CodSoft
Code.Create.Succeed

Build an AI model that can classify SMS messages as spam or legitimate. Use techniques like TF-IDF or word embeddings with classifiers like Naive Bayes, Logistic Regression, or Support Vector Machines to identify spam messages

Install Required Libraries

```
!pip install pandas scikit-learn nltk
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (3.8.1)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.5.0)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk) (8.1.7)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from nltk) (4.66.6)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

Load the Dataset

```
import pandas as pd

# Load the dataset
df = pd.read_csv('spam.csv', encoding='latin-1')
# Drop unnecessary columns and rename for clarity
df = df[['v1', 'v2']]
df.columns = ['label', 'message']

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

```
  label                                            message
0   ham  Go until jurong point, crazy.. Available only ...
1   ham                      Ok lar... Joking wif u oni...
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...
3   ham  U dun say so early hor... U c already then say...
4   ham  Nah I don't think he goes to usf, he lives aro...
```

Data Preprocessing

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

# Check for missing values
print(df.isnull().sum())

# Optional: Text preprocessing (removing punctuations, lowercasing)
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords

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

```
def preprocess_message(message):
    message = re.sub(r'\W', ' ', message)  # Remove special characters
    message = message.lower()  # Lowercase
    message = ' '.join(word for word in message.split() if word not in stop_words)  # Remove stopwords
    return message

df['message'] = df['message'].apply(preprocess_message)
```

```
label      0
message    0
dtype: int64
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Splitting the Dataset

```
from sklearn.model_selection import train_test_split

X = df['message']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Feature Extraction using TF-IDF

Now, we'll convert the text messages into numerical features using TF-IDF.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```

Model Training

```
from sklearn.naive_bayes import MultinomialNB
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report
```

```
# Initialize classifiers
nb = MultinomialNB()
lr = LogisticRegression(max_iter=1000)
svm = SVC(kernel='linear', probability=True)
```

Naive Bayes

```
# Train and evaluate Naive Bayes
nb.fit(X_train_tfidf, y_train)
y_pred_nb = nb.predict(X_test_tfidf)
accuracy_nb = accuracy_score(y_test, y_pred_nb)
print(f'Naive Bayes Accuracy: {accuracy_nb:.2f}')
print(classification_report(y_test, y_pred_nb, target_names=['ham', 'spam']))
```

```
Naive Bayes Accuracy: 0.97
              precision    recall  f1-score   support

         ham       0.97      1.00      0.98       942
        spam       1.00      0.83      0.91       173

    accuracy                           0.97      1115
   macro avg       0.98      0.91      0.94      1115
weighted avg       0.97      0.97      0.97      1115
```

Logistic Regression

```
# Train and evaluate Logistic Regression
lr.fit(X_train_tfidf, y_train)
y_pred_lr = lr.predict(X_test_tfidf)
accuracy_lr = accuracy_score(y_test, y_pred_lr)
print(f'Logistic Regression Accuracy: {accuracy_lr:.2f}')
print(classification_report(y_test, y_pred_lr, target_names=['ham', 'spam']))
```

```
Logistic Regression Accuracy: 0.96
              precision    recall  f1-score   support

         ham       0.95      1.00      0.97       942
        spam       0.98      0.73      0.84       173

    accuracy                           0.96      1115
   macro avg       0.97      0.86      0.91      1115
weighted avg       0.96      0.96      0.95      1115
```

SVM

```
# Train and evaluate SVM
svm.fit(X_train_tfidf, y_train)
y_pred_svm = svm.predict(X_test_tfidf)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print(f'SVM Accuracy: {accuracy_svm:.2f}')
print(classification_report(y_test, y_pred_svm, target_names=['ham', 'spam']))
```

```
SVM Accuracy: 0.98
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99       942
        spam       0.99      0.91      0.95       173

    accuracy                           0.98      1115
   macro avg       0.99      0.95      0.97      1115
weighted avg       0.98      0.98      0.98      1115
```

```
# Prediction function using Naive Bayes as default
def predict_message(message, model=nb):
    message_processed = preprocess_message(message)
    message_tfidf = vectorizer.transform([message_processed])
    prediction = model.predict(message_tfidf)
    return 'spam' if prediction[0] == 1 else 'ham'
```

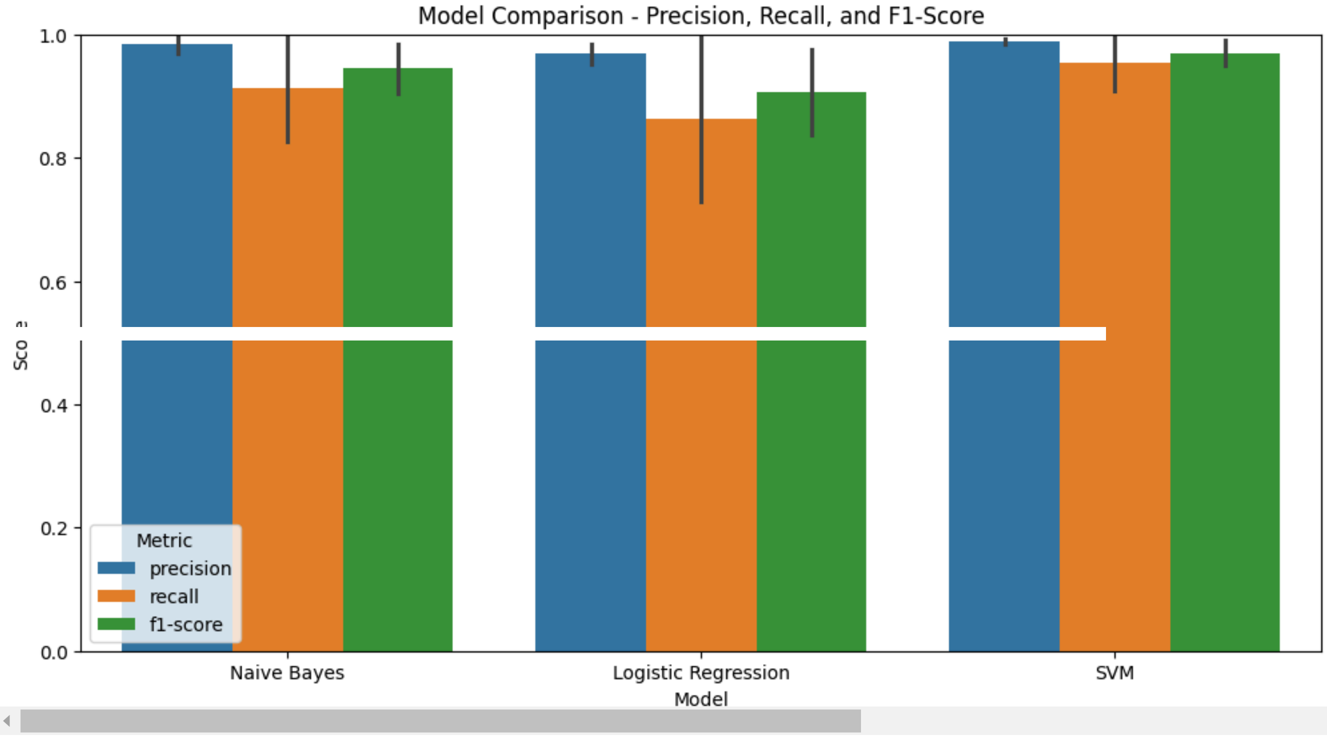
```
# Example usage
test_message = "You have 1 new message for dating. Call 0207-083-6089"
print("Naive Bayes prediction:", predict_message(test_message, nb))
print("Logistic Regression prediction:", predict_message(test_message, lr))
print("SVM prediction:", predict_message(test_message, svm))
```

```
Naive Bayes prediction: spam
Logistic Regression prediction: spam
SVM prediction: spam
```

Performance Visualisation

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report

# Generate classification reports for each model and convert them to DataFrames
def get_metrics_df(model_name, y_true, y_pred):
    report = classification_report(y_true, y_pred, output_dict=True, target_names=['ham', 'spam'])
    df = pd.DataFrame(report).transpose()
    df['model'] = model_name
    return df[['model', 'precision', 'recall', 'f1-score', 'support']]
```

```
# Generate predictions and get metrics for each model
nb_metrics = get_metrics_df('Naive Bayes', y_test, y_pred_nb)
lr_metrics = get_metrics_df('Logistic Regression', y_test, y_pred_lr)
svm_metrics = get_metrics_df('SVM', y_test, y_pred_svm)

# Combine all metrics into a single DataFrame
metrics_df = pd.concat([nb_metrics, lr_metrics, svm_metrics])

# Filter out the 'accuracy' and 'support' rows to focus on precision, recall, and f1-score
metrics_df = metrics_df[metrics_df.index.isin(['ham', 'spam'])]

# Plotting
plt.figure(figsize=(12, 6))
sns.barplot(x='model', y='value', hue='variable', data=pd.melt(metrics_df, id_vars=['model'], value_vars=['precision',
plt.title("Model Comparison - Precision, Recall, and F1-Score")
plt.xlabel("Model")
plt.ylabel("Score")
plt.ylim(0, 1)
plt.legend(title="Metric")
plt.show()
```


Model Comparison - Precision, Recall, and F1-Score
Sco
Metric
precision
recall
f1-score
Naive Bayes
Logistic Regression
SVM
Model