

Machine Learning

MOD006562

Faculty: Science and Engineering

School: Computing and Information Science

Academic Year: 2024/25

Trimester: 2

Student Name : Charlene Opara

Student ID: 2266484

Contents

[1 Introduction 5](#_Toc195251333)

[2 Description of the Dataset 5](#_Toc195251334)

[3 Data Analysis 6](#_Toc195251335)

[4 Preparing Data 7](#_Toc195251336)

[4.1 Data pre-processing 7](#_Toc195251337)

[4.2 Remove Stop Words 8](#_Toc195251339)

[4.3 Stemming 9](#_Toc195251340)

[4.4 Lemmatizer 9](#_Toc195251341)

[4.5 Vectorization 10](#_Toc195251342)

[5 Training and Evaluating Models 10](#_Toc195251343)

[5.1 Model 1 – Support Vector Machine Model 11](#_Toc195251344)

[5.1.1 Model Description 11](#_Toc195251345)

[.5.1.2 Evaluation Design 12](#_Toc195251347)

[5.1.3 Results. 13](#_Toc195251348)

[5.2 Model 2 – Gaussian Model 16](#_Toc195251349)

[5.2.1 Describe your model. 16](#_Toc195251350)

[5.2.2 Explain your evaluation design. 16](#_Toc195251352)

[5.2.3 Present and explain results. 16](#_Toc195251353)

[5.3 Model 3 – Multinomial Model 17](#_Toc195251354)

[5.3.1 Describe your model. 17](#_Toc195251355)

[5.3.2 Explain your evaluation design. 17](#_Toc195251356)

[5.3.3 Present and explain results. 18](#_Toc195251357)

[6 Evaluation Analysis 18](#_Toc195251359)

[7. References 20](#_Toc195251360)

# Introduction

The aim of this project is to determine the most accurate machine learning model for predicting the target values of newly generated chatbot responses. The primary goal is to evaluate and compare the performance of SOME classification models to identify which model best predicts the categories or intent behind chatbot responses. By training these models on a dataset that includes both primary features and target values, the project seeks to fine-tune the algorithms to optimize their performance for predicting unseen data. Through data preprocessing steps such as text cleaning, stop-word removal, and feature vectorization, the models will be trained and evaluated, providing insights into which machine learning approach offers the most accurate predictions for chatbot responses.

# Description of the Dataset

1. Targets:

* Targets are the outcomes or categories a predictive model aims to determine, acting as the dependent variables in a dataset. During training, the model examines the relationship between features (independent variables) and targets to identify patterns that generalize to new data. In this dataset, the target is the "categories" column.

2. Labels:

Labels are the actual outcomes or categories in a training dataset, serving as the ground truth for each data point. In supervised learning, the model compares its predictions to these labels to assess performance and adjust its parameters, evaluating the accuracy of the model. Labels are typically located in the "categories" column of the dataset.

3. Example:

An example is a single data item in a dataset, consisting of its features and, in the case of labeled datasets, its associated label or target.

4. Features:

Features are the input data or independent variables used to make predictions in a model. They represent measurable properties or characteristics of the examples being analyzed. The model uses these features to learn patterns that help predict the target values, so selecting relevant features is important for the model’s performance. In this dataset, the features include session\_id, message\_time, user\_message, chatbot\_response, response\_source, categories, and intent-name.

5. Primary Feature:

The primary feature in a dataset is the most important element for solving a specific problem, as it strongly influences the target variable. Identifying and focusing on this feature can enhance the model's efficiency and accuracy. These features form the foundation of a dataset's structure and play a critical role in building, training, and evaluating machine learning models.

# Data Analysis

This graph illustrates the number of times a label appeared in the dataset;

A graph of a number of people

Description automatically generated

# 4 Preparing Data

### 4.1 Data pre-processing

## This code preprocesses text data in a dataset by converting the text to lowercase, removing all punctuation, and whitespace by reducing multiple spaces to single spaces. It applies these preprocessing steps to the "chatbot\_response" column in the pandas DataFrame, ensuring that the text is cleaner and more consistent for analysis in machine learning tasks.**A computer screen shot of a black screen Description automatically generated**

### 4.2 Remove Stop Words

This Python script uses the **Natural Language Toolkit (NLTK)** to preprocess text by removing **stopwords** (common words like "the," "is," and "and") from the dataset column (chatbot\_response). First, it downloads The remove\_stop\_words() function tokenizes the input text, filters out stopwords, and rejoins the remaining words into a cleaned string (Bird et al., 2009). Finally, the script applies this function to the dataset using Pandas' apply()method (Pandas Development Team, 2023). This preprocessing step is crucial for NLP tasks, as it reduces noise and improves computational efficiency.A screenshot of a computer

Description automatically generated

### 4.3 Stemming

This Python script performs **text stemming** on the dataset's chatbot\_response column using NLTK (Natural Language Toolkit). The code first tokenizes each text entry into words using word\_tokenize(), then applies Porter stemming (PorterStemmer) to reduce words to their root forms (e.g., "running" → "run"). The stemmed words are rejoined into strings and stored back in the dataset (Bird et al., 2009; NLTK Team, 2023).

A black screen with white text

Description automatically generated

### 4.4 Lemmatizer

This Python script uses **NLTK's WordNetLemmatizer** to perform lemmatization on text data in the pandas DataFrame column (chatbot\_response). Lemmatization reduces words to their base or dictionary form (e.g., "running" → "run") to normalize text for NLP tasks (Bird et al., 2009). The lemmatize() function splits the input text into tokens, applies lemmatization , and rejoins the processed words into a string. The operation is applied to the entire column using Pandas' apply() (Pandas Development Team, 2023).A screen shot of a computer program

Description automatically generated

### 4.5 Vectorization

This code employs scikit-learn's TF-IDF vectorizer to transform text responses into numerical feature vectors. The TF-IDF approach calculates word importance by considering the occurrence in the dataset . By setting max\_features=1500, it retains only the most significant 1500 terms, optimizing dimensionality. The final output is converted to a dense NumPy array (toarray()) for compatibility with machine learning algorithms.

A screen shot of a computer program

Description automatically generated

# 5 Training and Evaluating Models

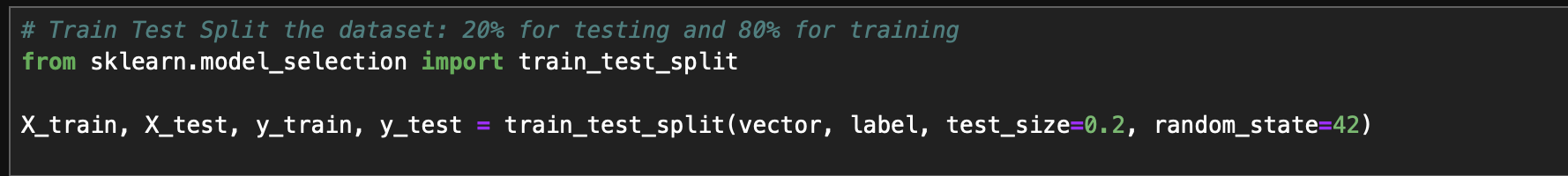
For this task I trained and tested three different models, the SVM model, Gaussian model and the Multinomial model.

​Support Vector Machines (SVMs) are supervised machine learning algorithms primarily employed for classification tasks, though they can also be adapted for regression, referred to as Support Vector Regression (SVR). The fundamental concept of SVM involves identifying the optimal decision boundary, or hyperplane, that best distinguishes between different classes in the feature space. This is accomplished by maximizing the margin between classes, which aids in preventing overfitting and enhancing accuracy. SVMs are versatile, effectively handling both linear and nonlinear data through the use of kernel functions, making them suitable for tasks such as text and image classification. Additionally, SVMs perform efficiently even with high-dimensional data or limited training samples.

Gaussian Naive Bayes (GNB) is a classification algorithm that assumes each feature follows a normal (Gaussian) distribution. This approach is particularly effective for continuous data (Geeksforgeeks, 2025). ​ GNB is efficient, requiring only the calculation of means and variances for each feature, which helps prevent overfitting and enhances accuracy. It often serves as a reliable baseline model, offering quick and effective results (Martins, 2025) .

The Multinomial Naive Bayes (MNB) classifier is a supervised machine learning algorithm commonly used for multiclass classification tasks, particularly in text classification scenarios. This model excels with high-dimensional, count-based features like word frequencies and naturally handles multiple classes. Its efficiency and effectiveness make it a popular choice for text classification tasks (Geekforgeeks, 2025). ​

The train-test split in my code used **80% of data for training** and **20% for testing**(test\_size=0.2). This division provides sufficient data for the model to learn effectively while retaining enough data to evaluate its performance accurately. ​

****

### 5.1 Model 1 – Support Vector Machine Model

### 5.1.1 Model Description

### The Support Vector Machine (SVM) is a supervised learning algorithm used for classification and regression tasks, known for its effectiveness in high-dimensional spaces (Cortes & Vapnik, 1995). It works by finding the optimal hyperplane that maximizes the margin between classes, improving generalization (Hearst et al., 1998). The model employs kernel functions (e.g., linear, polynomial) to handle non-linear decision boundaries by mapping data into higher-dimensional spaces (Boser et al., 1992). In this implementation, GridSearchCV optimized hyperparameters (C, kernel type, gamma) via cross-validation, selecting a polynomial kernel (degree=2) with C=10, achieving 65.81% accuracy. While SVMs are robust to overfitting (Vapnik, 1999), the moderate performance suggests potential feature engineering or alternative kernels (e.g., RBF) could enhance results

### .5.1.2 Evaluation Design

In this study, I divided our dataset into training and testing subsets, allocating 80% for training and 20% for testing. The training data was used to train and fine-tune the model using GridSearchCV, a method that exhaustively searches through all possible combinations of hyperparameters, such as the regularization parameter (C) and kernel type, to find the optimal settings. I employed three-fold cross-validation during this process to ensure the model's robustness by evaluating its performance on different subsets of the training data. To assess the model's effectiveness, we used accuracy as our performance metric, calculating the proportion of correct predictions on the test set.​ **A screen shot of a computer

Description automatically generated**

### 5.1.3 Results.

​ The model's selection of a polynomial kernel with a high regularization parameter (C=10) indicates the presence of non-linear patterns in the data. Achieving a test accuracy of approximately 66% suggests that while the model performs better than random guessing, there is substantial room for improvement.A black screen with white text

Description automatically generated

The provided code evaluates the performance of a classification model using two primary metrics: the classification report and the confusion matrix. The classification report, generated by classification report from scikit-learn, offers key performance indicators such as precision, recall, and F1-score for each class. Precision measures the accuracy of positive predictions, recall assesses the model's ability to identify all positive instances, and the F1-score provides a balance between precision and recall. These metrics are crucial for understanding the effectiveness of the model, especially in datasets with imbalanced classes. ​

Following this, the confusion matrix is computed using confusion\_matrix from scikit-learn, which compares the predicted labels (y\_pred) with the true labels (y\_test). This matrix displays the counts of true positives, true negatives, false positives, and false negatives, offering a comprehensive view of the model's performance across all classes (Bhandari, 2025).Visualizing the confusion matrix with a heatmap, as done in the code, aids in easily identifying areas where the model may be misclassifying certain classes.

In summary, the code utilizes scikit-learn's classification\_report and confusion\_matrix functions to provide a detailed evaluation of the classification model's performance, helping to identify strengths and areas for improvement.​

**A screenshot of a computer

Description automatically generated**

**A computer screen shot of a black screen with white text

Description automatically generated**

**A screenshot of a chatbot

Description automatically generated**

### 5.2 Model 2 – Gaussian Model

### 5.2.1 Describe your model.

### The Gaussian Naive Bayes (GaussianNB) classifier is a straightforward and efficient algorithm used for classification tasks involving continuous data. It operates under the assumption that each feature follows a normal (Gaussian) distribution within each class and that all features are independent of one another. By applying Bayes' theorem, the model calculates the probability of a data point belonging to a particular class based on the mean and variance of each feature for that class.

### 5.2.2 Explain your evaluation design.

The code evaluates two Naive Bayes classifiers,the GaussianNB and theMultinomialNB on the same dataset by splitting the data into training and testing sets using an 80/20 ratio, ensuring reproducibility with a fixed random state. Both models are trained on the training data, and their predictions are evaluated on the test set using accuracy as the metric. This approach provides a straightforward comparison of the classifiers' effectiveness under identical conditions.​

### 5.2.3 Present and explain results.

The Gaussian Naive Bayes (GaussianNB) classifier achieved an accuracy of approximately 57.35% on the dataset, indicating moderate performance. This outcome suggests that the model's assumptions—namely, that features are normally distributed and independent—may not hold true for this data, potentially affecting its effectiveness. Nonetheless, GaussianNB outperformed MultinomialNB in this scenario, implying that the dataset's continuous features are better suited to GaussianNB's design. Despite its simplicity and efficiency, GaussianNB's performance can decline if its core assumptions are violated, highlighting the importance of assessing data characteristics before model selection.​

A screenshot of a computer program

Description automatically generated

### 5.3 Model 3 – Multinomial Model

### 5.3.1 Describe your model.

The Multinomial model (often implemented as Multinomial Naive Bayes) is a probabilistic classifier based on Bayes’ theorem, designed for discrete/count-based data such as text classification (e.g., document categorization or spam detection) (Scikit-learn Developers, 2023). It assumes feature independence and uses frequency distributions to estimate class probabilities, making it efficient for high-dimensional datasets (McCallum & Nigam, 1998).

### 5.3.2 Explain your evaluation design.

The Multinomial Naive Bayes (MultinomialNB) classifier was evaluated on a dataset using an 80/20 train-test split, with a fixed random state to ensure consistent results. The model was trained on the training data and its performance was assessed on the test set by calculating accuracy, which measures the proportion of correct predictions. This process provides insight into the classifier's effectiveness under consistent conditions.​

### 5.3.3 Present and explain results.

# The Multinomial Naive Bayes (MultinomialNB) classifier achieved an accuracy of approximately 52.57% on the dataset, indicating that its performance was below expectations. This model is tailored for discrete or count-based data, such as word frequencies in text classification tasks. The lower accuracy suggests that the dataset may not align well with the model's assumptions—either the features are not discrete counts, or there is a weak relationship between features and labels.A screenshot of a computer program Description automatically generated

# 6 Evaluation Analysis

Reflecting on my model's performance, I observed that the final accuracy on the test set is displayed as a percentage, indicating how well the model generalizes to new data. The parameters selected by GridSearchCV, including C, kernel, gamma, and degree, were determined through cross-validation, ensuring the best combination for my dataset. A high accuracy suggests the model effectively captured patterns in the data, while a low accuracy might indicate issues such as insufficient data, poor feature selection, or overfitting/underfitting. Understanding these aspects helps me refine the model for better performance.​

In my approach, I began by converting categorical target values into numerical labels to make them suitable for the SVM.I split the data into training and testing sets, allocating 80% for training and 20% for testing, ensuring a balanced distribution for both training the model and evaluating its performance. To enhance accuracy, I standardized the features using scaling techniques, as SVMs are sensitive to the scale of input data. I adjusted the regularization parameter (C) to control the model's complexity; a lower C value simplifies the model, potentially leading to underfitting, while a higher C value allows for a more complex model, which could result in overfitting. Utilizing GridSearchCV, I performed cross-validation to identify the most effective hyperparameters, optimizing the model's performance.​

The code provided complements this approach by evaluating the model's performance through a classification report and a confusion matrix. The classification report offers metrics such as precision, recall, and F1-score for each class, providing a comprehensive understanding of the model's strengths and weaknesses. The confusion matrix visualizes the counts of true positives, true negatives, false positives, and false negatives, further aiding in the assessment of the model's performance across all classes. By analyzing these metrics, I can identify areas where the model may be misclassifying certain classes and make informed adjustments to improve its accuracy.​

In summary, combining thoughtful data preprocessing, meticulous parameter tuning, and thorough performance evaluation enables me to refine my SVM model, ensuring it aligns well with the specific characteristics of my dataset and generalizes effectively to new data.​

# 7. References

* Bird, S., Klein, E. and Loper, E., 2009. *Natural Language Processing with Python*. Available at: <https://www.nltk.org/book/> (Accessed 11/04/2025).
* Pandas, 2023. *Pandas Documentation*. Available at: <https://pandas.pydata.org/docs/>  Accessed 11/04/2025).
* NLTK, 2024. *Natural Language Toolkit*. Avaiable at: <https://www.nltk.org/> (Accessed 11 April 2025).
* Scikit Learn,2025.*TfidVectorizer*. Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html> (Accessed 11/04/2025).
* Boser, B.E., Guyon, I.M. and Vapnik, V.N., 1992. *A training algorithm for optimal margin classifiers. In Proceedings of the 5th Annual Workshop on Computational Learning Theory*, pp. 144–152. <https://doi.org/10.1145/130385.130401> (Accessed 11/04/2025).
* Scikit Learn,2025.*GaussianNB*. Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html> (Accessed 11/04/2025).
* Mitchell, M.T., 2020. *Generative and Discriminative Classidiers: Naiv Bayes and Logistic Regression.* Availble at: <https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf> (Accessed 11/04/2025).
* Scikit Learn, 2025.*Naive Bayes.* Available at: <https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html> (Accessed 11/04/2025).
* Geeksforgeeks, 2025.*Gaussian Naive Bayes.* Available at: <https://www.geeksforgeeks.org/gaussian-naive-bayes/?utm_source=chatgpt.com>l (Accessed 11/04/2025).
* Martins, C., 2025. *Gaussian Naive Bayes Explained With Scikit-Learn.* Available at:<https://builtin.com/artificial-intelligence/gaussian-naive-bayes?utm_source=chatgpt.com> (Accessed 11/04/2025).
* Geeksforgeeks, 2025. *Multinomial Naive Bayes.* Available at: <https://www.geeksforgeeks.org/multinomial-naive-bayes/?utm>(Accessed 11/04/2025).
* LabEx, 2025. *Optimizing Model Hyperparameters with GridSearchCV*

Available at: <https://labex.io/tutorials/ml-optimizing-model-hyperparameters-with-gridsearchcv-49219?utm_source=chatgpt.com> (Accessed 11/04/2025).