Practical – 6 -> Detecting spam or ham (non-spam) messages is a common Natural Language Processing (NLP) problem. It involves classifying text messages or emails into one of these two categories. Here's a step-by-step guide on how to build a basic spam detection model using Python and common NLP libraries like scikit-learn and NLTK.

1. Dataset Preparation

A dataset with labeled messages as either spam or ham. The most common dataset for this task is the SMS Spam Collection Dataset.

2. Data Preprocessing

- Convert all text to lowercase.
- Remove punctuation, numbers, and stop words (common words that don't contribute much to the meaning, like "and", "the", etc.).
- Tokenize the text (split into words).
- Stem or lemmatize the words (reduce words to their root form).

3. Feature Extraction

- Use methods like Bag of Words (BoW), Term Frequency-Inverse Document Frequency (TF-IDF), or word embeddings to convert text into numerical features.

4. Model Training

- Use machine learning models like Naive Bayes, Support Vector Machines, or Logistic Regression to train the classifier on the feature set.

5. Model Evaluation

- Evaluate the model using metrics like accuracy, precision, recall, and F1-score on a test dataset.

6. Implementation

 # Import necessary libraries

```
import pandas as pd

import numpy as np

import re

import nltk

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.naive_bayes import MultinomialNB
```

```
from sklearn.metrics import accuracy_score, classification_report

# Download NLTK stopwords
nltk.download('stopwords')
from nltk.corpus import stopwords

# Load the dataset
# If you have the dataset as a text file named 'SMSSpamCollection', use the following:
# df = pd.read_csv('SMSSpamCollection', sep='\t', names=['label', 'message'])

# Or download it using pandas directly (assuming it is hosted online).
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00228/smsspamcollection.zip"
df = pd.read_csv('SMSSpamCollection', sep='\t', names=['label', 'message'])

# Display first few rows
print(df.head())

# Define stopwords
stop_words = set(stopwords.words('english'))

# Function to preprocess the text
def preprocess_text(text):
    # Convert text to lowercase
    text = text.lower()
    # Remove non-alphabetic characters
    text = re.sub(r'\W', ' ', text)
    # Remove numbers
    text = re.sub(r'\d', ' ', text)
    # Remove single characters
```

```
    text = re.sub(r'\s+[a-zA-Z]\s+', ' ', text)
    # Remove multiple spaces
    text = re.sub(r'\s+', ' ', text)
    # Remove stopwords
    text = ' '.join(word for word in text.split() if word not in stop_words)
    return text

# Apply preprocessing to the messages
df['message'] = df['message'].apply(preprocess_text)

# Display some processed messages
print(df.head())

# Convert labels to binary (1 for spam, 0 for ham)
df['label'] = df['label'].map({'spam': 1, 'ham': 0})

# Feature extraction using TF-IDF Vectorizer
tfidf = TfidfVectorizer(max_features=3000)
X = tfidf.fit_transform(df['message']).toarray()
y = df['label'].values

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize and train the Naive Bayes model
model = MultinomialNB()
model.fit(X_train, y_train)

# Make predictions on the test set
```

```
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred, target_names=['ham', 'spam'])

# Print the results
print(f"Accuracy: {accuracy:.2f}")
print(f"\nClassification Report:\n{report}")
```