sfhzyszwz

February 1, 2025

1 Email-Detection

[30]:
```
# import the necc lib
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

[3]:
```
# read the data
df = pd.read_csv('/content/email_spam.csv')
```

[4]:
```
# check hte 5 top rows
df.head()
```

[4]:

	title	text	type
0	?? the secrets to SUCCESS	Hi James,\n\nHave you claim your complimentary…	spam
1	?? You Earned 500 GCLoot Points	\nalt_text\nCongratulations, you just earned\n…	not spam
2	?? Your GitHub launch code	Here's your GitHub launch code, @Mortyj420!\n …	not spam
3	[The Virtual Reward Center] Re: ** Clarifications	Hello,\n \nThank you for contacting the Virtua…	not spam
4	10-1 MLB Expert Inside, Plus Everything You Ne…	Hey Prachanda Rawal,\n\nToday's newsletter is …	spam

[ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   title   84 non-null     object
```

```
 1   text    84 non-null     object
 2   type    84 non-null     object
dtypes: object(3)
memory usage: 2.1+ KB
```

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

[ ]:
```
title    0
text     0
type     0
dtype: int64
```

2 Data Preparation

[5]:
```
# combining the 'title' and 'text' col
df['Text'] = df['title'] + ' ' + df['text']
df['Text']
```

[5]:
```
0     ?? the secrets to SUCCESS Hi James,\n\nHave yo…
1     ?? You Earned 500 GCLoot Points \nalt_text\nCo…
2     ?? Your GitHub launch code Here's your GitHub …
3     [The Virtual Reward Center] Re: ** Clarificati…
4     10-1 MLB Expert Inside, Plus Everything You Ne…
                            …
79    Your application for the position of  Child Pr…
80    Your Kilimall Account is Ready - Shopping Now!…
81    Your Steam account: Access from new web or mob…
82    Your uploaded document is rejected View In Bro…
83    You've Earned a Reward from Bard Explorers Ind…
Name: Text, Length: 84, dtype: object
```

[6]:
```
# drop the title and text
df.drop(['title', 'text'],inplace = True,axis =1)
```

2.1 Label Encoding

[7]:
```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['type'] = le.fit_transform(df['type'])
```

[8]:
```
df.head()
```

[8]:
```
   type                                               Text
0     1  ?? the secrets to SUCCESS Hi James,\n\nHave yo…
```

```
1     0  ?? You Earned 500 GCLoot Points \nalt_text\nCo…
2     0  ?? Your GitHub launch code Here's your GitHub …
3     0  [The Virtual Reward Center] Re: ** Clarificati…
4     1  10-1 MLB Expert Inside, Plus Everything You Ne…
```

[9]:
```
# defining X
X = df['Text']
X.head()
```

[9]:
```
0    ?? the secrets to SUCCESS Hi James,\n\nHave yo…
1    ?? You Earned 500 GCLoot Points \nalt_text\nCo…
2    ?? Your GitHub launch code Here's your GitHub …
3    [The Virtual Reward Center] Re: ** Clarificati…
4    10-1 MLB Expert Inside, Plus Everything You Ne…
Name: Text, dtype: object
```

[10]:
```
# define y
y = df['type']
y.head()
```

[10]:
```
0    1
1    0
2    0
3    0
4    1
Name: type, dtype: int64
```

[11]:
```
# train_test_split
from sklearn.model_selection import train_test_split
```

[12]:
```
# splitting the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4,␣
 ↪random_state = 42)
```

[13]:
```
# check the shape of X_train and y_train
print(X_train.shape)
print(y_train.shape)
```

```
(50,)
(50,)
```

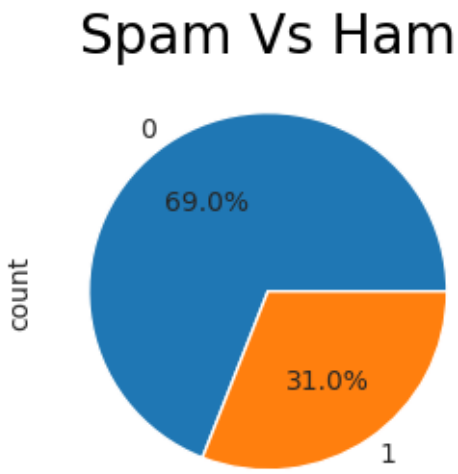
[14]:
```
# check the shape of X_test and y_test
print(X_test.shape)
print(y_test.shape)
```

```
(34,)
(34,)
```

[16]:
```
# convert dtype of y_train and y_test
y_train = y_train.astype(int)
y_test = y_test.astype(int)
```

[15]:
```
# plot the pie chart

plt.figure(figsize = (3,4))
sns.color_palette('deep')
sns.set_style('darkgrid')
y.value_counts().plot(kind='pie', autopct='%1.1f%%')
plt.title('Spam Vs Ham', color = 'black', size = 20)
plt.show()
```



3 Feature Extraction

[17]:
```
# import the TfidVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
```

[18]:
```
vectorizer = TfidfVectorizer(min_df = 1, stop_words = 'english', binary = True)
X_train_feature = vectorizer.fit_transform(X_train.astype(str))
X_test_feature = vectorizer.transform(X_test.astype(str))
```

[19]:
```
# Check if dimensions match
print(X_train_feature.shape)
print(X_test_feature.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(50, 1596)
(34, 1596)
(50,)
(34,)
```

[20]:
```
# Ensure the target variables are integers
y_train = y_train.astype(int)
y_test = y_test.astype(int)
```

4 SVM

[21]:
```
# import SVC
from sklearn.svm import SVC
```

[22]:
```
# define the model
svm = SVC(kernel = 'linear', class_weight='balanced')

# fit the model
svm.fit(X_train_feature, y_train)
```

[22]:
```
SVC(class_weight='balanced', kernel='linear')
```

[23]:
```
# predict the model
y_pred = svm.predict(X_test_feature)
y_pred
```

[23]:
```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

5 Evaluation

[24]:
```
# import accuracy score, classification report

from sklearn.metrics import accuracy_score, classification_report


accuracy = accuracy_score(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print('Accuracy', accuracy)
print()
print('class_report', class_report)
```

```
Accuracy 0.7352941176470589

class_report               precision    recall  f1-score   support
```

```
              0       0.72      1.00      0.84        23
              1       1.00      0.18      0.31        11

    accuracy                            0.74        34
   macro avg       0.86      0.59      0.57        34
weighted avg       0.81      0.74      0.67        34
```

6 Logistic Regression

[1]:
```
# import the logisitic regression

from sklearn.linear_model import LogisticRegression
```

[25]:
```
# initialize the model
lr_model = LogisticRegression()
```

[26]:
```
# train the model
lr_model.fit(X_train_feature, y_train)
```

[26]:
```
LogisticRegression()
```

[27]:
```
# predict the model
y_pred = lr_model.predict(X_test_feature)
```

7 Evaluation of LR

[32]:
```
# evaluation

accuracy_lr = accuracy_score(y_test, y_pred)
class_report_lr = classification_report(y_test, y_pred)

print('Accuracy:', accuracy_lr)
print()
print('class_report:', class_report_lr)
```

```
Accuracy: 0.6764705882352942

class_report:               precision    recall  f1-score   support

              0       0.68      1.00      0.81        23
              1       0.00      0.00      0.00        11

    accuracy                            0.68        34
   macro avg       0.34      0.50      0.40        34
```

```
weighted avg       0.46      0.68      0.55        34
```

8 Hyperparameter Tuning

[ ]:
```
from sklearn.model_selection import GridSearchCV
```

[ ]:
```
# define hyperparameters grid

para = {
    'C' : [0.1, 1, 10, 100],
    'kernel' : ['linear', 'rbf', 'poly'],
    'gamma' : ['scale', 'auto'],
    'degree' : [2,3,4]
    }
```

[ ]:
```
svm = SVC()

# intialize gridsearch
gs = GridSearchCV(svm, para, cv = 5, scoring = 'accuracy', n_jobs = -1)


# fit the model
gs.fit(X_train_feature, y_train)
```

[ ]:
```
GridSearchCV(cv=5, estimator=SVC(), n_jobs=-1,
             param_grid={'C': [0.1, 1, 10, 100], 'degree': [2, 3, 4],
                         'gamma': ['scale', 'auto'],
                         'kernel': ['linear', 'rbf', 'poly']},
             scoring='accuracy')
```

[ ]:
```
# get the best model from gridsearchcv

best_svm = gs.best_estimator_
best_svm
```

[ ]:
```
SVC(C=10, degree=2, kernel='linear')
```

[ ]:
```
# train the best model
best_svm.fit(X_train_feature, y_train)
```

[ ]:
```
SVC(C=10, degree=2, kernel='linear')
```

[ ]:
```
# predict the best svm

y_pred = best_svm.predict(X_test_feature)
y_pred
```

[ ]:
```
array([0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

[ ]:
```
# evaluation
print('accuracy_score : ', accuracy_score(y_test, y_pred))
```

```
accuracy_score :  0.7352941176470589
```

9 checking whether the mail is spam or ham

[ ]:
```
email_index = 63


# Get the email text using its index
email_text = [df.loc[email_index, 'Text']]

# Convert the text into numerical features
email_features = vectorizer.transform(email_text)

# Predict spam (1) or ham (0)
prediction = best_svm.predict(email_features)

print("Prediction:", "Spam" if prediction[0] == 1 else "Ham")
```

```
Prediction: Spam
```

10 check the multiple mails at once

[ ]:
```
email_index = [23, 66, 22]

# get the email text using its index
email_text = df.loc[email_index, 'Text'].tolist()

# convert the mail text into the numerical
email_features = vectorizer.transform(email_text)

# predict spam or ham
pred = best_svm.predict(email_features)

for i, predic in zip(email_index, pred):
  print('prediction : ', 'it is a spam email' if pred[0] == 1 else 'it is a ham␣
  ↪email')
```

```
prediction :  it is a spam email
prediction :  it is a spam email
prediction :  it is a spam email
```

11 Conclusion

This project compared SVM and Logistic Regression for email spam detector.

SVM has demonstrated the best accuracy as compare to the Logisitic model.

11.0.1 Feature Extraction

TF-IDF proved to be an effective feature extraction technique, enabling the models to capture important patterns in the email text.

11.0.2 Hyperparameter tuning

SVM, particularly after hyperparameter tuning, exhibited slightly better performance.

SVM accuracy : 74%

Logistic accuracy : 67%

[ ]: