**Spam SMS Detection Task**

**Objective**:

Develop a machine learning model that classifies SMS messages as either Spam or Ham (Not Spam).

**Dataset Overview:**

Contains SMS messages labeled as ‘spam’ or ‘ham’.

Label Message

Ham “Hi how are you?

Spam “Win a free vacation now! Text win to 12345”

**Example dataset:** SMS Spam Collection Dataset (UCI Machine Learning Repository)

**Expected Accuracy:** 95% - 98% depending on dataset size and model tuning.

**Expected Outcome:**

1)A robust spam classification model that accurately identifies spam messages.

2)Key Deliverables:

Cleaned and preprocessed dataset

Trained model (Naïve Bayes / Logistic Regression / SVM)

Performance metrics report

Model ready for deployment or integration into an SMS filtering system

**Python code:**

# Import Libraries

Import pandas as pd

Import numpy as np

Import re

Import nltk

From nltk.corpus import stopwords

From nltk.stem.porter import PorterStemmer

From sklearn.feature\_extraction.text import TfidfVectorizer

From sklearn.model\_selection import train\_test\_split

From sklearn.naive\_bayes import MultinomialNB

From sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Download stopwords

Nltk.download(‘stopwords’)

# Load Dataset

Df = pd.read\_csv(‘spam.csv’, encoding=’latin-1’)[[‘v1’, ‘v2’]]

Df.columns = [‘label’, ‘message’]

# Encode Labels

Df[‘label’] = df[‘label’].map({‘ham’: 0, ‘spam’: 1})

# Text Preprocessing Function

Ps = PorterStemmer()

Def clean\_text(text):

Text = re.sub(‘[^a-zA-Z]’, ‘ ‘, text).lower()

Text = text.split()

Text = [ps.stem(word) for word in text if not word in stopwords.words(‘english’)]

Return ‘ ‘.join(text)

# Apply Preprocessing

Df[‘clean\_message’] = df[‘message’].apply(clean\_text)

# TF-IDF Vectorization

Tfidf = TfidfVectorizer(max\_features=3000)

X = tfidf.fit\_transform(df[‘clean\_message’]).toarray()

Y = df[‘label’]

# Split Dataset

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=0)

# Model Training

Model = MultinomialNB()

Model.fit(X\_train, y\_train)

# Model Prediction

Y\_pred = model.predict(X\_test)

# Evaluation

Print(“Accuracy:”, accuracy\_score(y\_test, y\_pred))

Print(“\nClassification Report:\n”, classification\_report(y\_test, y\_pred))

Print(“\nConfusion Matrix:\n”, confusion\_matrix(y\_test, y\_pred))