welcome-to-colaboratory-1

September 22, 2024

Welcome to Colab!

(New) Try the Gemini API

Generate a Gemini API key

Talk to Gemini with the Speech-to-Text API

Gemini API: Quickstart with Python

Gemini API code sample

Compare Gemini with ChatGPT

More notebooks

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view and the command palette.

[5]:
```
import pandas as pd
import re
import nltk
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, confusion_matrix,␣
  ↪classification_report
import joblib

# Download necessary NLTK data
nltk.download('stopwords')
from nltk.corpus import stopwords

# Load the dataset
df = pd.read_csv("spam_ham_dataset.csv")

# Module 1: Text Preprocessing

# Function for cleaning the text
def clean_text(text):
    # Convert to lowercase
    text = text.lower()
    # Remove HTML tags
    text = re.sub(r'<[^>]*>', '', text)
    # Remove URLs
    text = re.sub(r'http\S+|www\S+|https\S+', '', text, flags=re.MULTILINE)
    # Remove non-alphabetical characters (keeping spaces)
    text = re.sub(r'[^a-z\s]', '', text)
    # Remove stopwords
    stop_words = set(stopwords.words('english'))
    text = ' '.join([word for word in text.split() if word not in stop_words])
    # Remove short words
    text = ' '.join([word for word in text.split() if len(word) > 2])
    return text

# Print the columns of the DataFrame to check their actual names
print(df.columns)

# Assuming the correct column name is 'text' based on common usage
df['cleaned_text'] = df['text'].apply(clean_text)

# Module 2: Feature Extraction and Model Training

# Features and Labels
X = df['cleaned_text']  # Cleaned email text
y = df['label']         # Label: spam=1, ham=0

# Feature Extraction using TF-IDF
vectorizer = TfidfVectorizer(max_features=5000)
X_tfidf = vectorizer.fit_transform(X)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2,␣
 ↪random_state=42)

# Train the Model: Using Multinomial Naive Bayes
model = MultinomialNB()
model.fit(X_train, y_train)

# Make predictions on the test set
y_pred = model.predict(X_test)

# Evaluate the Model
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# Additional Metrics
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```python
# Module 3: Model Saving and Deployment

# Save the trained model and vectorizer for deployment
joblib.dump(model, "spam_classifier_model.pkl")
joblib.dump(vectorizer, "tfidf_vectorizer.pkl")

# Load the model and vectorizer for future use
loaded_model = joblib.load("spam_classifier_model.pkl")
loaded_vectorizer = joblib.load("tfidf_vectorizer.pkl")

# Function to classify new incoming emails
def classify_email(email_text):
    # Clean and vectorize the email (use the same vectorizer as during training)
    cleaned_email = clean_text(email_text)
    email_vector = loaded_vectorizer.transform([cleaned_email])

    # Predict whether the email is spam or ham
    prediction = loaded_model.predict(email_vector)

    # Return the classification result
    return "Spam" if prediction == 1 else "Not Spam"

# Test the classification function with a new email
new_email = "Congratulations! You've won a $1000 gift card. Click here to claim␣
 ↪it."
result = classify_email(new_email)
print("The email is classified")
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Package stopwords is already up-to-date!
```

```
Index(['Unnamed: 0', 'label', 'text', 'label_num'], dtype='object')
Model Accuracy: 95.17%
[[714  28]
 [ 22 271]]
              precision    recall  f1-score   support

         ham       0.97      0.96      0.97       742
        spam       0.91      0.92      0.92       293

    accuracy                           0.95      1035
   macro avg       0.94      0.94      0.94      1035
weighted avg       0.95      0.95      0.95      1035

The email is classified
```