Document Classification System Report

1. Introduction

Document classification is a Natural Language Processing (NLP) task that automatically assigns predefined categories to textual data. It is widely used in applications such as spam detection, topic categorization, sentiment analysis, and automated document organization.

In this project, a machine learning–based document classification system was developed using classical machine learning algorithms. The objective was to build a model that can accurately classify text documents into their appropriate categories while running locally without relying on large language models or cloud APIs.

The system converts raw text into numerical features and uses supervised learning algorithms to learn patterns from labeled data. Multiple models were implemented and compared to determine the most suitable algorithm for this task.

2. Dataset

The dataset used in this project is the **20 Newsgroups dataset**, which was imported directly using the sklearn.datasets library.

This dataset is a well-known benchmark dataset commonly used for text classification research and experimentation. It contains approximately **20,000 text documents** categorized into **20 different topics**.

Each sample in the dataset consists of:

- a text document (email or post)
- a corresponding category label

The categories include topics such as:

- sports
- politics
- religion
- computers
- science
- automobiles

This dataset is suitable because:

- it is realistic text data
- it has multiple classes
- it is balanced enough for training
- it is widely used in academic research

Using a standard benchmark dataset ensures reproducibility and makes model performance comparable with existing methods.

3. Model Selection

Three classical machine learning models were selected and implemented:

1. Support Vector Machine (SVM)
2. Logistic Regression
3. Naive Bayes

These models were chosen because they are widely recognized as strong baseline algorithms for text classification tasks.

They were selected based on the following criteria:

- ability to handle high-dimensional data
- efficiency on sparse text features
- computational efficiency
- interpretability
- suitability for local execution

Deep learning models were intentionally avoided because the task requirements specified the use of classical machine learning methods.

4. Model Training and Optimization

Before training, text data must be converted into numerical form. This was done using **TF-IDF vectorization**.

TF-IDF (Term Frequency–Inverse Document Frequency) assigns importance scores to words based on:

- how frequently they appear in a document

- how rare they are across all documents

The vectorization configuration included:

- stop word removal
- maximum feature limit
- n-grams (single words + word pairs)
- sublinear term frequency scaling

These settings improve feature quality and help models learn more meaningful patterns.

Class imbalance was handled using techniques such as:

- class weighting (for SVM and Logistic Regression)
- prior probability adjustment (for Naive Bayes)

Hyperparameters were tuned manually to improve performance and stability. For example:

- SVM regularization parameter (C)
- Logistic Regression iterations
- Naive Bayes smoothing parameter

5. Results and Evaluation

Each model was trained using 80% of the dataset and tested on the remaining 20%.

Performance was evaluated using standard classification metrics:

- Accuracy
- Precision
- Recall
- F1-score

After training and testing, the models produced the following general results:

Model	**Performance**
SVM	Highest accuracy
Logistic Regression	Slightly lower
Naive Bayes	Moderate

The SVM model consistently outperformed the others and produced the most reliable predictions.

6. Metrics Used

Accuracy

Measures overall correctness of the model.

$$\textbf{Accuracy} = \frac{\boldsymbol{TP + TN}}{\boldsymbol{TP + TN + FP + FN}}$$

Where:

- TP = True Positives
- TN = True Negatives
- FP = False Positives
- FN = False Negatives

Precision

Measures how many predicted positives are actually correct.

$$\textbf{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FP}}$$

Interpretation:

Of all predicted positives, how many were correct?

Recall (Sensitivity)

Measures how many actual positives were correctly predicted.

$$\textbf{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP + FN}}$$

Interpretation:

Of all real positives, how many did the model detect?

F1 Score

Balances precision and recall.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Evaluation metric of SVM

```
Training SVM...

Accuracy: 0.9450928381962864
                          precision    recall  f1-score   support

             alt.atheism       0.96      0.96      0.96       160
           comp.graphics       0.87      0.93      0.90       195
 comp.os.ms-windows.misc       0.94      0.89      0.92       197
comp.sys.ibm.pc.hardware       0.84      0.87      0.86       196
   comp.sys.mac.hardware       0.91      0.92      0.92       193
          comp.windows.x       0.93      0.94      0.93       198
            misc.forsale       0.89      0.91      0.90       195
               rec.autos       0.95      0.95      0.95       198
         rec.motorcycles       0.98      0.96      0.97       199
      rec.sport.baseball       0.97      0.98      0.98       199
        rec.sport.hockey       0.99      0.98      0.99       200
               sci.crypt       0.98      0.97      0.98       198
         sci.electronics       0.91      0.93      0.92       197
                 sci.med       0.96      0.97      0.96       198
               sci.space       0.98      0.97      0.98       197
  soc.religion.christian       0.96      0.97      0.97       199
      talk.politics.guns       0.98      0.95      0.96       182
   talk.politics.mideast       1.00      0.97      0.99       188
      talk.politics.misc       0.97      0.96      0.96       155
      talk.religion.misc       0.95      0.89      0.92       126

                accuracy                           0.95      3770
               macro avg       0.95      0.94      0.95      3770
            weighted avg       0.95      0.95      0.95      3770
```

7.Confidence Score (Add This Section to Report)

Confidence Score

In addition to predicting the document category, the system also outputs a **confidence score** for each prediction. The confidence score represents how certain the model is about its prediction.

It is computed as:

Confidence Score = Highest Predicted Probability

This value ranges between **0 and 1**, where:

Score Range	**Meaning**
0.90 – 1.00	Very confident prediction
0.70 – 0.89	Reliable prediction
0.50 – 0.69	Moderate certainty
< 0.50	Low confidence

Why Confidence Score Is Important

Providing confidence scores improves model usability because:

- It indicates prediction reliability
- Helps detect uncertain predictions
- Allows decision systems to reject low-confidence outputs
- Improves transparency and interpretability

How Confidence Is Computed in This System

The Support Vector Machine model normally produces decision scores rather than probabilities. To obtain reliable probability estimates, a calibration technique was applied using:

CalibratedClassifierCV

This converts raw SVM scores into probability values using statistical scaling. The system then selects the highest probability as the confidence score.

7. Performance Analysis

The performance differences between models can be explained by how each algorithm works.

Naive Bayes

- Fast and simple
- Assumes word independence
- Works well but limited accuracy

Logistic Regression

- Learns weight importance for words
- More flexible than Naive Bayes
- Performs well on text data

Support Vector Machine

- Finds optimal boundary between categories
- Maximizes margin between classes
- Handles high-dimensional sparse data effectively

Text data is naturally:

- sparse
- high dimensional
- linearly separable

SVM is specifically designed for such data structures, which explains its superior performance.

8. Final Model Selection

After comparing all models, **Support Vector Machine (SVM)** was selected as the final model because:

- highest accuracy
- best generalization
- strong performance on sparse text features
- robust decision boundary

9. Conclusion

In this project, a document classification system was successfully developed using classical machine learning algorithms. The system converts raw text into numerical features using TF-IDF and applies supervised learning models to classify documents into predefined categories.

Multiple models were evaluated to ensure a fair comparison, and Support Vector Machine was selected as the best-performing model. The results demonstrate that classical machine learning techniques remain highly effective for text classification tasks, especially when combined with proper preprocessing and feature engineering.

The project shows that accurate document classification can be achieved efficiently using lightweight models that run locally without requiring heavy computational resources.

Deployment of model in streamlit : click the link to view

Document Classifier · Streamlit