## Spam Classifier – Full Evaluation

- **1. Project Objective**
- **Goal:** Build a machine learning model to classify SMS/email messages as **spam** (unwanted) or **ham** (normal).
- **Type of problem:** Text Classification.
- **Key ML concepts:** Text preprocessing, TF-IDF, Naive Bayes classifier.

---

- **2. Dataset**
- **Source:** spam.csv located at D:\Documents\ML 100 days\Proj-4\spam.csv.
- **Columns used:**
  - label: Spam or Ham
  - message: The text of the SMS/email
- **Exploration:**
  - Checked dataset shape, count of spam vs ham messages.
  - Spam messages are fewer than ham messages, which is typical in real-world datasets.

---

- **3. Text Preprocessing**

Machines cannot understand raw text. We need to convert it into numbers.

**Steps we performed:**

1. **Tokenization:** Split messages into words.
2. **Stopword Removal:** Removed common words like "the", "is", "at" that carry little meaning.

3. **TF-IDF (Term Frequency – Inverse Document Frequency):**
    - Converts text to numeric vectors.
    - Words appearing frequently in one message but not across all messages are weighted higher.
    - Example: "free" in spam → higher weight, "the" → low weight.

**Code used:**

```
from sklearn.feature_extraction.text import TfidfVectorizer


vectorizer = TfidfVectorizer(stop_words='english')

X_train_tfidf = vectorizer.fit_transform(X_train)

X_test_tfidf = vectorizer.transform(X_test)
```

---

- **4. Data Visualization**
- **WordCloud** to understand frequent words:
    - **Spam WordCloud:** Shows words like "free, win, offer"
    - **Ham WordCloud:** Shows words like "ok, call, meeting"

**Code used:**

```
from wordcloud import WordCloud

import matplotlib.pyplot as plt


spam_wc = WordCloud(width=600, height=400, background_color="black").generate(" ".join(spam_messages))

plt.imshow(spam_wc, interpolation="bilinear")
```

Visualization helps us **intuitively understand patterns** in spam vs ham messages.

---

- **5. Model Training**
- **Algorithm:** Multinomial Naive Bayes
- **Why Naive Bayes?**
    - Simple and fast for text classification.
    - Works well with high-dimensional features (many words).
    - Assumes independence between words (naive assumption).

**Code used:**

```
from sklearn.naive_bayes import MultinomialNB


model = MultinomialNB()

model.fit(X_train_tfidf, y_train)

y_pred = model.predict(X_test_tfidf)
```

---

- **6. Model Evaluation**
- **Accuracy:** ~97–99%
- **Confusion Matrix:**
    - **True Positive (TP):** Correctly predicted spam
    - **True Negative (TN):** Correctly predicted ham
    - **False Positive (FP):** Ham predicted as spam
    - **False Negative (FN):** Spam predicted as ham

**Code used:**

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report


print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

**Interpretation:**

- High accuracy shows the model can reliably distinguish spam from ham.
- Precision & recall can indicate if spam detection is more important than avoiding false alarms.

---

- **7. Testing Custom Messages**
- You can test the model with new messages:

```
test_messages = [
   "Congratulations! You won a free ticket to Bahamas. Claim now!",
   "Hi John, are we still meeting for lunch tomorrow?"
]


test_tfidf = vectorizer.transform(test_messages)
pred = model.predict(test_tfidf)
```

- Output:
  - First message → **Spam**
  - Second message → **Ham**

This demonstrates **real-world usage** of the classifier.

- **8. Summary & Key Takeaways**
- **Theory Applied:**
  - Text preprocessing, TF-IDF, Naive Bayes probability theory.
  - Understanding of spam patterns through visualization.
- **Skills Practiced:**
  - Pandas for data handling
  - Scikit-learn for ML models
  - Matplotlib & WordCloud for visualization
- **Outcome:**
  - Fully functional spam detection model.
  - Can predict new messages with high accuracy.
  - Clear understanding of preprocessing, feature extraction, and model evaluation.