Identify Potential Reasons:

1. **Data Quality:** First, the dataset used for training may show an imbalance, with some categories having significantly fewer examples than others. This can cause the model to lean towards the majority class, leading to biased predictions. And then, the dataset may contain noisy or mislabeled examples, which affects the model's ability to make accurate generalizations. Cleaning such noisy data is crucial to improve model performance.
2. **Model Choice:** Naive Bayes models are assumptions of features independence. In reality, features often exhibit some level of correlation, and this assumption may not hold true for all datasets. This can lead to suboptimal performance when the independence assumption is violated. Thats why to properly represent these complexity, a more complicated model could be required. Additionally, this model can have trouble understanding complex word connections or the context of technical jargon. Exploring different types of models like deep learning models that have a greater comprehension of context may be helpful. Lastly, the challenge at hand may surpass the capabilities of a simple model like Naive Bayes. Consideration of more advanced models, like ensemble methods or neural networks, may be warranted.
3. **Feature Extraction:** The features used for training might not fully capture the intricacies of support ticket content. Enhancing feature extraction, possibly by employing advanced techniques like word embeddings, can contribute to better model performance. The model may not take into account the sequential nature of text, missing out on crucial contextual information. Incorporating models that consider text sequences, such as RNN could be explored. The text may not have been preprocessed effectively, resulting in the inclusion of irrelevant information or noise. Improving preprocessing techniques, like removing stop words or stemming, can refine the quality of input data.
4. **Training Process:** The model may not fit a sufficient set of observations, making adequate generalization challenging, especially for tickets with technical details. It is important to enrich the training data set with a comprehensive set of observations. Overfitting may then have occurred, especially if the appropriate routines were not included in the model. Strategies such as changing the regularization parameters or detecting dropouts in neural networks can help to reduce overfitting.

Suggest Improvements:

1. **Data Preprocessing:** Use techniques like oversampling, undersampling, or creating class weight to address class imbalances. This guarantees that various ticket categories are fairly represented throughout the training procedure. Next, find and fix labeling mistakes, get rid of noise, and keep labeling consistent. A clean dataset is essential to the model's accurate learning.
2. **Feature Engineering:** Enhance text representation by utilizing advanced techniques like TF-IDF, word embeddings such as Word2Vec or pre-trained language models like BERT. Enhancing the model's comprehension of support ticket content, these techniques capture semantic relationships. Add other features later, including the duration of the ticket, the use of technical terminology, and other metadata. In order to give the model a more complete input, think about using features that record connections between words or context.
3. **Model Selection:** Consider advanced models like ensemble methods, neural networks, or deep learning architectures capable of capturing intricate patterns in support ticket data. These

models provide a more complex method of problem-solving. To combine predictions from several models and increase the system's overall predictive capacity, use model ensembling.

4. **Regularization and Hyper parameter Tuning:** Fine-tune hyper parameters, including learning rate and regularization, to strike an optimal balance between under-fitting and overfitting. This guarantees that the model will adjust to the complex features of the data. Experiment with diverse tokenization strategies and preprocessing techniques to effectively handle technical jargon and address overlapping themes in support ticket content. By this way the model's accuracy in ticket interpretation and classification may be improved.

Consideration of Additional Techniques:

1. **NLP Techniques:** Incorporate sentiment analysis into the model to help it not only classify support queries but also understand the users' emotional tone. The ability to handle client encounters with more sensitivity and efficiency depends on this extra layer of awareness. Use word embeddings that have already been pre-trained, such as Word2Vec, to improve the model's understanding of context and semantics. By using a methodology that goes beyond conventional techniques, the model is better able to understand finer points of language and increase the accuracy of its ticket classification.

Dealing with Technical Jargon and Overlapping Themes:

1. **Techniques:** Add transformer-based models to the system, such as GPT, BERT etc. Because these models are so good at interpreting context and subtle language, they are especially useful for translating technical support ticket jargon. Implement Ensemble of Models to create a strong classifier and combine more weak classifiers. So combine predictions from many models, each with a focus on a particular area of ticket classification, to create an all-encompassing ensemble approach.

Long-Term Strategy for Model Maintenance:

1. **Retraining:** In order to maintain the system up to date and flexible enough to adjust to changing language trends in support requests, emphasize the significance of routine model updates with fresh, relevant information. Accuracy metrics, false positives/negatives, and performance on certain ticket types should all be carefully assessed as part of this ongoing learning process.
2. **Dataset Updates:** Ensure that the training dataset is consistently updated with the most current support requests by implementing a reliable procedure for ongoing data collecting. By taking the initiative, the model is better able to adapt to changing customer language and stay updated.
3. **Monitoring**: Make sure to keep a close eye on important metrics including F1 score, recall, accuracy, and precision in classification. Maintaining the model's efficacy over time and identifying opportunities for improvement depends on this continuous evaluation. Integrate anomaly detection technologies to reinforce the model maintenance approach. This enables the

system to recognise changes in the distribution of data or unusual behavior in the model, resulting in timely modifications and optimisations.

4. **Version Control:** Tracking model changes requires the use of version control. Stability and possible interruptions are reduced by this procedure, which guarantees the ability to roll back to an earlier version in the event that new upgrades cause unanticipated problems. Github can be utilized for such purposes.