Results

Naïve Bayes achieved an accuracy of 0.851, precision of 0.820, recall of 0.655, F1 score of 0.728, and AUC of 0.911. It had the highest precision of the four classifiers, meaning that the comments it labeled as content concerns were usually correct. However, it identified only 65.5% of the actual content concerns. One possible explanation is the class prior: since about 70% of the training comments are miscellaneous, Naïve Bayes may favor the miscellaneous class for comments whose wording does not strongly indicate either class.

TABLE I
TEN-FOLD CROSS-VALIDATION RESULTS (MEAN ± STANDARD DEVIATION ACROSS FOLDS)

Classifier	Accuracy	Precision	Recall	F1 Score	AUC
Naïve Bayes	0.851 ± 0.008	$\mathbf{0.820} \pm 0.021$	0.655 ± 0.020	0.728 ± 0.015	0.911
SVM	$\mathbf{0.865} \pm 0.011$	0.792 ± 0.020	$\mathbf{0.756} \pm 0.026$	$\mathbf{0.774} \pm 0.019$	**0.927**
Decision Tree	0.770 ± 0.012	0.621 ± 0.019	0.628 ± 0.036	0.624 ± 0.024	0.730
Random Forest	0.852 ± 0.012	0.811 ± 0.026	0.672 ± 0.028	0.734 ± 0.023	0.923

Precision, recall, and F1 score are for the content concern (positive) class. AUC is computed from out-of-fold scores. The best value in each column is shown in bold.

B. SVM Results

The SVM achieved an accuracy of 0.865, precision of 0.792, recall of 0.756, F1 score of 0.774, and AUC of 0.927. It obtained the highest accuracy, recall, F1 score, and AUC of the four classifiers and was the only classifier to identify more than three quarters of the content concerns. Its precision was 0.02 to 0.03 lower than that of Naïve Bayes and Random Forest, while its recall was 0.08 to 0.10 higher.

C. Decision Tree Results

The Decision Tree achieved an accuracy of 0.770, precision of 0.621, recall of 0.628, F1 score of 0.624, and AUC of 0.730, which are the lowest values of the four classifiers on every metric. It also had the largest standard deviation in recall across folds (± 0.036), which suggests that its performance was more sensitive to the training data in each fold than that of the other classifiers.

D. Random Forest Results

The Random Forest achieved an accuracy of 0.852, precision of 0.811, recall of 0.672, F1 score of 0.734, and AUC of 0.923. Compared with the single Decision Tree, it improved the F1 score from 0.624 to 0.734 and the AUC from 0.730 to 0.923. Its AUC was within 0.004 of the SVM's, although its recall was 0.084 lower. A possible reason for this difference is discussed in Section VI-E.

V. AUC AND ROC CURVES

Fig. 1 shows the ROC curve for each classifier, which plots the true positive rate against the false positive rate as the decision threshold varies. The AUC can be interpreted as the probability that a randomly selected content concern receives a higher score than a randomly selected miscellaneous comment, where 0.5 corresponds to random guessing and 1.0 to perfect separation. Because AUC does not depend on a single threshold, it measures how well each classifier ranks comments independently of the cutoff used to compute the F1 score.

The SVM (AUC $= 0.927$), Random Forest (AUC $= 0.923$), and Naïve Bayes (AUC $= 0.911$) curves are close to one another and rise steeply near the origin. The SVM curve lies slightly above the other two for false positive rates between approximately 0.05 and 0.35, which is the range most relevant when only a small number of false alarms is acceptable.

The Decision Tree curve (AUC $= 0.730$) differs noticeably, consisting of two line segments joined at approximately

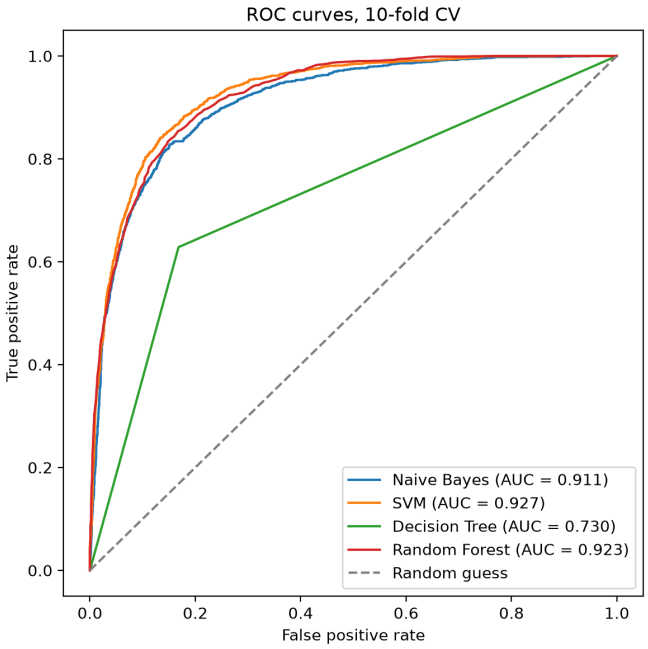

Fig. 1. ROC curves for the four classifiers, computed from out-of-fold scores in 10-fold cross-validation. The dashed diagonal represents random guessing.

$(0.17, 0.63)$. This shape is consistent with a tree that has no depth limit: such a tree tends to grow until its leaves are nearly pure, so most of its predicted probabilities are close to 0 or 1, leaving effectively one operating point. This point agrees with the values in Table I, with a recall of 0.628 and roughly 700 false positives among the 4,160 miscellaneous comments, which corresponds to a false positive rate of about 0.17.

VI. ANALYSIS AND COMPARISON OF CLASSIFIERS

A. Ranking of the Classifiers

Ranked by F1 score, the classifiers performed in the order SVM (0.774), Random Forest (0.734), Naïve Bayes (0.728), and Decision Tree (0.624). The SVM's F1 score was 0.040 higher than that of Random Forest and 0.046 higher than that of Naïve Bayes. Since the fold-to-fold standard deviations are approximately 0.02, these differences are about twice the typical variation between folds, which suggests that the SVM's advantage is not only a result of how the folds were split. A formal statistical significance test, however, was not performed. Random Forest and Naïve Bayes differed by only

0.006, which is well within one standard deviation, so the two appear roughly equivalent in F1 score, with Naïve Bayes slightly higher in precision and Random Forest slightly higher in recall.

B. Accuracy Versus F1 Score Under Class Imbalance

Because 69.5% of the comments are miscellaneous, a trivial classifier that labels every comment as miscellaneous would achieve 69.5% accuracy while identifying no content concerns. All four classifiers exceeded this baseline, but the Decision Tree's accuracy of 0.770 was only 7.5 percentage points above it. Accuracy also reduces the apparent differences between classifiers: the SVM outperformed Random Forest by 0.013 in accuracy but by 0.040 in F1 score, because most correct predictions come from the majority class. For this imbalanced dataset, the F1 score and AUC for the content concern class appear to be more informative than accuracy when comparing classifiers.

C. Effect of Sparse Text Features

YouTube comments are short, so each TF-IDF vector has only a small number of nonzero values relative to the size of the vocabulary. Poché et al. noted that informal language, such as slang, abbreviations, and misspellings, increases the number of features and makes comment vectors sparse [1, p. 200]. They also explained that SVMs tend to avoid overfitting in such high-dimensional spaces because they maximize the margin between classes [1, p. 200], which may explain why the SVM performed best in this replication as well. Naïve Bayes also produced a relatively high AUC (0.911), despite assuming that words occur independently of one another given the class, an assumption that does not hold for natural language.

D. Decision Tree Versus Random Forest

The Decision Tree splits on one feature (word) at a time and, without a depth limit, continues splitting until the training data is almost perfectly separated. This may cause it to memorize rare words that happen to occur in only one class, which would generalize poorly to unseen comments and is consistent with its low scores and high variance. The Random Forest reduces this effect by training each tree on a different bootstrap sample with a random subset of features and averaging their predictions. Averaging is expected to reduce the variance of individual trees and also produces a more continuous range of scores, which likely contributes to the large increase in AUC from 0.730 to 0.923.

E. Discrepancy Between AUC and F1 Score

Random Forest ranked comments nearly as well as the SVM (AUC of 0.923 compared with 0.927), yet its F1 score was 0.040 lower. One likely explanation is the decision threshold. The F1 score is measured at the default threshold, where a comment is labeled a content concern only if the forest's averaged probability exceeds 0.5. With 70% of the training data in the miscellaneous class, many content concerns may receive probabilities slightly below this threshold. Lowering the threshold would be expected to increase recall at some cost to precision. This was not tested in this report and remains a hypothesis for future work.

TABLE II
COMPARISON WITH POCHÉ ET AL. [1] (CONTENT CONCERN CLASS)

	Poché et al.			This study		
Classifier	P	R	F1	P	R	F1
Naïve Bayes	0.63	0.71	0.67	0.820	0.655	0.728
SVM	0.79	0.75	0.77	0.792	0.756	0.774
Decision Tree	not tested			0.621	0.628	0.624
Random Forest	not tested			0.811	0.672	0.734

P = precision, R = recall. Original values are from Table III of [1], p. 200.

F. Practical Implications

For the intended use case, a missed content concern may be more costly than a false alarm, because a creator cannot respond to a question or error report that they never see, while a false alarm only adds one more comment to review. Under this assumption, recall is more important than precision, and the SVM is the most suitable of the four classifiers, since it achieved the highest recall with only a small decrease in precision. Naïve Bayes may be preferable when false alarms are more costly, such as when automatically pinning a short list of questions for viewers to see.

To illustrate the SVM's performance, consider a video with 1,000 comments, of which approximately 305 would be content concerns given the class ratio in this dataset. Based on the SVM's average precision and recall, it would flag approximately 290 comments, of which about 230 would be content concerns and about 60 would be miscellaneous, while about 75 content concerns would be missed. The creator would therefore review about 29% of the comments while still seeing roughly three quarters of the content concerns.

VII. COMPARISON WITH THE ORIGINAL STUDY

A. Comparison of Results

Table II compares the results with the original study's Naïve Bayes and SVM configurations without stemming or stop-word removal, which are the closest to the setup used in this report.

The SVM results are very close to the original: precision, recall, and F1 score are all within 0.006 of the values reported by Poché et al. This was somewhat unexpected, since several implementation details differ. The original study used Weka's SMO implementation with a universal kernel [1, p. 199], whereas this replication used a linear SVM in scikit-learn. The original features were log-scaled term frequencies without IDF weighting or length normalization, as shown in Table II of the original paper [1, p. 200], whereas this replication used TF-IDF vectors normalized to unit length. The random assignment of comments to folds also differs. The similar F1 score despite these differences suggests that SVM performance on this dataset may depend more on the vocabulary of the comments than on the choice of kernel or toolkit, although

additional configurations would need to be tested to confirm this.

The Naïve Bayes results differ more. The F1 score of 0.728 is about 0.05 higher than the best Naïve Bayes configuration in the original study (0.68, with stemming) and about 0.06 higher than the configuration without preprocessing (0.67) [1, p. 200]. The balance between precision and recall also changed: precision was 0.19 higher (0.820 compared with 0.63), while recall was 0.055 lower (0.655 compared with 0.71).

B. Possible Reasons for the Differences

The cause of the Naïve Bayes difference was not tested directly, but the following factors may contribute:

1) *IDF weighting*: TF-IDF assigns larger values to words that occur in fewer comments and smaller values to words that occur in almost every comment, such as "the" or "video." Since Multinomial Naïve Bayes treats feature values similarly to word counts, more distinctive words may have greater influence on its predictions, which could account for the higher precision.
2) *Length normalization*: The TF-IDF vectors were normalized to unit length, so longer comments do not accumulate more evidence simply by containing more words. The original features were not normalized [1, p. 200].
3) *Implementation defaults*: Weka and scikit-learn differ in tokenization, smoothing, and the handling of unseen words, and Naïve Bayes may be more sensitive to these details than the SVM.

C. Limitations of the Comparison

Several factors limit this comparison. First, the original paper does not state whether its precision and recall refer to the content concern class or to Weka's weighted average across both classes. The close match for the SVM suggests that the values are comparable, but this cannot be confirmed. Second, the original abstract reports that the SVM achieved "an average accuracy of 77%" [1, p. 196], but the corresponding value in its results table is the F-measure [1, p. 200]; therefore, F1 scores were compared rather than accuracy, and the SVM's accuracy in this replication was 0.865. Third, the dataset used here contains 13 fewer comments because of the blank lines, which is unlikely to affect the results meaningfully. Finally, the original study evaluated only Naïve Bayes and SVM and did not report AUC, so the Decision Tree, Random Forest, and AUC results have no published values for comparison. Within these limits, the main classification result of the original study, an SVM F1 score of 0.77, was reproduced.

VIII. Limitations

This replication has several limitations. The dataset contains comments from only 12 videos collected in 2016, so the results may not generalize to other videos, programming topics, or more recent commenting behavior. All classifiers used default hyperparameters, and no decision threshold tuning was performed, so the reported results may underestimate what each classifier could achieve. The results also come from a single run of 10-fold cross-validation with one random seed, and no statistical significance tests were performed when comparing the classifiers.

IX. Conclusion

This report examined whether informative content-relevant comments on YouTube coding tutorials can be automatically identified and classified. Based on 10-fold cross-validation over 5,987 labeled comments, the results suggest that they can, with reasonable but imperfect performance. The linear SVM achieved the best results on most metrics, identifying 75.6% of content concerns with a precision of 79.2% (F1 $= 0.774$, AUC $= 0.927$), which closely matches the best result reported by Poché et al. Naïve Bayes and Random Forest ranked comments nearly as well, as shown by their AUC values, but identified fewer content concerns at the default threshold, while the Decision Tree performed worst on every metric, likely due to overfitting. Future work could test threshold tuning, hyperparameter optimization, and additional features such as word n-grams to determine whether recall can be improved.

References

[1] E. Poché, N. Jha, G. Williams, J. Staten, M. Vesper, and A. Mahmoud, "Analyzing user comments on YouTube coding tutorial videos," in *Proc. IEEE/ACM 25th Int. Conf. Program Comprehension (ICPC)*, 2017, pp. 196–206, doi: 10.1109/ICPC.2017.26.

[2] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, 2011.