Implementation of 3 Naive Bayesian model with with preprocessing on Mnist dataest

I have installed last vertion of acaconda since 1/11/2023 in base(root) enviroment saand launch Jupyter notebook I have called some libraries, various Naive Bayes models, and the Python Textual Analysis Library (PTL) to perform data preprocessing.

In [ ]:
```
from sklearn.naive_bayes import MultinomialNB,GaussianNB,BernoulliNB
from sklearn.metrics import accuracy_score
import os
import pandas as pd
from PIL import Image
import numpy as np
import pandas as pd
```

Here code get the directory and then

- 1.It resizes the images to the desired size (in this case, 13x13 pixels).
- 2.Converts them to grayscale.
- 3.Flattens the pixel values into a one-dimensional array for each image.
- 4.Stores these flattened arrays in the image_data list.
- 5.Converts the list of image arrays to a NumPy array.
- 6.Creates a pandas DataFrame from the NumPy array and saves it as a CSV file.
- and I have commented any of this step in the beginning of their code in addition I have used LANCZOS but there is orther filter like ANTIALIAS so I perfer it because i got an error with ANTIALIAS

In [ ]:
```
input_dir = "D:\\PyPo\\Term1\\Pattern\\Naive-Bayesian\\trainsetB"
output_csv_file = "output.csv"

# Create empty lists to store image data
images = []

for filename in os.listdir(input_dir):
    if filename.endswith(".jpg"):
        img = Image.open(os.path.join(input_dir, filename))
        img = img.convert("L")  # Convert to grayscale

        # Resize the image to the desired size
        target_size = (13, 13)
        img = img.resize(target_size, Image.Resampling.LANCZOS)  # Use LANCZOS resa

        # Flatten the pixel values and append to the list
        img_array = np.array(img).flatten()
        images.append(img_array)

# Convert the list of image arrays to a NumPy array
image_data = np.array(images)
```

```
# Create a DataFrame and save it as a CSV file
df = pd.DataFrame(image_data)
df.to_csv(output_csv_file, index=False)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_22108\3148738826.py:14: DeprecationWarn
ing: LANCZOS is deprecated and will be removed in Pillow 10 (2023-07-01). Use Resamp
ling.LANCZOS instead.
  img = img.resize(target_size, Image.LANCZOS)  # Use LANCZOS resampling filter
```

This code is also like previous one but for test data set

In [ ]:
```
input_dir = "D:\\PyPo\\Term1\\Pattern\\Naive-Bayesian\\testsetB"
output_csv_file = "outputtest.csv"

# Create empty lists to store image data
images = []

for filename in os.listdir(input_dir):
    if filename.endswith(".jpg"):
        img = Image.open(os.path.join(input_dir, filename))
        img = img.convert("L")  # Convert to grayscale

        # Resize the image to the desired size
        target_size = (13, 13)
        img = img.resize(target_size, Image.LANCZOS)  # Use LANCZOS resampling filt

        # Flatten the pixel values and append to the list
        img_array = np.array(img).flatten()
        images.append(img_array)

# Convert the list of image arrays to a NumPy array
image_data = np.array(images)

# Create a DataFrame and save it as a CSV file
df = pd.DataFrame(image_data)
df.to_csv(output_csv_file, index=False)
```

```
C:\Users\Lenovo\AppData\Local\Temp\ipykernel_22108\2371326841.py:14: DeprecationWarn
ing: LANCZOS is deprecated and will be removed in Pillow 10 (2023-07-01). Use Resamp
ling.LANCZOS instead.
  img = img.resize(target_size, Image.LANCZOS)  # Use LANCZOS resampling filter
```

Here I got working directory for next step

In [ ]:
```
working_directory=os.getcwd()
print (working_directory)
```

```
d:\PyPo\Term1\Pattern\Naive-Bayesian
```

In this step code read train and test data set whit cav format

In [ ]:
```
train=working_directory + '\output.csv'
test=working_directory + '\outputtest.csv'
```

```
train_data=pd.read_csv(train)
test_data=pd.read_csv(test)
```

Here code extracts the feature vectors and labels for training and testing from data stored in DataFrames. and displays the first few rows of the training data

In [ ]:
```
X_train = train_data.iloc[:, 1:].values
y_train = train_data.iloc[:, 0].values

X_test = test_data.iloc[:, 1:].values
train_data.head()
```

Out[ ]:

	0	1	2	3	4	5	6	7	8	9	...	159	160	161	162	163	164	165	166	167	168
0	0	0	0	1	5	5	6	0	3	7	...	2	2	0	0	0	0	0	0	0	0
1	0	0	0	2	3	0	0	0	3	5	...	0	0	0	0	3	2	0	0	0	0
2	0	0	0	2	4	6	6	4	3	3	...	1	4	4	18	1	0	0	0	0	0
3	0	0	0	1	3	3	3	2	2	1	...	0	0	0	0	8	28	7	2	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	1	0	7	204	63	0	4	1	0

5 rows × 169 columns

This part is a common data preprocessing step in machine learning, which is known as data normalization or scaling to improve the performance and convergence of some learning algorithm

In [ ]:
```
X_train = X_train / 255.0
X_test = X_test / 255.0
```

Here I creat 3 model of Naive Bayes models to be trained on dataset

In [ ]:
```
model_1 = MultinomialNB()
model_2=GaussianNB()
model_3=BernoulliNB()
```

So here I models are done be trianed

In [ ]:
```
model_1.fit(X_train, y_train)
model_2.fit(X_train, y_train)
model_3.fit(X_train, y_train)
```

Out[ ]:
```
▾ BernoulliNB
BernoulliNB()
```

Here for any moled a predict will be provided

In [ ]:
```
y_pred_1 = model_1.predict(X_test)
y_pred_2 = model_2.predict(X_test)
y_pred_3 = model_3.predict(X_test)
```

Here I test prediction of treaned model and reach to accuracy if them

In [ ]:
```
y_test = test_data.iloc[:, 0].values
accuracy_1 = accuracy_score(y_test, y_pred_1)
accuracy_2 = accuracy_score(y_test, y_pred_2)
accuracy_3 = accuracy_score(y_test, y_pred_3)
print(f'Accuracy of the MultinomialNB : {accuracy_1 * 100:.2f}%')
print(f'Accuracy of the GaussianNB    : {accuracy_2 * 100:.2f}%')
print(f'Accuracy of the BernoulliNB: {accuracy_3 * 100:.2f}%')
```

```
Accuracy of the MultinomialNB : 88.21%
Accuracy of the GaussianNB    : 78.65%
Accuracy of the BernoulliNB: 93.37%
```