NM_Project_2024 April 25, 2024

Naan Mudhalvan SchemeTNSDC – Machine Learning to Generative AI

Project:Insights from the SMS Spam Collection

RAJAVEL J

3rd Year – 2021503702

Madras Institute of Technology, Anna University

Introduction:

The SMS Spam Collection dataset is a valuable resource for studying spam messages in the context of SMS communication. With 5,574 messages tagged as either ham (legitimate) or spam, this dataset provides a rich source of information for researchers and practitioners interested in understanding and combating spam.

Problem Statement:

The problem of spam messages poses significant challenges in various communication channels, including SMS. Spam messages not only inconvenience users but also pose security risks and can lead to privacy breaches. The problem statement revolves around developing effective techniques to identify and differentiate between legitimate (ham) and spam messages in SMS communication.

Objective:

The primary objective of this research is to develop robust algorithms and models for accurately classifying SMS messages as ham or spam. By leveraging machine learning and natural language processing techniques, the aim is to create classifiers that can automatically detect spam messages with high precision and recall. Additionally, the research seeks to explore patterns and characteristics of spam messages to enhance understanding and inform future prevention strategies.

Import modules:

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

warnings.filterwarnings('ignore')

%matplotlib inline

ss Imbalancement

```
import pandas as pd
import numpy as np
import nltk
import re
from nltk.corpus import stopwords
```

```
from google.colab import files
uploaded = files.upload()
```

Loading the dataset:

```
import pandas as pd

# Read the CSV file into a DataFram
df = pd.read_csv('spam.csv', encodi

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

	v1	v2	Unnamed: 2	Unnam
0	ham	Go until jurong point, crazy.. Available only ...	NaN	
1	ham	Ok lar... Joking wif u oni...	NaN	
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	
3	ham	U dun say so early hor... U c already then say...	NaN	
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	

	Unnamed: 3	Unnamed: 4
0	NaN	NaN
1	NaN	NaN
2	NaN	NaN
3	NaN	NaN
4	NaN	NaN

```
[ ] # get necessary columns for processing
    df = df[['v2', 'v1']]
    # df.rename(columns={'v2': 'messages', 'v1': 'label'}, inplace=True)
    df = df.rename(columns={'v2': 'messages', 'v1': 'label'})
    df.head()
```

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

Preprocessing the dataset:

```
[ ]
    # check for null values
    df.isnull().sum()
```

```
messages    0
label       0
dtype: int64
```

```
def clean_text(text):
    # convert to lowercase
    text = text.lower()
    # remove special characters
    text = re.sub(r'[^0-9a-zA-Z]', ' ', text)
    # remove extra spaces
    text = re.sub(r'\s+', ' ', text)
    # remove stopwords
    text = " ".join(word for word in text.split() if word not in STOPWORDS)
    return text
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
[ ] # clean the messages
    df['clean_text'] = df['messages'].apply(clean_text)
    df.head()
```

	messages	label	clean_text
0	Go until jurong point, crazy.. Available only ...	ham	go jurong point crazy available bugis n great ...
1	Ok lar... Joking wif u oni...	ham	ok lar joking wif u oni
2	Free entry in 2 a wkly comp to win FA Cup fina...	spam	free entry 2 wkly comp win fa cup final tkts 2...
3	U dun say so early hor... U c already then say...	ham	u dun say early hor u c already say
4	Nah I don't think he goes to usf, he lives aro...	ham	nah think goes usf lives around though

Input Split:

```
[ ] X = df['clean_text']
    y = df['label']
```

Model Training:

from sklearn.pipeline import Pipeline

from sklearn.model_selection import train_test_split, cross_val_score

from sklearn.metrics import classification_report

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer, TfidfTransformer

def classify(model, X, y):

  # train test split x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42, shuffle=True, stratify=y)

  # model training pipeline_model = Pipeline([('vect', CountVectorizer()),

    ('tfidf', TfidfTransformer()), ('clf', model)])

  pipeline_model.fit(x_train, y_train)

  print('Accuracy:', pipeline_model.score(x_test, y_test)*100)

```
#   cv_score = cross_val_score(model, X, y, cv=5)
#    print("CV Score:", np.mean(cv_score)*100)
   y_pred = pipeline_model.predict(x_test)
   print(classification_report(y_test, y_pred)
```

```
[ ] from sklearn.linear_model import LogisticRegression
    model = LogisticRegression()
    classify(model, X, y)
```

```
[ ] Accuracy: 96.8413496051687
                precision    recall  f1-score   support

           ham       0.97      1.00      0.98      1206
          spam       0.99      0.77      0.87       187

      accuracy                           0.97      1393
     macro avg       0.98      0.88      0.92      1393
  weighted avg       0.97      0.97      0.97      1393
```

```
[ ] from sklearn.naive_bayes import MultinomialNB
    model = MultinomialNB()
    classify(model, X, y)
```

```
[ ] Accuracy: 96.69777458722182
                precision    recall  f1-score   support

           ham       0.96      1.00      0.98      1206
          spam       1.00      0.75      0.86       187

      accuracy                           0.97      1393
     macro avg       0.98      0.88      0.92      1393
  weighted avg       0.97      0.97      0.96      1393
```

```
[ ] from sklearn.svm import SVC
    model = SVC(C=3)
    classify(model, X, y)
```

```
[ ] Accuracy: 98.27709978463747
                precision    recall  f1-score   support

           ham       0.98      1.00      0.99      1206
          spam       1.00      0.87      0.93       187

      accuracy                           0.98      1393
     macro avg       0.99      0.94      0.96      1393
  weighted avg       0.98      0.98      0.98      1393
```

```
[ ] from sklearn.ensemble import RandomForestClassifier
    model = RandomForestClassifier()
    classify(model, X, y)
```

```
[ ] Accuracy: 97.5592246949031
                precision    recall  f1-score   support

           ham       0.97      1.00      0.99      1206
          spam       1.00      0.82      0.90       187

      accuracy                           0.98      1393
     macro avg       0.99      0.91      0.94      1393
  weighted avg       0.98      0.98      0.97      1393
```

Conclusion:

In conclusion, the SMS Spam Collection dataset offers a valuable opportunity to explore and address the challenges posed by spam messages in SMS communication. By leveraging this dataset and employing advanced techniques in machine learning and natural language processing, researchers can contribute to the development of more effective spam detection systems. Ultimately, such efforts can help improve user experience, enhance security, and mitigate the impact of spam in SMS communication.