In [1]:
```python
import numpy as np
import seaborn as sns
import pandas as pd
import matplotlib.pyplot as plt
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, classification_report, confusion_matrix, pre
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
```

In [4]:
```python
df = pd.read_csv(r"C:\Users\Simi\Downloads\email spam.csv",encoding='latin-1', engine =
```

In [5]:
```python
df.head()
```

Out[5]:

	v1	v2
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...

In [6]:
```python
df.tail()
```

Out[6]:

	v1	v2
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

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

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

In [8]:
```python
df.size
```

Out[8]:
```
11144
```

In [9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

In [10]:
```
df.columns = ["label", "message"]
```

In [11]:
```
df.head()
```

Out[11]:

	label	message
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...

In [12]:
```
df.isnull().values.any()
```

Out[12]:
```
False
```

In [13]:
```
df.isnull().sum()
```

Out[13]:
```
label      0
message    0
dtype: int64
```

In [14]:
```
df.duplicated().values.any()
```

Out[14]:
```
True
```

In [15]:
```
df.duplicated().sum()
```

Out[15]:
```
403
```

In [16]:
```
df.drop_duplicates(inplace=True)
```

In [17]:
```
df.describe()
```

Out[17]:

	label	message
count	5169	5169
unique	2	5169
top	ham	Go until jurong point, crazy.. Available only ...
freq	4516	1

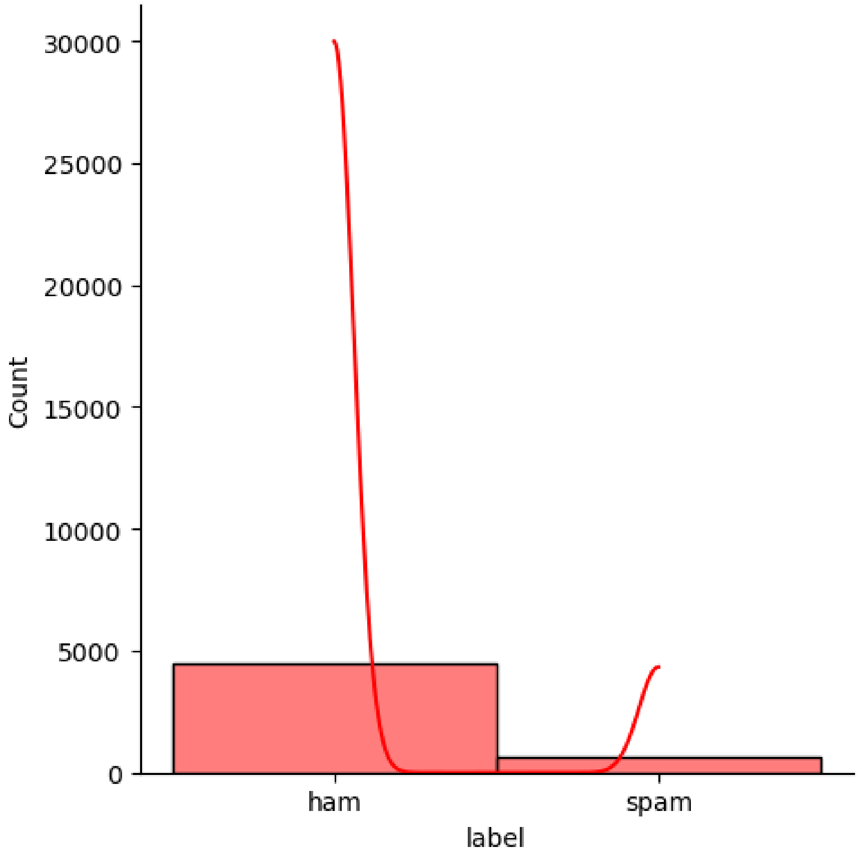
In [18]:

```
sns.displot(df.label, kde =True, color = "red")
```

```
C:\Users\Simi\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The fig
ure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[18]:

```
<seaborn.axisgrid.FacetGrid at 0x215bf03d590>
```



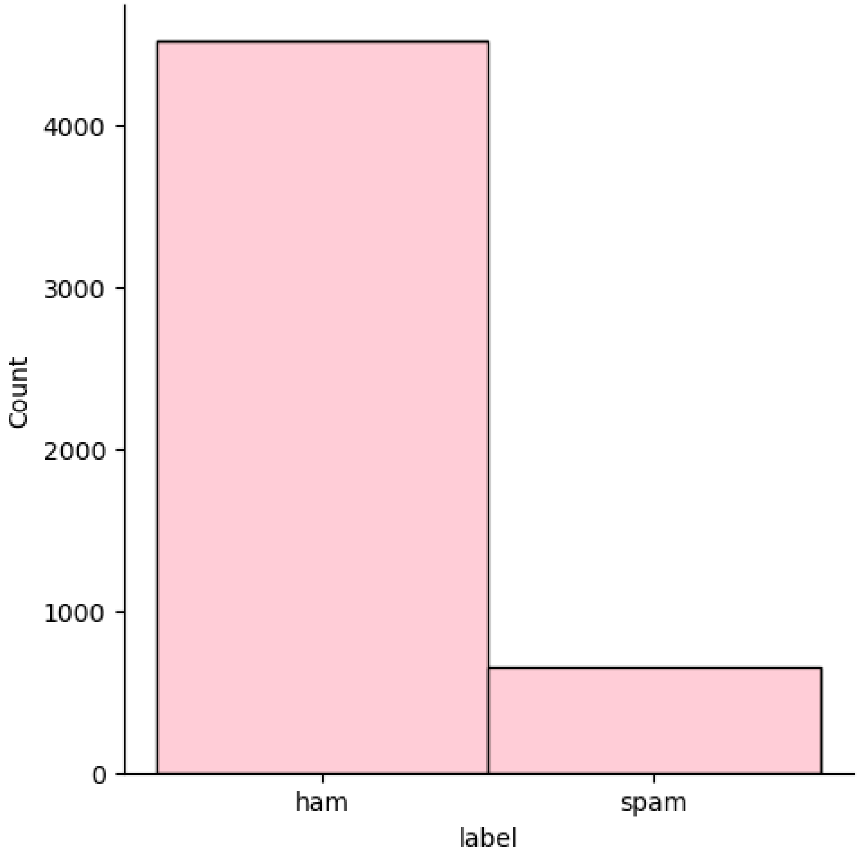
In [19]:

```
sns.displot(df.label, color = "pink")
```

```
C:\Users\Simi\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The fig
ure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x215bf0e80d0>
```



In [20]:

```
encoder = LabelEncoder()
df["label"] = encoder.fit_transform(df["label"].values)
```

In [21]:

```
vectorizer = TfidfVectorizer()
```

In [22]:

```
X = vectorizer.fit_transform(df["message"])
X.toarray()
```

Out[22]:

```
array([[0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [23]:

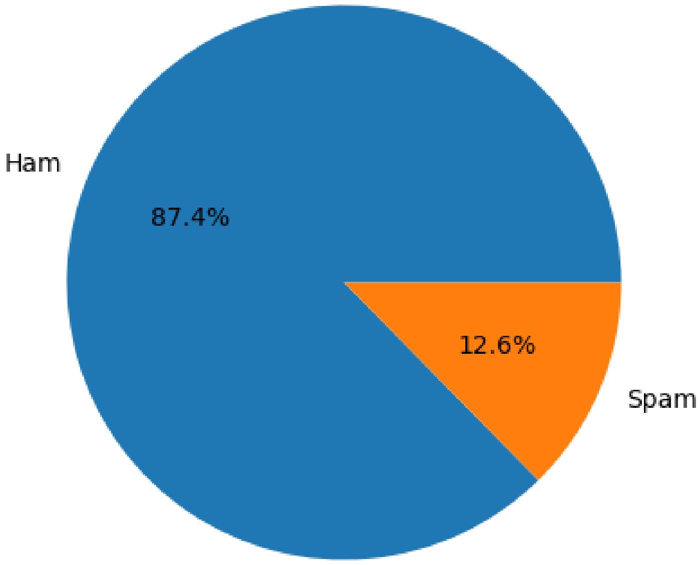
```
y = df["label"]
```

In [24]:

```
p, k = len(df[df["label"] == 0]), len(df[df["label"] == 1])
```

In [25]:

```
label = np.array(["Ham", "Spam"])
values = np.array([p, k])
plt.figure(figsize=(5, 5))
plt.pie(values, labels=label, autopct="%.1f%%")
plt.show()
```



In [26]:

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

In [27]:

```
naive_bayes_model = MultinomialNB()
```

In [28]:

```
naive_bayes_model.fit(X_train, y_train)
```

Out[28]:

```
▾ MultinomialNB
MultinomialNB()
```

In [29]:

```
nb_predictions = naive_bayes_model.predict(X_test)
```

In [30]:

```
print("Naive Bayes Model:")
print(confusion_matrix(y_test, nb_predictions))
print(classification_report(y_test, nb_predictions))
print("Accuracy: ", accuracy_score(y_test, nb_predictions))
print("r2_Score: ", r2_score(y_test, nb_predictions))
print("Precision_score: ", precision_score(y_test, nb_predictions))
print("Recall_score: ", recall_score(y_test, nb_predictions))
print("f1_score: ", f1_score(y_test, nb_predictions))
```

```
Naive Bayes Model:
[[889   0]
 [ 46  99]]
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       889
           1       1.00      0.68      0.81       145

    accuracy                           0.96      1034
   macro avg       0.98      0.84      0.89      1034
weighted avg       0.96      0.96      0.95      1034

Accuracy:  0.9555125725338491
r2_Score:  0.6310150886311625
Precision_score:  1.0
Recall_score:  0.6827586206896552
f1_score:  0.8114754098360656
```

In [31]:

```
logistic_regression_model = LogisticRegression()
```

In [32]:

```
logistic_regression_model.fit(X_train, y_train)
```

Out[32]:

```
▾ LogisticRegression
LogisticRegression()
```

In [33]:

```
lr_predictions = logistic_regression_model.predict(X_test)
```

In [34]:

```
print("Logistic Regression Model:")
print(confusion_matrix(y_test, lr_predictions))
print(classification_report(y_test, lr_predictions))
print("Accuracy: ", accuracy_score(y_test, lr_predictions))
print("r2_Score: ", r2_score(y_test, lr_predictions))
print("Precision_score: ", precision_score(y_test, lr_predictions))
print("Recall_score: ", recall_score(y_test, lr_predictions))
print("f1_score: ", f1_score(y_test, lr_predictions))
```

```
Logistic Regression Model:
[[886   3]
 [ 43 102]]
              precision    recall  f1-score   support

           0       0.95      1.00      0.97       889
           1       0.97      0.70      0.82       145

    accuracy                           0.96      1034
   macro avg       0.96      0.85      0.90      1034
weighted avg       0.96      0.96      0.95      1034

Accuracy:  0.9555125725338491
r2_Score:  0.6310150886311625
Precision_score:  0.9714285714285714
Recall_score:  0.7034482758620689
f1_score:  0.8160000000000001
```

In [35]:

```
svm_model = SVC(kernel='linear')
```

In [36]:

```
svm_model.fit(X_train, y_train)
```

Out[36]:

```
▾         SVC
SVC(kernel='linear')
```

In [37]:

```
svm_predictions = svm_model.predict(X_test)
```

In [38]:

```
print("Support Vector Machine (SVM) Model:")
print(confusion_matrix(y_test, svm_predictions))
print(classification_report(y_test, svm_predictions))
print("Accuracy: ", accuracy_score(y_test, svm_predictions))
print("r2_Score: ", r2_score(y_test, svm_predictions))
print("Precision_score: ", precision_score(y_test, svm_predictions))
print("Recall_score: ", recall_score(y_test, svm_predictions))
print("f1_score: ", f1_score(y_test, svm_predictions))
```

```
Support Vector Machine (SVM) Model:
[[886   3]
 [ 14 131]]
              precision    recall  f1-score   support

           0       0.98      1.00      0.99       889
           1       0.98      0.90      0.94       145

    accuracy                           0.98      1034
   macro avg       0.98      0.95      0.96      1034
weighted avg       0.98      0.98      0.98      1034

Accuracy:  0.9835589941972921
r2_Score:  0.8636360110158644
Precision_score:  0.9776119402985075
Recall_score:  0.903448275862069
f1_score:  0.939068100358423
```

In [ ]:

```
```