```
import json
import csv
import pandas as pd
from urllib.request import urlretrieve

# Downloading dataset from url
url = (
    "https://gist.githubusercontent.com/"
    "farhaan-settyl/ecf9c1e7ab7374f18e4400b7a3d2a161/"
    "raw/f94652f217eeca83e36dab9d08727caf79ebdecf/dataset.json"
    )
filename = "dataset.json"
urlretrieve(url, filename)

# Convert JSON file to CSV
def json_to_csv(json_file, csv_file):
    with open(json_file, 'r') as f:
        data = json.load(f)

    # Assuming JSON data is a list of dictionaries
    keys = data[0].keys() if data else []

    # write into a CSV file.
    with open(csv_file, 'w', newline='') as f:
        writer = csv.DictWriter(f, fieldnames=keys)
        writer.writeheader()
        writer.writerows(data)

json_path = "./dataset.json"
csv_path = "./dataset.csv"

# Example usage
json_to_csv(json_path, csv_path)

# Read CSV file into pandas DataFrame
df = pd.read_csv(csv_path)
```

```
# Get column names from the DataFrame
column_names = df.columns.tolist()

# Print the column names
print(column_names)
```

```
['externalStatus', 'internalStatus']
```

```
# Extract unique classes from the "internalStatus" column
unique_classes = df["internalStatus"].unique()

# Print the unique classes
for cl in unique_classes:
  print(cl)

print(len(unique_classes))
```

```
Port Out
Inbound Terminal
Port In
Departure
Arrival
Gate In
Loaded on Vessel
Gate Out
On Rail
Off Rail
Empty Return
In-transit
Outbound Terminal
Empty Container Released
Unloaded on Vessel
15
```

```
import pandas as pd
import nltk
from nltk.corpus import stopwords
nltk.download('stopwords')

# Read the CSV file into a Pandas DataFrame
df = pd.read_csv("/content/dataset.csv")

# Select the columns with text data
text_columns = ["externalStatus", "internalStatus"]

# Remove rows with missing values in the selected columns
df = df.dropna(subset=text_columns)

# Convert all text data to lowercase
for column in text_columns:
  df[column] = df[column].str.lower()

# Remove punctuation from the text data
for column in text_columns:
  df[column] = df[column].str.replace('[^\w\s]', '')

# Remove stop words from the text data
stop_words = set(stopwords.words('english'))
for column in text_columns:
  df[column] = df[column].apply(
      lambda x: ' '.join([word for word in x.split() if word not in stop_words])
      )

# Print the preprocessed data
print(df)
```

```
                                        externalStatus  \
0                                                 port
1                                             terminal
2                                                 port
3     vessel departure first pol (vessel name : tian...
4     vessel arrival final pod (vessel name : tian f...
...                                                 ...
1217                               import loaded rail
1218                        full transshipment loaded
1219                        full transshipment loaded
1220                             export loaded vessel
1221                                     empty shipper

                internalStatus
0                         port
1              inbound terminal
2                         port
3                    departure
4                      arrival
...                        ...
1217             loaded vessel
1218             loaded vessel
1219             loaded vessel
1220             loaded vessel
1221  empty container released

[1222 rows x 2 columns]
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
from sklearn.feature_extraction.text import CountVectorizer

# Create a CountVectorizer object
vectorizer = CountVectorizer()

# Fit the vectorizer to the
# "externalStatus" column
vectorizer.fit(df["externalStatus"])

# Transform the "externalStatus" column
# into a bag-of-words representation
external_status_bow = vectorizer.transform(df["externalStatus"])

# Print the bag-of-words representation
print(external_status_bow)
```

```
  (0, 106)      1
  (1, 121)      1
  (2, 106)      1
  (3, 48)       1
  (3, 62)       1
  (3, 63)       1
```

```
(3, 70)       1
(3, 90)       1
(3, 105)      1
(3, 122)      1
(3, 134)      2
(4, 30)       1
(4, 61)       1
(4, 63)       1
(4, 70)       1
(4, 90)       1
(4, 104)      1
(4, 122)      1
(4, 134)      2
(5, 47)       1
(6, 66)       1
(7, 11)       1
(7, 80)       1
(7, 83)       1
(7, 115)      1
:     :
(1213, 64)    1
(1213, 81)    1
(1213, 127)   1
(1214, 60)    1
(1214, 81)    1
(1214, 134)   1
(1215, 56)    1
(1215, 116)   1
(1216, 73)    1
(1216, 109)   1
(1216, 129)   1
(1217, 73)    1
(1217, 81)    1
(1217, 109)   1
(1218, 64)    1
(1218, 81)    1
(1218, 127)   1
(1219, 64)    1
(1219, 81)    1
(1219, 127)   1
(1220, 60)    1
(1220, 81)    1
(1220, 134)   1
(1221, 56)    1
(1221, 116)   1
```

```
tsize = 0.2
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier as rfclass
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    external_status_bow,
    df["internalStatus"],
    test_size=tsize
    )

# Create a Random Forest model
model1 = rfclass(n_estimators=10)

# Train the model on the training data
model1.fit(X_train.toarray(), y_train)

# Predict the labels for the test data
y_pred = model1.predict(X_test.toarray())

# Calculate the accuracy of the model
accuracy1 = accuracy_score(y_test, y_pred)

# Print the accuracy
print(accuracy1)
```

```
0.9918367346938776
```

```
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    external_status_bow,
    df["internalStatus"],
    test_size=tsize
    )

# Create a Gaussian Naive Bayes model
model2 = GaussianNB()

# Train the model on the training data
model2.fit(X_train.toarray(), y_train)

# Predict the labels for the test data
y_pred = model2.predict(X_test.toarray())

from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier as dtclass
from sklearn.metrics import accuracy_score

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(
    external_status_bow,
    df["internalStatus"],
    test_size=tsize
    )

# Create a Decision Tree Model
model3 = dtclass(criterion="log_loss")

# Train the model on the training data
model3.fit(X_train.toarray(), y_train)

# Predict the labels for the test data
y_pred = model3.predict(X_test.toarray())

# Calculate the accuracy of the model
accuracy3 = accuracy_score(y_test, y_pred)

# Print the accuracy
print(accuracy3)

import graphviz
dot_data = tree.export_graphviz(model3, out_file=None)
graph = graphviz.Source(dot_data)
graph.render("port")
```

```
0.9877551020408163
'port.pdf'
```