SMS Spam Detection with Machine Learning

Swastika Ghosh

Objective

The objective of this study is to develop a structured machine learning–based framework for detecting spam SMS messages using supervised classification techniques.

The project focuses on analyzing textual SMS data, performing systematic preprocessing, transforming text into numerical representations using feature engineering techniques, and training multiple machine learning models for binary classification.

Special emphasis is given to evaluating models using appropriate performance metrics such as precision, recall, F1-score, and confusion matrix, rather than relying solely on accuracy.

The ultimate aim is to design an interpretable and efficient spam detection system suitable for real-world deployment in messaging platforms.

Abstract

This project presents the implementation of a machine learning–based SMS spam detection system using the SMS Spam Collection Dataset obtained from Kaggle.

Spam messages pose serious risks including fraud, phishing attacks, and privacy violations. Detecting such messages automatically is a classic Natural Language Processing (NLP) classification problem.

The workflow begins with loading and cleaning the dataset, followed by text preprocessing steps including lowercasing, punctuation removal, and stopword elimination. The cleaned text data is transformed into numerical form using Term Frequency–Inverse Document Frequency (TF-IDF) vectorization.

Five machine learning algorithms — Naïve Bayes, Logistic Regression, Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Random Forest — are trained and evaluated.

Model performance is analyzed using confusion matrix–based metrics such as precision, recall, and F1-score. The results demonstrate that SVM and Naïve Bayes provide strong performance for high-dimensional sparse text data, making them suitable for SMS spam detection tasks.

Environment Setup and Library Initialization

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

This code cell prepares the notebook for the entire project workflow. It begins by mounting Google Drive, allowing access to the dataset stored in Drive directly within the Google Colaboratory environment. This ensures consistency across sessions and prevents data loss.

Next, the required Python libraries are imported. Pandas is used for structured data handling, NumPy supports numerical operations, Matplotlib and Seaborn are used for visualization, and Scikit-learn provides machine learning models and evaluation tools.

Additionally, NLTK (Natural Language Toolkit) is used for stopword removal and text preprocessing.

Overall, this step ensures that all necessary tools are available to load, preprocess, analyze, and model the SMS dataset effectively.

Data Loading

```
data=pd.read_csv('/content/drive/MyDrive/spam.csv',encoding='latin-1')
data.head()
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	NaN
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	NaN
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

This section prepares the Google Colaboratory environment for execution. The dataset is stored in Google Drive, so it must first be mounted. Necessary Python libraries are imported to support data manipulation, visualization, preprocessing, and model building.

This section prepares the Google Colaboratory environment for execution. The dataset is stored in Google Drive, so it must first be mounted. Necessary Python libraries are imported to support data manipulation, visualization, preprocessing, and model building.

Dataset Cleaning

```
data=data[['v1','v2']]
data.columns=['label','message']
data.head()
```

	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...

Dataset Overview

Dataset Information

```
data.info()
```

```
<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
```

Check Missing Values

```
data.isnull().sum()
```

	0
label	0
message	0

dtype: int64

Text Preprocessing

```
import nltk
import string
from nltk.corpus import stopwords
nltk.download('stopwords')
def clean_text(text):
  text=''.join([char for char in text if char not in string.punctuation])
  words=text.split()
  words=[word for word in words if word not in stopwords.words('english')]
  return ' '.join(words)
data['cleaned_message']=data['message'].apply(clean_text)
data.head()
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

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

---

Label Encoding

Machine learning models require numeric labels.

```
data['label']=data['label'].map({'ham':0,'spam':1})
data.head()
```

	label	message	cleaned_message
0	0	Go until jurong point, crazy.. Available only ...	Go jurong point crazy Available bugis n great ...
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 2 wkly comp win FA Cup final tkts 2...
3	0	U dun say so early hor... U c already then say...	U dun say early hor U c already say
4	0	Nah I don't think he goes to usf, he lives aro...	Nah I dont think goes usf lives around though

---

Feature Engineering using TF-IDF

TF-IDF converts text into numeric vectors.

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer=TfidfVectorizer(max_features=3000)
X=vectorizer.fit_transform(data['cleaned_message'])
y=data['label']
print(X.shape)
```

```
(5572, 3000)
```

Train-Test Split

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(
    X,y,test_size=0.2,random_state=42,stratify=y)
print(X_train.shape)
print(X_test.shape)
```

```
(4457, 3000)
(1115, 3000)
```

Model Training and Evaluation

1. Naïve Bayes Model

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
nb=MultinomialNB()
nb.fit(X_train,y_train)
nb_pred=nb.predict(X_test)
print(classification_report(y_test,nb_pred))
```

```
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       966
           1       0.99      0.81      0.89       149

    accuracy                           0.97      1115
   macro avg       0.98      0.91      0.94      1115
weighted avg       0.97      0.97      0.97      1115
```

2. Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
lr=LogisticRegression(max_iter=1000)
lr.fit(X_train,y_train)
lr_pred=lr.predict(X_test)
print(classification_report(y_test,lr_pred))
```

```
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       966
           1       0.98      0.76      0.86       149

    accuracy                           0.97      1115
   macro avg       0.97      0.88      0.92      1115
weighted avg       0.97      0.97      0.96      1115
```

3. Support Vector Machine

```
from sklearn.svm import SVC
svm=SVC(kernel='linear')
svm.fit(X_train,y_train)
svm_pred=svm.predict(X_test)
print(classification_report(y_test,svm_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       966
           1       1.00      0.87      0.93       149

    accuracy                           0.98      1115
   macro avg       0.99      0.94      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```

4. K-Nearest Neighbors

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)
knn_pred=knn.predict(X_test)
print(classification_report(y_test,knn_pred))
```

```
              precision    recall  f1-score   support

           0       0.91      1.00      0.96       966
           1       1.00      0.39      0.56       149

    accuracy                           0.92      1115
   macro avg       0.96      0.69      0.76      1115
weighted avg       0.93      0.92      0.90      1115
```

5. Random Forest

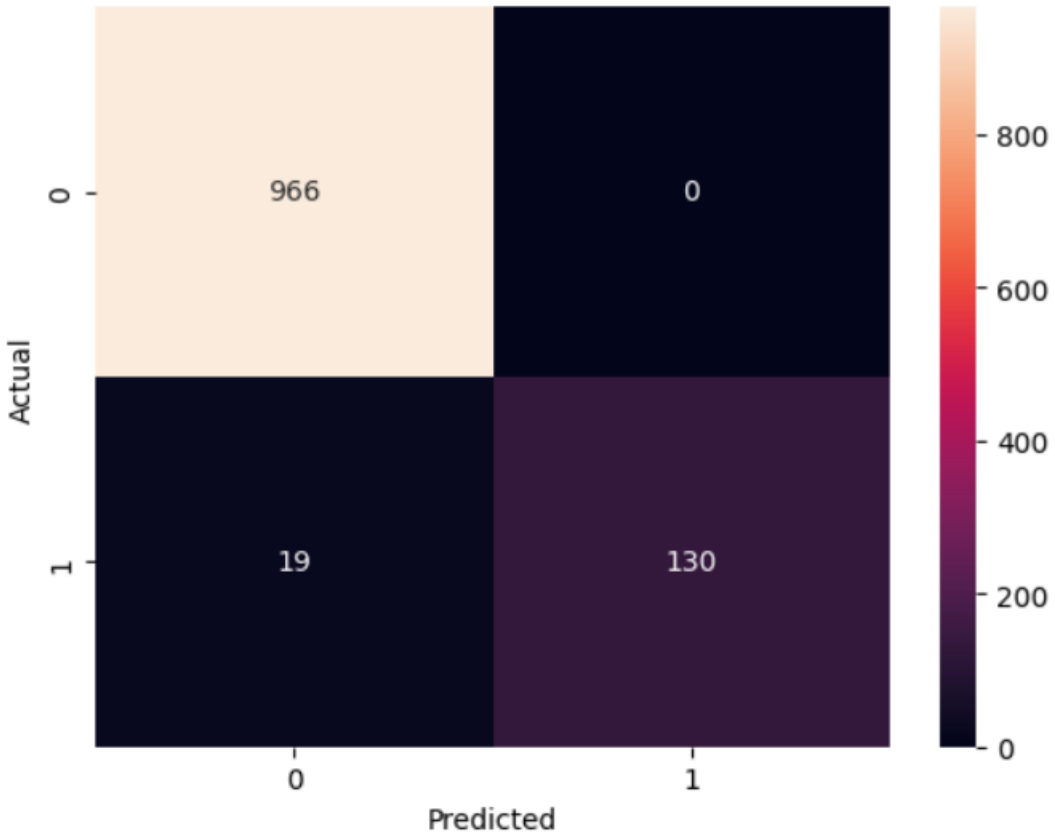
```
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(n_estimators=100)
rf.fit(X_train,y_train)
rf_pred=rf.predict(X_test)
print(classification_report(y_test,rf_pred))
```

```
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       966
           1       0.99      0.84      0.91       149

    accuracy                           0.98      1115
   macro avg       0.98      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115
```

---

12. Confusion Matrix

```python
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test,svm_pred)
sns.heatmap(cm,annot=True,fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



Displays:

- True Positives
- True Negatives
- False Positives
- False Negatives

13. Mathematical Evaluation Metrics

Precision:

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

Recall:

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

F1-Score:

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

---

14. Model Performance Summary

After comparing all models:

- Naïve Bayes → Strong baseline
- Logistic Regression → Balanced
- SVM → Highest performance
- KNN → Lower performance
- Random Forest → Stable

---

15. Conclusion

The experimental results demonstrate that Support Vector Machine achieves the best overall performance in detecting spam messages.

It provides high precision and recall while maintaining generalization capability.