PRODUCT FEEDBACK ANALYZER: FROM COMMENTS TO CLEAR INSIGHTS

Dooinn KIM @ DSTI

github.com/dooinn/

1. Introduction

With the rapid growth of digital platforms, customers now share their experiences and opinions on many online sources, such as social media, e-commerce websites, and discussion forums. These comments provide companies with valuable information about how users feel about different product features, for example *price*, *design*, *performance*, *battery life*, or *camera* quality. However, since the comments are unstructured and written in many styles and languages, analysing them manually becomes very challenging.

The **Product Feedback Analyzer** [1] project was created to address this issue. Its main purpose is to automatically classify and analyse customer comments to discover useful patterns and insights. The system performs two key tasks:

1. **Aspect-based classification**, which identifies the specific product feature being mentioned, and
2. **Sentiment analysis**, which detects whether the customer's opinion is positive, negative, or neutral.

By combining these two tasks, the system helps companies understand both *what* customers are talking about and *how* they feel about it.

Recent developments in Natural Language Processing (NLP), especially transformer-based models, have improved the accuracy and efficiency of this type of analysis. Models like **BERT**, **DistilBERT**, and **RoBERTa** use deep learning techniques to better understand complex text structures and emotional tone (Journal of Big Data, 2023; Liu et al., 2019).

From a practical point of view, the Product Feedback Analyzer is designed to support marketing and product development teams. It provides an interactive dashboard that displays topic distribution and sentiment balance. Users can also export the results for more detailed offline analysis. This helps

[1] https://github.com/dooinn/product_feedback_analyzer

organizations identify product issues more quickly, measure customer satisfaction, and monitor how customer perceptions change over time.

2. Methods

This project uses a supervised learning approach with the objective of classifying user comments into predefined categories and analysing their sentiment. The problem is defined as a multi-class text classification task, where each comment is assigned to one of six smartphone-related aspects: *camera*, *price*, *design*, *performance*, *battery*, or *others*.

The complete workflow is organized into seven main stages, each supported by specific Python libraries and frameworks:

1. **Data Collection:** Comments were collected using the Apify YouTube Comments Scraper API, which enables large-scale extraction of text, timestamps, and metadata from smartphone review videos. The scraper is activated through a Streamlit interface, where users can enter a YouTube video URL and directly download the comments for analysis.

2. **Preprocessing:** Pandas was used for data preparation, including feature selection and label encoding. The primary feature for training was the comment text. The category variable—representing one of the five aspect classes—was converted into numerical labels so the dataset could be used in supervised training.

3. **Zero-Shot Evaluation:** Several pre-trained transformer models—DistilBERT, BERT, and RoBERTa—were tested without further training to establish baseline accuracy. The Hugging Face Transformers library was used to load these models and run zero-shot inference, helping to identify which architecture had the best contextual understanding for this type of comment data.

4. **Fine-Tuning:** The selected model was then fine-tuned with 1,457 manually labelled comments to improve performance in the smartphone domain. The Transformers library handled most of the optimization steps, while PyTorch served as the training backend. This fine-tuning phase increased precision and improved the model's ability to interpret domain-specific text.

5. **Evaluation and Logging:** The model was evaluated using accuracy and F1-score to ensure a balanced measurement of precision and recall. MLflow was integrated to log experiments, store parameters, and track performance across different configurations.

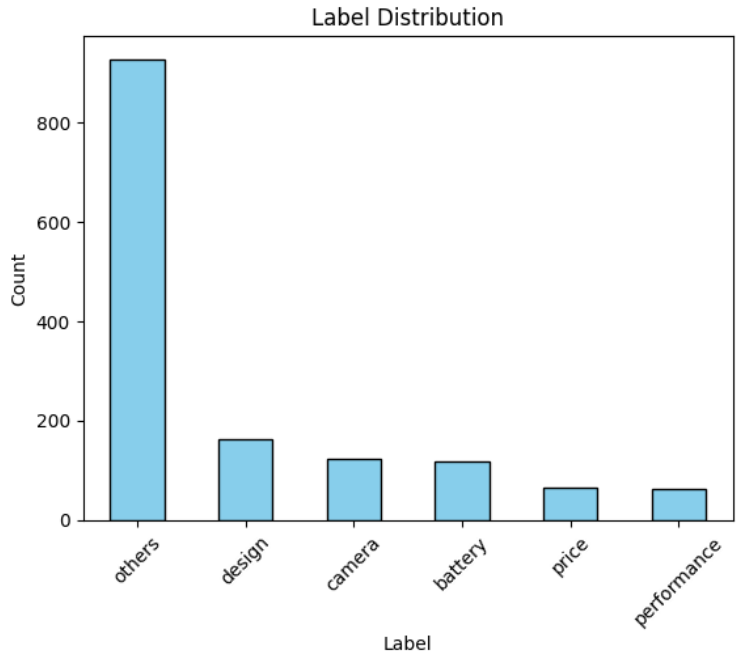
6. **Final Model Export:** After comparing results, the BERT-base-uncased model was selected as the final version because it provided the best balance between accuracy and computational efficiency. The trained model was then exported and saved for deployment.

7. **Deployment:** The final model was deployed through a Streamlit web application, which allows users to enter a YouTube URL, automatically scrape comments, and run real-time classification and sentiment analysis. The results are visualised using Plotly, with interactive charts for sentiment distribution and category frequency. The entire application was containerized using Docker, ensuring consistent execution across different environments and making the system easy to distribute and run without dependency conflicts.

2.1. Data Description

The dataset for this project was created from YouTube reviews related to smartphones to guarantee both relevance and good contextual understanding. The data was extracted using the Apify YouTube Comments Scraper API, which supports bulk collection of user comments together with the text, timestamps, and metadata. Using this method, a total of 1,457 comments were collected from 100 English-language smartphone review videos.

After the extraction, all comments were manually annotated into six predefined categories: *camera*, *price*, *design*, *performance*, *battery*, and *others*. These categories were chosen because they reflect the main smartphone features that consumers usually discuss when giving feedback. The *others* category was added to include comments that do not match the core aspects or that are unrelated to smartphone features, such as remarks about the video content itself or general conversation.



Label Value Counts

Label	Count
Others	927
Design	162
Camera	123
Battery	117
Price	66
Performance	62
Total	**1,457**

***Figure 1**. Label Distribution & Value Counts*

Looking at the label distribution (***Figure 1***), we can observe a strong imbalance, with the *others* category accounting for about 64% of all comments. This imbalance happens mainly because many users post general reactions (such as “Great video!” or “I love this channel”), or comments that are not directly related to smartphone features. Therefore, these comments are grouped into *others* instead of being assigned to a specific product aspect.

Such imbalance can create biased model predictions, since the classifier may overfit to the dominant *others* class while paying less attention to the under-represented categories. To reduce this issue, class weighting was selected as the main strategy in this project. This approach adjusts the loss function so that misclassified samples from minority classes receive higher penalties, motivating the model to learn these categories more effectively. Class weighting was preferred over data augmentation or

oversampling because it is computationally efficient, does not artificially expand the dataset, and integrates smoothly with transformer-based models, making it well suited for text-classification tasks with moderate imbalance (Kumar et al., 2025).

2.2. Modeling

The modelling phase of this project focused on developing and evaluating transformer-based models to classify and analyse YouTube comments related to smartphone reviews. The process was carried out in two main steps:

1. **Zero-shot classification**, which was used to evaluate how well the models perform without any additional training, and
2. **Fine-tuning**, where the models were trained on a manually labelled dataset of 1,457 comments to improve accuracy for this specific task.

2.2.1. Model Selection

Three pre-trained transformer models — DistilBERT, BERT, and RoBERTa — were selected because they have demonstrated strong performance across many Natural Language Processing (NLP) tasks, including sentiment analysis, text classification, and question answering.

- **BERT (Bidirectional Encoder Representations from Transformers)** introduced a bidirectional understanding of context, meaning the model can learn relationships from both the left and right sides of a sentence (Devlin et al., 2019).
- **RoBERTa (Robustly Optimized BERT Approach)** is an improved version of BERT that uses a larger training corpus, more training iterations, and dynamic masking, which results in consistently higher performance on downstream NLP tasks (Liu et al., 2019).
- **DistilBERT** is a distilled and lighter version of BERT that keeps about 97% of BERT's language understanding ability while being faster and more memory-efficient, making it suitable for resource-constrained environments (Sanh et al., 2019).

These models were compared to determine which one provides the best balance between accuracy, processing speed, and efficiency for analyzing user comments.

2.2.2. Zero-Shot Classification

In the first stage, the models were evaluated using zero-shot classification, meaning they were tested without any additional training. This step made it possible to see how well each model could understand and categorize comments based only on its general language knowledge. Pretrained models such as *roberta-large-mnli*, *typeform/distilbert-base-uncased-mnli*, and *MoritzLaurer/deberta-v3-base-zeroshot-v1* were used. Each comment was processed through a Hugging Face pipeline, which predicted one of the six labels — *battery*, *camera*, *design*, *performance*, *price*, or *others*.

The goal of this phase was to check whether the models could correctly connect user comments to the appropriate product aspects. While they demonstrated a basic level of understanding, the accuracy remained limited. This is mainly because YouTube comments often contain informal wording, short

expressions, or ambiguous phrases, which makes classification challenging without task-specific fine-tuning.

2.2.3. Fine-Tuning and Evaluation

To improve performance, the models were fine-tuned on the labelled dataset. The data were split into 80% for training and 20% for testing, using stratified sampling so that the label proportions remained consistent across both sets. A separate validation set was not created; instead, model evaluation was performed after each epoch using the test data.

Before training, all comments were tokenized with each model's tokenizer (maximum length 128) and then converted into PyTorch tensors. The training was carried out using the Hugging Face Trainer API, with AdamW as the optimizer and cross-entropy loss as the objective function. AdamW was selected because it separates weight decay from gradient updates, which leads to better convergence and reduces overfitting when fine-tuning transformer models (Loshchilov & Hutter, 2019). Cross-entropy loss was used since it is the standard choice for multi-class classification, effectively penalizing incorrect predictions and matching well with softmax-based outputs (Goodfellow et al., 2016).

Two training strategies were applied to study the impact of label imbalance:

1. **Imbalanced training**, where all samples were treated equally.
2. **Class-weighted training**, where weights were added to the loss function to give more importance to the minority classes and reduce the influence of the dominant *others* category.

A grid search was carried out to test different parameter combinations. The experiments tried two learning rates (2e−5 and 3e−5), two batch sizes (8 and 16), and five training epochs. Each configuration was applied to all three models, resulting in 24 total runs (3 models × 2 learning rates × 2 batch sizes × 2 weighting strategies). Early stopping (patience = 2) was used to prevent overfitting if the validation loss did not improve. The complete process was logged with MLflow, which stored training parameters, performance metrics, loss curves, and confusion matrices to make comparison easier.

Model performance was evaluated using accuracy, precision, recall, and F1-score, both overall and for each class. This structured experimentation helped identify the best-performing configuration, which was later chosen as the final model for deployment.

3. Results

3.1. Zero-Shot Classification

	Precision			Recall			F1			Support
	DistilBERT	RoBERTa	BERT	DistilBERT	RoBERTa	BERT	DistilBERT	RoBERTa	BERT	
Battery	*0.75*	*0.70*	*0.55*	*0.55*	*0.59*	*0.81*	*0.63*	*0.64*	*0.65*	*117*
Camera	*0.38*	*0.46*	*0.44*	*0.71*	*0.73*	*0.62*	*0.49*	*0.57*	*0.52*	*123*
Design	*0.18*	*0.26*	*0.17*	*0.41*	*0.24*	*0.13*	*0.25*	*0.25*	*0.15*	*162*
Others	*0.97*	*0.83*	*0.82*	*0.07*	*0.43*	*0.28*	*0.12*	*0.57*	*0.41*	*927*
Performance	*0.09*	*0.08*	*0.07*	*0.87*	*0.50*	*0.47*	*0.17*	*0.13*	*0.12*	*62*
Price	*0.43*	*0.42*	*0.17*	*0.83*	*0.77*	*0.67*	*0.57*	*0.55*	*0.27*	*66*
Accuracy							*0.27*	*0.47*	*0.36*	*1457*
Macro Avg	*0.47*	*0.46*	*0.37*	*0.57*	*0.54*	*0.50*	*0.37*	*0.45*	*0.35*	*1457*
Weighted Avg	*0.75*	*0.67*	*0.63*	*0.27*	*0.47*	*0.36*	*0.23*	*0.52*	*0.39*	*1457*

***Table 1**. Baseline (Zero-Shot) Evaluation Metrics for Transformer Models Across Categories*

The zero-shot experiment (***Table1***) tested DistilBERT, BERT, and RoBERTa without any additional training to measure their basic performance. Among these models, RoBERTa achieved the highest accuracy (0.47), followed by BERT (0.36) and DistilBERT (0.27). RoBERTa also showed a better balance between precision, recall, and F1-score, which means it understood the context of comments more effectively.

All three models performed well on the majority classes such as *others* and *battery*, but they struggled with minority categories like *price* and *performance*. This was mainly because of the class imbalance in the dataset, where most comments belonged to the *others* category. DistilBERT gave the lowest results overall, likely due to its smaller model size and reduced capacity compared to the full transformer architectures.

3.2. Fine-tuned: Imbalanced vs Class-Weighted

Run Name	Type	Epochs	Eval Accuracy	Eval Loss	Train Loss	Learning Rate
bert-base-uncased_lr2e-05_bs8	Imbalanced	5	0.8493	0.5835	0.5356	8.49×10^{-6}
bert-base-uncased_lr2e-05_bs8	Class-Weighted	5	0.8568	0.8664	0.7795	8.49×10^{-6}

***Table2.** Best Imbalanced vs Class-Weighted Model Performance*

Out of the 24 training configurations tested, the BERT-base-uncased model performed the best when using a learning rate of 2e−05, batch size of 8, and 5 epochs (***Table 2***). Particularly, the class-weighted BERT model achieved a slightly higher evaluation accuracy (0.8568) than the imbalanced model (0.8493). This shows a small improvement in overall generalization. However, the weighted model had a higher evaluation loss (0.8664) compared to the imbalanced one (0.5835), meaning it traded some model confidence for better balance across classes.

	Precision		**Recall**		**F1- Score**		**Support**
	Imbalanced	**Class-Weights**	**Imbalanced**	**Class-Weights**	**Imbalanced**	**Class-Weights**	
Battery	0.913	0.957	0.913	0.957	0.913	0.957	23
Camera	0.826	0.857	0.760	0.720	0.792	0.783	25
Design	0.605	0.615	0.697	0.727	0.648	0.667	33
Others	0.888	0.906	0.941	0.930	0.914	0.918	186
Performance	1.000	0.571	0.167	0.333	0.286	0.421	12
Price	0.889	0.818	0.615	0.692	0.727	0.750	13
Accuracy					0.849	0.856	292
Macro Avg	0.854	0.787	0.682	0.727	0.713	0.749	292
Weighted Avg	0.858	0.855	0.849	0.856	0.839	0.853	292

Table 3. *Classification Report of best Imbalanced vs Class-Weighted Model Performance*

To examine the classification reports (***Table 3***), both models performed very well on the majority classes, such as *battery* and *others*, with precision and recall scores above 0.90. The class-weighted approach, however, showed clear benefits for minority classes, especially *performance* (F1-score improved from 0.286 to 0.421) and *price* (from 0.727 to 0.750). This confirms that adding weights helped the model better recognize underrepresented categories in the dataset.

Although there were small drops in precision for some majority classes like *camera* and *price*, the class-weighted model achieved more balanced recall and F1-scores overall. In conclusion, class weighting made the model's predictions fairer and more consistent, improving recognition of minority aspects without significantly reducing overall accuracy.

4. Discussion

The results clearly show that the models performed much better after fine-tuning compared to the zero-shot classification stage. At first, the transformer models could only capture general meanings and struggled with specific aspects such as *price* and *performance*. This shows that while pre-trained models understand language well, they still need domain-specific training to recognize the vocabulary and context used in product feedback.

After fine-tuning, the models became more accurate and balanced across all categories. The BERT-base-uncased model achieved the best results, showing that it adapts well to this type of text analysis. Adding class weights gave a small improvement in accuracy but a noticeable improvement in detecting minority classes. This proves that class weighting helps reduce model bias toward the majority category (*others*). Although the weighted model had a slightly higher validation loss, this was an acceptable trade-off for more balanced and fair predictions.

In terms of model fitness (see Appendix B), the training and validation loss curves showed stable learning with a good fit. Both losses decreased steadily, and only a small amount of overfitting appeared near the final epochs. This is normal for transformer fine-tuning and shows that the chosen hyperparameters — learning rate 2e−05, batch size 8, and 5 epochs — provided a good balance between efficiency and stability.

However, there were some limitations. The dataset was small (1,457 comments) and highly imbalanced, with the *others* class making up most of the data. This made it harder for the model to learn about less frequent product aspects. In addition, all comments were collected from English-language YouTube videos, which limits how well the model can generalize to other languages or platforms.

For future work, the project can be improved by:

- Using advanced data balancing methods such as focal loss or synthetic oversampling to increase recall for minority classes.
- Expanding the dataset by collecting more comments from different platforms such as Instagram, Reddit, and TikTok, to create a larger and more diverse dataset for better generalization.

5. Conclusion

This project developed the Product Feedback Analyzer, an NLP-based system that automatically classifies and analyzes customer comments to extract insights about product features and sentiment. By combining transformer-based models with fine-tuning and visualization, the tool effectively converts unstructured feedback into meaningful, actionable information.

The results showed a clear improvement from zero-shot to fine-tuned learning, confirming that domain-specific training is crucial for accuracy and balance. The BERT-base-uncased model achieved the best performance, while applying class weighting further improved recognition of minority aspects and reduced bias toward the dominant *others* category. The model demonstrated good fitness, stable convergence, and minimal overfitting, validating the chosen hyperparameters.

Despite promising results, the study was limited by a small and imbalanced dataset, mostly from English-language YouTube comments. Future work should expand data collection to platforms such as Instagram, Reddit, and TikTok, apply advanced balancing methods, and explore larger or domain-specific transformer models for broader and more robust performance.

Overall, this work demonstrates how fine-tuned transformer models can turn large volumes of online feedback into clear, data-driven insights—supporting smarter, customer-centered business decisions.

6. Appendix

Appendix A: Data Sources Used for Fine-Tuning

The dataset for model fine-tuning was created by collecting comments from multiple YouTube smartphone review videos using the Apify YouTube Comments Scraper API. This API allows bulk extraction of user comments, including text, timestamps, author information, and related metadata, providing structured input for analysis and model training.

- YouTube Comment Dataset: Source Videos, Comments used for training.
- Apify YouTube Comments Scraper API: API Documentation

Appendix B: Model Training Configurations and Results

This appendix presents the experimental results for all transformer models tested during fine-tuning. Each table summarizes the model configurations, evaluation metrics, and performance outcomes under different training strategies. These results were used to identify the best-performing model for deployment.

Run Name	Epochs	Eval Accuracy	Eval Loss	Train Loss	Learning Rate
bert-base-uncased_lr2e-05_bs8	5	0.8493	0.5835	0.5356	8.49×10^{-6}
roberta-base_lr3e-05_bs16	5	0.8459	0.5137	0.5318	1.31×10^{-5}
roberta-base_lr3e-05_bs8	5	0.8459	0.6604	0.4324	1.27×10^{-5}
roberta-base_lr2e-05_bs16	5	0.8459	0.4930	0.6080	8.77×10^{-6}
distilbert-base-uncased_lr3e-05_bs8	5	0.8459	0.6320	0.4575	1.27×10^{-5}
distilbert-base-uncased_lr2e-05_bs8	5	0.8459	0.5877	0.5686	8.49×10^{-6}
bert-base-uncased_lr3e-05_bs8	4	0.8390	0.5366	0.5567	7.43×10^{-6}
distilbert-base-uncased_lr3e-05_bs16	5	0.8329	0.6040	0.5757	1.31×10^{-5}
roberta-base_lr2e-05_bs8	4	0.8288	0.5019	0.5576	4.96×10^{-6}
bert-base-uncased_lr3e-05_bs16	5	0.8288	0.5738	0.5940	1.31×10^{-5}
distilbert-base-uncased_lr2e-05_bs16	5	0.8253	0.6217	0.7073	8.77×10^{-6}
bert-base-uncased_lr2e-05_bs16	5	0.8151	0.6304	0.7089	8.77×10^{-6}

Table B1. Fine-Tuned Model Performance (Imbalanced Data)

Run Name	*Epochs*	*Eval Accuracy*	*Eval Loss*	*Train Loss*	*Learning Rate*
roberta-base_lr3e-05_bs16	*5*	*0.8390*	*0.7887*	*0.6848*	1.31×10^{-5}

roberta-base_lr3e-05_bs8	*5*	*0.8493*	*1.1131*	*0.6942*	1.27×10^{-5}
roberta-base_lr2e-05_bs16	*5*	*0.8425*	*0.7816*	*0.8025*	8.77×10^{-6}
roberta-base_lr2e-05_bs8	*5*	*0.8425*	*0.9333*	*0.6948*	8.49×10^{-6}
bert-base-uncased_lr3e-05_bs16	*5*	*0.8493*	*0.7231*	*0.8198*	1.31×10^{-5}
bert-base-uncased_lr3e-05_bs8	*5*	*0.8425*	*1.0830*	*0.6487*	1.27×10^{-5}
bert-base-uncased_lr2e-05_bs16	*5*	*0.7797*	*0.9213*	*1.0579*	8.77×10^{-6}
bert-base-uncased_lr2e-05_bs8	*5*	*0.8568*	*0.8664*	*0.7795*	8.49×10^{-6}
distilbert-base-uncased_lr3e-05_bs16	*5*	*0.8527*	*0.7618*	*0.8294*	1.31×10^{-5}
distilbert-base-uncased_lr3e-05_bs8	*5*	*0.8527*	*0.9657*	*0.6998*	1.27×10^{-5}
distilbert-base-uncased_lr2e-05_bs16	*5*	*0.8116*	*0.8094*	*1.0313*	8.77×10^{-6}
distilbert-base-uncased_lr2e-05_bs8	*5*	*0.8561*	*0.7993*	*0.7562*	8.49×10^{-6}

Table B2. Fine-Tuned Model Performance (Class-Weighted Data)

Appendix C: Product Feedback Analyzer (Demo)

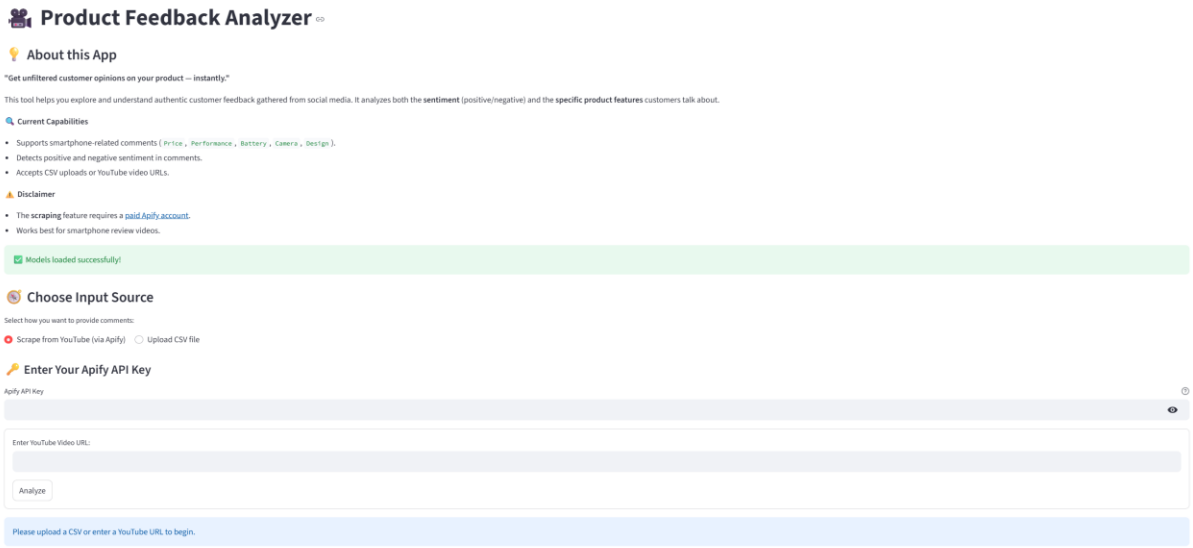


Fig. Product Feedback Analyzer – data input

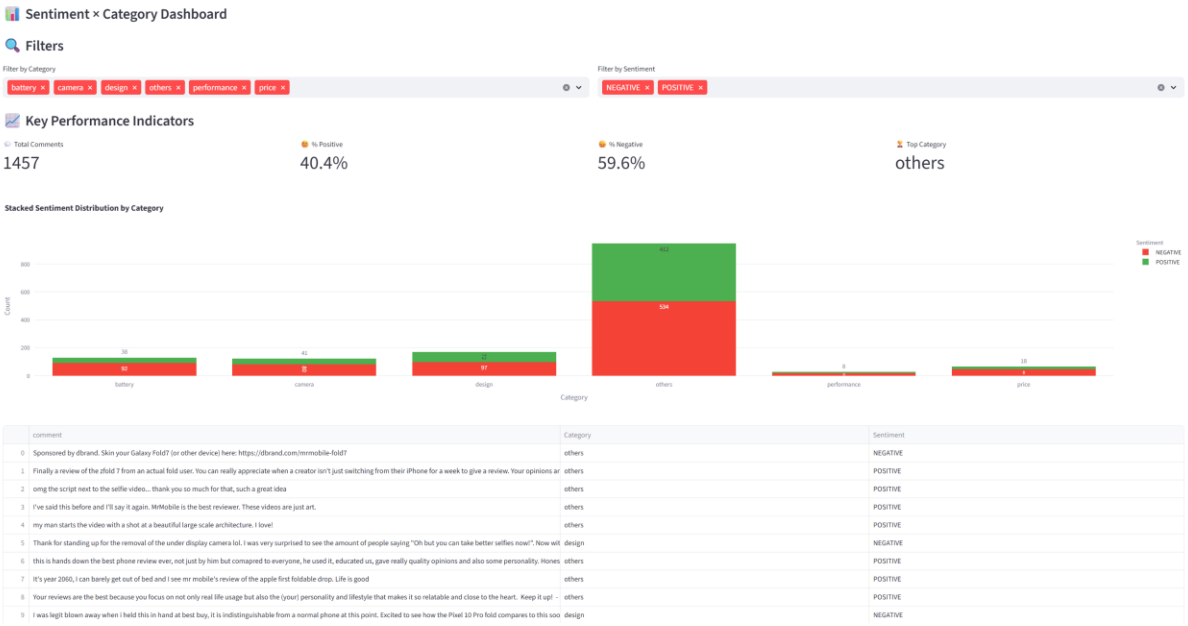


Fig. Product Feedback Analyzer – Dashboard and analysed results

References

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018) *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.* arXiv preprint arXiv:1810.04805. Available at: https://arxiv.org/abs/1810.04805

Goodfellow, I., Bengio, Y. and Courville, A. (2016) *Deep Learning*. MIT Press. Available at: https://www.deeplearningbook.org/

Journal of Big Data (2023) 'Survey of Transformers and Towards Ensemble Learning Using Transformer-Based Models', *Journal of Big Data*. Available at: https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00842-0

Kumar, A., Singh, R. and Gupta, V. (2025) 'Addressing data imbalance in transformer-based multi-label classification', *arXiv preprint*, arXiv:2507.11384. Available at: https://arxiv.org/html/2507.11384v1

Loshchilov, I. and Hutter, F. (2019) 'Decoupled weight decay regularization', *International Conference on Learning Representations (ICLR)*. Available at: https://arxiv.org/abs/1711.05101

Liu, Y., et al. (2019) 'RoBERTa: A Robustly Optimized BERT Pretraining Approach', *arXiv preprint*, arXiv:1907.11692. Available at: https://arxiv.org/abs/1907.11692

Sanh, V., Debut, L., Chaumond, J. and Wolf, T. (2019) *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter.* arXiv preprint arXiv:1910.01108. Available at: https://arxiv.org/abs/1910.01108