Spam Detection Model Documentation

1. Introduction

Spam detection is a classification problem where messages are categorized as either spam (unwanted messages) or ham (legitimate messages). This document provides a detailed overview of the data processing, exploratory data analysis (EDA), model building, and improvements used in this project.

2. Data Cleaning

2.1 Dataset Loading

The dataset `spam.csv` is loaded using pandas with `latin-1` encoding.

2.2 Dropping Unnecessary Columns

Several unnamed columns that are not useful for classification are removed:

```
df.drop(columns=['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], inplace=True)
```

2.3 Renaming Columns

The columns are renamed for better readability:

```
df.rename(columns={'v1': 'target', 'v2': 'text'}, inplace=True)
```

2.4 Encoding Target Labels

Label encoding is applied to convert categorical labels into numerical form:

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

df['target'] = encoder.fit_transform(df['target'])
```

2.5 Handling Missing and Duplicate Values

- Checking and removing null values:

```
df.isnull().sum()
```

- Checking and removing duplicate values:

```
df.duplicated().sum()
```

```
df = df.drop_duplicates(keep='first')
```

3. Exploratory Data Analysis (EDA)

EDA helps in understanding the distribution and characteristics of the dataset.

3.1 Data Distribution

The distribution of spam and ham messages is visualized using a pie chart:

```
plt.pie(df['target'].value_counts(), labels=['ham', 'spam'], autopct='%0.2f',
colors=['red', 'green'])
```

```
plt.show()
```

3.2 Feature Engineering

New features are created:

- `num_characters`: Character count in each message
- `num_words`: Word count
- `num_sentences`: Sentence count

```
df['num_characters'] = df['text'].apply(len)
```

```
df['num_words'] = df['text'].apply(lambda x: len(nltk.word_tokenize(x)))
```

```
df['num_sentences'] = df['text'].apply(lambda x: len(nltk.sent_tokenize(x)))
```

3.3 Correlation Heatmap

A heatmap is plotted to analyze feature correlations:

```
sns.heatmap(df.corr(numeric_only=True), annot=True)
```

3.4 Histogram Plots

Distribution of message length in spam and ham messages:

```
sns.histplot(df[df['target']==0]['num_characters'])

sns.histplot(df[df['target']==1]['num_characters'], color='red')

plt.show()
```

4. Data Preprocessing

Text data is transformed through:

- Lowercasing
- Tokenization
- Removing special characters, stopwords, and punctuation
- Stemming

```
from nltk.stem.porter import PorterStemmer

from nltk.corpus import stopwords

import string

ps = PorterStemmer()


def transform_text(text):

    text = text.lower()

    text = nltk.word_tokenize(text)

    y = [i for i in text if i.isalnum()]

    text = [i for i in y if i not in stopwords.words('english') and i not in
string.punctuation]

    text = [ps.stem(i) for i in text]
```

```
    return ' '.join(text)


df['transformed_text'] = df['text'].apply(transform_text)
```

5. Model Building

5.1 Feature Extraction

TF-IDF vectorization is used to convert text into numerical form:

```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(max_features=3000)

X = tfidf.fit_transform(df['transformed_text']).toarray()

y = df['target'].values
```

5.2 Splitting Data

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

5.3 Model Training and Evaluation

Different models are trained and evaluated:

```
from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score, confusion_matrix, precision_score

mnb = MultinomialNB()

mnb.fit(X_train, y_train)

y_pred = mnb.predict(X_test)

print(accuracy_score(y_test, y_pred))

print(confusion_matrix(y_test, y_pred))
```

```
print(precision_score(y_test, y_pred))
```

6. Model Comparison

Multiple classifiers are tested:

```
from sklearn.ensemble import RandomForestClassifier

clfs = {'NB': MultinomialNB(), 'RF': RandomForestClassifier(n_estimators=50, random_state=2)}

def train_classifier(clf, X_train, y_train, X_test, y_test):

    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)

    return accuracy_score(y_test, y_pred), precision_score(y_test, y_pred)
```

7. Model Saving

The best model is saved using Pickle:

```
import pickle

pickle.dump(tfidf, open('vectorizer.pkl', 'wb'))

pickle.dump(mnb, open('model.pkl', 'wb'))
```

8. Conclusion

This project successfully builds a spam detection model using NLP techniques, EDA, and machine learning models. The best-performing model is stored for future predictions.