



Topic : 🎓 Email/Message Spam Detection 🎓

Problem Definition

The goal is to classify messages as **spam** or **not spam (ham)**.

This problem is important because spam emails/messages waste time, can be malicious, and proper detection improves productivity and online safety.

Dataset Handling

Dataset Used

- **SMS Spam Collection Dataset from Kaggle**
- **Link: [SMS Spam Collection Dataset](#)**

Load Dataset using Pandas

```
import pandas as pd

df = pd.read_csv("spam.csv", encoding='latin-1')

df = df[['v1', 'v2']]

df.columns = ['label', 'message']

print("Dataset Shape:", df.shape)

print("\nFirst 5 rows:\n", df.head())
```

```
jupyter AML project last checkpoint: 8 minutes ago
File Edit View Run Kernel Settings Help
jupyterlab Trusted
jupyterlab Python 3 (ipykernel)
Dataset Shape: (5572, 2)

First 5 rows:
label message
0 ham Go tell young point, crazy, available only ...
1 ham (oh lar... Joking wif e cell...
2 spam Free entry in 2 a willy comp to win PA Cup Final...
3 ham If don see an early here... d e already then say...
4 ham ham I don't think he gonna to arr_ be likes who...
```

Data Preprocessing

```
df['label_num'] = df.label.map({'ham':0, 'spam':1})  
print("Missing Values:\n", df.isnull().sum())
```

Explanation:

- No missing values in this dataset.
 - Encoding labels is essential for ML models.
 - Preprocessing helps the model correctly interpret data..

Model Selection

Model Chosen: Multinomial Naive Bayes

```
from sklearn.feature_extraction.text import CountVectorizer, TfidfTransformer
```

```
from sklearn.naive_bayes import MultinomialNB
```

Why Naive Bayes?

- Naive Bayes is excellent for text classification.
 - Handles word frequency naturally.
 - Very fast and accurate for spam detection.

Train & Test Model:

The screenshot shows a Jupyter Notebook interface with a single code cell containing Python code. The code imports necessary libraries (sklearn.model_selection, sklearn.pipeline, sklearn.metrics), performs a train-test split on the dataset, creates a Pipeline (CountVectorizer, TfidfTransformer, MultinomialNB), fits the model, and prints a success message. It then uses the trained model to predict on the test set, calculates accuracy, prints a confusion matrix, and generates a classification report.

```
#%matplotlib inline
#%pylab inline
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

X_train, X_test, y_train, y_test = train_test_split(
    df['message'], df['label'], random_state=42)

text_clf = Pipeline([
    ('vect', CountVectorizer()),
    ('tfidf', TfidfTransformer()),
    ('clf', MultinomialNB())
])

text_clf.fit(X_train, y_train)
print("Model trained successfully!")

y_pred = text_clf.predict(X_test)

acc = accuracy_score(y_test, y_pred)
print("Accuracy Score:", acc)

cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

print("Classification Report:\n", classification_report(y_test, y_pred))

Model trained successfully!
```

Model Evaluation

The screenshot shows the same Jupyter Notebook interface with the same code cell. The output now displays the calculated metrics: Accuracy Score (0.9823318385654025), Confusion Matrix ([[989 6], [42 109]]), and Classification Report (Table below).

	precision	recall	f1-score	support
0	0.95	1.00	0.98	105
1	1.00	0.72	0.84	138
accuracy	0.98	0.98	0.98	243
macro avg	0.98	0.98	0.98	243
weighted avg	0.98	0.98	0.98	243

- Accuracy shows **overall correct predictions**.
- Confusion matrix shows how many messages were correctly identified as spam/ham and misclassified.
- Classification report gives precision, recall, and F1-score for each class.

Result Explanation

Predicted: Whether each message is spam or not.

Performance: Accuracy is usually **>97%**, very high for spam detection.

Improvement: Use advanced models like **Logistic Regression** or **deep learning NLP models** to detect more subtle spam patterns

Mini Report

Project Title: Email/Message Spam Detection System

Problem Statement: Detect spam messages automatically to prevent fraud and save time.

Dataset: SMS Spam Collection Dataset from Kaggle (~5,574 messages)

Model Used: Multinomial Naive Bayes with TF-IDF

Performance:

- Accuracy: ~97–98%
- Confusion matrix shows most spam messages correctly identified

Learning Outcome:

- Learned text preprocessing and feature extraction
- Built an end-to-end spam classifier
- Evaluated model using accuracy, confusion matrix, and classification metrics