6612_Assignment1

September 18, 2024

CSCI6612 - Visual Analytics

Assignment 1

Fall 2024

[ ]:

[ ]:

We will be using Crowdmark this term for submitting and marking assignments. Crowdmark allows markers to provide specific comments in place on the submitted documents, leading to better quality feedback.

1. You will **receive an email** from the Crowdmark mailer with a button "Go to assignment" and a link to use if the button does not work for you.
2. You need to submit your answer to **each question separately**.
3. Your answer to a math question should be as **Image/PDF file**. It can be a photo of a handwritten solution or a pdf file of a typeset solution. The choice is up to you. The format will not affect your mark, as long as handwriting is neat and very readable. Please note that your submitted image **should not be rotated**. Rotaded images will not be marked, but they will automatically receive a zero.
4. Your answer to a coding/experimentation question will be **in two parts** (i.e. two questions). The first part is an **Image/PDF answer**, where you upload the PDF printout of your python notebook (pdf file). The second part is a text answer question, to which you attach your python notebook (ipynb file). Please do not zip any of the files you upload to Crowdmark.

0.1 Submitting your solution as a group

You are allowed to work on the assignment individually or as a group of 2 students. Here are the crowdmark rules for a group submition:

Once the assignment has been distributed, students will receive an email with a link to view the assignment.

- Students will be able to add or edit group members before submitting.
- Students can only add members who have not already been added to another group. Once a student is added to a group, the submission page is shared among the group members and any member can submit work on behalf of the group.

- Students will receive an email notification that they have been added to a group by a particular user.
- Students also have access to the submission log in their portflio. The log includes the addition or removal of group members and submission activity for each individual user.
- Once a group assignment has been submitted, students will not be able to edit the group members. Students will need to contact their instructor to make any changes to group members after submission. Note: if students have submitted individually, moving students to a group will clear their individual submissions.

0.2 Instructions for generating a PDF printout of a Jupyter notebook.

There are 3 ways to save a Jupyter Notebook as a PDF file: 1. Using Jupyter Notebook 2. Using `nbconvert` Command-Line Tool. 3. Using print to a PDF file in the browser (**DON'T USE**)

1. **Using Jupyter Notebook**
    1. Install Jupyter Notebook. If you install Anaconda, Jupyter Notebook will be included in the Anaconda Navigator.
    2. Open the notebook in Jupyter.
    3. Select "File" from the menu bar, then "Download as", then "PDF via LaTeX". See the image below.
    4. The notebook will be converted to a PDF file and downloaded to your computer.

2. **Using `nbconvert` Command-Line Tool**
    1. Install nbconvert.
    2. Open the terminal and navigate to the folder containing the notebook.
    3. Enter the following command: `jupyter nbconvert --to pdf <notebook filename>`.
    4. The notebook will be converted to a PDF file and saved in the same folder.
3. **Using print to a PDF file in the browser (DON'T USE)**
    - The produced PDFs can't be parsed and hard to mark, so please don't use this method. If for any reasons you have a hard time with producing a PDF don't hasitate to contact the TA.

Note: If you don't want to install any software localy, you can use a Dal FCS-hosted Jupyter Web Interface Service: https://timberlea.cs.dal.ca:8000/. Should you run into any errors, please ask the CS HelpDesk for help.

Note: in solving the math questions, aim for general (symbolic) solutions and substitute the specific numbers at the end. This demonstrates a solid understanding of the key concepts. You can answer the math questions in two ways: * **Use LaTeX to typeset the equations**. Section H of this LaTeX reference sheet is a good reference. Here is another LaTeX reference sheet. The equations in the questions are typeset in LaTeX, so you can use them as examples. * **Use neat handwriting**, scan your solution using AdobeScan, or Dropbox on your mobile phone, and upload the image file to the corresponding Crowdmark Image/PDF question.

Your answers to the experimental questions should be in a solution notebook, in the form of code and text cells, using markdown for your text responses. **You should also include in the**

notebook the results of running your code. This means that you must not clear the output produced by your program.

The marking criteria are described in rubrics in the syllabus.

You can submit multiple editions of your assignment. Only the last one will be marked. It is recommended to upload a complete submission, even if you are still improving it, so that you have something into the system if your computer fails for whatever reason.

IMPORTANT: PLEASE NAME YOUR PYTHON NOTEBOOK FILE AS: * `<LAST_NAME>-<FIRST_NAME>_<PARTNER_LAST_NAME>-<PARTNER_FIRST_NAME>-Assignment-N.ipynb`

for example: **Milios-Evangelos-Assignment-1.ipynb** \

0.3 QUESTION 1

In this question, you will experiment with classification algorithms from Sklearn API on the 20 newsgroups dataset. Make use of code examples available in the sklearn documentation, and clearly reference the URLs of the classes / methods you reused. Use the default parameters of the methods you use where possible.

1. Load the dataset.

[1]:
```
from sklearn.datasets import fetch_20newsgroups_vectorized

newsgroups_data = fetch_20newsgroups_vectorized()

print(f"Top 5 rows: {newsgroups_data.data[:5]}")
```

```
Top 5 rows:   (0, 5022) 0.017109647770728872
  (0, 5886)     0.017109647770728872
  (0, 6214)     0.017109647770728872
  (0, 6216)     0.017109647770728872
  (0, 6281)     0.017109647770728872
  (0, 6286)     0.017109647770728872
  (0, 6324)     0.017109647770728872
  (0, 6331)     0.017109647770728872
  (0, 6403)     0.017109647770728872
  (0, 11391)    0.017109647770728872
  (0, 13930)    0.017109647770728872
  (0, 15094)    0.017109647770728872
  (0, 15251)    0.017109647770728872
  (0, 15530)    0.017109647770728872
  (0, 16731)    0.017109647770728872
  (0, 20228)    0.017109647770728872
  (0, 26214)    0.017109647770728872
  (0, 26806)    0.017109647770728872
  (0, 27436)    0.017109647770728872
  (0, 27618)    0.017109647770728872
  (0, 27645)    0.017109647770728872
  (0, 27901)    0.017109647770728872
```

```
  (0, 28012)    0.05132894331218662
  (0, 28146)    0.41063154649749295
  (0, 28421)    0.034219295541457743
  :     :
  (4, 111322)   0.06468462273531508
  (4, 112421)   0.06468462273531508
  (4, 114440)   0.19405386820594528
  (4, 114455)   0.06468462273531508
  (4, 114520)   0.06468462273531508
  (4, 114579)   0.06468462273531508
  (4, 114625)   0.06468462273531508
  (4, 114646)   0.12936924547063017
  (4, 114692)   0.06468462273531508
  (4, 114731)   0.06468462273531508
  (4, 115475)   0.12936924547063017
  (4, 116636)   0.06468462273531508
  (4, 116665)   0.06468462273531508
  (4, 116722)   0.06468462273531508
  (4, 118280)   0.06468462273531508
  (4, 119714)   0.06468462273531508
  (4, 123292)   0.06468462273531508
  (4, 123796)   0.06468462273531508
  (4, 124026)   0.06468462273531508
  (4, 124046)   0.19405386820594528
  (4, 124616)   0.12936924547063017
  (4, 124946)   0.06468462273531508
  (4, 125074)   0.06468462273531508
  (4, 128402)   0.32342311367657545
  (4, 128420)   0.06468462273531508
```

Explain your code here.

This code loads the pre-processed version of the 20 Newsgroups dataset from sklearn.datasets library, where the raw text is transformed into numerical features using the TF-IDF vectorization. This allows the data to be readily utilized for machine learning algorithms without requiring further preprocessing. By printing the dataset, I am ensuring that the dataset has been correctly loaded and getting a preliminary understanding of its structure.

2. Split the data set into a training set and a test set. You may find this utility helpful.

[2]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(newsgroups_data.data,␣
 ↪newsgroups_data.target, test_size=0.2, random_state=42)

print(f"Training data shape: {X_train.shape}")
print(f"Test data shape: {X_test.shape}")
print(f"Training target shape: {y_train.shape}")
print(f"Test target shape: {y_test.shape}")
```

```
Training data shape: (9051, 130107)
Test data shape: (2263, 130107)
Training target shape: (9051,)
Test target shape: (2263,)
```

Explain your code here.

In this code, I first import the train_test_split function from sklearn.model_selection to split the dataset into training and testing subsets. Then I use this function to divide the dataset, with 80% allocated for training and 20% for testing, specified by the test_size=0.2 argument. By setting random_state=42, I ensure that the data is split consistently every time the code is run, making the results reproducible. After splitting, I print the shapes of the training and testing datasets, both for the features (X_train and X_test) and their corresponding target labels (y_train and y_test), to confirm that the split has been performed correctly and that the data is ready for further processing in the machine learning pipeline.

3. Train a AdaBoost Classifier with the training test, and estimate its accuracy using the test set.

[3]:
```python
from sklearn.ensemble import AdaBoostClassifier
from sklearn.metrics import accuracy_score

clf = AdaBoostClassifier()

clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy * 100:.2f}%")
```

```
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(

Accuracy: 54.62%
```

Explain your code here.

In this code, I first import the AdaBoostClassifier from sklearn.ensemble and accuracy_score from sklearn.metrics. I then initialize the AdaBoost classifier and train it using the training data with clf.fit(). After training, I use the classifier to make predictions on the test set with clf.predict(). To evaluate how well the classifier performed, I calculate the accuracy of these predictions using accuracy_score() and print the result as a percentage. This gives a measure of how accurately the AdaBoost classifier can predict the categories of the new data.

4. Generate ten different classifiers using ten different splits. Compute the mean and standard deviation of the ten different values you obtain for accuracy. Optionally, you can use the KFold generator (intended for cross validation).

[4]:
```
import numpy as np

accuracies = []

for i in range(10):
    X_train, X_test, y_train, y_test = train_test_split(newsgroups_data.data,␣
 ↪newsgroups_data.target, test_size=0.2)

    clf = AdaBoostClassifier()

    clf.fit(X_train, y_train)

    y_pred = clf.predict(X_test)

    accuracy = accuracy_score(y_test, y_pred)

    accuracies.append(accuracy)

mean_accuracy = np.mean(accuracies)
std_accuracy = np.std(accuracies)

print(f"Mean: {mean_accuracy * 100:.2f}%")
print(f"Standard deviation: {std_accuracy * 100:.2f}%")
```

```
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
```

```
  warnings.warn(
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
/local/pkg/python/root-python-3.12/lib/python3.12/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
```

```
Mean: 53.67%
Standard deviation: 0.81%
```

Explain your code here. What insight do you obtain from the mean and standard deviation you calculated?

In this code, I perform a series of experiments to evaluate the performance of the AdaBoost classifier. I start by initializing an empty list to store the accuracy results. Then, I repeat the following steps 10 times: first, I split the dataset into training and testing sets, then train the AdaBoost classifier on the training data, and finally predict the outcomes for the test set. After each prediction, I compute the accuracy and store it in the list. Once all splits are done, I calculate the mean and standard deviation of the accuracies to get an overall sense of how well the classifier performs across different splits. The mean accuracy gives the average performance, while the standard deviation indicates how much the performance varies.

Insight: The standard deviation of 0.81% indicates that the model is pretty consistent on 10 runs. Which on average scored 53.67%.

5. Use PCA to reduce the dimensionality of the dataset down to two dimensions.

[5]:
```python
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
```

```
X_reduced = pca.fit_transform(newsgroups_data.data.toarray())

print(f"Shape of the reduced dataset: {X_reduced.shape}")
```

```
Shape of the reduced dataset: (11314, 2)
```

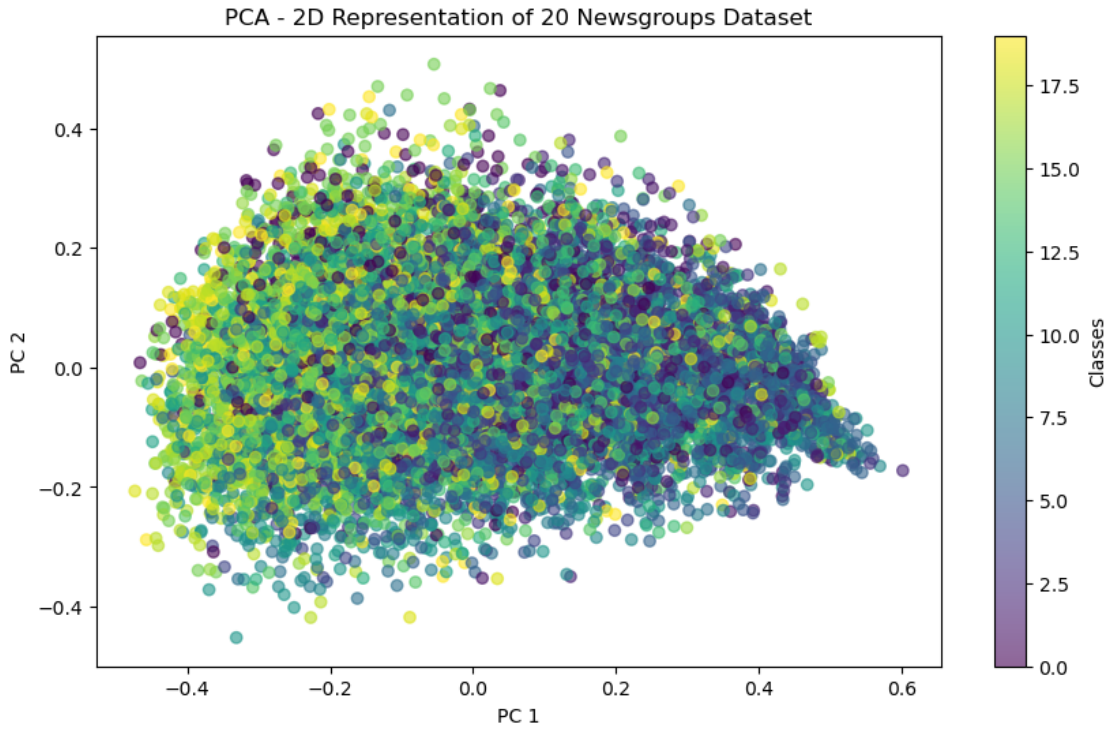
Explain your code here.

In this code, I use Principal Component Analysis (PCA) to reduce the dimensionality of the dataset to 2 dimensions. First, I create a PCA object with n_components=2 to specify that I want to reduce the data to two features. I then apply PCA to the dataset after converting it from a sparse matrix to a dense format using toarray(). The transformed data, now reduced to two dimensions, is stored in X_reduced. Finally, I print the shape of this reduced dataset.

6. Plot a scatterplot of the dataset using the two-dimensional representation you obtained in the previous step. Do you see a visual separation between the classes in the scatterplot?

[6]:
```
import matplotlib.pyplot as plt

y = newsgroups_data.target

plt.figure(figsize=(10, 6))
scatter = plt.scatter(X_reduced[:, 0], X_reduced[:, 1], c=y, cmap='viridis',␣
 ↪alpha=0.6)

plt.colorbar(scatter, label="Classes")
plt.title("PCA - 2D Representation of 20 Newsgroups Dataset")
plt.xlabel("PC 1")
plt.ylabel("PC 2")
plt.show()
```

```
Matplotlib is building the font cache; this may take a moment.
```



Explain your code here. Describe your insights from the scatterplot.

In this code, I create a scatter plot to visualize the 2D representation of the 20 Newsgroups dataset after applying PCA. I use plt.scatter() to plot the data points, where the colors represent different classes in the dataset, using the 'viridis' colormap for better distinction. A color bar is added to the plot to indicate which colors correspond to which classes. The plot is then labeled with a title and axis labels to describe the PCA components clearly. Finally, plt.show() displays the scatter plot, providing a visual insight into how the data is distributed in the reduced 2D space.

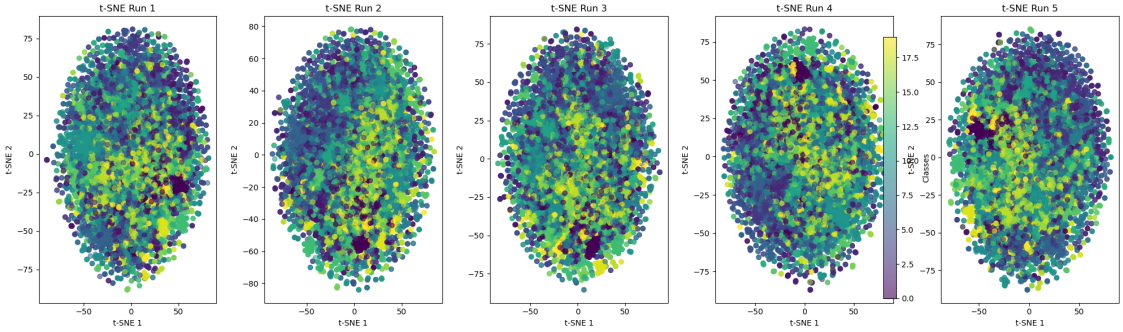
Insights: The scatter plot shows a dense cloud of points without any clear separation between different groups. This means the classes overlap significantly. There are no distinct clusters where points of one class are visibly separated from others, which indicates that in this two-dimensional representation, the dataset is not easily distinctly visible by class.

7. Use t-SNE (see link1 and link2) to reduce the dimensionality of the dataset down to two dimensions. Plot a scatterplot of the dataset using the 2D representation you obtained with t-SNE. Repeat five times, using five runs of t-SNE. Do you see a visual separation between the classes in the scatterplot?

[7]:
```
from sklearn.manifold import TSNE

fig, axes = plt.subplots(1, 5, figsize=(20, 6))

for i in range(5):
    tsne = TSNE(n_components=2, random_state=i)
```

```
    X_tsne_reduced = tsne.fit_transform(newsgroups_data.data.toarray())
    y = newsgroups_data.target

    scatter = axes[i].scatter(X_tsne_reduced[:, 0], X_tsne_reduced[:, 1], c=y,␣
 ↪cmap='viridis', alpha=0.6)
    axes[i].set_title(f"t-SNE Run {i+1}")
    axes[i].set_xlabel("t-SNE 1")
    axes[i].set_ylabel("t-SNE 2")

plt.colorbar(scatter, ax=axes, label="Classes")
plt.tight_layout()
plt.show()
```

```
/tmp/ipykernel_3531215/1295485882.py:16: UserWarning: This figure includes Axes
that are not compatible with tight_layout, so results might be incorrect.
  plt.tight_layout()
```



Explain your code here. Describe your insights from the scatterplot.

In this code, I use t-SNE to visualize the 20 Newsgroups dataset in 2 dimensions and create multiple scatter plots to observe how the data distribution can vary. I first set up a figure with five subplots to display the results. For each of the five runs, I apply t-SNE with a different random seed (random_state=i) to reduce the probability of redundant data. Each run's results are plotted as a scatter plot on separate subplots. I use colors to represent different classes in the dataset and add a color bar to indicate the class labels. The plots are titled accordingly and labeled to describe the t-SNE components. Finally, I adjusted the layout for clarity and displayed all five scatter plots together.

Insights: Across all five runs, the t-SNE projections show a similar overall "elliptical" or "oval" structure with significant overlap between the classes. There are no clearly distinguishable clusters, though I can see some denser areas where points are grouped closely together. As a result, the information separating the classes might be distributed across more complex dimensions that are not fully captured by the two components.

Compared to the PCA scatterplot I generated earlier, t-SNE has produced a more compact and interpretable structure. However, even with t-SNE, I don't see any distinct class separations, which reflects the complexity of the 20 Newsgroups dataset.