pr4-spam-gs

November 2, 2024

[1]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages
(2.2.2)
Requirement already satisfied: numpy>=1.22.4 in /usr/local/lib/python3.10/dist-
packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-
packages (from pandas) (2024.2)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
```

[2]:
```
import numpy as np
import pandas as pd
```

[5]:
```
df = pd.read_csv('SMSSpamCollection',sep='\t',names=['label','text'])
```

[6]:
```
df.head()
```

[6]:

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

[7]:
```
df.shape
```

[7]:
```
(5572, 2)
```

[8]:
```
!pip install nltk
```

```
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages
(3.8.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages
```

```
(from nltk) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages
(from nltk) (1.4.2)
Requirement already satisfied: regex>=2021.8.3 in
/usr/local/lib/python3.10/dist-packages (from nltk) (2024.9.11)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages
(from nltk) (4.66.6)
```

[9]:
```
import nltk
```

[10]:
```
nltk.download('stopwords')
```

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

[10]:
```
True
```

[11]:
```
sent = 'How are you friends?'
```

[14]:
```
nltk.download('punkt')   #this is optional required in collab not juypter
```

```
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

[14]:
```
True
```

[15]:
```
from nltk.tokenize import word_tokenize
word_tokenize(sent)
```

[15]:
```
['How', 'are', 'you', 'friends', '?']
```

[16]:
```
from nltk.corpus import stopwords
swords = stopwords.words('english')
```

[18]:
```
#swords
```

[19]:
```
clean = [word for word in word_tokenize(sent) if word not in swords]
```

[20]:
```
clean
```

[20]:
```
['How', 'friends', '?']
```

[21]:
```
# Stemming words with NLTK
from nltk.stem import PorterStemmer
ps = PorterStemmer()
clean = [ps.stem(word) for word in word_tokenize(sent)
         if word not in swords]
clean
```

[21]:
```
['how', 'friend', '?']
```

[23]:
```
sent = 'Hello friends! How are you? We will learning python today'
```

[24]:
```
def clean_text(sent):
    tokens = word_tokenize(sent)
    clean = [word for word in tokens if word.isdigit() or word.isalpha()]
    clean = [ps.stem(word) for word in clean
         if word not in swords]
    return clean
    #print(clean)
```

[25]:
```
clean_text(sent)
```

[25]:
```
['hello', 'friend', 'how', 'we', 'learn', 'python', 'today']
```

[26]:
```
# Pre-processing
from sklearn.feature_extraction.text import TfidfVectorizer
```

[27]:
```
tfidf = TfidfVectorizer(analyzer=clean_text)
```

[28]:
```
x = df['text']
y = df['label']
```

[29]:
```
x_new = tfidf.fit_transform(x)
```

[30]:
```
x.shape
```

[30]:
```
(5572,)
```

[31]:
```
x_new.shape
```

[31]:
```
(5572, 6513)
```

[35]:
```
# Instead of using get_feature_names(), use get_feature_names_out()
tfidf.get_feature_names_out()
```

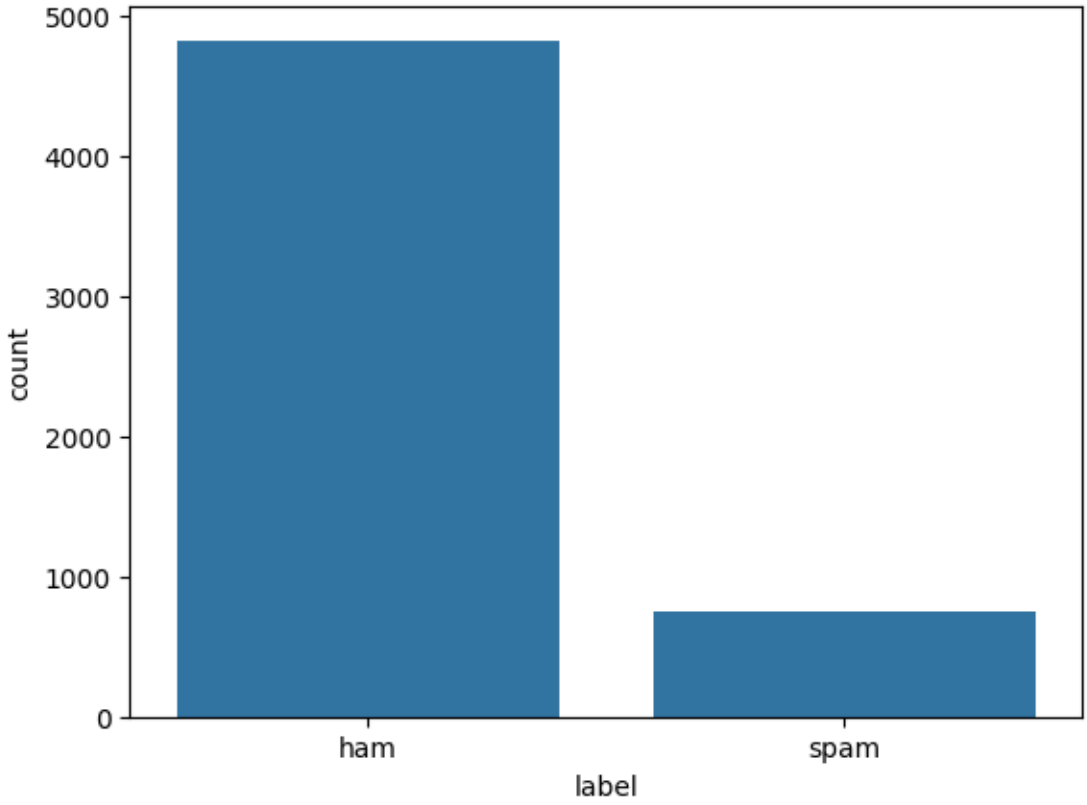
[35]:
```
array(['0', '008704050406', '0089', …, 'zyada', 'é', 'ü'], dtype=object)
```

[36]:
```
y.value_counts()
```

[36]:
```
label
ham     4825
spam     747
Name: count, dtype: int64
```

[37]:
```
import seaborn as sns
sns.countplot(x=y)
```

[37]:
```
<Axes: xlabel='label', ylabel='count'>
```



[38]:
```
#cross validation
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x_new,y,test_size=0.
↪25,random_state=0)
```

[39]:
```
x_train.shape
```

[39]:
```
(4179, 6513)
```

[41]:
```
x_test.shape
```

[41]:
```
(1393, 6513)
```

[42]:
```
y_train.shape
```

[42]:
```
(4179,)
```

[44]:
```
y_test.shape
```

[44]:
```
(1393,)
```

[45]:
```
from sklearn.naive_bayes import GaussianNB
```

[46]:
```
nb = GaussianNB()
```

[47]:
```
nb.fit(x_train.toarray(),y_train)
```

[47]:
```
GaussianNB()
```

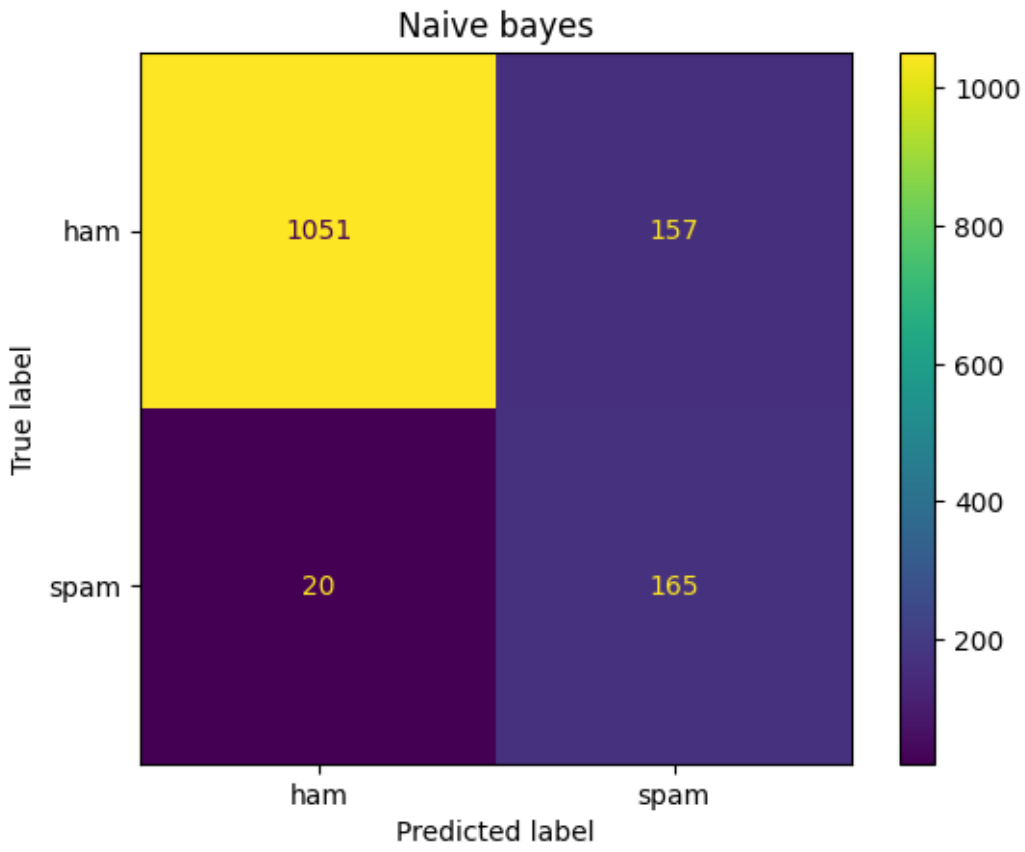
[48]:
```
y_pred_nb = nb.predict(x_test.toarray())
```

[49]:
```
y_test.value_counts()
```

[49]:
```
label
ham     1208
spam     185
Name: count, dtype: int64
```

[50]:
```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
```

[51]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_nb)
plt.title('Naive bayes')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_nb)}")
print(classification_report(y_test,y_pred_nb))
```



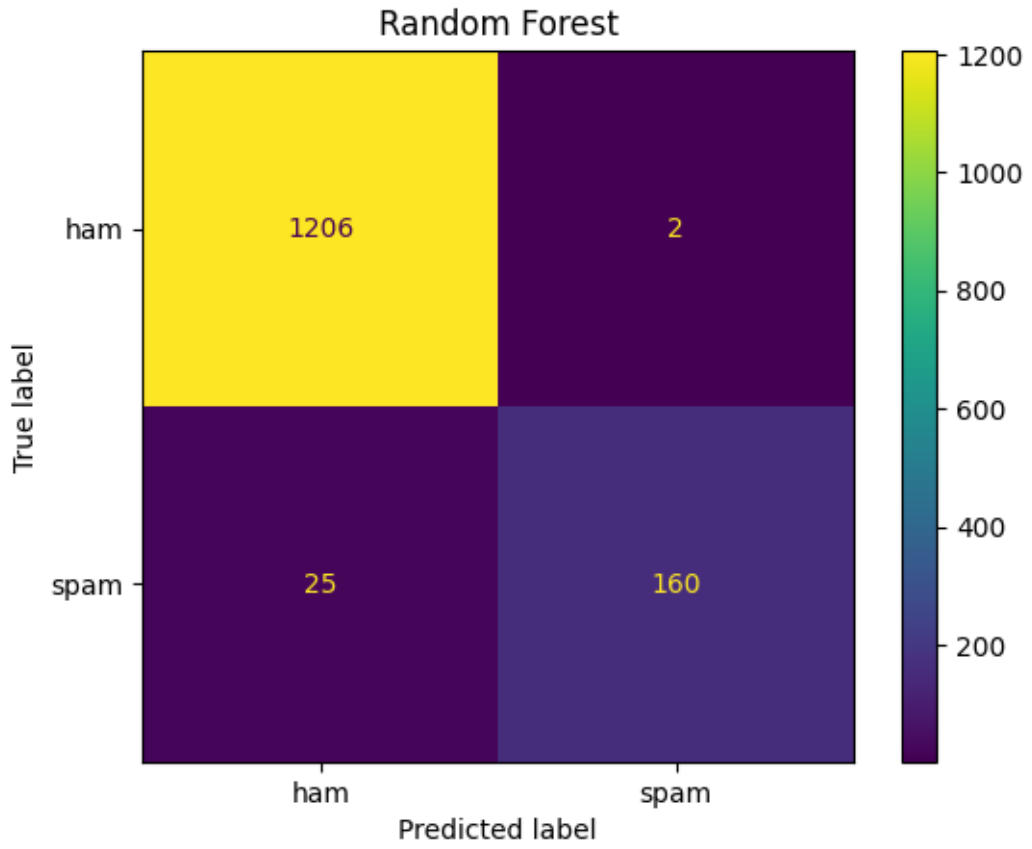

```
Accuracy is 0.8729361091170137
              precision    recall  f1-score   support

         ham       0.98      0.87      0.92      1208
        spam       0.51      0.89      0.65       185

    accuracy                           0.87      1393
   macro avg       0.75      0.88      0.79      1393
weighted avg       0.92      0.87      0.89      1393
```

[53]:
```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(random_state=0)
rf.fit(x_train,y_train)
```

[53]:
```
RandomForestClassifier(random_state=0)
```

[54]:
```
y_pred = rf.predict(x_test) #float
```

[58]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
plt.title('Random Forest')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred)}")
print(classification_report(y_test,y_pred))
```

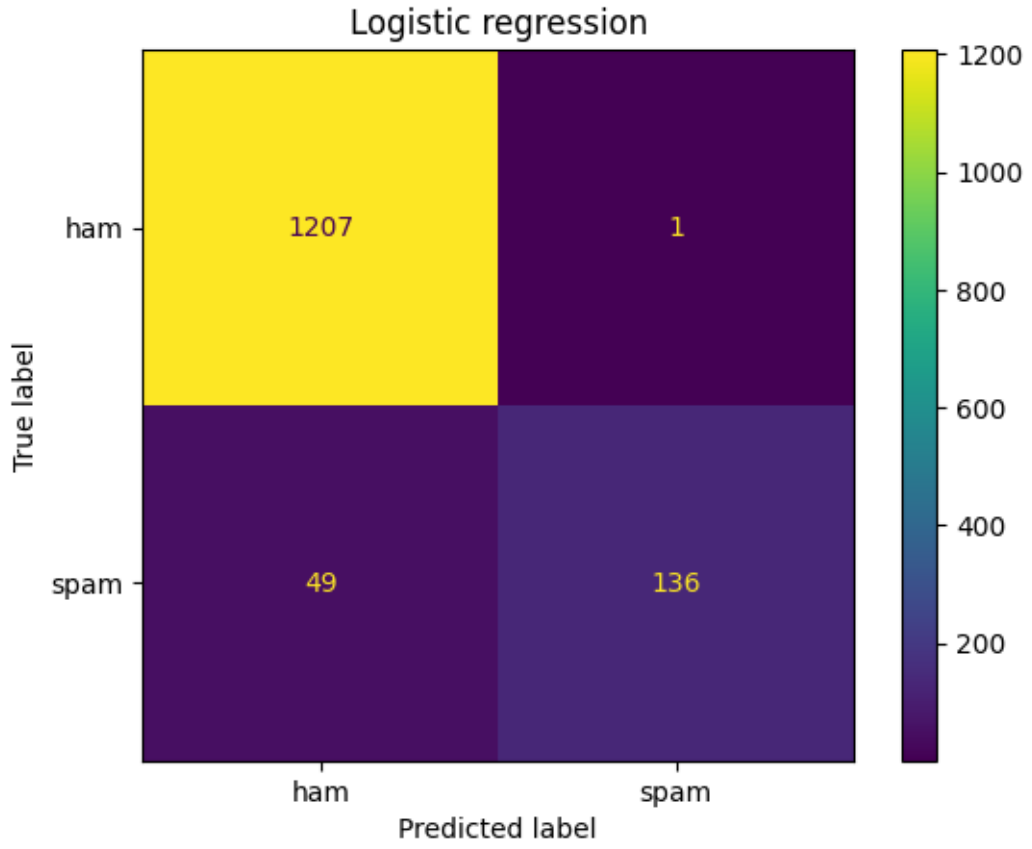



```
 Accuracy is 0.9806173725771715
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99      1208
        spam       0.99      0.86      0.92       185

    accuracy                           0.98      1393
   macro avg       0.98      0.93      0.96      1393
weighted avg       0.98      0.98      0.98      1393
```

[59]:
```
from sklearn.linear_model import LogisticRegression
model_lr = LogisticRegression(random_state=1)
```

```
model_lr.fit(x_train,y_train)
y_pred_lr = model_lr.predict(x_test)
```

[60]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred_lr)
plt.title('Logistic regression')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred_lr)}")
print(classification_report(y_test,y_pred_lr))
```




```
 Accuracy is 0.9641062455132807
              precision    recall  f1-score   support

         ham       0.96      1.00      0.98      1208
        spam       0.99      0.74      0.84       185

    accuracy                           0.96      1393
   macro avg       0.98      0.87      0.91      1393
weighted avg       0.97      0.96      0.96      1393
```

==========

Hyper parameter tunning

[61]:
```
from sklearn.model_selection import GridSearchCV
```

[66]:
```
params = {

    'criterion':['gini', 'entropy','log_loss'],
     'max_features': ['sqrt','log2'],
     'random_state': [0,1,2,3,4],
    'class_weight':['balanced','balanced_subsample']
}
```

[67]:
```
grid = GridSearchCV(rf, param_grid=params, cv=5, scoring='accuracy')
```

[68]:
```
grid.fit(x_train,y_train)
```

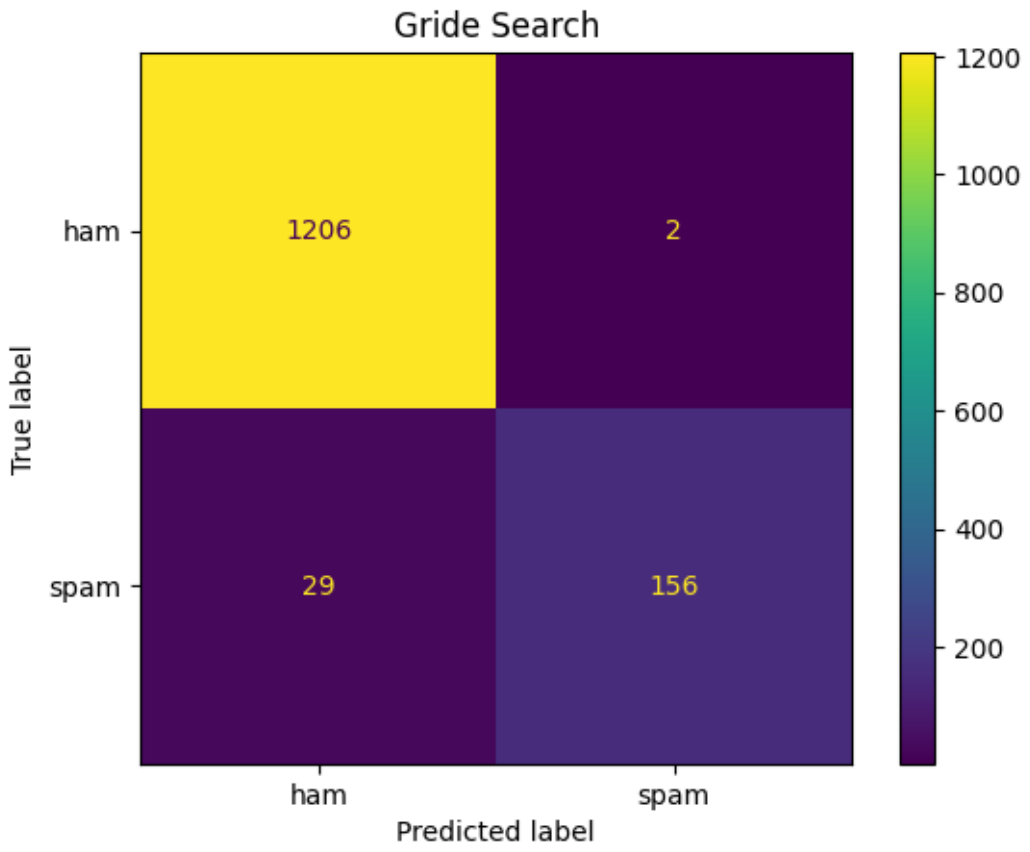
```
/usr/local/lib/python3.10/dist-packages/numpy/ma/core.py:2820: RuntimeWarning:
invalid value encountered in cast
  _data = np.array(data, dtype=dtype, copy=copy,
```

[68]:
```
GridSearchCV(cv=5, estimator=RandomForestClassifier(random_state=0),
             param_grid={'class_weight': ['balanced', 'balanced_subsample'],
                         'criterion': ['gini', 'entropy', 'log_loss'],
                         'max_features': ['sqrt', 'log2'],
                         'random_state': [0, 1, 2, 3, 4]},
             scoring='accuracy')
```

[70]:
```
rf = grid.best_estimator_
```

[71]:
```
y_pred = rf.predict(x_test)
```

[72]:
```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
plt.title('Gride Search')
plt.show()
print(f" Accuracy is {accuracy_score(y_test,y_pred)}")
print(classification_report(y_test,y_pred))
```




```
Accuracy is 0.9777458722182341
              precision    recall  f1-score   support

         ham       0.98      1.00      0.99      1208
        spam       0.99      0.84      0.91       185

    accuracy                           0.98      1393
   macro avg       0.98      0.92      0.95      1393
weighted avg       0.98      0.98      0.98      1393
```

[ ]: