```
#Upload the dataset
```

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

Choose files dataset.csv.zip

dataset.csv.zip(application/x-zip-compressed) - 1118402 bytes, last modified: 21/10/2025 - 100% done

```
Saving dataset.csv.zip to dataset.csv.zip
```

```
#load the dataset
```

```
import pandas as pd
```

```
#Read the data
```

```
import pandas as pd
df=pd.read_csv('/content/dataset.csv', encoding='latin1')
df=df.iloc[:,:2]
df.columns=['label','message']
df.head()
df.info()
df['label'].value_counts()
```

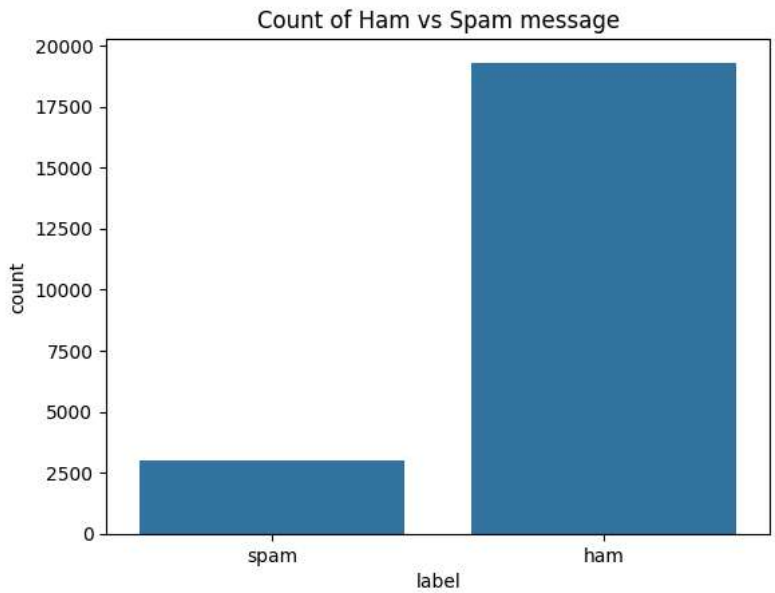
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22287 entries, 0 to 22286
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   label    22287 non-null  object
 1   message  22287 non-null  object
dtypes: object(2)
memory usage: 348.4+ KB
```

label	count
ham	19299
spam	2988

dtype: int64

```
Basic Distribution & Balance
```

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.countplot(x='label',data=df)
plt.title('Count of Ham vs Spam message')
plt.show()
```



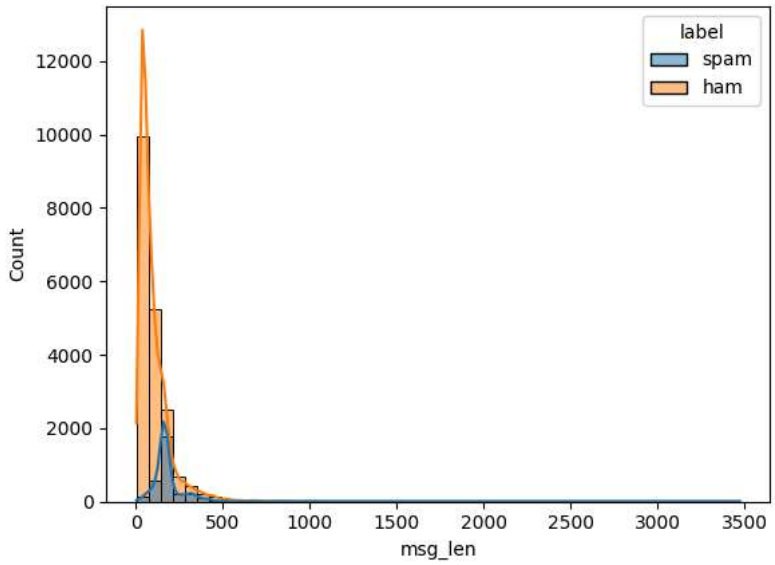
```
df['label'].value_counts(normalize=True)*100
```

label	proportion
ham	86.593081
spam	13.406919

dtype: float64

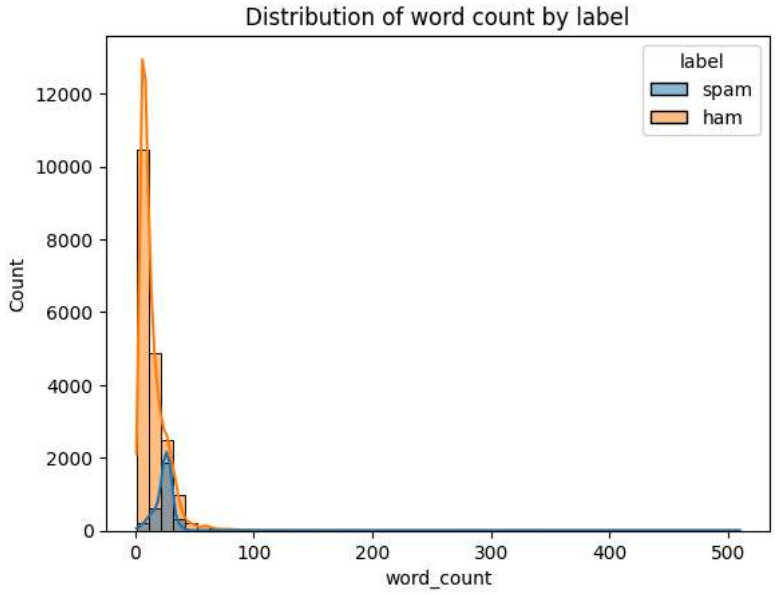
Message Length

```
df['msg_len']=df['message'].apply(len)
sns.histplot(data=df, x='msg_len', hue='label', kde=True, bins=50)
plt.show()
```



word count

```
df['word_count']=df['message'].apply(lambda x:len(x.split()))
sns.histplot(data=df, x='word_count', hue='label', kde=True, bins=50)
plt.title('Distribution of word count by label')
plt.show()
```



Text cleanning & pre-processing

```
import string
def clean_text(text):
```

```
    text = text.lower()
    text = text.translate(str.maketrans('', '', string.punctuation))
    return text

df['clean_msg'] = df['message'].apply(clean_text)
```

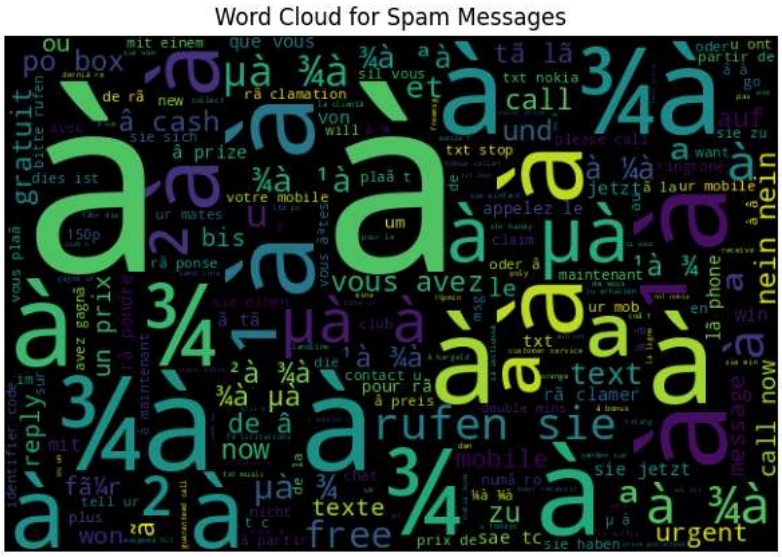
```
df['num_digits'] = df['message'].apply(lambda x: sum(c.isdigit() for c in x))
df['has_url'] = df['message'].apply(lambda x: 1 if 'http' in x or 'www' in x else 0)
```

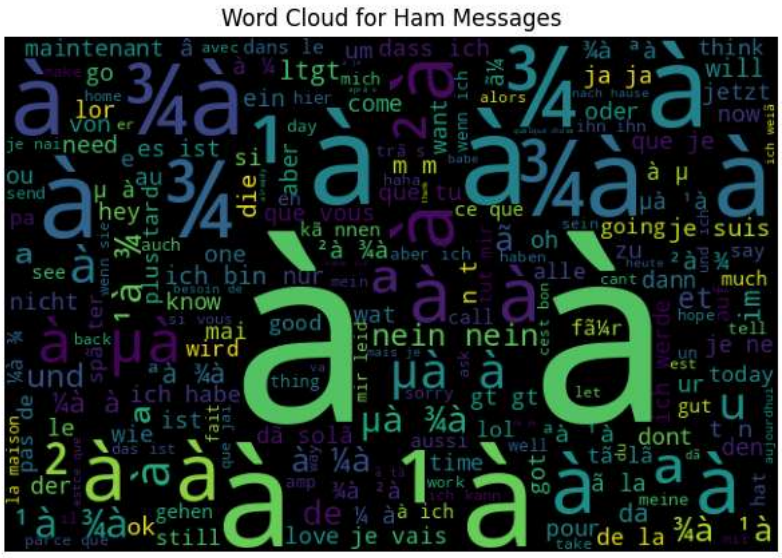
Word Frequency & visualization

```
from wordcloud import WordCloud

spam_text = ' '.join(df[df['label']=='spam']['clean_msg'])
ham_text  = ' '.join(df[df['label']=='ham']['clean_msg'])

wc_spam = WordCloud(width=600, height=400).generate(spam_text)
wc_ham  = WordCloud(width=600, height=400).generate(ham_text)

plt.figure(figsize=(10,5))
plt.imshow(wc_spam)
plt.axis('off')
plt.title('Word Cloud for Spam Messages')
plt.show()

plt.figure(figsize=(10,5))
plt.imshow(wc_ham)
plt.axis('off')
plt.title('Word Cloud for Ham Messages')
plt.show()
```

Most common words

```
from wordcloud import WordCloud

spam_text = ' '.join(df[df['label']=='spam']['clean_msg'])
ham_text  = ' '.join(df[df['label']=='ham']['clean_msg'])

wc_spam = WordCloud(width=600, height=400).generate(spam_text)
wc_ham  = WordCloud(width=600, height=400).generate(ham_text)

plt.figure(figsize=(10,5))
plt.imshow(wc_spam)
plt.axis('off')
plt.title('Word Cloud for Spam Messages')
plt.show()

plt.figure(figsize=(10,5))
plt.imshow(wc_ham)
plt.axis('off')
plt.title('Word Cloud for Ham Messages')
plt.show()
```





Comparative Statistics & Insights

```
df.groupby('label')[['msg_len','word_count','num_digits','has_url']].mean()
```

label	msg_len	word_count	num_digits	has_url
ham	98.956423	14.559355	1.406135	0.000674
spam	177.493307	24.150602	15.344712	0.129183

Check for Duplicates & Noise

```
df.duplicated().sum()
```

```
np.int64(1774)
```

Class Balance & Train-Test Split Strategy

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    df['clean_msg'], df['label'],
    test_size=0.2, stratify=df['label'], random_state=42)
```

Modelling & Baseline Performance

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.pipeline import Pipeline
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report, confusion_matrix

pipe = Pipeline([
    ('tfidf', TfidfVectorizer(ngram_range=(1,2))),
    ('nb',   MultinomialNB())
])
pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
print(classification_report(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         ham       0.93      1.00      0.96      3860
        spam       1.00      0.51      0.67       598

    accuracy                           0.93      4458
   macro avg       0.96      0.75      0.82      4458
weighted avg       0.94      0.93      0.92      4458

[[3860    0]
 [ 295  303]]
```