Text Classification

Pureti Anusha#1, Krishnaveni I #2, Sushma V #3, Mrudula K #4, Sai Swetha M #5, Navya Kranthi T #6

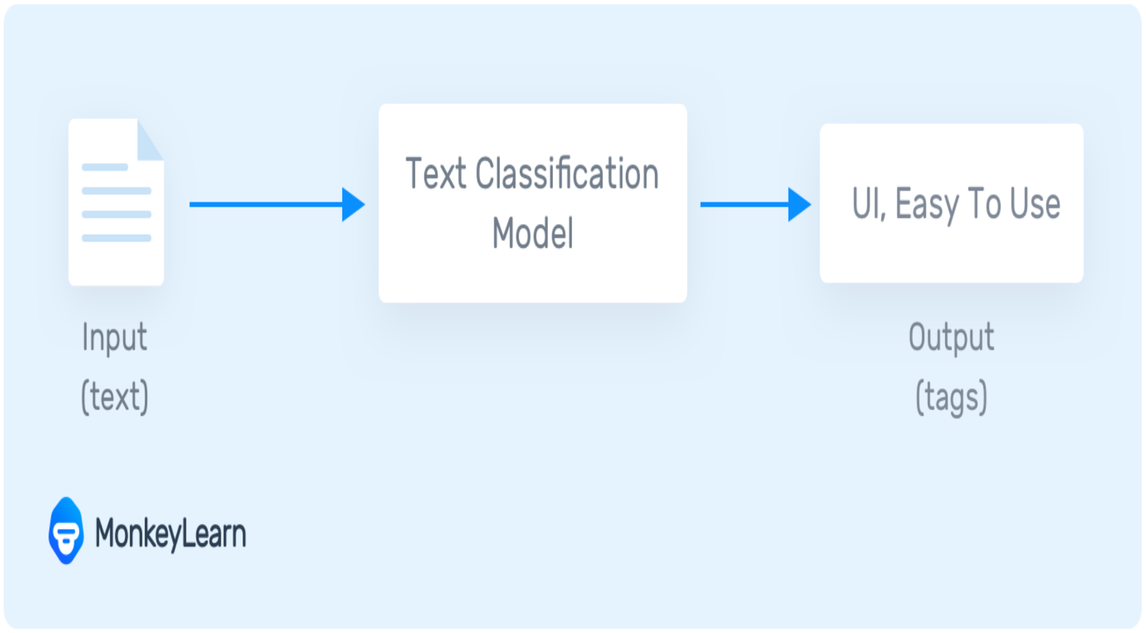
#1 Associate Professor, Dept of Computer Science and Engineering, Qis College of Engineering and Technology, Ongole
#2 Student, Dept of Computer Science and Engineering, Qis College of Engineering and Technology, Ongole
#3 Student, Dept of Computer Science and Engineering, Qis College of Engineering and Technology, Ongole
#4 Student, Dept of Computer Science and Engineering, Qis College of Engineering and Technology, Ongole
#5 Student, Dept of Computer Science and Engineering, Qis College of Engineering and Technology, Ongole
#6 Student, Dept of Computer Science and Engineering, Qis College of Engineering and Technology, Ongole

Abstract
Organizations are increasingly interested in classifying texts or parts thereof into categories, as this enables more effective use of their information. Manual procedures for text classification work well for up to a few hundred documents. However, when the number of documents is larger, manual procedures become laborious, time-consuming, and potentially unreliable. Techniques from text mining facilitate the automatic assignment of text strings to categories, making classification expedient, fast, and reliable, which creates potential for its application in organizational research. The purpose of this article is to familiarize organizational researchers with text mining techniques from machine learning and statistics. We describe the text classification process in several roughly sequential steps, namely training data preparation, preprocessing, transformation, application of classification techniques, and validation, and provide concrete recommendations at each step. To help researchers develop their own text classifiers, the R code associated with each step is presented in a tutorial. The tutorial draws from our own work on job vacancy mining. We end the article by discussing how researchers can validate a text classification model and the associated output.

1.Introduction

Text classification is the process of assigning tags or categories to text according to its content. It's one of the fundamental tasks in natural language processing with broad applications such as sentiment analysis, topic labeling, spam detection, and intent detection. Unstructured data in the form of text is everywhere: social media, emails, chats, web pages, support tickets, survey responses, and more. Text can be an extremely rich source of information, but extracting insights from it can be hard and time-consuming due to its unstructured nature[1]. Machine learning text classification can help businesses automatically structure and analyze their text, quickly and cost-effectively, to automate processes and enhance data-driven decisions. Text classification (a.k.a. text categorization or text tagging) is the task of assigning a set of predefined categories to open-ended. Text classifiers can be used to organize, structure, and categorize pretty much any kind of text – from documents, medical studies and files, and all over the web. For example, new articles can be organized by topics; support tickets can be organized by urgency; chat conversations can be organized by language; brand mentions can be organized by sentiment; and so on. "The user interface is quite straightforward and easy to use." A classifier can take this text as an input, analyze its content, and then automatically assign relevant tags, such as UI and Easy To Use that represent this text:

It's estimated that around 80% of all information is unstructured, with text being one of the most common types of unstructured data. Because of the messy nature of text, analyzing, understanding, organizing, and sorting through text data is hard and time-consuming, so most companies fail to use it to its full potential[2]. This is where text classification with machine learning comes in. Using text classifiers, companies can automatically structure all manner of relevant text, from emails, legal documents, social media, chatbots, surveys, and more in a fast and cost-



effective way. This allows companies to save time analyzing text data, automate business processes, and make data-driven business decisions.
Some of the top reasons:

Scalability

Manually analyzing and organizing is slow and much less accurate.. Machine learning can automatically analyze millions of surveys, comments, emails, etc., at a fraction of the cost, often in just a few minutes. Text classification tools are scalable to any business needs, large or small.

Real-time analysis There are critical situations that companies need to identify as soon as possible and take immediate action (e.g., PR crises on social media). Machine learning text classification can follow your brand mentions constantly and in real time, so you'll identify critical information and be able to take action right away.

Consistent criteria

Human annotators make mistakes when classifying text data due to distractions, fatigue, and boredom, and human subjectivity creates inconsistent criteria. Machine learning, on the other hand, applies the same lens and criteria to all data and results. Once a text classification model is properly trained it performs with unsurpassed accuracy.

Cleaning the text:

The first step that we have to do is to prepare and clean the dataset. Cleaning the dataset is an essential step to remove any meaningless words or useless terms like hashtags, mentions, punctuations, and many more. To clean the text, we can utilize libraries like re for removing terms with patterns and NLTK to remove words

2.Related Work

Text classification can be used in a broad range of contexts such as classifying short texts (e.g., tweets, headlines, chatbot queries, etc.) or organizing much larger documents (e.g., customer reviews, news articles,legal contracts, longform customer surveys, etc.). Some of the most well-known examples of text classification include sentiment analysis, topic labeling, language detection, and intent detection.

Sentiment Analysis

Perhaps the most popular example of text classification is sentiment analysis (or opinion mining): the automated process of reading a text for opinion polarity (positive, negative, neutral, and beyond). Companies use sentiment classifiers on a wide range of applications, like product analytics, brand monitoring, market research, customer support, workforce analytics, and much more.

Sentiment analysis allows you to automatically analyze all forms of text for the feeling and emotion of the writer.

Topic Labeling

Another common example of text classification is topic labeling, that is, understanding what a given text is talking about. It’s often used for structuring and organizing data, such as organizing customer feedback by topic or organizing news articles by subject.

Language Detection

Language detection is another great example of text classification, that is, the process of classifying incoming text according to its language. These text classifiers are often used for routing purposes (e.g., route support tickets according to their language to the appropriate team).

Intent Detection

Intent detection or intent classification is another great use case for text classification that analyzes text to understand the reason behind feedback. Maybe it's a complaint, or maybe a customer is expressing intent to purchase a product. It's used for customer service, marketing email responses, generating product analytics, and automating business practices. Intent detection with machine learning can read emails and chatbot conversations and automatically route them to the correct department.

Social Media Monitoring:

According to Hootsuite, nearly half of Americans have interacted with companies or institutions on at least one of their social media networks. All of these interactions represent a lot of actionable insights for any business. Just on Twitter alone, users send 500 million tweets every day. Furthermore, surveys show that 83% of customers who comment or complain on social media expect a response the same day, with 18% expecting it to come immediately. What are people complaining about when they mention a particular brand? What are they praising? How have they reacted to a specific message or campaign?

The answers to these questions can be found within the sea of data available on social media, but without the help of machine learning text analysis, making sense of all this data would be extremely costly, time-consuming, and much less accurate – if even possible. Fortunately, with machine learning you can analyze social media data in a scalable, cost-effective, downright easy way.

Brand Monitoring

Online conversations around a brand and its competitors heavily influence consumers. Some blogs, forums, review sites, and influencers are becoming even more important than traditional outlets. According to MineWhat, 81% of buyers conduct online research before making a purchase. Consumers care immensely about what people are saying online about a brand – BrightLocal states that 85% of consumers trust online reviews as much as personal recommendations.

Customer Service

Building a good customer experience is one of the foundations of a sustainable and growing company. According to Hubspot, people are 93% more likely to be repeat customers at companies with excellent customer service. The study also unveiled that 80% of respondents said they had stopped doing business with a company because of a poor customer experience.

Text classification can help support teams provide a stellar experience by automating tasks that are better left to computers, saving precious time that can be spent on more important things.

For instance, text classification is often used for automating ticket routing and triaging. Imagine a global company that provides customer support in several languages; this involves the process of assigning tickets based on the ticket's language. To do this, a person is needed to manually assign the ticket to the correct team who can understand and reply to the customer in the right language[4],[5]. With text classification, instead of using humans you can use a language detection classifier to do this task for you.

Text classification can also be used for routing support tickets to a teammate with specific product expertise. For instance, if a customer writes in asking about refunds, you can automatically assign the ticket to the teammate with permission to perform refunds. This will ensure the customer gets a quality response more quickly. Without the need to triage every single ticket, support teams can work more efficiently and reduce response times. You'll always know that tickets have been automatically routed to the most appropriate team.

Support teams can also use text classification to automatically detect the urgency of a support ticket and prioritize accordingly. By using machine learning to set priorities, you can ensure your team is always working on the most urgent tickets, every time.

Companies are also leveraging text classification for getting insights from support conversations, thus improving their reporting and analytics. You can even use customer complaint classification to find complaints wherever they may exist, from all over the web.

Example: Analyzing customer support interactions on Twitter

There are different trends around how to deal with customers in social media. Some support teams try to appear hip and cool while others project a more professional appearance. But, which approach is better received by customers?

To find out, we experimented with keyword extraction and sentiment analysis to analyze 200,000+ customer interactions with Verizon, T-Mobile, AT&T, and Sprint on Twitter. First, we analyzed the most relevant keywords in all these tweets and found out that each carrier has its unique customer service approach. For instance, T-Mobile has a friendlier and more personal approach, with every support representative signing each message with their name, while Verizon tweets are very dry and professional. Then, we performed sentiment analysis on the data, and the results suggest that a friendlier take on social media elicits more positive responses:

3 Automated text classification methods

3.1. Conceptual overview of text classification

We distinguish between sentiment and content classification, which are both of particular interest for marketing applications. The former involves predicting the emotion of an unlabeled text document such as the following exemplary movie review, drawn from one of our datasets: "All in all, a great disappointment". In sentiment classification, the goal of the text classification methods would be to detect the emotion conveyed through this text and to correctly classify it as negative. Lexicon-based or supervised machine learning methods are the two major approaches to accomplish this task (see Appendix A for an overview of text classification problems and approaches). Supervised machine learning methods learn either sentiment or custom content categories based on manually labeled text data and inductively construct classifiers based on observed patterns without requiring manual coding of classification rules (Dumais et al., 1998). This makes them flexible in understanding grammatical construction specific to certain domains. In comparison, lexicon-based methods (e.g., NRC or LIWC) require expert-crafted dictionaries, consisting of elaborate word lists and associated labels to classify text documents (Mohammad & Turney, 2010; Pennebaker, Boyd, Jordan, & Blackburn, 2015). These are often generic across domains but can be extended by custom word lists. If no suitable dictionary is available, researchers must create their own (e.g., Hansen, Kupfer, & Hennig-Thurau, 2018). As the creation of such dictionaries is cumbersome, lexical methods such as LIWC are most commonly used for two-class sentiment classification because several off-the-shelf dictionaries exist (e.g., Hennig-Thurau et al., 2015; Ordenes et al., 2017). While these methods are quick and easy to employ, they also come with drawbacks. For example, LIWC may struggle to correctly predict the negative sentiment of a post like the previous example, as the individual words "great" and "disappointment" point in two opposite emotional directions unless such phrases are included in the dictionary. In contrast, machine learning methods can learn that the word pair "great disappointment" indicates negative sentiment without the need to curate a dictionary a priori.

In comparison, content classification refers to the task of assigning custom category labels to new text documents, e.g., to automatically detect that YouTube comments such as "Subscribe to my channel" have commercial rather than user interest background. Such tasks are potentially easier than the extraction of emotion, which often requires higher context understanding, e.g., irony, playing a larger role (Das & Chen, 2007). In contrast to all other methods we study, latent Dirichlet allocation (LDA, Blei, Andrew, & Jordan, 2003) is an unsupervised machine learning method originally developed and applied for knowledge discovery purposes (Humphreys & Wang, 2017). In marketing research, LDA is most commonly used for explorative topic modeling or latent topic identification (e.g., Puranam, Narayan, & Kadiyali, 2017; Zhang, Moe, & Schweidel, 2017). Such types of analyses have different objectives and are conceptually and empirically not comparable to the remaining methods in terms of accurately recovering researcher-defined class labels and are therefore beyond the scope of this investigation.

3.2. Algorithmic approaches and characteristics of text classification methods

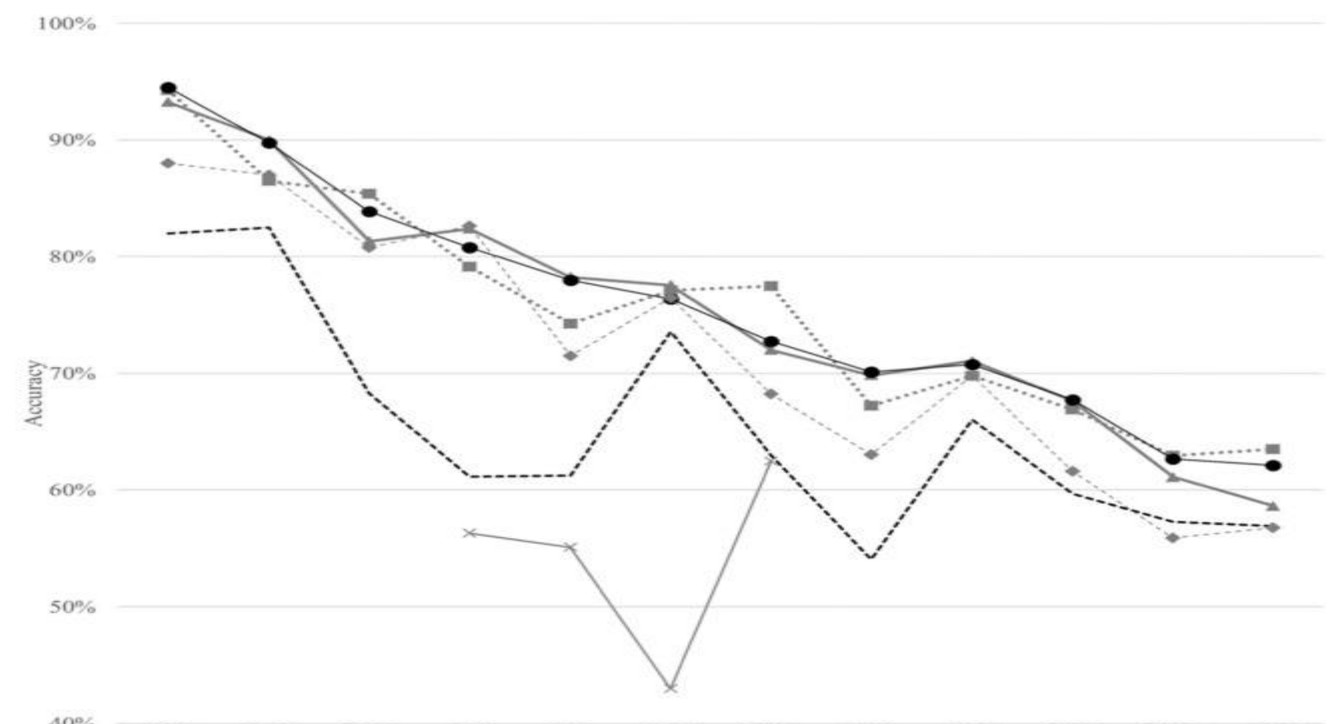
In total, we test a set of ten text classification methods due to their conceptually different algorithmic approaches, their use and relevance for marketing research, and their proven performance in other disciplines. This includes five machine learning methods, i.e., ANN, kNN, NB, RF, and SVM, as well as five lexicon-based methods, i.e., AFINN (Nielsen, 2011), BING (Hu & Liu, 2004), LIWC (Pennebaker et al., 2015), NRC Emotion Lexicon (Mohammad & Turney, 2010), and Valence Aware Dictionary for Sentiment Reasoning (VADER, Hutto & Gilbert, 2014). While ANN, RF, and SVM are discriminative classifiers, NB is a generative, probabilistic classifier. In contrast, kNN is a non-parametric classifier, belonging to the family of proximity-based algorithms. We explain each of these approaches in more detail next. ANN are the most highly parametrized method in our comparison. Neurons, which are connected to the input layer, inductively learn patterns from training data to allow predictions on test data (Efron & Hastie, 2016). The simplest form of ANN consists of only one input and output layer (perceptrons). The number of units in the output layer corresponds to each of the possible classes. Current computational capabilities enable the inclusion of multiple hidden layers in between (e.g., LeCun, Bengio, & Hinton, 2015; Sebastiani, 2002). The number of nodes in the hidden layer is linked to the complexity of the classification task (Detienne, Detienne, & Joshi, 2003). As common text classification problems represent linearly separable class structures in high-dimensional space (Aggarwal & Zhai, 2012), single-layer ANN with a non-linear activation function are most frequently applied for text classification (e.g., Moraes, Valiati, & Neto, 2013).

Due to their flexible structure, ANN can be considered particularly versatile, performing well across different classification tasks, which is likely relevant when handling noisy social media data. Moreover, ANN can learn subtle text patterns. This can be important for sentiment problems, where the link between individual word features and the class may be more complex compared with content classification tasks. However, this ability to adapt to even contradictory data and potentially better recognition of higher context tends to negatively affect the computational costs of ANN (Sebastiani, 2002). The more complex the network typology, the higher the computational time both in the training and prediction phase. While RF can be easily parallelized, ANN are more difficult to multi-thread, posing a larger optimization problem. Moreover, given their large number of parameters and complex structure, ANN are difficult to interpret intuitively and require expert knowledge for parameter tuning. kNN is a lazy learning algorithm with no offline training phase (Yang, 1999). All training documents are stored and computation is deferred to the prediction phase (Sebastiani, 2002). For each test document, kNN ranks the nearest neighbors of the labeled examples from the training set and uses the categories of the highest-ranked neighbors to derive a class assignment. The more near neighbors with the same category, the higher the confidence in that prediction (Yang & Liu, 1999). Computing the respective distances between all test and training documents makes kNN computationally costly when applied to high-dimensional, sparse text data (Aggarwal & Zhai, 2012), especially if the training set is large (Sebastiani, 2002). Moreover, as a non-parametric method, kNN suffers from the curse of dimensionality (Bellmann, 1961), requiring an exponentially larger number of training examples to generalize well for many features. This makes kNN prone to overfit in-sample and predict poorly out-of-sample. Thus, relative performance of kNN is likely lower for longer texts with many features and, in turn, more favorable relative to all other methods for shorter texts. NB is one of the simplest probabilistic classifier models (Yang, 1999). The classifier estimates a class-conditional document distribution P(d| c) from the training documents and applies Bayes' rule to estimate P(c| d) for test documents, where the documents are modeled using their terms. To efficiently compute the conditional probabilities, NB assumes all features to be independent. This naïve assumption can provide a reasonable trade-off between performance and computational costs. Domingos and Pazzani (1997) find that NB can also perform well when features are interdependent. In addition, Netzer et al. (2016) argue that the resulting generative model is easy to interpret and explain. Moreover, NB as a generative classifier may be recommended for smaller sample sizes due to its inherent regularization, making it less likely to overfit compared with discriminative classifiers (e.g., Domingos, 2012; Ng & Jordan, 2002). However, NB is not capable of modeling interaction effects among features. Thus, we expect it to perform relatively well for problems with strong individual signal words and straightforward relationships between the text features and the respective classes, e.g., for simple forms of promotion content detection (Yang, Nie, Xu, & Guo, 2006) and two-class sentiment classification exhibiting strong polarity.

4 Results

Comparison of method performances across all datasets and classification tasks

To facilitate interpretation and in the interest of parsimony, we group all similar datasets and compare similar sample size resulting in 12 distinct types of social media text data. Specifically, we aggregate across languages since we do not detect a significant impact of language on classification performance for any method. Fig. 1 summarizes the resulting accuracies for all methods. The performance of the lexicon-based methods is reported for all two-class sentiment problems. Only two out of the five dictionaries perform slightly better than the weakest machine learning algorithm (kNN) and that occurs only in three instances. However, these few instances where one of the dictionary approaches exceeds a machine learning method is due to the poor performance of kNN for these datasets having >15% lower accuracy than the best performing approach. Since none of the dictionaries achieve a performance close to the winning approaches, we summarize these as the average performance in Fig. 1 and provide all details in Appendix B.



	SMS	YTU	ROT	AMR	YEL	AMT	IMD	FBK	CBP	TWS	TWC	CBC
ANN	93.3	90.0○	81.3	82.4○	78.3○	77.6○	72.0	69.8○	71.0○	67.7○	61.1	58.6
kNN	82.0	82.5	68.3	61.1	61.3	73.6	63.0	54.0	66.0	59.7	57.3	56.9
NB	94.3○	86.5	85.4○	79.1	74.3	77.1○	77.5○	67.2	69.8	66.9○	63.0○	63.5○
RF	94.5○	89.8○	83.9	80.8	78.0○	76.4	72.8	70.1○	70.8○	67.7○	62.7○	62.1
SVM	88.0	87.0	80.8	82.6○	71.5	76.4	68.3	63.0	69.8	61.6	55.9	56.8
DICT				56.3	55.1	43.0	62.5					

5 Conclusion

This article provided an overview of TC and a tutorial on how to conduct actual TC on the problem of job task information extraction from vacancies. We discussed and demonstrated the different steps in TC and highlighted issues surrounding the choices of features, classification algorithms, and evaluation metrics. We also outlined ways to evaluate and validate the resulting classification models and prediction from these models. TC is an empirical enterprise where experimentation with choices of representation, dimensionality reduction, and classification techniques is standard practice. By building several classifiers and comparing them, the final classifier is chosen based on repeated evaluation and validation. Thus TC is not a linear process; one has to revisit each step iteratively to examine how choices in each step affects succeeding steps. Moreover, classifiers evolve in the presence of new data. TC is a wide research field and there are many other techniques that were not covered here.

6 References:

[1] Pedregosa et al. 2011. Scikit-learn: Machine Learning in Python, JMLR 12, pp. 2825–2830.

[2] Manning et al. 2011. Introduction to Information Retrieval. Cambridge University Press, pp. 234–265.

[3] McCallum A. and Nigam K. 1998. A comparison of event models for Naive Bayes text classification. Proc. AAAI/ICML-98 Workshop on Learning for

Text Categorization, pp. 41–48.
[4] Khalid, I. A. 2020. Create A Simple Search Engine Using Python. Towards Data Science. https://towardsdatascience.com/create-a-simple-search-engine-using-python-412587619ff5
[5] https://www.kaggle.com/c/nlp-getting-started
[6] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Authors Profile

Pureti Anusha, M.Tech., working as Associate Professor of CSE Department in QIS College of Engineering and Technology (Autonomous), Ongole, Andhra Pradesh, India.

Krishnaveni I pursuing B Tech in Computer Science Engineering from QIS College of Engineering and Technology (Autonomous & NAAC 'A' Grade), Ponduru Road, Vengamukkalapalem, Ongole, Prakasam Dist, Affiliated to Jawaharlal Nehru Technological University, Kakinada.

Sushma I pursuing B Tech in Computer Science Engineering from QIS College of Engineering and Technology (Autonomous & NAAC 'A' Grade), Ponduru Road, Vengamukkalapalem, Ongole, Prakasam Dist, Affiliated to Jawaharlal Nehru Technological University, Kakinada.

Mrudula K pursuing B Tech in Computer Science Engineering from QIS College of Engineering and Technology (Autonomous & NAAC 'A' Grade), Ponduru Road, Vengamukkalapalem, Ongole, Prakasam Dist, Affiliated to Jawaharlal Nehru Technological University, Kakinada.

Sai Swetha M pursuing B Tech in Computer Science Engineering from QIS College of Engineering and Technology (Autonomous & NAAC 'A' Grade), Ponduru Road, Vengamukkalapalem, Ongole, Prakasam Dist, Affiliated to Jawaharlal Nehru Technological University, Kakinada.

Navya Kranthi T pursuing B Tech in Computer Science Engineering from QIS College of Engineering and Technology (Autonomous & NAAC 'A' Grade), Ponduru Road, Vengamukkalapalem, Ongole, Prakasam Dist, Affiliated to Jawaharlal Nehru Technological University, Kakinada.