Spam Classifier ML Project

Project Overview

A machine learning-based spam SMS classifier that distinguishes between spam and ham (non-spam) messages using the Multinomial Naive Bayes algorithm. This project demonstrates fundamental natural language processing (NLP) techniques and machine learning implementation for text classification.

Features

- **Text Preprocessing**: Automatic data cleaning and transformation
- **Machine Learning Model**: Multinomial Naive Bayes classifier
- **Feature Extraction**: Count Vectorization for text-to-numeric conversion
- **Performance Evaluation**: Accuracy scoring and prediction capabilities
- **Real-time Prediction**: Classify new messages instantly

Installation & Setup

Prerequisites

- Python 3.7+
- Google Colab or local Python environment
- Required libraries: pandas, scikit-learn

Installation Steps

1. **Upload the dataset**:

python

from google.colab import files

uploaded = files.upload()

2. **Install required packages** (if not already installed):

python

!pip install pandas scikit-learn

Dataset Information

- **Source**: SMS Spam Collection Dataset
- **Size**: 5,574 messages
- **Classes**:
    - Ham (legitimate): 4,827 messages

- Spam: 747 messages

- **Features**: Message text and label

Project Structure

text

```
spam-classifier/
├── spam.csv              # Dataset file
├── spam_classifier.ipynb    # Main project file
└── README.md             # Documentation
```

Implementation

1. Data Loading & Exploration

python

```python
import pandas as pd
df = pd.read_csv("spam.csv")
df.head()
```

2. Data Preprocessing

- Check for missing values: df.isnull().sum()
- Convert labels to numerical values:

python

```python
df['label_num'] = df['label'].map({'ham':0, 'spam':1})
```

3. Train-Test Split

python

```python
from sklearn.model_selection import train_test_split
X = df['message']        # Features
y = df['label_num']      # Target
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

4. Feature Extraction

python

```
from sklearn.feature_extraction.text import CountVectorizer

cv = CountVectorizer()

X_train_cv = cv.fit_transform(X_train)

X_test_cv = cv.transform(X_test)
```

5. Model Training

python

```
from sklearn.naive_bayes import MultinomialNB

model = MultinomialNB()

model.fit(X_train_cv, y_train)
```

6. Model Evaluation

python

```
from sklearn.metrics import accuracy_score

y_pred = model.predict(X_test_cv)

accuracy = accuracy_score(y_test, y_pred)

print(f"Model Accuracy: {accuracy:.2%}")
```

Usage

Making Predictions

python

```
# Test with new message

message = ["Congratulations! You won a free iPhone"]

message_cv = cv.transform(message)

prediction = model.predict(message_cv)


# Convert prediction back to label

result = "spam" if prediction[0] == 1 else "ham"

print(f"The message is classified as: {result}")
```

Example Predictions

python

```
test_messages = [
```

```
    "Hey, are we still meeting tomorrow?",
    "FREE entry to win £1000 prize! Text NOW!",
    "Your package will arrive today at 3 PM"
]


for msg in test_messages:
    msg_cv = cv.transform([msg])
    pred = model.predict(msg_cv)[0]
    label = "spam" if pred == 1 else "ham"
    print(f"Message: {msg}")
    print(f"Classification: {label}\n")
```

Model Performance

- **Algorithm**: Multinomial Naive Bayes
- **Accuracy**: Typically achieves 98%+ accuracy
- **Training Time**: Fast training (seconds)
- **Prediction Time**: Real-time classification

Key Components

1. CountVectorizer

- Converts text documents to matrix of token counts
- Handles preprocessing internally (lowercasing, tokenization)
- Creates vocabulary from training data

2. Multinomial Naive Bayes

- Particularly suited for discrete features (word counts)
- Efficient for text classification
- Handles multiple classes naturally

3. Train-Test Split

- 80% training data, 20% testing data
- Random state for reproducibility
- Stratified sampling maintains class distribution

Results

The model achieves high accuracy in distinguishing between spam and ham messages. Example output:

text

```
Model Accuracy: 98.47%

Message: "Congratulations! You won a free iPhone" → Classification: spam

Message: "Hey, are we still meeting tomorrow?" → Classification: ham
```

Extending the Project

Additional Features

1. **TF-IDF Vectorization**:

python

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer()
```

2. **Multiple Algorithms**:

python

```
from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier
```

3. **Enhanced Evaluation**:

python

```
from sklearn.metrics import classification_report, confusion_matrix
```

Web Interface (Future Enhancement)

python

```
from flask import Flask, request, render_template

import joblib


app = Flask(__name__)

model = joblib.load('spam_model.pkl')


@app.route('/')

def home():
```

```python
    return render_template('index.html')


@app.route('/predict', methods=['POST'])
def predict():
    message = request.form['message']
    # Prediction logic here
    return render_template('result.html', prediction=prediction)
```

Troubleshooting

Common Issues

1. **File Upload Error**: Ensure spam.csv is in the correct directory
2. **Shape Mismatch**: Always use transform() not fit_transform() on test data
3. **Memory Issues**: Reduce dataset size or use sparse matrices

Solutions

python

```python
# Check data dimensions
print(f"Training shape: {X_train_cv.shape}")
print(f"Test shape: {X_test_cv.shape}")


# Verify label distribution
print(df['label'].value_counts())
```

Future Improvements

- Implement TF-IDF vectorization
- Add multiple machine learning algorithms
- Create web interface with Flask
- Deploy as REST API
- Add model persistence with joblib
- Implement cross-validation
- Add hyperparameter tuning
- Include more comprehensive evaluation metrics

Dependencies

python

pandas>=1.3.0

scikit-learn>=1.0.0

Contributing

1. Fork the repository
2. Create a feature branch
3. Commit your changes
4. Push to the branch
5. Create a Pull Request.

Contact

- **Developer**: SANJANA. N
- **Email**: sanjana28224@gmail.com