Only for Reference and basic understanding of flow

The breast cancer dataset is available in the sklearn.datasets module and can be loaded using load_breast_cancer(). implement a Naive Bayes Classifier on the Breast Cancer Dataset using python's sklearn library. Assume that all features of datasets are continuous variables and must be used for building model. Perform following task. 1. load data set 2. Split data for training and testing 3. Build a model 4. Predict labels of test data

```
from sklearn.datasets import load_breast_cancer

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score


# Load the dataset

data = load_breast_cancer()

X = data.data

y = data.target


# Split the data

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


# Build the model

model = GaussianNB()

model.fit(X_train, y_train)


# Predict labels for the test set

y_pred = model.predict(X_test)


# Evaluate the model

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

Implement a Gaussian Naive Bayes classifier to predict whether a patient has diabetes based on various health metrics of PIMA.csv dataset . The dataset consists of information from 768 female Pima Indians aged 21 and older, initially gathered by the National Institute of Diabetes and Digestive and Kidney Diseases. Target variable: Diabetes (binary, 0 or 1) Attributes: Pregnancies, OGTT (Oral Glucose Tolerance Test), Blood pressure, Skin thickness, Insulin, BMI (Body Mass Index), Age, Pedigree diabetes function. 1. Load the Dataset from a CSV file. Provide summary statistics for the dataset. 2. Split the data into training and testing sets (80% train, 20% test). 3. Instantiate a Gaussian Naive Bayes model and fit it on the training data. 4. Predict diabetes status for the test set. Discuss any biases in the dataset and how they may affect model performance.

```
import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import GaussianNB

from sklearn.metrics import accuracy_score


# Load the dataset

data = pd.read_csv('PIMA.csv')  # Ensure PIMA.csv is in your working directory


# Display summary statistics

print(data.describe())


# Define features and target

X = data.drop(columns=['Diabetes'])

y = data['Diabetes']


# Split data

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


# Build the model

model = GaussianNB()

model.fit(X_train, y_train)


# Predict and evaluate
```

```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

Build a Naive Bayes classifier to analyze the sentiment of movie reviews (positive or negative). The IMDB_Dataset.csv typically contains two main columns: Review: The text of the movie review. Sentiment: A label indicating the sentiment of the review, usually categorized as 'positive' or 'negative'. Load the dataset and inspect the structure. Clean the text data by removing punctuation, converting to lowercase, and tokenizing the reviews.Use CountVectorizer or TfidfVectorizer to convert the text data into a numerical format suitable for model training. Split the dataset and Create a Multinomial Naive Bayes model and fit it on the training data. Predict the sentiment of the test reviews.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score

# Load dataset
data = pd.read_csv('IMDB_Dataset.csv')

# Preprocess text data
data['Review'] = data['Review'].str.replace('[^\w\s]', '').str.lower()

# Convert text to numerical format
vectorizer = CountVectorizer(stop_words='english')
X = vectorizer.fit_transform(data['Review'])
y = data['Sentiment'].map({'positive': 1, 'negative': 0})

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build and evaluate the model
```

```
model = MultinomialNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

Use sklearn.datasets.fetch_20newsgroups() dataset (all sets) to classify newsgroup documents into their respective categories using a Multinomial Naive Bayes classifier. Load the dataset and explore the categories available. Preprocess the text data using CountVectorizer to convert text documents into a matrix of token counts. Split the dataset and Create a Multinomial Naive Bayes model. Fit the model on the training data and print accuracy.

```
from sklearn.datasets import fetch_20newsgroups

from sklearn.model_selection import train_test_split

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score


# Load dataset

newsgroups = fetch_20newsgroups(subset='all')

X = newsgroups.data

y = newsgroups.target


# Convert text to numerical format

vectorizer = CountVectorizer(stop_words='english')

X_vec = vectorizer.fit_transform(X)


# Split data

X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2, random_state=42)


# Build and evaluate the model
```

```
model = MultinomialNB()

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```