VILNIUS GEDIMINAS TECHNICAL UNIVERSITY

FACULTY OF FUNDAMENTAL SCIENCES

DEPARTMENT OF INFORMATION TECHNOLOGY

Arailym Issayeva
ITfuc-21

RESEARCH PAPER

Title of the research paper: Automated sentiment evaluation of study quality feedback survey open questions answers

Supervisor in the university: Prof. Simona Ramanauskaitė

Vilnius, 2023

Abstract

This research paper presents a comprehensive evaluation of student feedback at Vilnius Tech, with a specific emphasis on analyzing open-ended responses. To streamline the evaluation process, we propose an innovative solution employing automated sentiment analysis techniques. Through rigorous testing, we found that utilizing naïve Bayes resulted in an impressive accuracy of 72.27%. By automating sentiment detection, educators can gain deeper insights into student experiences, enabling the enhancement of educational outcomes. This approach also facilitates efficient evaluation of lecturer performance and harnesses valuable suggestions from students, positively impacting the university community.

1. Introduction

In our research paper, we concentrate on evaluating student feedback obtained through the Vilnius Tech evaluation system, which collects feedback from students after each semester. Currently, only closed-ended feedback is considered, but studies indicate that this approach may be influenced by the "halo effect" and susceptible to falsification. For example, Clayson & Haley (2011) found that up to 31% of respondents admitted to providing false information on evaluation forms, compared to only 19.4% for written comments.

To gain a more comprehensive understanding of student perception, it is essential to analyze open-ended feedback. Written comments allow students to freely express themselves, providing valuable insights into their thoughts and experiences (Stupans et al., 2016). Ignoring this valuable source of information means missing out on crucial feedback.

However, the volume of written feedback can be overwhelming (Brockx et al., 2012; Rajput et al., 2016), with over 60,000 responses to analyze each semester from more than 10,000 students. Manually assessing this volume would be impractical and could introduce subjective biases. Therefore, automating the analysis of open-ended feedback is essential to enhance the evaluation process.

To address these challenges, we propose an automated solution. Our methodology includes gathering a sample of around 300 feedback sentences from exchange students. We preprocess the collected data to eliminate irrelevant information and standardize the text. Subsequently, we apply sentiment analysis techniques, utilizing Jupyter Lab as the Integrated Development Environment (IDE) while coding with Python.

The Python programming language provides various libraries for sentiment analysis techniques, enabling us to objectively assign sentiment labels (neutral, negative, or positive) to each written feedback. Through this automated approach, we can efficiently analyze the sentiments conveyed in the feedback data and systematically assess the quality of the study.

.

2. Related Works

2.1 Sentiment analysis

Sentiment analysis is a method that helps you analyze and understand the tone or emotion of a particular text (Bing, 2012), such as written survey feedback in our case, which can be positive, negative, or neutral.

Sentiment analysis is used in educational institutions to analyze open-ended textual feedback from students, providing valuable insights beyond numerical ratings. It helps uncover specific problems, challenges, and suggestions that may not be captured by Likert-scale questions. This deeper understanding enables lecturers and academic administrators to address issues and make improvements (Ren, 2023). By considering students' preferences and needs, institutions can enhance the overall learning experience by tailoring their curriculum and teaching methods accordingly.

The sentiment extraction and classification of given reviews can be primarily accomplished using two methods: the Lexicon-Based Approach and the Machine Learning Approach (Li et al., 2020). These two approaches aim to identify whether the sentiment of the reviews is positive, negative, or neutral.

2.2 Machine learning-based approach

Machine learning involves training computers to recognize sentiment patterns in written feedback through large sets of labeled data. This training data includes written feedback texts with labeled sentiment labels (positive, negative, or neutral). Then mathematical algorithms analyze the data and fine-tune their parameters to create a sentiment analysis model.Once the model is trained, it can accurately predict the sentiment of new feedback (test data), enabling efficient analysis of extensive textual data. In the provided Figure 1 below, you can see a graphical representation of how the machine learning process works for sentiment analysis.

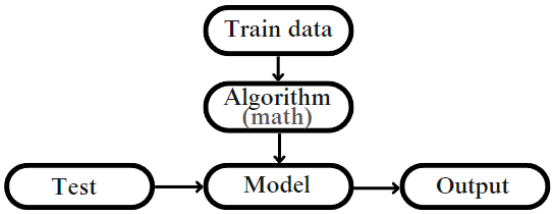


Fig. 1. *Sentiment Analysis using Machine Learning: Workflow Diagram*

According to Bonaccorso (2017), there are two main classifications for the machine learning approach: supervised learning and unsupervised learning, as shown in Figure 2 below.

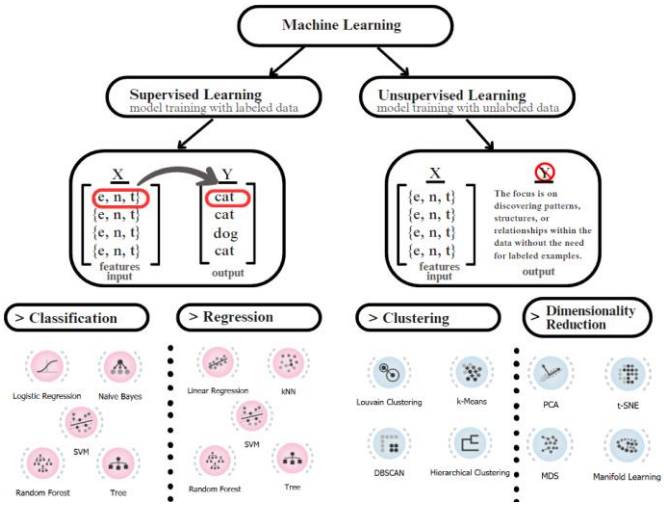


Fig. 2. *Categorization of Machine Learning-Based Approaches*

2.2.1 Supervised learning

In supervised learning, the algorithm is trained using labeled data, which means the training dataset has both its input features and associated output labels. The basic goal of supervised learning is to develop a mapping function that can correctly predict the output labels for data that is unseen, based on the features of the input data.

There are two primary types of supervised learning: Regression and Classification. In Regression, the algorithm predicts a continuous output, like house prices based on features such as area, number of rooms, and location. On the other hand, Classification involves assigning input data to predefined categories. A classic example is email spam detection, where the model classifies emails as spam or non-spam based on various features.

To further understand categorization in SL, let's look at real-life example, dividing creatures into dog or cat categories. Think of a dataset with different animal examples, each with specific features (X) like the number of eyes, the type of nose, and whether or not they have tails, as well as labels (Y) indicating whether they are cats or dogs. Using this labeled dataset (Fig. 2), we train a supervised learning model by feeding it with the features (X) and related labels (cat or dog). The model gains the ability to spot trends in features linked to each animal species throughout training.
After training, the model is able to categorize fresh instances of animals according to their features. For instance, if we input the trained model with a new instance(Xnew) with two eyes, a wet nose, and a tail, it will correctly predict that the animal falls within the "Dog" category. The model makes this prediction based on the patterns it discovered from the labeled data during training. The model is able to correctly forecast the category of new, unseen instances.

Commonly used algorithms in supervised learning for tasks like sentiment analysis include Naive Bayes, Support Vector Machines (SVM), and Random Forests.

A. Naive Bayes Classification

The Naive Bayes Classifier is a probabilistic machine learning algorithm widely used in sentiment analysis. In

this context, **features** (words) are represented as **'X,'** and sentiment **labels** (negative, neutral, positive) are represented as **'Y.'**

$$P(\text{label} \mid \text{feature}) = \frac{P(\text{feature} \mid \text{label})\, P(\text{label})}{P(\text{feature})}$$

Fig. 3. *Bayes Theorem in Sentiment Analysis*

To apply the Bayes theorem to sentiment analysis, we use the "naive" assumption, assuming each word in the text is independent of others given the sentiment label. This simplification enables efficient probability calculation. In sentiment analysis, we consider multiple features (words) in the text, but for clarity, let's initially explain using a single feature.

Suppose we already know that the word "admire" has occurred, and we wish to determine if it expresses a negative, positive, or neutral sentiment. We begin with the Bayes theorem formula, starting with the probability of being negative, denoted as P(negative | 'admire'). This is equal to the probability of 'admire' given its negative (P('admire' | negative)) multiplied by the probability of being negative (P(negative)) and divided by the probability of 'admire' occurring (P('admire')).

$$P(y \mid x) = \frac{P(x \mid y)\, P(y)}{P(x)}$$

$$P(y = -1 \mid \text{"admire"}) = \frac{P(\text{"admire"} \mid \text{negative})\, P(\text{negative})}{P(\text{"admire"})}$$

P("admire" | negative) - > Using the trained data, we determine how many times the word "admire" occurs in negative sentiment texts.
P(negative) - > We calculate it using the number of negative rows divided by the total number of rows in the trained data.

To calculate the sentiment for the word "admire," we need trained data. For example, we find the probability of the word 'admire' appearing in negatively labeled sentences (P('admire' | negative)) and the proportion of negative-labeled sentences in the training data (P(negative)). Similarly, we will calculate these probabilities for positive and neutral sentiments.

By analyzing in which sentiment category the word 'admire' predominantly appears, we can determine its sentiment. The common dividing part, P('admire'), can be eliminated from the calculation.

Let's calculate the sentiment for the word "admire."

We have 1000 rows of data, consisting of 300 negative rows, 600 positive rows, and 100 neutral rows. The probabilities are as follows:

P(y = -1) = 0.3 (Probability of a negative label)

P(y = 1) = 0.6 (Probability of a positive label)

P(y = 0) = 0.1 (Probability of a neutral label)

The probabilities of the feature "admire" belonging to each sentiment category are as follows:

Probability of the feature "admire" being negative:

0.2 * 0.3 = 0.06

Probability of the feature "admire" being positive:

0.7 * 0.6 = 0.42

Probability of the feature "admire" being neutral:

0.1 * 0.1 = 0.01

To label the sentiment, we use the formula argmax (P(y=i | x)). In our case, the highest probability is associated with a positive sentiment, so we label it as a positive sentiment.

In the previous explanation, we discussed a single feature, which was the sentiment of a single word. Now, we'll handle multiple features because we aim to determine the sentiment of a given text, which contains multiple independent words. We'll use an expanded formula that considers all the features (words) in the text.

$$P(y \mid x) \propto P(x \mid y)\, P(y)$$

$$P(x_1, x_2, x_3, \ldots x_i \mid y)$$

$$P(x_1 \mid y)\, P(x_2 \mid y, x_1)\, P(x_3 \mid y, x_2 x_1) \ldots$$

Let's break down the expression P(x1 | y) * P(x2 | y, x1) for a clearer understanding. For example, P('good' | y = 1) * P('nice' | y = 1, 'good') calculates the probability of the word 'good' appearing in positively labeled data, assuming it's positive. Then, it determines the probability of the word 'nice' appearing, given that 'good' has already appeared in a positive sentiment context. This process continues, and the words become interdependent, making the expression extensive and computationally inefficient.

We use the independent assumption of the Naive Bayes algorithm in practice to deal with this problem. We refer to it as "naive" since it implies that the data are independent of one another. This assumption enables us

to compute the probability of each word independently for each sentiment category, greatly simplifying and improving the calculating procedure.

It's important to note that the absence of a specific word in the training dataset, like "amazing," doesn't necessarily mean the sentiment is negative or neutral. By continuously training our Naive Bayes classifier with more data, we can improve its ability to handle unseen words and make more accurate predictions.

2.2.2 Unsupervised learning

In Unsupervised Learning, outcomes aren't predefined, and the system generates insights through pattern recognition from input data. It explores and identifies hidden insights independently, unlike supervised learning that relies on labeled data. Unsupervised learning includes two main types: Clustering, which groups similar data points together based on intrinsic features, and Dimensionality Reduction, which reduces the number of variables or features in the data while retaining essential information. Unsupervised learning applies to Market Segmentation, Topic Modeling, and various other fields. While sentiment analysis often relies on supervised learning, unsupervised learning is essential for pattern discovery in large datasets without prior outcome knowledge. Popular algorithms in unsupervised learning include k-means clustering, hierarchical clustering, Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE), and Gaussian Mixture Models (GMM).

2.3 Lexicon-based approach

Lexicon-Based Sentiment Analysis calculates polarity scores for positive and negative words in a text using a predefined sentiment lexicon (e.g., "good" as positive, "bad" as negative). Unlike Machine Learning methods, it doesn't need extensive labeled data for training, making it fast and efficient. This independence from training data allows for a simple and quick way to determine sentiment in texts without time-consuming model training.

Lexicon-Based Sentiment Analysis can be classified into two main types: Dictionary-Based and Corpus-Based (Rajput et al. 2016).

Dictionary-Based Lexicon: In this approach, a sentiment lexicon is created by forming a database of positive and negative words from an initial set. The lexicon may also include synonyms and antonyms. When analyzing a text, the method matches the words in the text with the entries in the sentiment lexicon to determine their sentiment polarity, whether positive, negative, or neutral. The sentiment scores of matched words are combined to calculate the overall sentiment of the text. Example: Let's calculate the sentiment score for the feedback: "I admire this lecturer; he teaches in a clear and easy-to-follow manner."

The four words "admire," "clear," "easy," and "follow" are all positive. No negative words. 12 words total.
Sentiment Score ≈ (4 - 0) / 12 ≈ 0.3.
This feedback has a sentiment score of about 0.3, which indicates a slightly positive sentiment.

In the Corpus-Based Lexicon approach, sentiment lexicons are derived from statistical analysis of a large text corpus, assigning positive and negative labels based on word co-occurrence. These lexicons are then used to calculate sentiment scores for text analysis.

2.3 Limitations of Lexicon-Based, Machine Learning, and Hybrid Approaches in Sentiment Analysis

Sentiment analysis approaches have limitations that can impact accuracy and effectiveness:
Lexicon-Based Approach: Relies on sentiment lexicons, which may lack comprehensiveness or be outdated, missing subtle nuances. Struggles with handling negations and sarcasm without contextual understanding.

Machine Learning-Based Approach: Requires extensive labeled training data, which can be time-consuming and costly. Performance depends on the quality and representativeness of training data, leading to potential bias.

Hybrid Approach: Combines machine learning and lexicon-based methods but faces challenges from both. Introducing complexities in implementation and tuning.

Being aware of these limitations is crucial for obtaining accurate and meaningful results in sentiment analysis.

3. Research Methodology

3.1 Dataset

We will describe our methodology for sentiment analysis in this chapter using the dataset "exchange_students_feedback.csv." The dataset includes 224 written feedback entries from exchange students. Our main goal is to analyze the sentiments expressed in these feedbacks using two different methods: a lexicon-based approach with VADER and an alternative machine learning approach with the Naive Bayes classifier. Before we perform sentiment analysis, we need to preprocess the raw text data, so it becomes structured and suitable for analysis.

To conduct sentiment analysis on the exchange students' feedback dataset, we will use the Python programming language along with popular libraries such as NLTK (Natural Language Toolkit) and scikit-learn. The NLTK library will help us with text preprocessing tasks, including tokenization and stopword removal. Additionally, we will leverage the VADER lexicon from the NLTK library to perform the lexicon-based sentiment analysis. For the machine learning approach, we will utilize the Naive Bayes algorithm implemented in the scikit-learn library.

We'll perform the analysis using Jupyter Lab, an interactive environment for executing code, visualizing data, and documenting our methodology and results.

3.2 Data Pre-processing

In order to arrange the unstructured text data for analysis, we first break down each feedback sentence into a set of tokens. Before dealing with the complete CSV data, we will use a simple example of raw text from our dataset to walk through the text processing steps so that we can understand text pre-processing.

3.2.1 Tokenization

Let's consider an example from the 10th row of the dataset, which consists of three sentences.

```
#Tokenization
import nltk #text processing library for human language data.
example = '''He is a great teacher that cares about his students.
He is aware that we have a lot of work to do so he has always been flexible and considered with us.
I would like to meet more teachers like him.'''
```

Tokenization is the first step in the text processing pipeline. The `sent_tokenize` function breaks down the lengthy text into separate sentences, allowing us to analyze the sentiments expressed in each sentence independently. After dividing the text into sentences, we proceed to the tokenization of individual words using the `word_tokenize` function, and we also perform lowercasing and punctuational mark removal to extract necessary information and filter out noise. The results of this process are displayed below.

```
['he', 'is', 'a', 'great', 'teacher', 'that', 'cares', 'about', 'his', 'students', '']
['he', 'is', 'aware', 'that', 'we', 'have', 'a', 'lot', 'of', 'work', 'to', 'do', 'so'
', 'he', 'has', 'always', 'been', 'flexible', 'and', 'considered', 'wit
h', 'us', '']
['i', 'would', 'like', 'to', 'meet', 'more', 'teachers', 'like', 'him', '']
```

3.2.2 Stopwords removal

Stopwords are common words in a language that lack significant meaning. Removing them reduces noise and highlights relevant words in NLP tasks. The code below uses NLTK's stopwords corpus to create a set called "sw" containing common English stopwords. Stop words are removed from a list of word tokens, and the results are displayed below.

```
['great', 'teacher', 'cares', 'students', '.']
['aware', 'lot', 'work', 'always', 'flexible', 'considered', 'us', '.']
['would', 'like', 'meet', 'teachers', 'like', '.']
```

3.2.3 Stemming

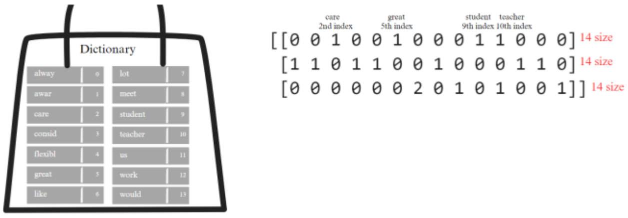
The code uses NLTK's PorterStemmer to perform stemming, reducing words to their base form by removing prefixes and suffixes. This simplifies and groups similar words in the `stemmed_word_tokens` list, showing the transformation to their basic forms.

```
[['great', 'teacher', 'care', 'student', '.'], 5 features
 ['awar', 'lot', 'work', 'alway', 'flexibl', 'consid', 'us', '.'], 8 features
 ['would', 'like', 'meet', 'teacher', 'like', '.']]  6 features
```

3.2.3 Converting words into a vector (Bag-of-Words)

Bag of Words (BoW) is a fundamental technique in NLP that helps computers understand and process text. Since computers cannot comprehend words directly, BoW converts text into numbers for analysis.

To address the challenge of different text lengths, as evident from the arrays shown above, the Bag of Words (BoW) technique utilizes a fixed-size dictionary. This dictionary assigns unique numbers to each word in the corpus. The BoW representation is achieved by counting the occurrences of words from the dictionary in a given text. This process results in a fixed-size array, with each element indicating the frequency of a specific word in the text. Words not present in the text have corresponding zero values. For a visual representation, please refer to the figure provided below.




```
{'great': 5, 'teacher': 10, 'care': 2, 'student': 9, 'awar': 1, 'lot': 7, 'work': 12, 'alwa
y': 0, 'flexibl': 4, 'consid': 3, 'us': 11, 'would': 13, 'like': 6, 'meet': 8}
14
```

With the text processing completed, let's explore the opportunities for sentiment analysis using two approaches: VADER, a lexicon-based method, and Naive Bayes, a machine learning-based method. Let's delve into these options further in the next paragraph.

3.3 Used Methods

3.3.1 Lexicon-based approach

The code uses VADER (Valence Aware Dictionary and sEntiment Reasoner) lexicon from NLTK for sentiment analysis. VADER is helpful for short texts without a pre-trained dataset.

The code imports the `Sentiment Intensity Analyzer` class from NLTK's VADER module and initializes an instance to analyze the sentiment of the given text.
The code iterates through a list of sentences and applies the `analyzer.polarity_scores(sentence)` method. This method returns a dictionary with sentiment scores, including the 'compound' score, representing the overall sentiment of the sentence. The 'compound' score ranges from -1 to 1, where -1 indicates highly negative sentiment, 1 indicates highly positive sentiment, and values close to 0 indicate a more neutral sentiment.

The sentiment label is assigned using a threshold approach. Sentences with a 'compound' score greater than 0.05 are classified as positive, those with a score lower than -0.05 as negative, and scores between -0.05 and 0.05 as neutral. The threshold of 0.05 balances precision and recall in sentiment analysis.

In the upcoming chapter, we'll discuss finding the most efficient threshold. Testing various thresholds with labeled data helps determine the optimal threshold for achieving the desired balance between precision and recall, ensuring accurate and reliable results while minimizing misclassifications.
The code prints each sentence with its sentiment scores and label for a clear overview of the sentiment analysis results.

```
Sentence: great teacher care student .
Sentiment Scores: {'neg': 0.0, 'neu': 0.215, 'pos': 0.785, 'compound': 0.8074}
Sentiment Label: positive
----------
Sentence: awar lot work alway flexibl consid us .
Sentiment Scores: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Sentiment Label: neutral
----------
Sentence: would like meet teacher like .
Sentiment Scores: {'neg': 0.0, 'neu': 0.375, 'pos': 0.625, 'compound': 0.6124}
Sentiment Label: positive
```

Fig. 4. *Sentiment Analysis Using the Vader Lexicon*

3.3.1 Machine learning-based approach

The Machine Learning approach in sentiment analysis requires a pre-existing dataset for training. This method is advantageous over using VADER lexicon when dealing with the ever-evolving nature of internet and social media communication, as VADER may struggle to keep up with new expressions and trends. However, the Machine Learning approach has a limitation in that it requires a larger amount of labeled data to obtain better levels of accuracy in its predictions.

Explanation of the code:
We load the necessary libraries: 'pandas' for data handling, 'CountVectorizer' for feature extraction, 'MultinomialNB' for the Naive Bayes classifier, and 'accuracy_score' for model performance evaluation. After preprocessing and converting the words to numerical vectors, then we train a Naive Bayes model for sentiment analysis using 80% of the data from the 'labeled_feedback.csv' file. For testing, we provide a single raw text input (example) and predict its sentiment label. As an output, we get predicted sentiment and confidence score indicating the certainty of the prediction.

```
Predicted Sentiment for the text is : 1
Confidence: 0.9997334579674919
```

Fig. 5. *Sentiment Analysis Using the Naïve Bayes Classifier*

4. Results and Discussion

Now let's look at the results that we obtained when working with the CSV file/dataset. As you may recall from the third chapter, we stated that we will test our lexicon-based approach in various thresholds. Here are the outcomes we achieved.

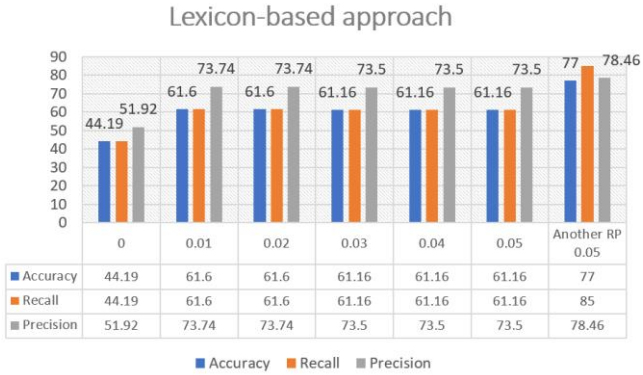


Fig. 6. *Lexicon-Based Sentiment Analysis with Various Thresholds.*

The results show that thresholds 0.01 and 0.02 yielded higher accuracy, precision, and recall in Vader lexicon sentiment analysis. However, the achieved accuracy of 61.6% is relatively low, indicating potential limitations in predicting sentiments using this approach.

Let's focus on Naive Bayes, which achieved an impressive accuracy of 72.27% in sentiment analysis, surpassing the lexicon-based method Vader and MonkeyLearn tool. After splitting the data, 80% was allocated for training (180 rows of text), and the remaining 44 rows were used for testing. The model's precision is 73.87%. In comparison, MonkeyLearn, an alternative sentiment analysis tool, achieved an accuracy of 63.83% and a precision of 65.26%, slightly lower than Naive Bayes.

Previous studies have shown promising results for Naive Bayes. Singh et al. (2017) achieved an accuracy of 85.127% with a small training dataset, while Pang et al. (2002) obtained an accuracy of 78.7% in sentiment analysis on movie reviews using Naive Bayes.

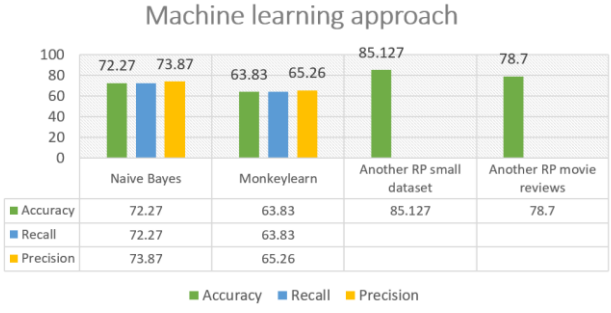


Fig. 7. *Machine learning-based approach for Sentiment Analysis*

5. Conclusions

In conclusion, our findings show that machine learning outperforms lexicon-based approaches for sentiment analysis. Working with the JupyterLab IDE allowed us to have a better knowledge of machine learning ideas.

In the future, we intend to improve our sentiment analysis model by testing it with university datasets and using greater training data for more accurate predictions. In addition, we intend to investigate additional machine learning algorithms, such as Support Vector Machines (SVM) and Random Forest, to improve the performance of our model.

I am glad to share all the code and the detailed explanation of the code, along with the results obtained from the sentiment analysis on our student's feedback dataset. Please find the attachments here.

6. References

1. Clayson, D. E., & Haley, D. A. (2011). Are students telling us the truth? A critical look at the student evaluation of teaching. *Marketing Education Review, 21(2)*, 101–112. https://goo.su/9tf10
2. Stupans, I., McGuren, T., & Babey, A. M. (2016). Student evaluation of teaching: A study exploring student rating instrument free-form text comments. *Innovative Higher Education, 41*(1), 33–42. https://goo.su/US3r
3. Brockx, B., Van Roy, K., & Mortelmans, D. (2012). The student as a commentator: Students' comments in student evaluations of teaching. *Procedia-Social and Behavioral Sciences, 69*, 1122–1133. https://goo.su/RHE27d
4. Rajput, Q., Haider, S., & Ghani, S. (2016). Lexicon-based sentiment analysis of teachers' evaluation. *Applied Computational Intelligence and Soft Computing*, 1–12. https://goo.su/yped7
5. Bing, L. (2012). Sentiment Analysis and Opinion Mining (Synthesis Lectures on Human Language Technologies)
6. Li, W., Jin, B., & Quan, Y. (2020). Review of research on text sentiment analysis based on deep learning. *Open Access Library Journal, 7*, 1–8. https://doi.org/10.4236/oalib.1106174
7. Ren, P., Yang, L. & Luo, F. Automatic scoring of student feedback for teaching evaluation based on aspect-level sentiment analysis. *Educ Inf Technol* **28**, 797–814 (2023). https://goo.su/GvCQ
8. Quratulain Rajput, Sajjad Haider, Sayeed Ghani, "Lexicon-Based Sentiment Analysis of Teachers' Evaluation", *Applied Computational Intelligence and Soft Computing*, vol. 2016, Article ID 2385429, 12 pages, 2016. https://doi.org/10.1155/2016/2385429
9. Pang B, Lee L, Vaithyanathan S (2002) Thumbs up? In: Proceedings of the ACL-02 conference on empirical methods in natural language processing—EMNLP '02. https://doi.org/10.3115/1118693.1118704
10. Singh, J., Singh, G. & Singh, R. Optimization of sentiment analysis using machine learning classifiers. *Hum. Cent. Comput. Inf. Sci*. 7, 32 (2017). https://doi.org/10.1186/s13673-017-0116-3
11. Bonaccorso G (2017) Machine learning algorithms. Packt Publishing Ltd.
12. Nasim, Zarmeen & Rajput, Quratulain & Haider, Sajjad. (2017). Sentiment analysis of student feedback using machine learning and lexicon-based approaches. 1-6. 10.1109/ICRIIS.2017.8002475. https://goo.su/pRyz
13. Solanki, M. S. (2019). Sentiment analysis of text using rule based and natural language toolkit. International Journal of Innovative Technology and Exploring Engineering (IJITEE), 8(12S). https://goo.su/sIp2T
14. Chusovlyanov, D. S. (2014). Machine learning for sentiment analysis and classification of texts into several classes. https://goo.su/UwkeaH
15. Bonta, V., Kumaresh, N., & Janardhan, N. (2019). A comprehensive study on lexicon based approaches for sentiment analysis. *Asian Journal of Computer Science and Technology*, 8(S2), 1-6. https://goo.su/mRlkQ