Import Libraries

```
import pandas as pd
import numpy as np
import string
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer

from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import LinearSVC
from sklearn.metrics import accuracy_score, precision_score,
recall_score, classification_report

nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to C:\Users\Sandipan
[nltk_data]     Jana\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!

True
```

Load Dataset

```
df = pd.read_csv("D:/downloads 2/spam.csv")  # or full path if needed
df.columns = ['label', 'text']
df['label'] = df['label'].map({'ham': 0, 'spam': 1})
df.head()
```

```
   label                                               text
0      0                   Hey, are we still meeting later?
1      1  Congratulations! You've won a $1000 Walmart gi...
2      0                 I'll call you later when I'm free.
3      1  URGENT! Your account has been compromised. Ver...
4      0      Don't forget to bring the documents tomorrow.
```

Preprocess Text

```
stemmer = PorterStemmer()
stop_words = set(stopwords.words('english'))

def clean_text(text):
```

```
    text = text.lower()
    text = re.sub(r'\d+', '', text)  # remove numbers
    text = text.translate(str.maketrans('', '', string.punctuation))
# remove punctuation
    tokens = text.split()
    tokens = [stemmer.stem(word) for word in tokens if word not in
stop_words]
    return ' '.join(tokens)

df['clean_text'] = df['text'].apply(clean_text)
```

TF-IDF Vectorization

```
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['clean_text']).toarray()
y = df['label']
```

Train/Test Split

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

Train Models

Naive Bayes

```
nb_model = MultinomialNB()
nb_model.fit(X_train, y_train)
nb_preds = nb_model.predict(X_test)
```

SVM

```
svm_model = LinearSVC()
svm_model.fit(X_train, y_train)
svm_preds = svm_model.predict(X_test)
```

Evaluation Function

```
def evaluate_model(y_true, y_pred, model_name):
    print(f"\n=== {model_name} ===")
    print("Accuracy:", accuracy_score(y_true, y_pred))
    print("Precision:", precision_score(y_true, y_pred))
    print("Recall:", recall_score(y_true, y_pred))
    print("Classification Report:\n", classification_report(y_true,
y_pred))

evaluate_model(y_test, nb_preds, "Naive Bayes")
evaluate_model(y_test, svm_preds, "SVM")
```

```
=== Naive Bayes ===
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2


=== SVM ===
Accuracy: 1.0
Precision: 1.0
Recall: 1.0
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      1.00      1.00         1

    accuracy                           1.00         2
   macro avg       1.00      1.00      1.00         2
weighted avg       1.00      1.00      1.00         2
```

Save the Model for Reuse

```
import joblib
joblib.dump(svm_model, 'spam_classifier.pkl')
joblib.dump(tfidf, 'tfidf_vectorizer.pkl')
```

```
['tfidf_vectorizer.pkl']
```

Spam vs Ham Count (Bar Chart)

```

```