AI3011: Lab Assignment Report

Lab Assignment 08

Pratik Rana
U20220108

PLAKSHA UNIVERSITY

Declaration

I, Pratik, certify that this project is my own work, based on my personal study and research and that I have acknowledged all material and sources used in its preparation, whether they be books, articles, reports, lecture notes, and any other kind of document, electronic or personal communication. I also certify that this project has not previously been submitted for assessment in any academic capacity and that I have not copied in part or whole or otherwise plagiarised the work of other persons. I confirm that I have identified and declared all possible conflicts that I may have.

Dated: 13 - 03 - 2023

Pratik Rana

Answers

Q1: Provide three applications of Semi-supervised learning.
Ans. Three Applications:

- Speech recognition
- Image classification
- Classify sentiments in text data

Q2: What are the three assumptions of Semi-supervised learning?
Ans. Three Assumptions:

- Cluster assumption: Points that are close to each other in the input space are likely to belong to the same class.
- Manifold assumption: The data lie approximately on a low-dimensional manifold embedded in the input space.
- Continuity assumption: The decision boundary between classes is smooth and doesn't change abruptly within small neighbourhoods.

Q3: What is the significance of each of the above three assumptions in Semi-supervised Learning?
Ans. Significance of using the above three assumptions in Semi-supervised Learning:

- Cluster assumption helps in leveraging unlabelled data by assuming that points close to each other are likely to belong to the same class.
- Manifold assumption aids in extrapolating information from unlabelled data by assuming that the data distribution lies on a low-dimensional manifold.
- Continuity assumption assists in making predictions smoother and more accurate by assuming that the decision boundary between classes doesn't change abruptly.

Q4: How does the Co-training method differ from the Self-training method of Semi-supervised learning? Which one usually performs better for accuracy and such performance metrics?
Ans.

- Co-training uses **multiple views** of the data and trains separate classifiers on each view, while Self-training iteratively trains a single classifier on the entire dataset, including both labelled and unlabelled data.
- Co-training requires initial labelled data for each view, while Self-training starts with a small amount of labelled data and **iteratively** adds pseudo-labelled data.
- Co-training **performs better** when there are multiple informative views of the data, while Self-training may perform better when the dataset has a clear structure and the classifier can effectively exploit the unlabelled data.

Q5: How to evaluate the performance of a semi-supervised learning method while training?
Ans. Monitoring performance metrics such as accuracy, precision, recall, and F1 score on a validation set. Cross-validation and hold-out validation are common evaluation techniques.

References:

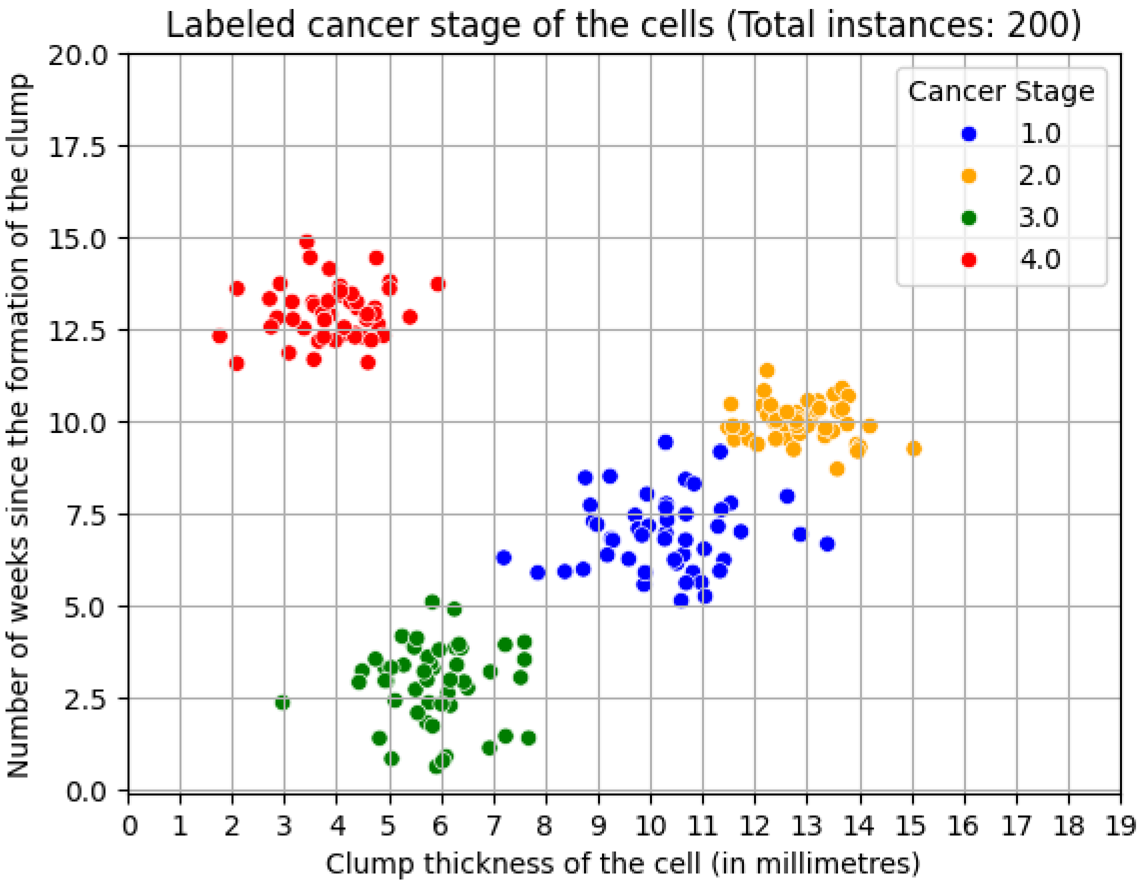
1. **Lecture Slides**

Plots:

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
from sklearn.semi_supervised import SelfTrainingClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, classification_report, accu
```

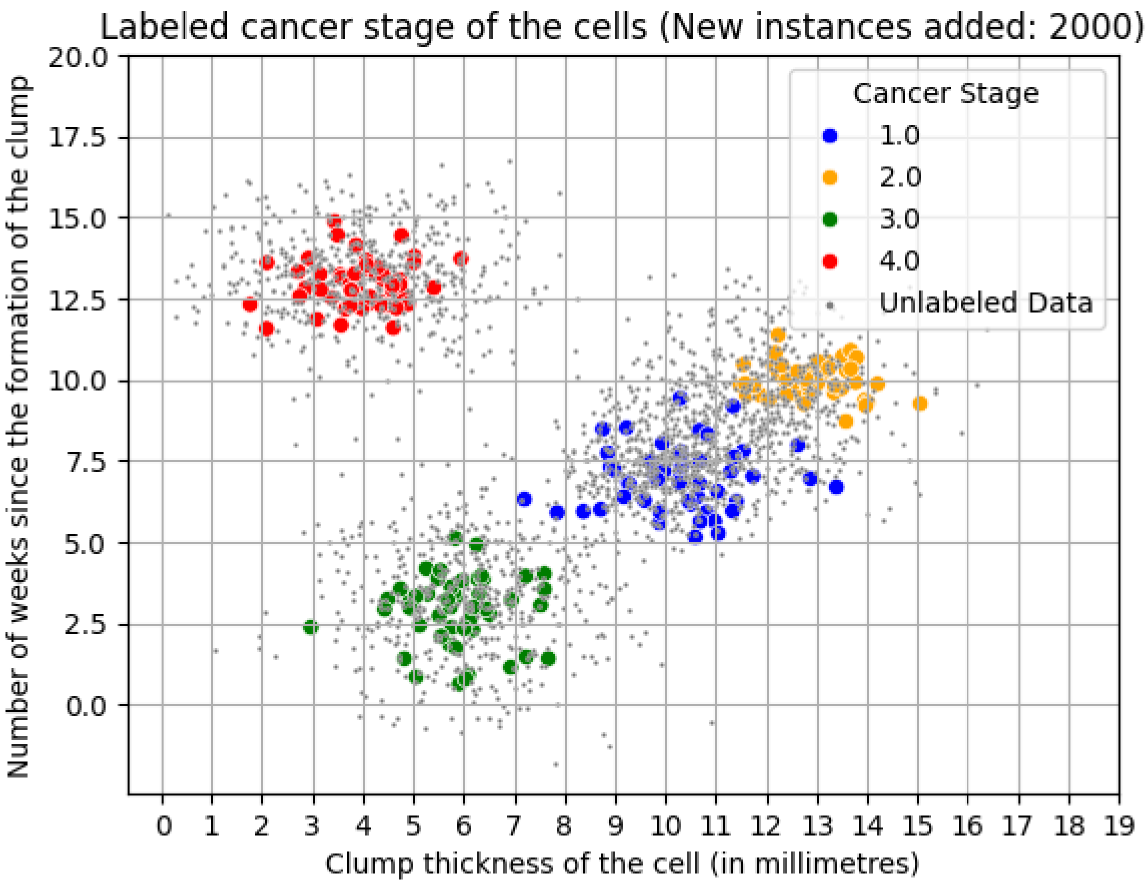
In [ ]:
```
df=pd.read_excel("data 1.xlsx")
palette={1: 'blue', 2:'orange', 3: 'green', 4:'red'}
```

In [ ]:
```
sns.scatterplot(data=df, x='Clump thickness', y='No of week', hue=df['Can
plt.grid(True)
plt.xlabel("Clump thickness of the cell (in millimetres)")
plt.ylabel("Number of weeks since the formation of the clump")
plt.title("Labeled cancer stage of the cells (Total instances: 200)")
plt.xticks(np.arange(0, 20))
plt.yticks(np.arange(0, 21, 2.5))
plt.legend(title='Cancer Stage', loc='upper right')
plt.show()
```



In [ ]:
```
sns.scatterplot(data=df, x='Clump thickness', y='No of week', hue=df['Can
sns.scatterplot(data=df, x='Clump thickness_new', y='No of week_new', col
plt.scatter([], [], color='grey', label='Unlabeled Data', s=3)
plt.grid(True)
plt.xlabel("Clump thickness of the cell (in millimetres)")
plt.ylabel("Number of weeks since the formation of the clump")
plt.title("Labeled cancer stage of the cells (New instances added: 2000)"
plt.xticks(np.arange(0, 20))
plt.yticks(np.arange(0, 21, 2.5))
```

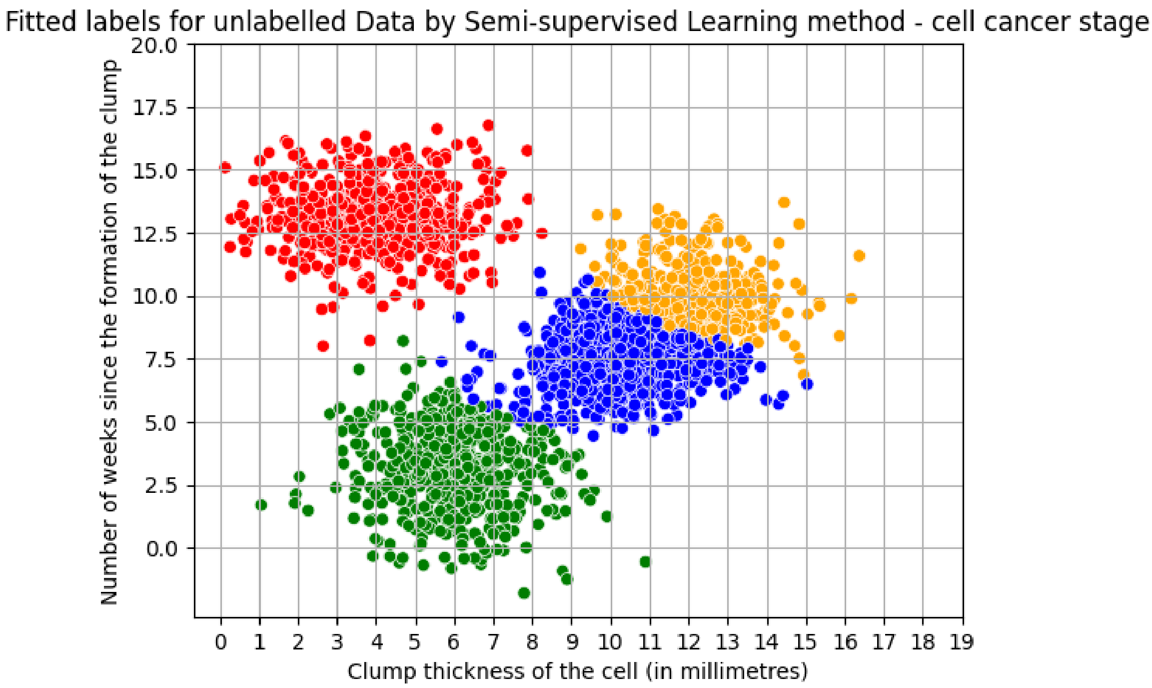
```
plt.legend(title='Cancer Stage', loc='upper right')
plt.show()
```



In [ ]:

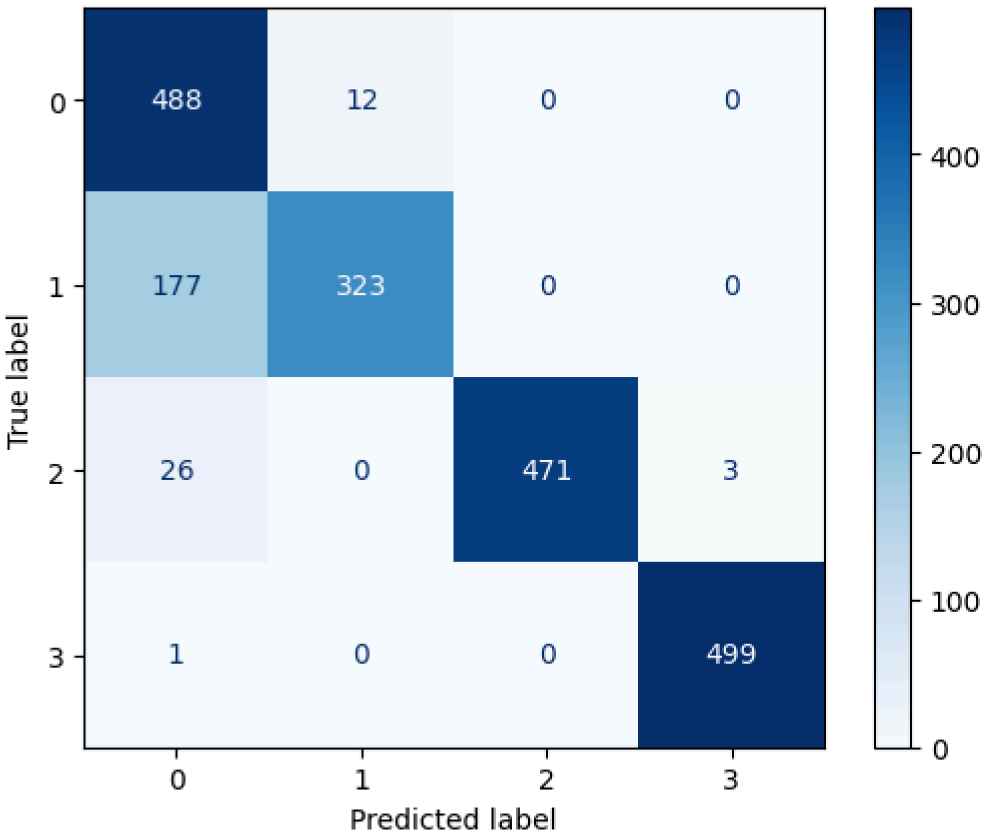
```
x=df[['Clump thickness', 'No of week']][:200]
y=df['Cancer stage'][:200]

knn=KNeighborsClassifier()
ssm=SelfTrainingClassifier(base_estimator=knn)
ssm.fit(x, y)

sns.scatterplot(data=df, x='Clump thickness', y='No of week', hue=df['Can
df[['Clump thickness', 'No of week']]=df[['Clump thickness_new', 'No of w

y_train=df[['Clump thickness', 'No of week']]
y_pred=ssm.predict(y_train)

sns.scatterplot(data=df, x='Clump thickness', y='No of week', hue=y_pred,
plt.grid(True)
plt.xlabel("Clump thickness of the cell (in millimetres)")
plt.ylabel("Number of weeks since the formation of the clump")
plt.title("Fitted labels for unlabelled Data by Semi-supervised Learning
plt.xticks(np.arange(0, 20))
plt.yticks(np.arange(0, 21, 2.5))
plt.legend().set_visible(False)
plt.show()
```

```
/home/pratik/.local/lib/python3.10/site-packages/sklearn/semi_supervised/_
self_training.py:217: UserWarning: y contains no unlabeled samples
  warnings.warn("y contains no unlabeled samples", UserWarning)
```



In [ ]:
```
cm=confusion_matrix(df['True cancer stage'], y_pred)

ConfusionMatrixDisplay(cm).plot(cmap="Blues")
```

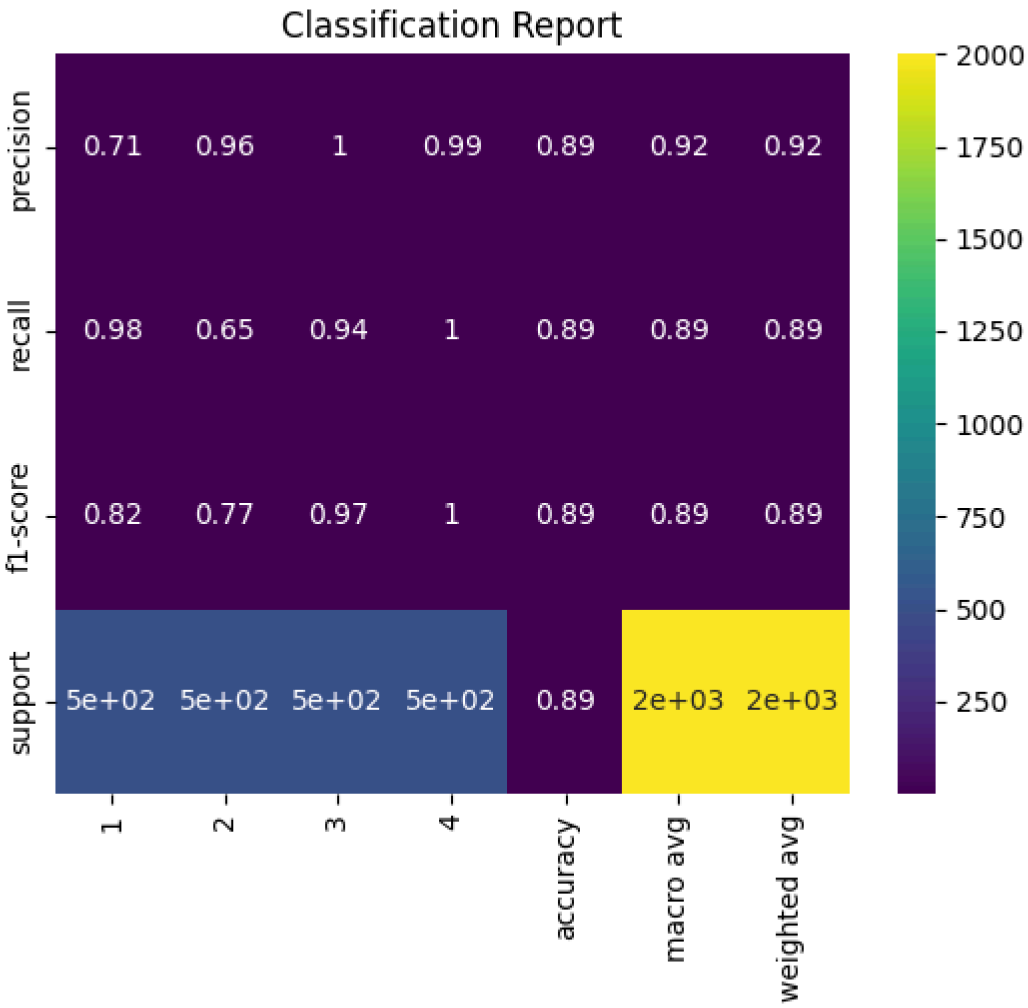
Out[ ]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x78e1
6bdef310>
```



In [ ]:
```
print(f'Classification Report : {classification_report(df["True cancer st
report_dict = classification_report(df["True cancer stage"], y_pred, outp
report_df = pd.DataFrame(report_dict)
sns.heatmap(report_df, annot=True, cmap='viridis')
plt.title('Classification Report')
```

```
plt.show()
print(f'Accuracy Score: {accuracy_score(df["True cancer stage"], y_pred)}
print(f'Balanced Accuracy Score: {balanced_accuracy_score(df["True cancer
```

```
Classification Report :               precision    recall  f1-score   supp
ort

           1       0.71      0.98      0.82       500
           2       0.96      0.65      0.77       500
           3       1.00      0.94      0.97       500
           4       0.99      1.00      1.00       500

    accuracy                           0.89      2000
   macro avg       0.92      0.89      0.89      2000
weighted avg       0.92      0.89      0.89      2000
```




```
Accuracy Score: 0.8905
Balanced Accuracy Score: 0.8905000000000001
```