May 30th, 2025

Making Sense of Support: Topic Modeling and Priority Classification for Omnilytics

Final paper

Natural Language Processing & Text Analytics

Program: MSc in Business Administration and Data Science

Authors: Michele Daconto (176193), Ludovico Francia (177119)

Examiners: Daniel Hardt, Rajani Singh, Ole Torp Lassen

Number of characters: 25,452

Number of pages: 15

Link to Colab Notebook: here

Academic Year 2024/2025

Table of Contents

Abstract

This project presents an analysis of 28,587 multilingual customer support tickets from Omnilytics - a tech company - through the application of topic modeling and supervised classification techniques. To uncover the underlying thematic structure of the tickets, we employed Latent Dirichlet Allocation (LDA) and BERTopic. Both models produced robust and interpretable results, identifying 4 recurring themes, with BERTopic achieving higher coherence score and demonstrating superior performance.

In the second phase of the study, we developed several classification pipelines aimed at predicting ticket priority. These combined a range of vectorization methods (Bag of Words, TF-IDF, Word2Vec, OpenAI embeddings) with different classifiers (Naïve Bayes, Logistic Regression, Random Forest, Neural Networks). While the priority classification task proved more challenging—likely due to the inherent subjectivity of urgency labels—the results remain meaningful and actionable, offering valuable guidance for optimizing ticket triage processes.

These natural language techniques enabled us to support the company in optimizing the management of its ticketing system by extracting meaningful insights from the text contained in customer support requests.

Keywords: Ticketing – Topic Modeling – Priority Classification – BoW – TF-IDF – Word2Vec – OpenAI

1. Introduction

In this paper we analyzed a dataset of support tickets sourced from Kaggle and referencing a tech company that we have renamed *Omnilytics*.

Omnilytics is a B2B tech company offering a broad suite of digital solutions including software, systems integration, digital marketing, and data security. In the past year, the company has experienced strong revenue growth driven by a consistent increase in its customer base. This commercial success of Omnilytics has inevitably led to a significant increase in the volume of support requests received from customers.

Because the company has not had sufficient time to expand its support team, it is facing a period in which it is struggling to handle customer requests effectively and in a timely manner. This situation has alerted company executives as it threatens the company's willingness to put customer satisfaction at the center of its operations in order to persist on the growth path. Inadequate ticket management can in fact lead to a significant reduction in the quality perceived by customers regarding the solutions offered by the company, and thus it is a problem that company executives want to solve.

Lacking the budget to expand the support team, the company chose to rely on a team of experienced Natural Language Processing consultants with a dual objective:

1. Analyze support tickets to identify recurring themes.
2. Build an automated system to rank the priority of tickets to ensure adequate response times for the most urgent cases.

In this paper we stepped into the shoes of these consultants to support Omnilytics in improving its customer support operations.

1.1 Our NLP Approach to Omnilytics' Challenges

To address the two problems posed by Omnilytics, we structured the project into several phases, as illustrated in Fig. 1 below:

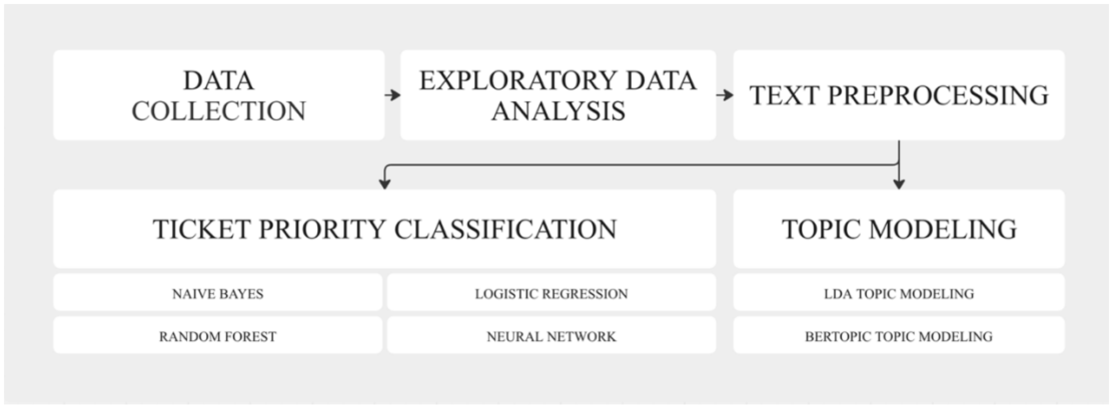


Figure 1 – *Workflow Diagram*

Our work began with a data collection phase where we obtained the dataset of support tickets already managed by the company. We then conducted an *Exploratory Data analysis* (EDA) phase where we sought to understand the structure and type of data in order to fully understand the data available to answer the two problems. To adapt the text variables to the needs of the different natural language processing techniques used, we subjected these variables to an in-depth *text preprocessing* process.

To identify the most critical topic areas and thus the recurring themes arising in support tickets, we decided to use topic modeling techniques. In parallel, to automate the prioritization of tickets, we used supervised classification models. In both cases, our approach was characterized by the use of varied techniques and models in order to be able to identify the best solution to address the two problems posed by Omnilytics.

1.2 Literature Review

Several researchers have studied the use of natural language processing techniques in support ticket management. Regarding the ticket classification problem, Grace (2025) implements a classification system based on different types of text representations (BoW, TF-IDF, Word2Vec, and Doc2Vec) and different classification models such as Naive Bayes, SVM, Random Forest, and RNN. Taking a cue from this approach, our work experiments not only with Naive Bayes and Random Forest but also with different models such as Logistic Regression and simple neural networks. In addition, we explored the use of OpenAI embeddings, as they can offer greater semantic depth than traditional models. Concerning the identification of recurring themes in tickets, Wiranto and Uswatunnisa (2023) focus on the application of LDA topic modeling. In our work, we proposed the use of both LDA and BERTopic, supported by the GPT-4 model for the automatic naming of extracted topics.

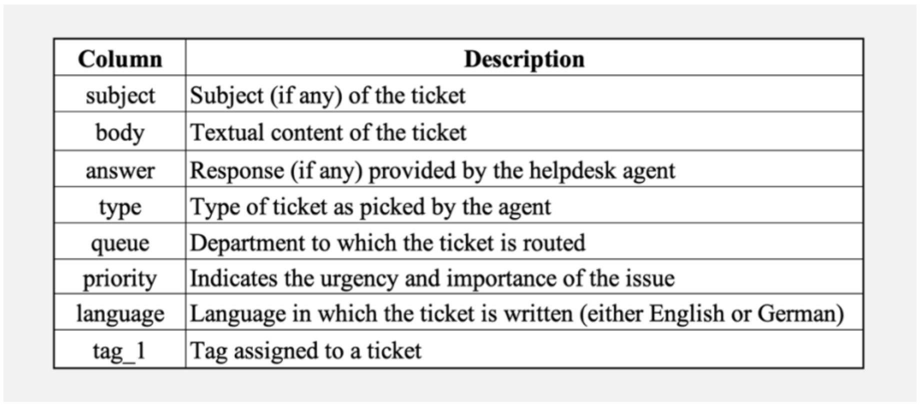
2. Data & Preparation

2.1 Dataset Description

The dataset used for our analysis is available at the following link. Within the ZIP archive, the relevant CSV file is the one prefixed with *'aa-'*, which was updated and corrected as of April 2025.

The archive contains 28,587 support requests submitted by customers of a tech company (*Omnilytics*, a fictitious name) through a dedicated ticketing system. The dataset description does not specify the time frame in which the tickets were submitted or the size of the company.

Each ticket includes information structured across the following columns (see Fig. 2).



Column	Description
subject	Subject (if any) of the ticket
body	Textual content of the ticket
answer	Response (if any) provided by the helpdesk agent
type	Type of ticket as picked by the agent
queue	Department to which the ticket is routed
priority	Indicates the urgency and importance of the issue
language	Language in which the ticket is written (either English or German)
tag_1	Tag assigned to a ticket

Figure 2 – *Dictionary of relevant columns*

2.2 Missing Values & Columns Removal

The dataset originally included seven additional columns (*tag_2* through *tag_8*) representing further tags associated with each ticket. Along with the *version*[1] column, we chose to exclude these fields due to a high proportion of missing values and their limited contribution to the overall information content.

As for the columns relevant to our analysis, they contain no missing values with the exception of *subject* (3,838 - cases where the ticket author did not provide a subject) and *answer* (only 7 missing entries).

[1] This column contains the software's version of the ticketing system, and it has no utility for our analysis.

No duplicate records were found.

2.3 Exploratory Data Analysis

After verifying the presence of missing values using the *get_col_overview()* function, we explored the unique values within each column and visualized their distributions using *plot_col_distribution()*.

This process allowed us to gain a deeper understanding of the dataset, leading to the following insights.

- Listed in order of frequency, the possible ticket *types* are *Incident* (40.1%), *Request* (28.6%), *Problem* (21.0%) and *Change* (10.2%) (see Fig. A in Appendix).
- Tickets can be routed to one of ten departments (*queues*). The most frequently assigned are *Technical Support* (29.3%) and *Product Support* (18.4%), while the least requested are *General Inquiry* (1.4%) and *Human Resources* (2.0%) (see Fig. B in the Appendix).
- The possible priority levels assigned to a ticket, in order of frequency, are *medium* (40.3%), *high* (39.1%), and *low* (20.6%) (see Fig. C in the Appendix).
- Tickets can be written in two languages: English (57.2%) and German (42.8%) (see Fig. D in the Appendix).
- There are 116 possible tags (*tag_1*) that can be used to label a ticket. The most frequently used are *Security* (20.6%) and *Bug* (18.7%) (see Fig. E in Appendix).

We concluded the exploratory data analysis with two bar charts aimed at examining how ticket types are distributed across each priority level (see Fig. F in the Appendix) and across departments (*queues*) (see Fig. G in the Appendix). Monitoring changes in these distributions will be especially useful when the dataset is updated and new tickets are added.

3. Text Preprocessing

The dataset used in our work contains three main textual features that characterize support tickets: *subject* (the subject of the ticket, which is not always present), *body* (main content of the ticket), *answer* (response provided by the support). In order to extract meaningful information from these texts, it was necessary to apply several text preprocessing operations aimed at improving the semantic and syntactic quality of the text by removing noise.

This text preprocessing phase is essential to ensure the reliability of the results. Being a bag-of-words approach, *Latent Dirichlet Allocation (LDA)* is a topic modeling technique that produces results which are strongly influenced by the quality of the input text. Clean and consistent input text enables the identification of well-defined and interpretable topics. Similarly, some text representations used in supervised classification such as *CountVectorizer, TF-IDF* and *Word2Vec* are also highly affected by noise and redundancies in the input text. For this reason, after the exploratory data analysis phase, we proceeded to perform an in-depth text preprocessing process.

The pipeline adopted for text feature cleaning is depicted in Fig. 3:

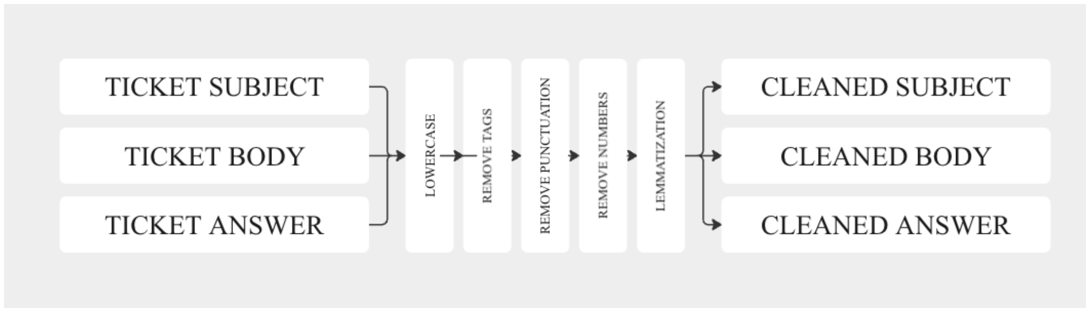


Figure 3 – *Text Preprocessing pipeline*

Texts were converted to lowercase to standardize word representation. This was followed by the removal of textual components that lacked meaningful information for our analysis, such as tags (e.g., "<name>"), punctuation, and numbers.

To reduce lexical fragmentation and improve semantic coherence, the text underwent a *lemmatization* process to reduce each word to its base form (lemma). Without this step, the results of topic modeling and classification would have been compromised by the informational noise caused by terms that are semantically identical but appear in different morphological forms. Lemmatization was performed using the *SpaCy* library, which enabled us to remove *stopwords* and apply lemmatization based on part-of-speech tagging.

During the exploratory data analysis phase, we also identified the presence of tickets written in both English and German. To accommodate the multilingual nature of the dataset, we applied language-specific preprocessing using two separate models: *"en_core_web_md"* for English language tickets and *"de_core_news_md"* for German language tickets.

Finally, we constructed a combined field that merges the ticket's subject (if any) with its body to enrich the contextual information provided to the models. This features merging allows us to increase the informational content of the input text, benefiting both the topic modeling and classification tasks.

3.1 OpenAI Embeddings

To overcome the limitations of bag-of-words approaches and Word2Vec (which generates embedding only at the word level), we used the OpenAI API, specifically the *"text-embedding-3-large"* embedding model to generate dense, numerical vector representations of support ticket content.

This model is designed to capture the semantic and contextual structure of the language by training on massive amounts of multilingual data. The model is able to capture the similarity between texts even when the words used are different and even if the language is different.

Due to the nature of the model, excessive preprocessing that would compromise the informational quality of the text must be avoided. Therefore, the text used to get the embedding vector underwent a minimal preprocessing process aimed at ensuring consistent formatting of the text without applying lemmatization and stopwords removal.

4. Topic Modeling

To address Omnilytics' first objective - analyzing support tickets to identify recurring themes - we developed two topic modeling models in order to discover latent topics within the support tickets: LDA and BERTopic. Latent Dirichlet Allocation (LDA) topic modeling is a probabilistic model based on bag-of-words. BERTopic, on the other hand, uses embeddings and clustering algorithms.

To evaluate the quality of the results we used the *coherence score (c_v),* which measures the semantic coherence of the words contained in each topic. The score ranges between zero and one. The higher the values and the greater the interpretability of the results of the model.

Having a bag-of-words approach, LDA is sensitive to the vocabulary structure and thus using multi-language tickets as model input would generate biased results. For simplicity, we limited our analysis to English-language tickets only.

After training the two models, we used the GPT-4.1 model to assign names to each of the topics extracted from LDA and BERTopic. To return a name, the model received in the input prompt 10 main keywords and 10 ticket examples for each topic.

4.1 LDA Topic Modeling

Latent Dirichlet Allocation is a topic modeling model that represents each document as a combination of latent topics and each topic as a distribution of words. LDA is a bag-of-words approach and therefore does not take into account the order of words within a text but relies solely on their co-occurrence.

To be trained LDA requires as a parameter the number of arguments to be extracted. Using *pyLDAvis*, which allows us to visualize the separation between different topics, we found that 4 topics represented a good balance with distinct clusters. The recurrent topics found by the LDA model were:

1. Technical Issue Troubleshooting and Resolution (42.9% of tickets)
2. Digital Marketing Strategy Optimization (12.8% of the tickets)
3. Hospital Medical Data Security Breaches (19.4% of tickets)
4. SaaS Integration and Analytics Support (24.9% of tickets)

The distribution of tickets by topic is shown in Fig. 4:

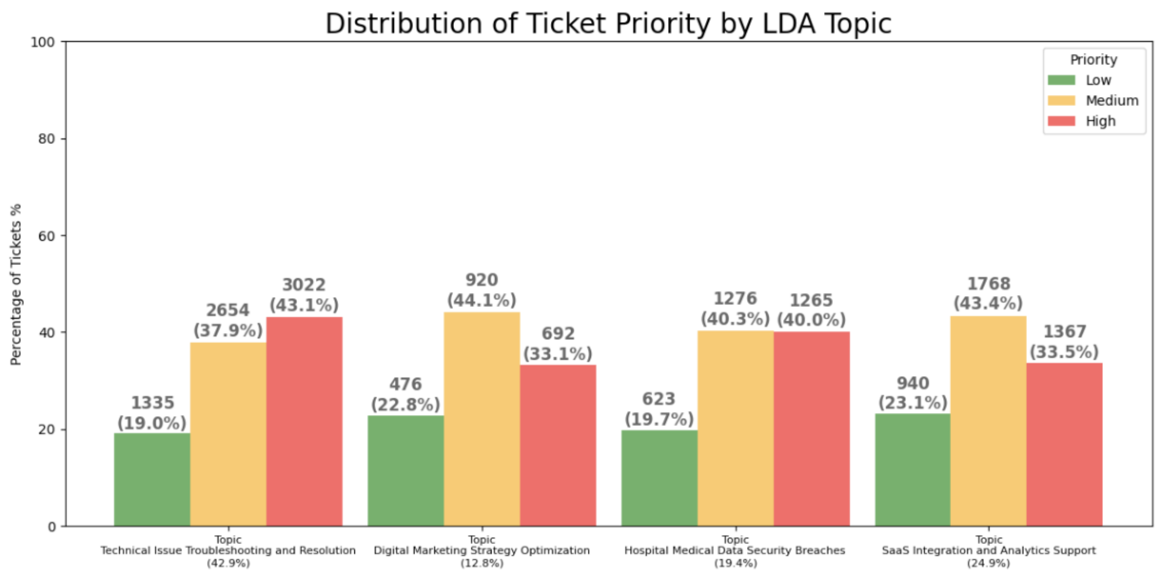


Figure 4 – *Distribution of Ticket Priority by LDA Topic*

4.2 BERTopic Modeling

BERTopic, on the other hand, does not use word frequency but instead uses embeddings in combination with the HDBSCAN clustering algorithm to group similar texts. Overcoming the limitations of the bag-of-words approach, the use of embeddings allows the model to capture the contextual meaning of texts and perform more precise clustering.

BERTopic classified 4% of the tickets as belonging to the outliers group (not assigned to any topic) and identified 4 main topics:

1. Technical Issue Troubleshooting and Resolution (41.3% of the tickets)
2. Digital Strategy & Integration Support (12.9% of the tickets)
3. Hospital Medical Data Security Breach (20.3% of tickets)
4. Medical Data Security and Privacy (25.5% of tickets)

The distribution of tickets by topic is shown in Fig. 5:

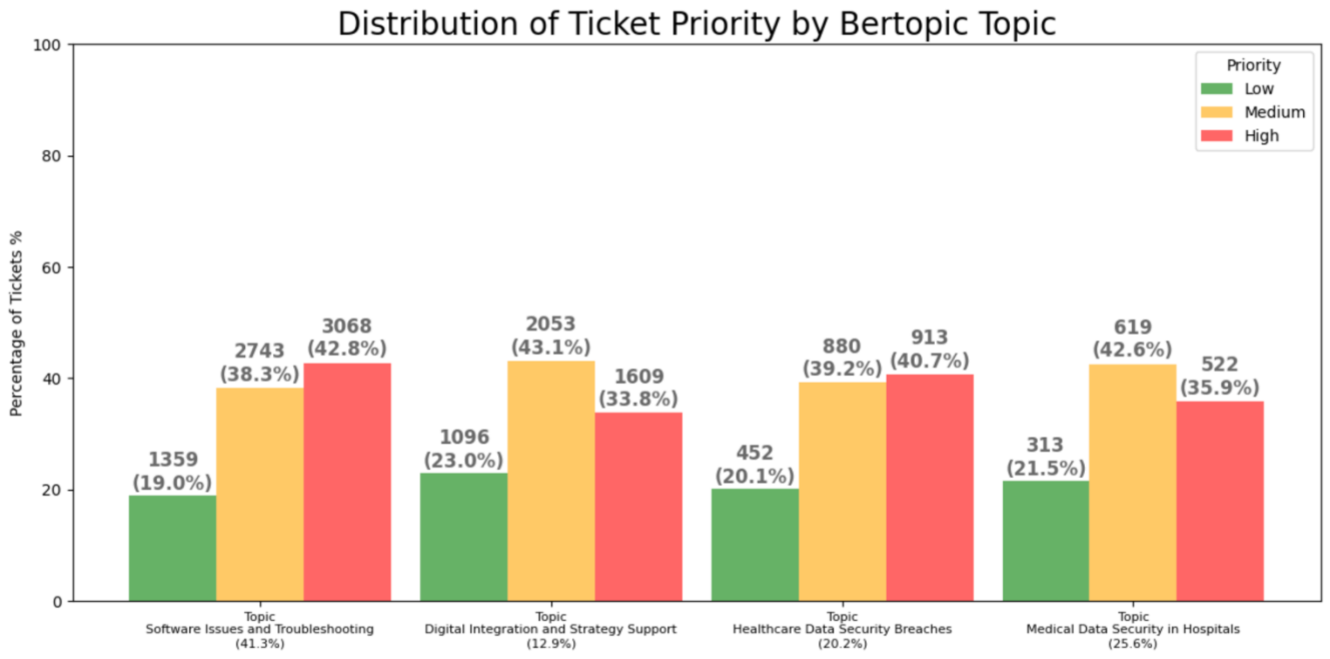


Figure 5 – *Distribution of Ticket Priority by BERTopic Topic*

4.3 Model Comparison and Insights

The LDA model achieved a coherence score of 0.6707, while BERTopic achieved a score of 0.7656 indicating better quality and interpretability of the extracted topics. Thus, the use of embeddings enabled better preservation of the semantic relationships present between words by overcoming the limitations of the bag-of-words approach.

Interestingly, both models identified three similar recurring themes in the support tickets:

1. Technical Issue Troubleshooting and Resolution
2. Hospital Medical Data Security Breaches
3. A topic related to digital strategy: Digital Strategy and Integration Support in BERTopic and Digital Marketing Strategy Optimization in LDA.

Both models attribute more than 40% of the tickets to the topic “Technical Issue Troubleshooting and Resolution”. Omnilytics should therefore try to focus its efforts in this category: for example, it could try to increase the technical documentation provided to customers, improve the quality of diagnostic tools and self-service support channels.

4.4 Strategic Implications for Omnilytics

In addition to providing Omnilytics with the main recurring themes within support tickets, this analysis provides Omnilytics with an analytical tool to be able to continuously monitor customer support. Using these models, the company can:

1. Monitor changes in ticket volume in each topic over time, becoming able to identify early increases in requests in specific areas.
2. Identify new emerging topics by observing changes within the BERTopic outlier category. An increase in outlier cases may indicate the emergence of new needs.
3. Analyze the priority distribution of tickets within each topic. An increase in high priority tickets within a topic may indicate a critical deterioration in the solutions offered by Omnilytics.

5. Priority Classification

To address Omnilytics' second objective - predicting the priority level of each ticket - we developed and tested several algorithms using a supervised learning approach.

The motivation behind this effort is to support helpdesk staff across departments by providing a machine learning tool capable of identifying which tickets should be addressed with greater urgency.

As with topic modeling, we limited the classification task to tickets written in English. Including German tickets would have introduced several vectorization issues, as the pre-trained Google News Word2Vec model only supports English vocabulary[2]. Additionally, Bag-of-Words approaches - both standard and TF-IDF - would lose effectiveness by treating semantically equivalent words in different languages as distinct features.

To ensure a fair comparison between the performance of different vectorization strategies and classification models, we split the English dataset into three subsets:

- Training set (70%): 11,436 tickets
- Validation set (10%): 1,634 tickets
- Test set (20%): 3,268 tickets

All models were trained on the same training set and evaluated on the same validation and test sets, ensuring consistency and comparability across experiments.

5.1 Feature Extraction

To test a variety of approaches, we experimented with different types of input for our classification algorithms. More specifically we transformed the *subject* and *body* fields – merged and cleaned in the *text_combined_cleaned* column – using four different vectorization methods.

Countvectorizer is a classical bag-of-words approach that transforms text into a vector of word counts. It builds a vocabulary from the training corpus and represents each ticket by the frequency of each word in that vocabulary. To reduce noise, we used the max_df parameter to remove words that appear in more than 85% of the tickets.

[2] OpenAI, instead, can work also with German.

Tfidfvectorizer is also a bag-of-words method, but instead of simple counts, it applies a weighting scheme that reflects the importance of a word in a given document relative to the entire corpus. It downweights common words across tickets and upweights words that are frequent in a specific ticket but rare across the corpus.

Word2Vec, unlike bag-of-words approaches, capture semantic relationships between words by positioning similar words close to each other in a continuous vector space. Specifically, we used the pre-trained *Google News* model, which provides word-level embeddings. Since the model does not produce sentence-level representations, we aggregated word vectors using two strategies:

- a maximum vector, which takes the maximum value per dimension across all word vectors in a ticket (highlighting the most semantically dominant terms);
- an average vector, which computes the mean of all word vectors (capturing the general context).

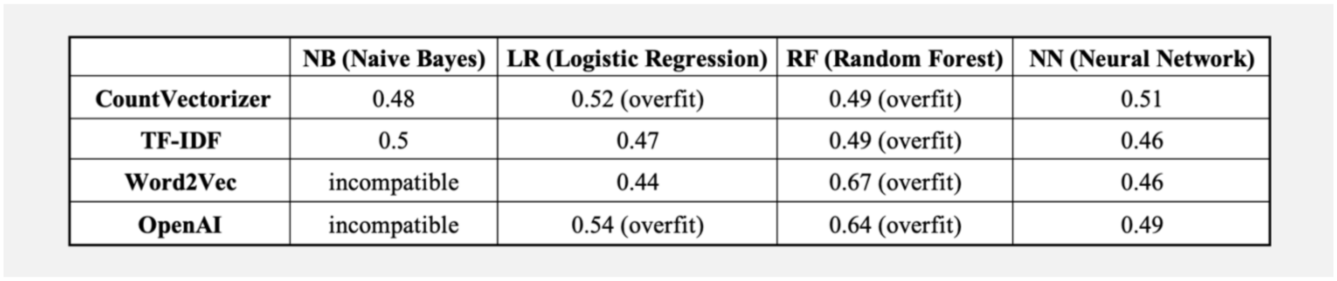
By concatenating these two, we obtained a 600-dimensional vector per ticket, balancing salient keywords with broader context.

OpenAI text Embeddings were generated using the *text-embedding-3-large* model via the OpenAI API. Unlike Word2Vec, this model provides document-level embeddings directly, returning a single 3072-dimensional vector for each input text. Thanks to its transformer-based architecture and extensive training data, this method offers richer and more robust semantic representations than traditional word-level embeddings.

Using these four different types of textual representations allowed us to explore how various vectorization techniques interact with different classification models and influence their performance.

5.2 Classification Results

For the priority prediction task, we evaluated four machine learning algorithms: Naive Bayes, Logistic Regression, Random Forest, and a Neural Network. Each classifier was tested across four different text representation strategies, resulting in a total of 14 unique vectorizer-classifier pipelines. The performance of each configuration is summarized in Fig. 6.



	NB (Naive Bayes)	LR (Logistic Regression)	RF (Random Forest)	NN (Neural Network)
CountVectorizer	0.48	0.52 (overfit)	0.49 (overfit)	0.51
TF-IDF	0.5	0.47	0.49 (overfit)	0.46
Word2Vec	incompatible	0.44	0.67 (overfit)	0.46
OpenAI	incompatible	0.54 (overfit)	0.64 (overfit)	0.49

Figure 6 – *Classification Report of all 14 pipelines*

As shown, the pipelines performed similarly, with accuracy consistently around 0.5. Since each classifier was initialized only once, overfitting is especially likely when using embedding-based vectorizers, which tend to provide richer and more complex feature representations. The *(overfit)* label was added when a clear overfitting is detected, particularly when training accuracy is greater than the validation one by more than 8%.

Naive Bayes assumes that features are statistically independent, which is incompatible with dense semantic embeddings (such as Word2Vec or OpenAI embeddings), where vector components are highly interdependent. For this reason, Naive Bayes was applied only to the bag-of-words representation (*CountVectorizer)* and to the TF-IDF representation. While TF-IDF led to a slightly higher accuracy (0.50 vs. 0.48), the model trained with CountVectorizer achieved a higher weighted F1-score (0.48 vs. 0.45), suggesting a better balance in predicting across the three priority levels (see Fig. 7).

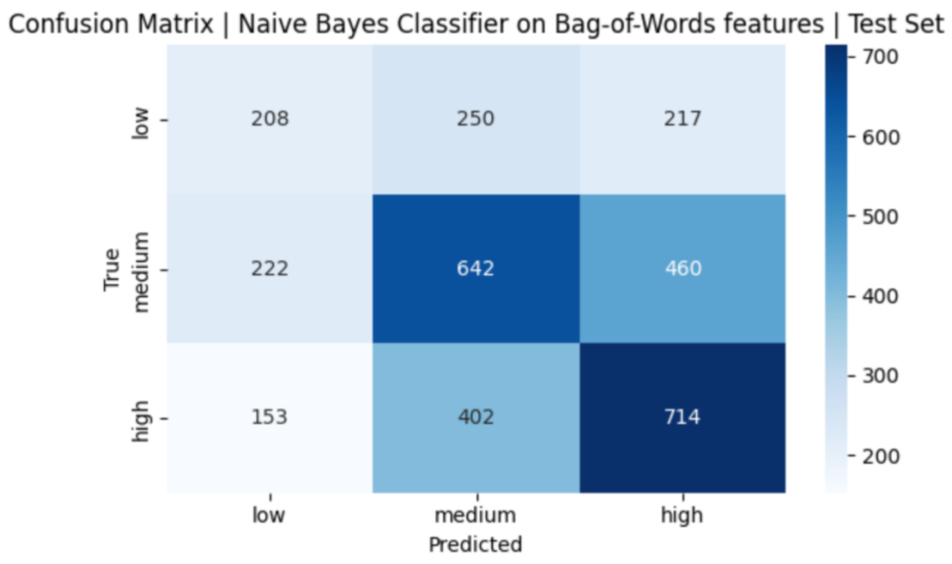


Figure 7 – *Confusion Matrix of NB on BoW (CountVectorizer)*

Logistic Regression was trained on all four types of input representations. Compared to Naive Bayes, we observed a general improvement in accuracy, but this came at the cost of increased overfitting. For all inputs, the accuracy on the training set was significantly higher than on the test set, with a gap consistently exceeding 8 percentage points. LR performed the best on TF-IDF features (see Confusion Matrix, Fig. H in Appendix).

Random Forest models were also trained on all four input types and similarly exhibited substantial overfitting. The accuracy on the training set exceeded that of the test set by over 13%, and by more than 30% when using embedding-based inputs. This highlights the model's tendency to overfit high-dimensional feature spaces without sufficient generalization. RF performed the best on classic BoW features (see Confusion Matrix, Fig. I in Appendix).

Neural Networks demonstrated a lower tendency to overfit, thanks to the use of regularization and dropout layers. However, the overall accuracy remained modest, hovering around 50% across all input types. While the overfitting problem was partially addressed, the model struggled to capture enough discriminative features to yield strong predictive performance. NN performed the best on OpenAI features (see Confusion Matrix, Fig. L in Appendix).

6. Conclusion

As we reach the conclusion of this paper, we can now review the steps we have taken to address the two objectives set by Omnilytics' leadership (see Introduction).

To identify the four main recurring themes within the received tickets, we employed two models: Latent Dirichlet Allocation (LDA), which relies on a BoW approach, and BERTopic, which leverages OpenAI embeddings and clustering algorithms. The quality of the generated topics was assessed using the *coherence score*, which clearly favored the BERTopic model (0.77 vs. 0.67). This performance gap is partly attributable to the use of OpenAI embeddings.

Despite their methodological differences, both models identified common key themes, including a medical/hospital-related topic, a theme linked to digital strategy, and one concerning technical issues - the latter being the most prevalent, encompassing approximately 40% of all tickets.

The results of the ticket priority classification task were less satisfactory and robust, yet still acceptable. The pipelines we developed outperformed a random classifier[3], and therefore we believe these models can still provide meaningful value to Omnilytics by assisting helpdesk staff in prioritizing which tickets to address first.

6.1 Limitations & Future Scenarios

A first limitation of the current approach is the use of *text_combined_cleaned* as the feature variable. This merges the ticket subject and body by simply appending the first at the beginning of the latter. This method may unintentionally reduce the importance of the subject, as its words are treated indistinguishably from the rest of the body. However, the subject often contains key summarizing information and keywords that are particularly valuable for classification and topic modeling. Future work could benefit from treating the subject and body separately or assigning greater weight to the subject content.

Another constraint of our analysis is that we had to exclude tickets in German. As the company continues to grow, a natural step would be to expand the research to include those tickets and hopefully tickets in other languages too. This applies more to topic modeling, as for classification we already have OpenAI which can generate embeddings starting from texts in many languages.

This work lays a solid foundation for intelligent ticket analysis within Omnilytics. While full automation remains out of reach for now, the tools developed can meaningfully support human decision-making and offer a scalable path toward smarter, multilingual support systems in the future.

[3] Since we have 3 priority classes, a random classifier would have an expected accuracy of 0.33.

References

[1] Wiranto and Uswatunnisa (2022). Topic Modeling for Support Ticket using Latent Dirichlet Allocation | Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi). Jurnal.iaii.or.id.
https://jurnal.iaii.or.id/index.php/RESTI/article/view/4542/680

[2] Grace, A. (2025, May 16). Leveraging Natural Language Processing (NLP) for Intelligent Incident Ticket Classification.
https://www.researchgate.net/publication/391423910

[3] OpenAI. (n.d.). *Vector Embeddings - OpenAI API*. Platform.openai.com.
https://platform.openai.com/docs/guides/embeddings

[4] OpenAI. (n.d.). *Text and Prompting - OpenAI API*. Platform.openai.com.
https://platform.openai.com/docs/guides/text

[5] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space.
https://arxiv.org/abs/1301.3781

Appendix

Fig. A – *Type* column distribution

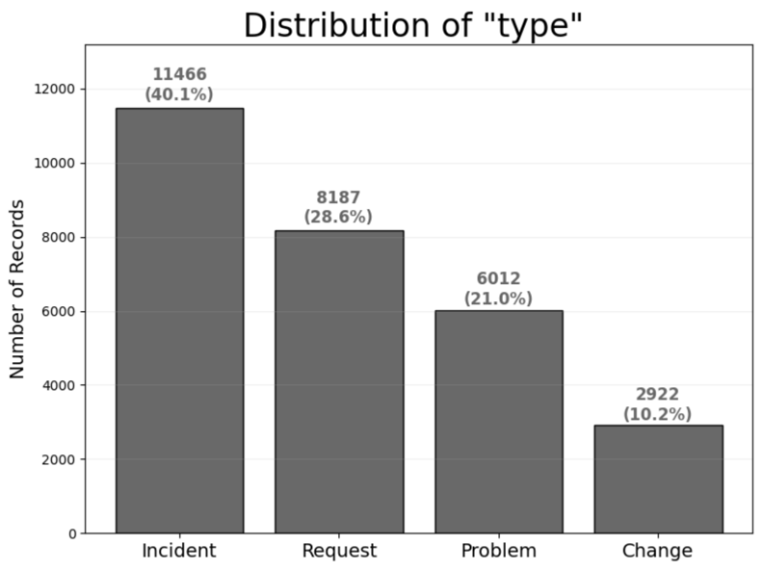


Fig. B – *Queue* column distribution

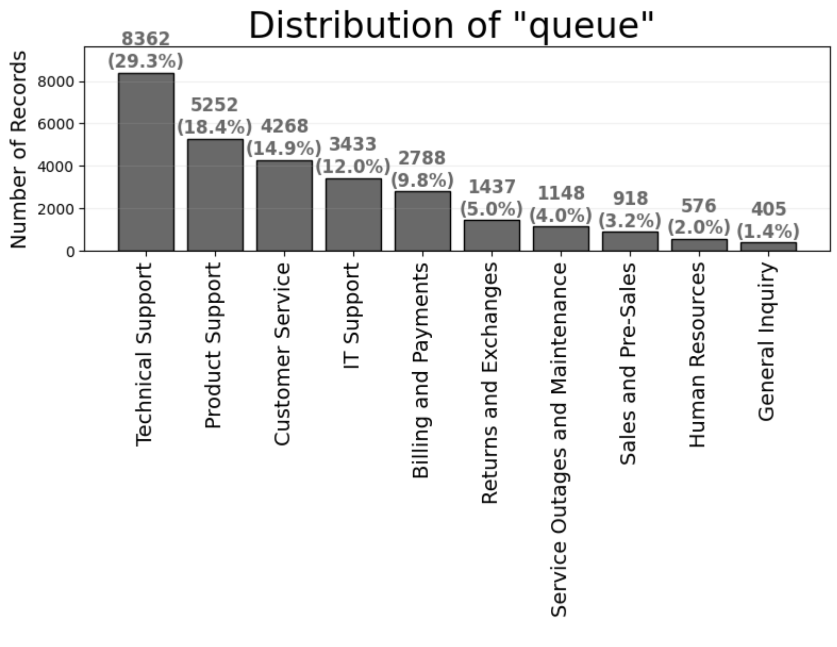


Fig. C – *Priority* column distribution

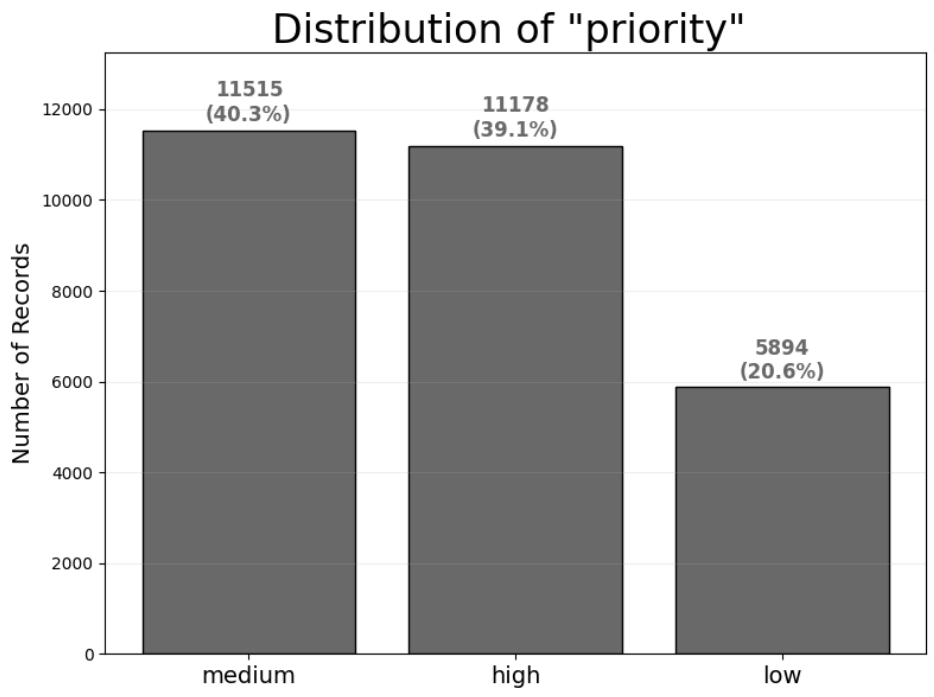


Fig. D – *Language* column distribution

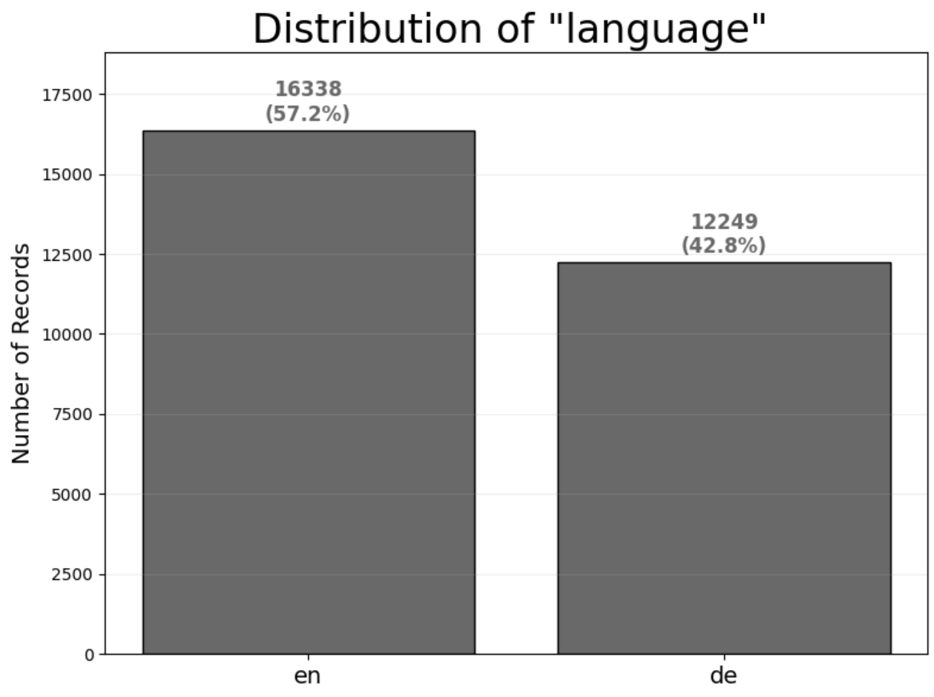


Fig. E – *Tag_1* column distribution

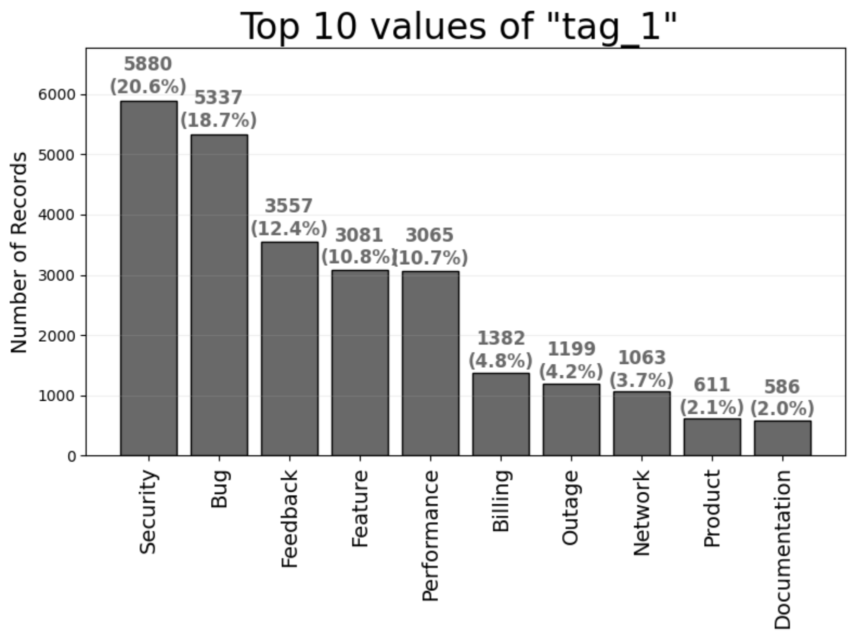


Fig. F – Type distribution by priority

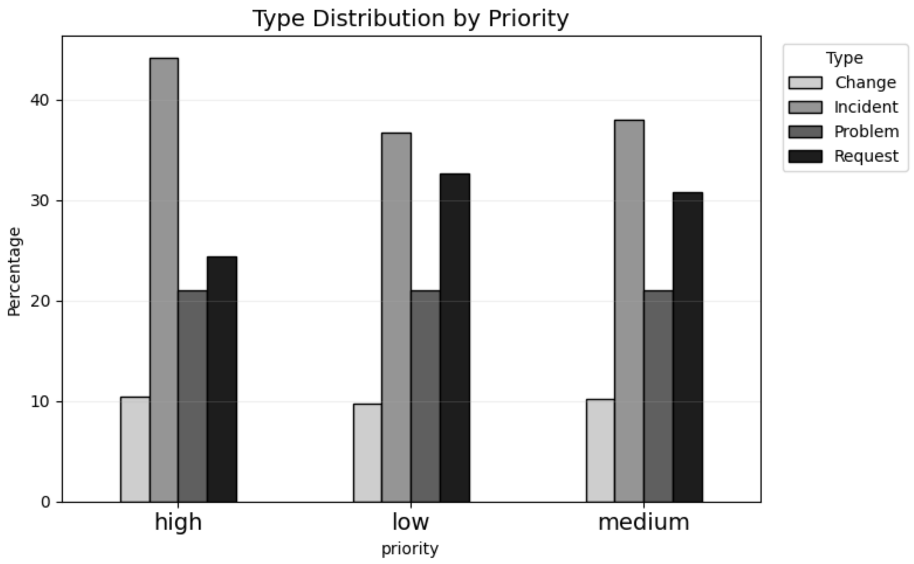


Fig. G – Type distribution by Queue

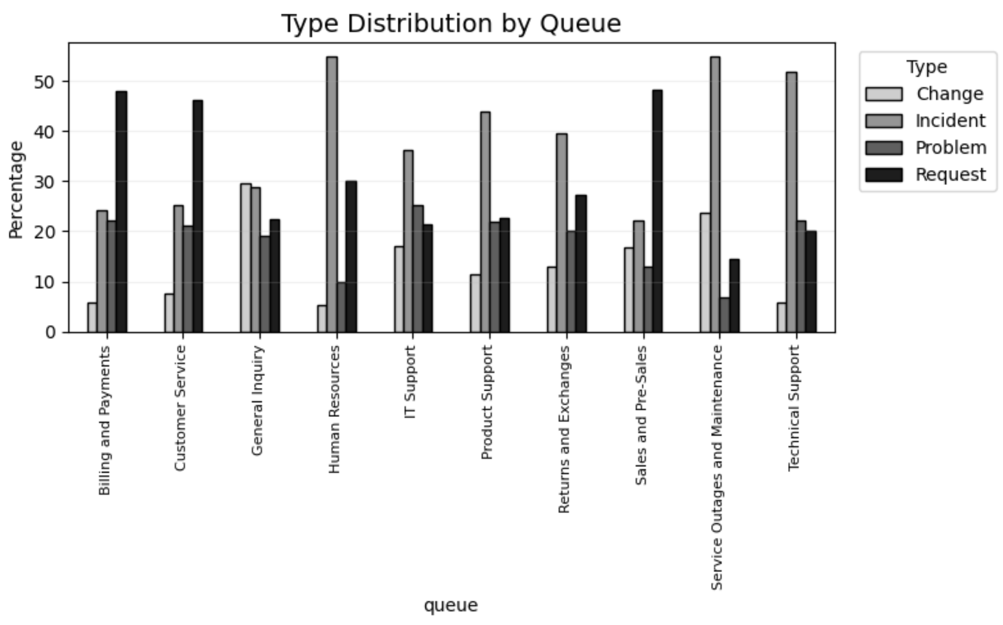


Fig. H – Confusion Matrix of LR on TF-IDF features

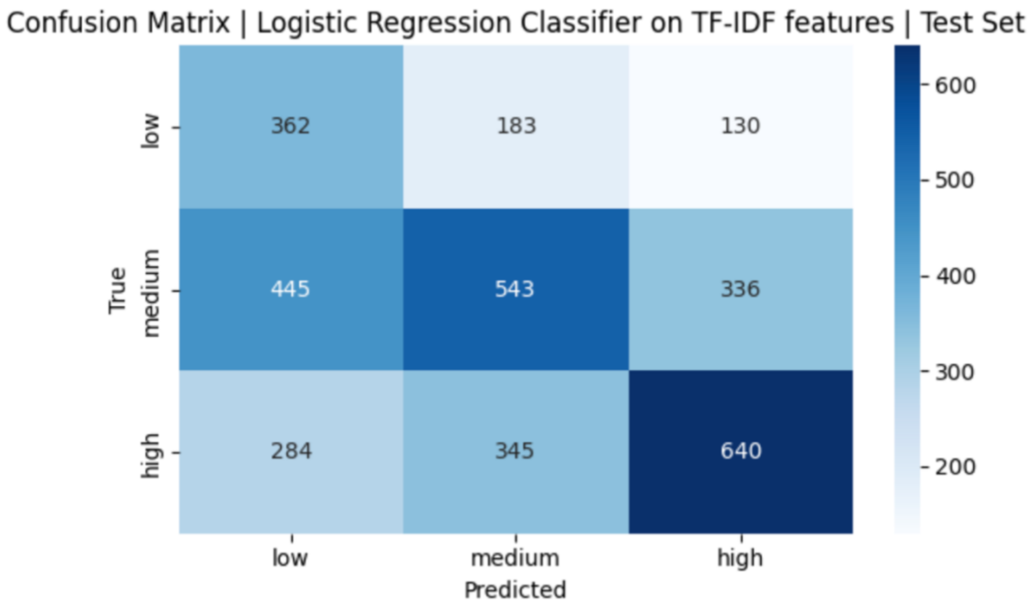


Fig. I – Confusion Matrix of RF on classic BoW features

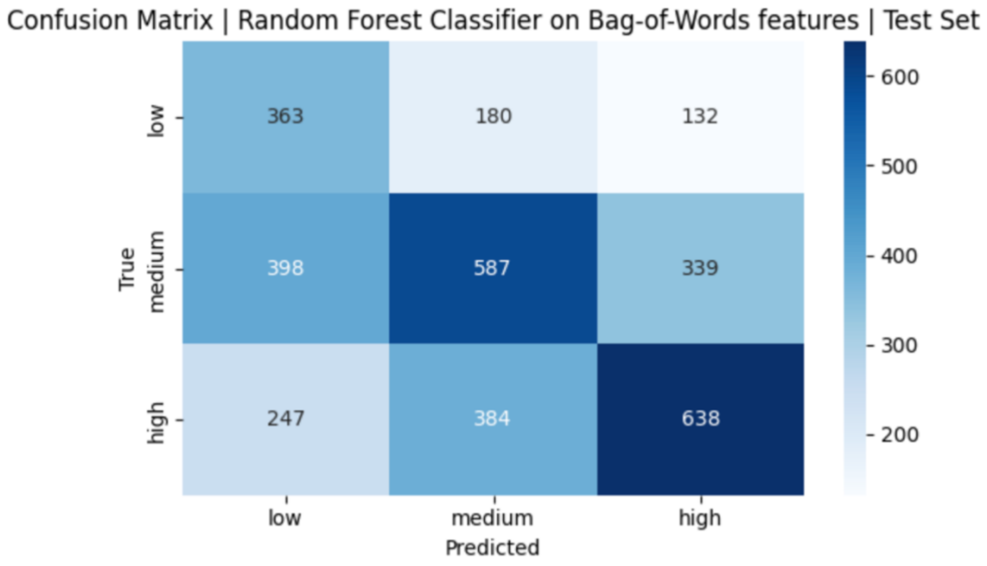
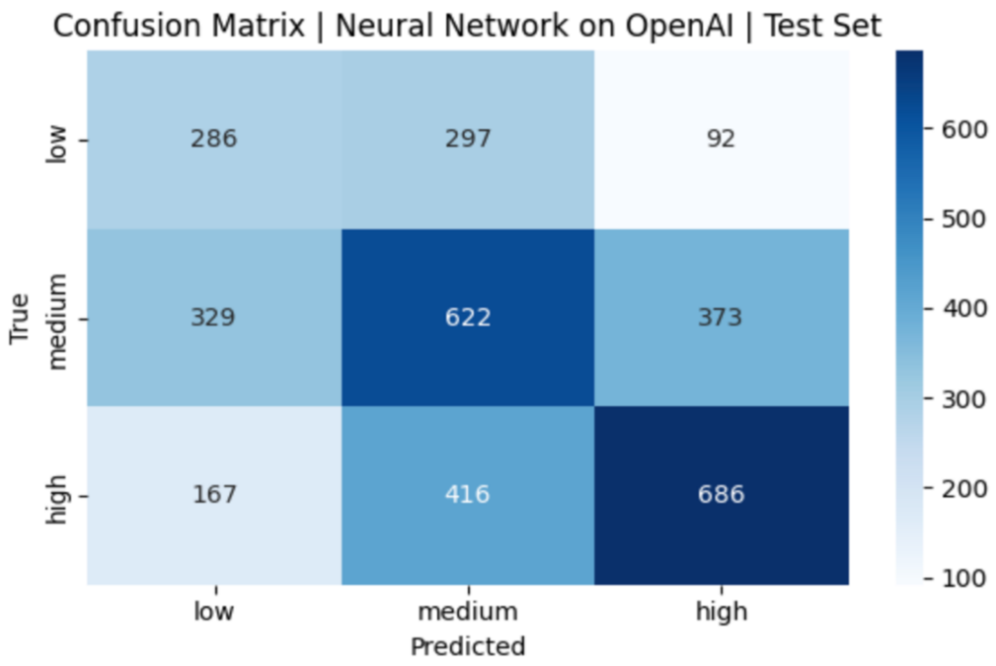


Fig. L – Confusion Matrix of NN on OpenAI features



Confusion Matrix | Neural Network on OpenAI | Test Set
True
low
medium
high
Predicted
low
medium
high
286
297
92
329
622
373
167
416
686
600
500
400
300
200
100