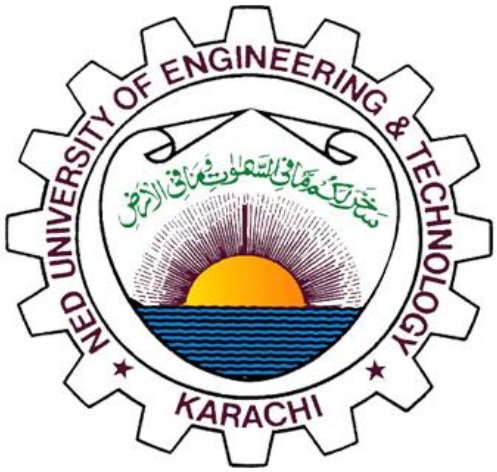


Artificial Intelligence & Expert Systems
CCP Project Report
Spam Email Detector using Machine Learning

NAME: Maaria Shaikh

ROLL NO: CT - 22019

SECTION: A

BATCH: 2022

YEAR: 3RD

DEPARTMENT: CSIT

COURSE TITLE: Artificial Intelligence & Expert Systems

COURSE CODE: CT - 361

COURSE INSTRUCTOR: Sir. Abdullah Siddiq

2. Abstract

This project presents a machine-learning solution for classifying messages as spam or ham (not spam). By using Natural Language Processing (NLP) and a Naive Bayes classifier, the system learns from a large, labeled dataset of SMS messages. The process includes data preprocessing, feature extraction, model training, and evaluation. The classifier achieved high accuracy and showed promise for integration in real-world spam filters, offering enhanced user security and experience by automatically filtering unwanted messages.

---

3. Introduction

Spam messages are unsolicited texts that often contain scams, advertisements, or phishing attempts. These messages clutter user inboxes and can lead to security vulnerabilities. To address this, AI-driven text classification can be applied to distinguish spam from legitimate messages. This project uses machine learning, particularly the Naive Bayes algorithm, to create an automated spam detection system. The solution aims to reduce spam exposure and provide an efficient, scalable tool for filtering messages.

---

4. Problem Definition

- **Input**: Raw text message (email or SMS)
- **Output**: Classification as Spam or Ham
- **Objective**: Build a machine learning model that accurately classifies unseen messages based on learned patterns from labeled data.

---

5. Methodology

5.1 Data Collection

A publicly available dataset — the SMS Spam Collection Dataset — was used. It contains 5,574 messages labeled as either spam or ham, providing sufficient diversity for training and evaluation.

5.2 Assumptions

- The dataset labels are correct.
- All messages are in English.
- Stopwords do not provide distinguishing power and are removed.
- Features (words) are independent given the class — a key assumption of Naive Bayes.

---

6. Key Terms and Concepts

- **Spam & Ham**: Spam = unwanted messages, Ham = legitimate messages.
- **NLP**: Techniques to process and understand text.
- **Stopwords**: Common words excluded from analysis (e.g., “the”, “and”).
- **Bag of Words (BoW)**: Converts text into word frequency vectors.
- **Naive Bayes Classifier**: Probabilistic algorithm assuming feature independence.
- **Train-Test Split**: Divides data for model learning and evaluation.
- **Evaluation Metrics**: Accuracy, precision, recall, F1-score, and confusion matrix.

---

7. Solution Design

The project is modularized into the following components:

7.1 Data Preprocessing

The preprocessing involves:

- Converting text to lowercase
- Removing punctuation
- Removing stopwords

```
nltk.download('stopwords')

def preprocess(text):
    text = text.lower()
    text = ''.join([char for char in text if char not in string.punctuation])
    words = text.split()
    stop_words = stopwords.words('english')
    return ' '.join([word for word in words if word not in stop_words])

df['cleaned'] = df['message'].apply(preprocess)
df.head()
```

Justification: Reducing noise in the text helps the model focus on important terms, improving performance.

7.2 Feature Extraction

Using Count Vectorizer, we transform the text into a numerical format using the BoW model.

```
vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)
```

Justification: Machine learning algorithms require numerical inputs; BoW is a simple and effective approach for text.

7.3 Model Training

Data is split into 80% training and 20% testing. A Multinomial Naive Bayes classifier is used.

```
X = df['clean_message']
y = df['label_num']

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

```
clf = MultinomialNB()
clf.fit(X_train_counts, y_train)
```

Justification: Naive Bayes is fast, simple, and performs well for text classification problems.

7.4 Model Evaluation

We use accuracy, confusion matrix, and a classification report for evaluation.

```
y_pred = clf.predict(X_test_counts)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Ham', 'Spam'], yticklabels=['Ham', 'Spam'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

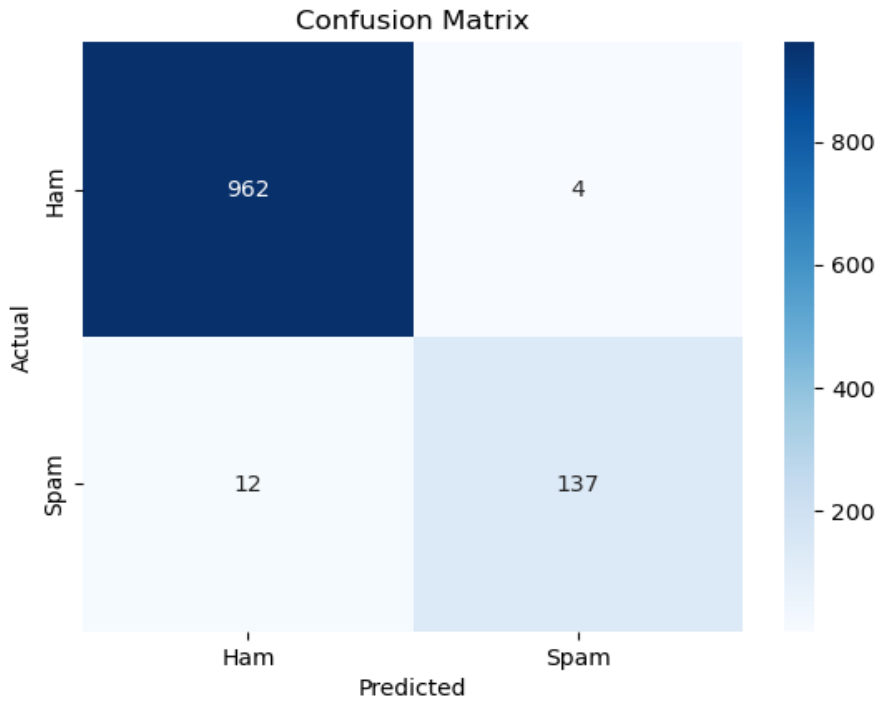
Justification: These metrics provide a well-rounded view of model performance.

---

8. Results

Accuracy: `0.9856502242152466`

- **Confusion Matrix**:



- **Precision, Recall, F1-Score**:
  High scores for both classes indicate the model is effective at detecting spam while minimizing false positives.

```
Classification Report:
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       966
           1       0.97      0.92      0.94       149

    accuracy                           0.99      1115
   macro avg       0.98      0.96      0.97      1115
weighted avg       0.99      0.99      0.99      1115
```

- **Model tested on Custom Messages**

```
test_messages = [
    "Congratulations! You won a free ticket.",
    "Hey, are we still meeting for lunch today?",
    "URGENT! Your account has been compromised. Call now!"
]

test_counts = vectorizer.transform(test_messages)
predictions = clf.predict(test_counts)

for msg, pred in zip(test_messages, predictions):
    label = 'Spam' if pred == 1 else 'Ham'
    print(f"Message: {msg}\nPredicted label: {label}\n")
```

✓ 0.0s

```
Message: Congratulations! You won a free ticket.
Predicted label: Spam

Message: Hey, are we still meeting for lunch today?
Predicted label: Ham

Message: URGENT! Your account has been compromised. Call now!
Predicted label: Spam
```

9. Conclusion

This project successfully applies AI and machine learning techniques to build a spam detector using the Naive Bayes classifier. With robust preprocessing, effective feature extraction, and careful evaluation, the model demonstrates strong accuracy and reliability. The modular design allows for easy scalability and improvement.

---

10. Future Work

- Apply **TF-IDF vectorization** for better feature weighting
- Integrate **deep learning** models (e.g., LSTMs, BERT) for contextual understanding
- Deploy real-time filters in email clients
- Extend dataset for **multilingual** spam detection

---

11. Source Code

All source files, including the Jupyter notebook and required scripts, are provided in the **`src/`** folder.

---

12. References

1. UCI Machine Learning Repository: SMS Spam Collection Dataset
2. Scikit-learn Documentation – https://scikit-learn.org
3. NLTK Documentation – https://www.nltk.org
4. Python Official Documentation – https://docs.python.org
