Name : Atharv Hanumant Admile

Batch no: 06

1. Basic Text Classification Project

Objective: Build a simple text classification model and compare 2–3 algorithms.

Dataset:

- SMS Spam Collection Dataset

Steps:

1. Text Cleaning (lowercase, remove punctuation)
2. Remove stopwords
3. Convert text using CountVectorizer and TF-IDF

Models to Use:

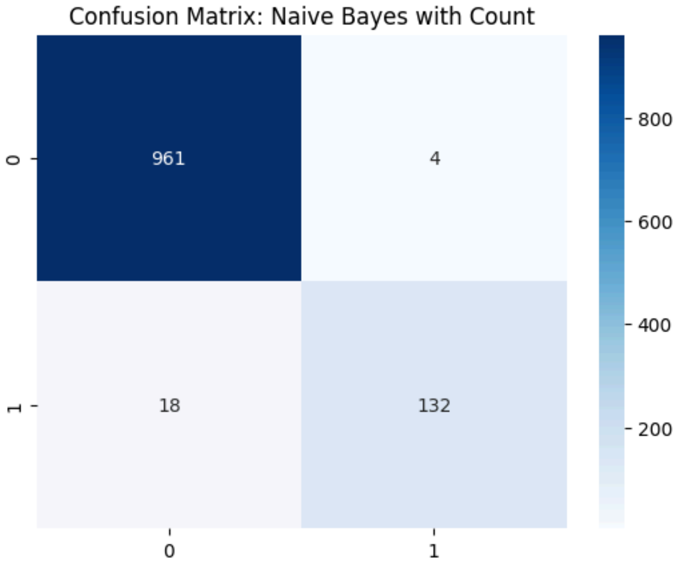
- Naive Bayes
- Logistic Regression
- Support Vector Machine (Optional)

Evaluation:

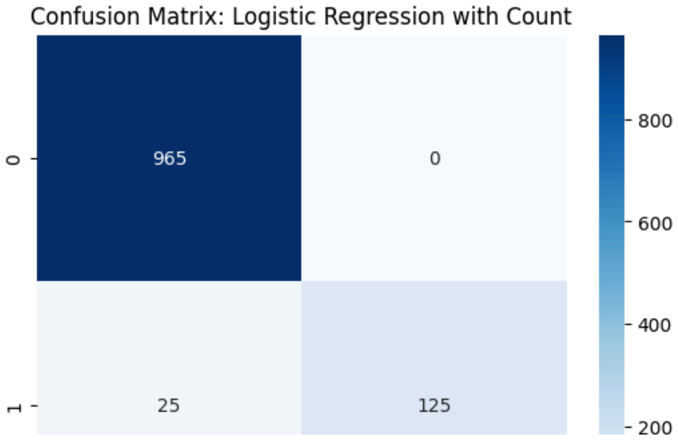
- Accuracy
- Confusion Matrix
- Classification Report Bonus:
- Compare CountVectorizer vs TF-IDF results
- Show top important words for each class

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import matplotlib.pyplot as plt
import seaborn as sns

nltk.download('stopwords')

# Load dataset
url = 'https://raw.githubusercontent.com/mohitgupta-omg/Kaggle-SMS-Spam-Collection-Dataset-/master/spam.csv'
df = pd.read_csv(url, encoding='latin-1')
df = df[['v1', 'v2']]
df.columns = ['label', 'text']
df['label'] = df['label'].map({'ham': 0, 'spam': 1})

# Text cleaning
def clean_text(text):
    text = text.lower()
    text = ''.join([char for char in text if char not in string.punctuation])
    words = text.split()
    words = [word for word in words if word not in stopwords.words('english')]
    return ' '.join(words)

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

# Split data
X = df['clean_text']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Function to train and evaluate
def train_evaluate(vectorizer, model, name, vec_name):
    X_train_vec = vectorizer.fit_transform(X_train)
    X_test_vec = vectorizer.transform(X_test)
    model.fit(X_train_vec, y_train)
    y_pred = model.predict(X_test_vec)
    acc = accuracy_score(y_test, y_pred)
    print(f"{name} with {vec_name} - Accuracy: {acc:.4f}")
    print(classification_report(y_test, y_pred))
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f"Confusion Matrix: {name} with {vec_name}")
    plt.show()
    return acc, vectorizer, model

# Vectorizers
count_vec = CountVectorizer()
tfidf_vec = TfidfVectorizer()

# Models
nb = MultinomialNB()
lr = LogisticRegression()
svm = SVC(kernel='linear')  # Optional

# Evaluate with CountVectorizer
print("Using CountVectorizer:")
acc_nb_count, count_vec, nb_count = train_evaluate(count_vec, nb, "Naive Bayes", "Count")
acc_lr_count, _, lr_count = train_evaluate(count_vec, lr, "Logistic Regression", "Count")
acc_svm_count, _, svm_count = train_evaluate(count_vec, svm, "SVM", "Count")

# Evaluate with TF-IDF
print("\nUsing TF-IDF:")
acc_nb_tfidf, tfidf_vec, nb_tfidf = train_evaluate(tfidf_vec, nb, "Naive Bayes", "TF-IDF")
acc_lr_tfidf, _, lr_tfidf = train_evaluate(tfidf_vec, lr, "Logistic Regression", "TF-IDF")
acc_svm_tfidf, _, svm_tfidf = train_evaluate(tfidf_vec, svm, "SVM", "TF-IDF")

# Bonus: Compare vectorizers
data = {
    'Model': ['Naive Bayes', 'Logistic Regression', 'SVM'],
    'CountVectorizer Accuracy': [acc_nb_count, acc_lr_count, acc_svm_count],
    'TF-IDF Accuracy': [acc_nb_tfidf, acc_lr_tfidf, acc_svm_tfidf]
}
comparison_df = pd.DataFrame(data)
print("\nComparison:")
print(comparison_df)

# Bonus: Top important words for each class (using Logistic Regression with TF-IDF)
feature_names = tfidf_vec.get_feature_names_out()
coef = lr_tfidf.coef_[0]
top_spam = [feature_names[i] for i in coef.argsort()[-10:][::-1]]
top_ham = [feature_names[i] for i in coef.argsort()[:10]]
print("\nTop words for Spam:", top_spam)
print("Top words for Ham:", top_ham)
```

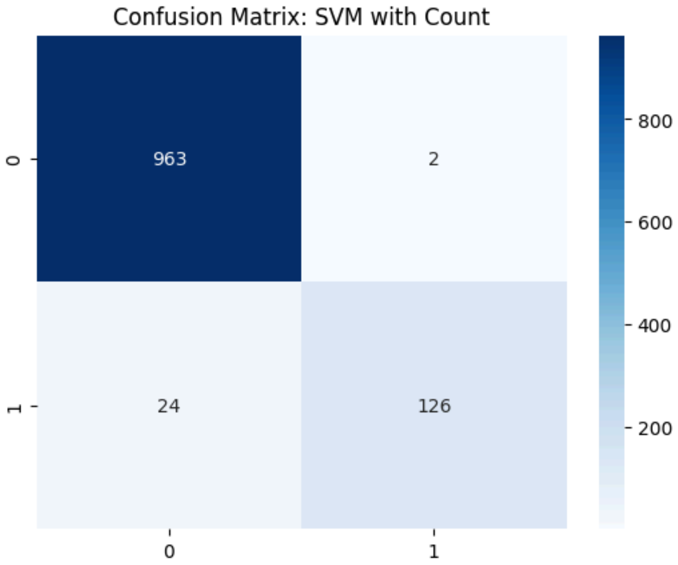
```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.12/dist-packages (1.6.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.12/dist-packages (2.2.2)
Requirement already satisfied: nltk in /usr/local/lib/python3.12/dist-packages (3.9.1)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.12/dist-packages (3.10.0)
Requirement already satisfied: seaborn in /usr/local/lib/python3.12/dist-packages (0.13.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (2.
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.1
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.12/dist-packages (from scikit-learn) (1.
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.12/dist-packages (from scikit-lea
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.12/dist-packages (from pandas)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.12/dist-packages (from pandas) (2025.3)
Requirement already satisfied: click in /usr/local/lib/python3.12/dist-packages (from nltk) (8.3.1)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.12/dist-packages (from nltk) (2025.11.
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-packages (from nltk) (4.67.3)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (1
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (26
Requirement already satisfied: pillow>=8 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (11.3.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.12/dist-packages (from matplotlib) (3
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.8.2-
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
Using CountVectorizer:
Naive Bayes with Count - Accuracy: 0.9803
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       965
           1       0.97      0.88      0.92       150

    accuracy                           0.98      1115
   macro avg       0.98      0.94      0.96      1115
weighted avg       0.98      0.98      0.98      1115
```



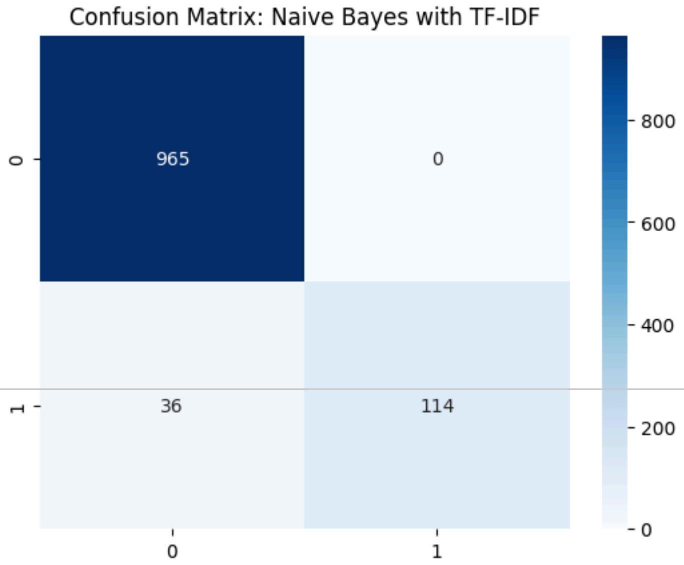
```
Logistic Regression with Count - Accuracy: 0.9776
              precision    recall  f1-score   support

           0       0.97      1.00      0.99       965
           1       1.00      0.83      0.91       150

    accuracy                           0.98      1115
   macro avg       0.99      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115
```



```
SVM with Count - Accuracy: 0.9767
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       965
           1       0.98      0.84      0.91       150

    accuracy                           0.98      1115
   macro avg       0.98      0.92      0.95      1115
weighted avg       0.98      0.98      0.98      1115
```


Confusion Matrix: SVM with Count

```
Using TF-IDF:
Naive Bayes with TF-IDF - Accuracy: 0.9677
              precision    recall  f1-score   support

           0       0.96      1.00      0.98       965
           1       1.00      0.76      0.86       150

    accuracy                           0.97      1115
   macro avg       0.98      0.88      0.92      1115
weighted avg       0.97      0.97      0.97      1115
```


Confusion Matrix: Naive Bayes with TF-IDF

```
Logistic Regression with TF-IDF - Accuracy: 0.9498
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       965
           1       0.96      0.65      0.78       150

    accuracy                           0.95      1115
   macro avg       0.95      0.82      0.87      1115
weighted avg       0.95      0.95      0.95      1115
```

Confusion Matrix: Logistic Regression with TF-IDF