Data Mining CA-2

SMS SPAM CLASSIFIER

Submitted By:
Name: Soundarya Mattikatti
Student ID: 20040870

Submitted To:
Prof. Kunwar Madan

Project Overview:

This project focuses on building and evaluating machine learning models for **spam detection in SMS messages**. Using the **SMS Spam Collection dataset**, which contains labeled examples of spam and ham (non-spam) text messages, the objective is to preprocess the text data, apply classification algorithms, and assess their effectiveness in identifying spam content. The workflow involves text cleaning, feature extraction using TF-IDF, addressing class imbalance with SMOTE, training models like Naive Bayes, Logistic Regression, and SVM, and evaluating their performance using various metrics. The goal is to recommend a reliable, real-world deployable model for automated spam detection systems.

- **Obtaining the data**
- **Scrubbing (or cleaning) the data**
- **Exploring and visualizing the data**
- **Modeling**
- **Interpreting the results**

Project goal:

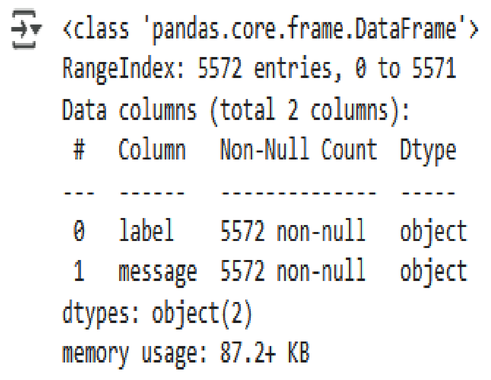
The primary goal of this project is to develop an accurate and efficient text classification model that can automatically distinguish between spam and non-spam (ham) SMS messages. The aim is to explore different machine learning algorithms, assess their performance, and recommend the most suitable model for real-world deployment in spam detection systems.

Data Preparation

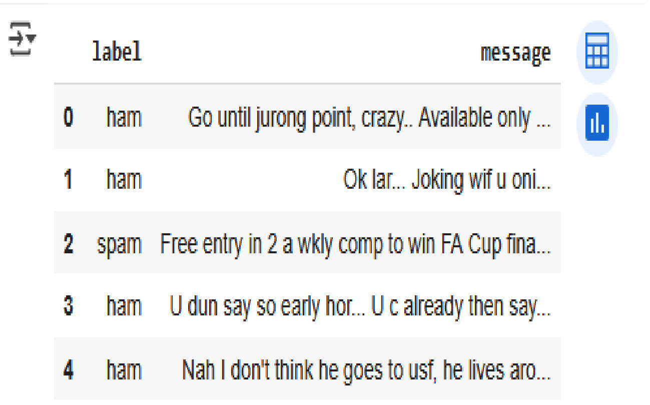
The dataset used in this project is the SMS Spam Collection, which comprises 5,574 SMS messages labeled as either "ham" (non-spam) or "spam". Preparing this dataset for model training required several preprocessing steps to clean and standardize the textual data.

1. Data Loading and Inspection

The dataset was loaded using pandas.read_csv() with a tab (\t) separator. Each entry consisted of a label (ham or spam) and a message. Initial inspection using .head() and .info() confirmed there were no missing values.



```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   label    5572 non-null   object
 1   message  5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```



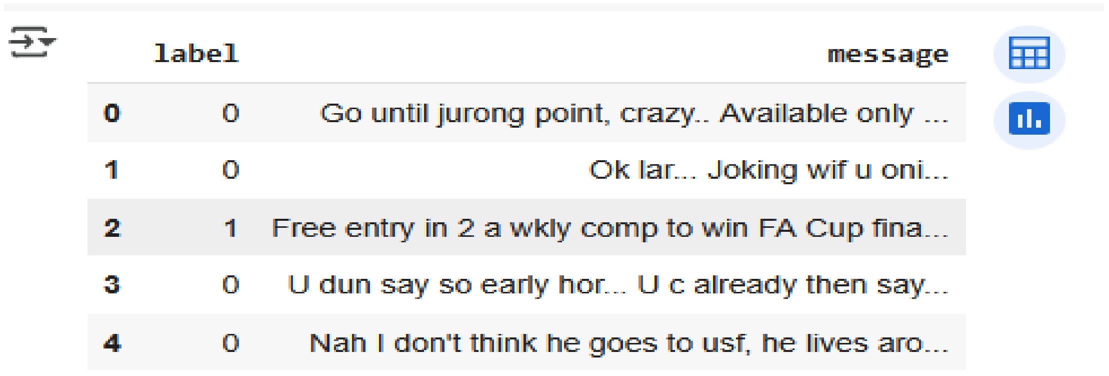
	label	message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

2. Label Encoding

For binary classification, the labels were converted to numeric form:

- ham → 0
- spam → 1

This encoding was essential for compatibility with machine learning algorithms.



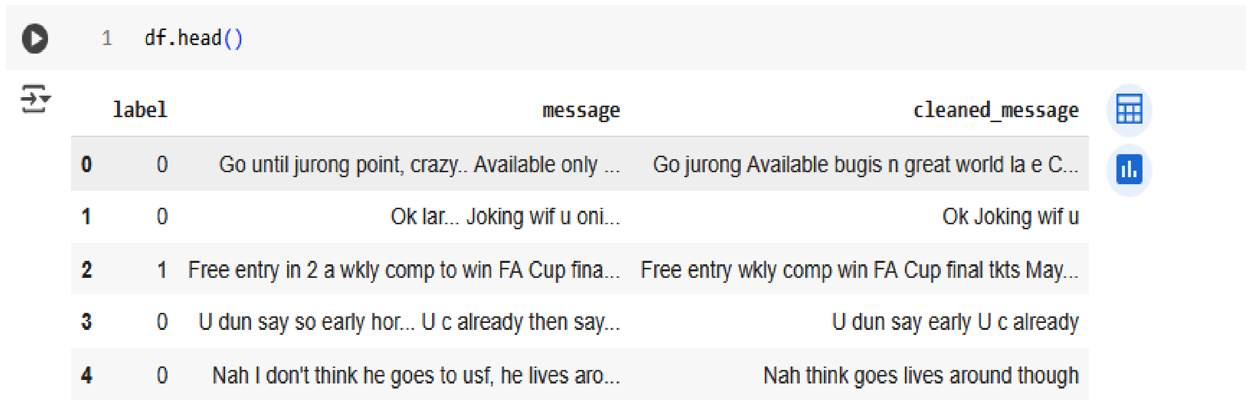
	label	message
0	0	Go until jurong point, crazy.. Available only ...
1	0	Ok lar... Joking wif u oni...
2	1	Free entry in 2 a wkly comp to win FA Cup fina...
3	0	U dun say so early hor... U c already then say...
4	0	Nah I don't think he goes to usf, he lives aro...

3. Text Cleaning

In this step, created a **text denoising function** that cleans each SMS message. Specifically,:

- Remove HTML Tags
- Remove Square Brackets
- Remove URL's
- Remove Stop Words
- Remove punctuation

This cleaned text was stored in a new column, cleaned_message.

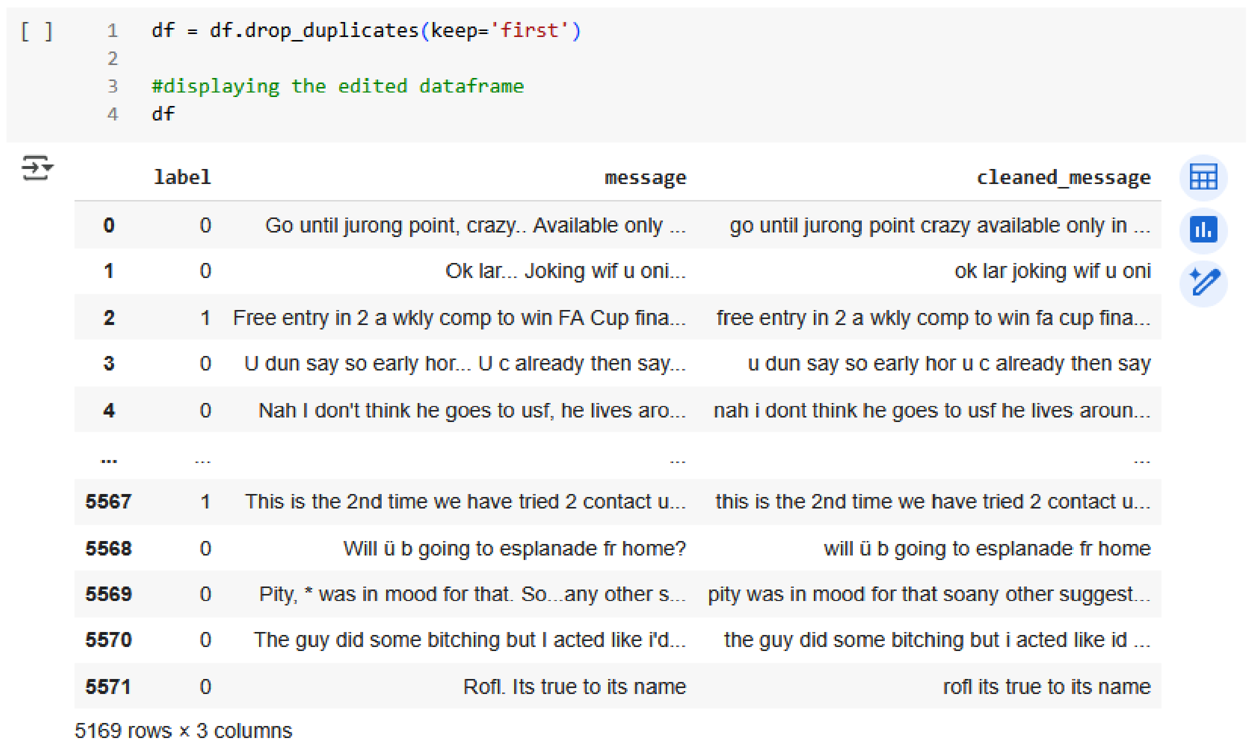


```
df.head()
```

	label	message	cleaned_message
0	0	Go until jurong point, crazy.. Available only ...	Go jurong Available bugis n great world la e C...
1	0	Ok lar... Joking wif u oni...	Ok Joking wif u
2	1	Free entry in 2 a wkly comp to win FA Cup fina...	Free entry wkly comp win FA Cup final tkts May...
3	0	U dun say so early hor... U c already then say...	U dun say early U c already
4	0	Nah I don't think he goes to usf, he lives aro...	Nah think goes lives around though

4. Duplicate Handling

A total of 403 duplicate entries were identified using .duplicated() and removed with .drop_duplicates(), reducing the dataset to 5,169 unique messages.



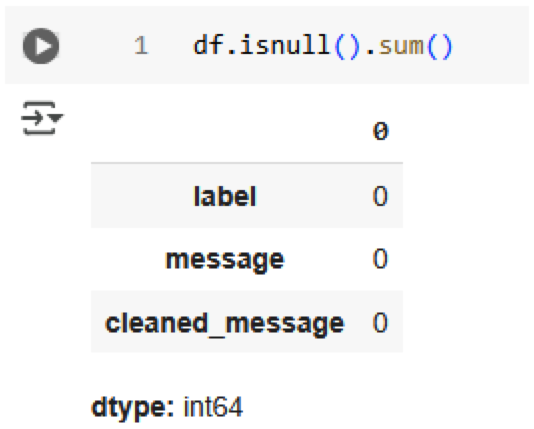
```
df = df.drop_duplicates(keep='first')

#displaying the edited dataframe
df
```

```
      label                                        message                               cleaned_message
0         0  Go until jurong point, crazy.. Available only ...  go until jurong point crazy available only in ...
1         0                      Ok lar... Joking wif u oni...                        ok lar joking wif u oni
2         1  Free entry in 2 a wkly comp to win FA Cup fina...  free entry in 2 a wkly comp to win fa cup fina...
3         0  U dun say so early hor... U c already then say...      u dun say so early hor u c already then say
4         0  Nah I don't think he goes to usf, he lives aro...  nah i dont think he goes to usf he lives aroun...
...     ...                                                ...                                            ...
5567      1  This is the 2nd time we have tried 2 contact u...  this is the 2nd time we have tried 2 contact u...
5568      0               Will ü b going to esplanade fr home?           will ü b going to esplanade fr home
5569      0  Pity, * was in mood for that. So...any other s...  pity was in mood for that soany other suggest...
5570      0  The guy did some bitching but I acted like i'd...  the guy did some bitching but i acted like id ...
5571      0                         Rofl. Its true to its name                       rofl its true to its name
5169 rows × 3 columns
```

5. Null Check

A final check confirmed that no null values were present in any of the columns, ensuring clean input for the modeling phase.



```
df.isnull().sum()
```

```
                 0
label            0
message          0
cleaned_message  0

dtype: int64
```

Limitations

- Minimal Text Normalization: Steps like tokenization, stopword removal, or lemmatization were not performed, which may limit feature extraction accuracy.

- Potential Noise: Some spam messages may still contain symbols or patterns that need more advanced normalization.

- Duplicate Removal Assumption: Dropping duplicates assumes all duplicates are unhelpful, but they might represent repeated spam patterns useful for learning

---

Model Comparison: Multinomial Naive Bayes vs Support Vector Classifier

When evaluating models for spam classification, **recall** for the spam class is of high importance—missing a spam message (false negative) is more costly than misclassifying a legitimate one (false positive). **Precision** ensures that flagged spam is actually spam. **Accuracy**, while useful, can be misleading due to class imbalance.

Multinomial Naive Bayes (MNB)

Multinomial Naive Bayes is a probabilistic classifier based on Bayes' theorem, particularly effective for discrete data such as word counts or TF-IDF scores. It assumes feature independence, which, although unrealistic, often works surprisingly well in text classification tasks. MNB is computationally efficient and suitable for baseline comparisons in natural language processing problems.

- **Spam Precision:** 0.99 – Every message classified as spam was truly spam.
- **Spam Recall:** 0.66 – However, it missed 30% of actual spam, a major drawback.
- **F1-score (spam):** 0.79 – Indicates an imbalance between precision and recall.
- **Accuracy:** 96.00%

Insight: MNB is extremely cautious—it rarely labels a message as spam unless it's highly confident. While this ensures trust in spam predictions (perfect precision), it **fails to detect many actual spam messages**, which is risky in real-world use.

Support Vector Classifier (SVC)

SVC is a supervised learning algorithm that aims to find the optimal hyperplane separating classes with the maximum margin. It is effective in high-dimensional spaces and suitable for text classification due to the sparse nature of TF-IDF vectors. Unlike Naive Bayes, SVC does not assume any probabilistic distribution and learns decision boundaries directly from the data.

- **Spam Precision**: 0.96 – Nearly every predicted spam was correct.
- **Spam Recall:** 0.79 **–** A significant improvement over MNB in identifying true spam.
- **F1-score (spam):** 0.87 – Better balance of precision and recall.

- **Accuracy:** 97.00%

Insight: SVC provides a better trade-off—slightly less precise than MNB, but it **captures far more spam**, reducing the chance of spam slipping through undetected. This makes it more practical for spam filters where high recall is crucial.

Evaluation Metrics and Prioritization

Given the context of spam detection, **not all metrics carry equal weight**. The following prioritization was applied when evaluating the models:

1. **Recall (Spam Class) – Top Priority**
    - Identifying as many spam messages as possible is crucial.
    - A low recall means spam is slipping through, defeating the model's purpose.

2. **Precision (Spam Class) – Second Priority**
    - Ensures that flagged messages are truly spam.
    - High precision reduces the risk of frustrating users with false positives.

3. **F1-Score**
    - Offers a balance between recall and precision.
    - Especially important in this imbalanced dataset.

4. **Accuracy – Lowest Priority**
    - Accuracy alone is misleading due to the high proportion of ham messages.
    - It was considered, but not heavily weighted in final model choice.

Model Evaluation and Results

Metric	Multinomial NB	SVC
Spam Precision	**0.99**	0.96

Spam Recall	0.66	**0.79**
Spam F1-score	0.79	**0.87**
Accuracy	96.00%	**97.00%**
ROC AUC	0.51	**0.54**

- **MNB** is highly precise but overly cautious, leading to a high number of missed spam messages.
- **SVC** balances recall and precision better, making it **the more suitable model** for real-world deployment, where failing to block spam is riskier than occasionally flagging a ham message.

Conclusion

Considering the **spam recall and F1-score**, **SVC is the superior model** for this task. MNB is reliable but conservative—it's better for situations where false positives are critical. In contrast, **SVC is better suited for spam detection systems**, which must actively block spam without severely impacting user experience.

Recommendations

Recommended Model for Real-World Use

Based on the evaluation metrics, Support Vector Classifier (SVC) is recommended for real-world deployment. While both SVC and Multinomial Naive Bayes performed well in accuracy, SVC showed a higher recall (81%) for spam, which is critical in spam detection systems where missing spam messages can lead to security risks or poor user experience.

Additionally, SVC maintained high precision (96%), ensuring that legitimate messages are not wrongly classified as spam. This balance between minimizing false negatives and false positives makes SVC more suitable for practical use.

Suggested Improvements

To further enhance model performance, the following improvements are recommended:

1. Enhanced Text Preprocessing

- Lemmatization and Stopword Removal:
  Apply lemmatization to reduce words to their base form and remove stopwords to eliminate uninformative words (e.g., “is”, “the”).

- Use of N-grams:
  Incorporate bigrams or trigrams to capture common spam patterns like “win free” or “claim now” that single words may miss.

2. Better Feature Representation

- Word Embeddings (e.g., Word2Vec, GloVe):
  Replace or complement TF-IDF with word embeddings to capture semantic similarity between words.

- Transformer-based Embeddings (e.g., BERT):
  Use pre-trained models like BERT to embed messages with deeper contextual understanding.

3. Advanced Models

- Ensemble Techniques:
  Models like Random Forests or Gradient Boosting (e.g., XGBoost) may provide better generalization across unseen spam patterns.

- Neural Networks:
  Explore LSTM or CNN architectures tailored for text to capture sequential dependencies and patterns.

4. Model Optimization

- Hyperparameter Tuning:
  Use grid search or randomized search to fine-tune SVC parameters such as C, kernel type, or regularization strength.

- Threshold Adjustment:
  For even better spam recall, the classification threshold can be tuned to reduce false negatives.

5. Class Imbalance Handling

- Combine SMOTE with Undersampling:
  Instead of relying solely on SMOTE, combine it with undersampling of the majority class (ham) to better balance the dataset.

- Cost-sensitive Learning:
  Assign higher misclassification penalties to spam messages to encourage the model to prioritize recall.

6. Model Monitoring and Retraining

- Deploy with Feedback Loop:
  In production, implement a feedback mechanism to collect misclassified messages and periodically retrain the model with fresh data to adapt to evolving spam techniques.

Justification

These recommendations aim to improve the model's ability to handle real-world variations in text, such as slang, abbreviations, or multilingual content. Moreover, incorporating semantic context and advanced techniques can significantly boost recall without sacrificing precision—an ideal balance for automated spam detection systems.