```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'email-spam-detection-dataset-classification:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-sets%2F1961542%2F32358

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
  os.symlink(KAGGLE_INPUT_PATH, os.path.join("..", 'input'), target_is_directory=True)
except FileExistsError:
  pass
try:
  os.symlink(KAGGLE_WORKING_PATH, os.path.join("..", 'working'), target_is_directory=True)
except FileExistsError:
  pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
              with ZipFile(tfile) as zfile:
                zfile.extractall(destination_path)
            else:
              with tarfile.open(tfile.name) as tarfile:
                tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

```
Downloading email-spam-detection-dataset-classification, 215935 bytes compressed
[==================================================] 215935 bytes downloaded
```

```
Downloaded and uncompressed: email-spam-detection-dataset-classification
Data source import complete.
```

```
import numpy as np
import pandas as pd
import nltk
```

```
sms=pd.read_csv('../input/email-spam-detection-dataset-classification/spam.csv', encoding='latin-1')
sms.head()
```

| | v1 | v2 | Unnamed: 2 | Unnamed: 3 | Unnamed: 4 |
|---|---|---|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... | NaN | NaN | NaN |
| 1 | ham | Ok lar... Joking wif u oni... | NaN | NaN | NaN |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... | NaN | NaN | NaN |
| 3 | ham | U dun say so early hor... U c already then say... | NaN | NaN | NaN |

Next steps: Generate code with sms  View recommended plots

```
sms=sms.drop(["Unnamed: 2","Unnamed: 3","Unnamed: 4"],axis=1)
sms=sms.rename(columns={"v1":"label","v2":"text"})
sms.head()
```

| | 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... |

Next steps: Generate code with sms  View recommended plots

```
print(" no of rows", len(sms))
```

```
 no of rows 5572
```

```
sms.label.value_counts()
```

```
ham     4825
spam     747
Name: label, dtype: int64
```

```
sms.describe()
```

| | label | text |
|---|---|---|
| count | 5572 | 5572 |
| unique | 2 | 5169 |
| top | ham | Sorry, I'll call later |
| freq | 4825 | 30 |

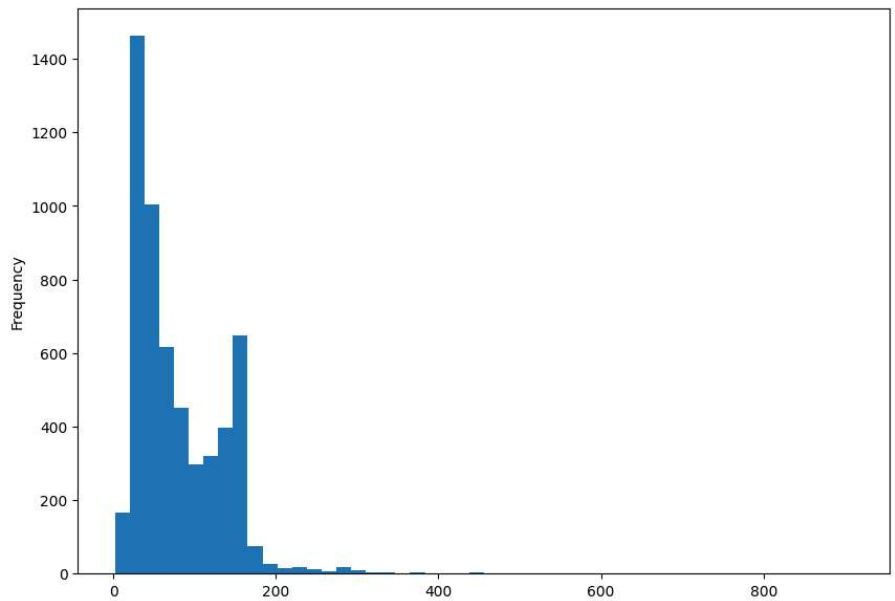
```
sms['length']=sms['text'].apply(len)
sms.head()
```

| | label | text | length |
|---|---|---|---|
| 0 | ham | Go until jurong point, crazy.. Available only ... | 111 |
| 1 | ham | Ok lar... Joking wif u oni... | 29 |
| 2 | spam | Free entry in 2 a wkly comp to win FA Cup fina... | 155 |
| 3 | ham | U dun say so early hor... U c already then say... | 49 |
| 4 | ham | Nah I don't think he goes to usf, he lives aro... | 61 |

Next steps: Generate code with sms  View recommended plots

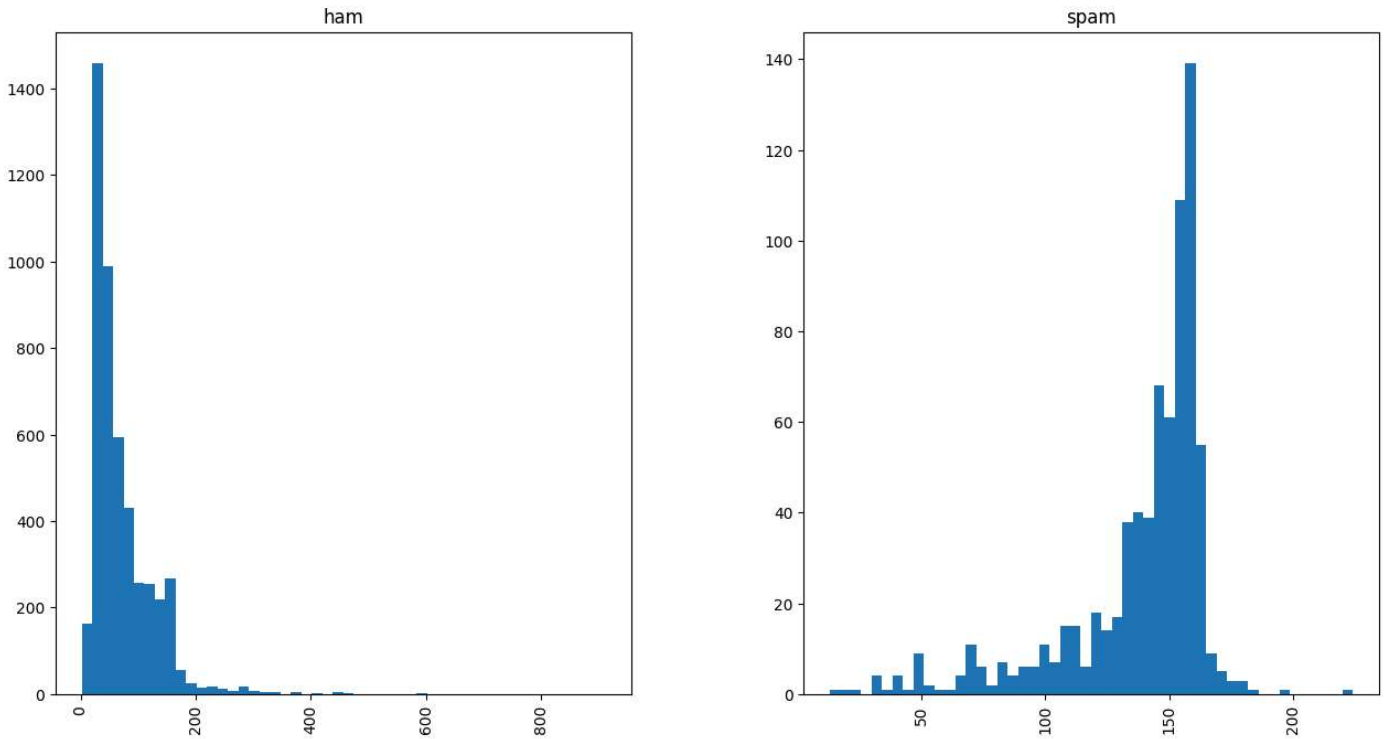
```
import matplotlib.pyplot as plt
import seaborn as sns

sms['length'].plot(bins=50, kind='hist',figsize=(10,7))
```

```
<Axes: ylabel='Frequency'>
```




```
sms.hist(column='length', by='label', bins=50, figsize=(15,8))
```

```
array([<Axes: title={'center': 'ham'}>, <Axes: title={'center': 'spam'}>],
      dtype=object)
```




```
sms.loc[:,'label']=sms.label.map({'ham':0, 'spam':1})
sms.head()
```

```
<ipython-input-11-2d476ed5f859>:1: DeprecationWarning: In a future version, `df.iloc[:, i] = newvals` will attempt to set the values inp
  sms.loc[:,'label']=sms.label.map({'ham':0, 'spam':1})
```

| | label | text | length |
|---|---|---|---|
| **0** | 0 | Go until jurong point, crazy.. Available only ... | 111 |
| **1** | 0 | Ok lar... Joking wif u oni... | 29 |
| **2** | 1 | Free entry in 2 a wkly comp to win FA Cup fina... | 155 |
| **3** | 0 | U dun say so early hor... U c already then say... | 49 |
| **4** | 0 | Nah I don't think he goes to usf, he lives aro... | 61 |

Next steps: Generate code with sms  View recommended plots

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split

count=CountVectorizer()
input=['REMINDER FROM O2: To get 2.50 pounds free call credit and details of great offers pls reply 2 this text with your valid name, house n

text=count.fit_transform(sms['text'])

x_train, x_test, y_train, y_test= train_test_split(text, sms['label'], test_size=0.20, random_state=1)
text
```

```
<5572x8672 sparse matrix of type '<class 'numpy.int64'>'
        with 73916 stored elements in Compressed Sparse Row format>
```

```
print(x_train.shape)
print(x_test.shape)

input=text[5571]

from sklearn.neural_network import MLPClassifier

model=MLPClassifier()
model.fit(x_train, y_train)
```

```
▾ MLPClassifier
MLPClassifier()
```

```
prediction=model.predict(x_test)
print(prediction)
```

```
[0 0 0 ... 0 0 0]
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print("Multinomial NB")
print("Accuracy score: {}". format(accuracy_score(y_test, prediction)) )
print("Precision score: {}". format(precision_score(y_test, prediction)) )
print("Recall score: {}". format(recall_score(y_test, prediction)))
print("F1 score: {}". format(f1_score(y_test, prediction)))
```

```
Multinomial NB
Accuracy score: 0.9919282511210762
Precision score: 1.0
Recall score: 0.935251798561151
F1 score: 0.966542750929368
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print("Bernoulli NB")
print("Accuracy score: {}". format(accuracy_score(y_test, prediction)) )
print("Precision score: {}". format(precision_score(y_test, prediction)) )
print("Recall score: {}". format(recall_score(y_test, prediction)))
print("F1 score: {}". format(f1_score(y_test, prediction)))
```

```
Bernoulli NB
Accuracy score: 0.9919282511210762
Precision score: 1.0
Recall score: 0.935251798561151
F1 score: 0.966542750929368
```

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

print("MLP Classifier")
print("Accuracy score: {}". format(accuracy_score(y_test, prediction)) )
print("Precision score: {}". format(precision_score(y_test, prediction)) )
```