Programming Assignment - 5

```
# Import required packages
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```

Question 1

Setup a logistic regression model on the data at [adultUCI](http://archive.ics.uci.edu/ml/datasets/Adult). Discuss the performance of your model using appropriate statistics. Use dummy variables to handle categorical variables.

1. Prepare the data. Create dummy variables for categorical variables. See this
2. Get feature matrix X, and target variable y (>50k or <50k)
3. Split data into training and testing
4. Normalize data using MinMaxScaler
5. Creat a LogisticRegression object for modeling
6. Train the model with training data
7. Compare the precision, recall, and F1-score on the train and test data.
8. Improve the performance of the model on the test dataset.

```
url = "http://archive.ics.uci.edu/ml/machine-learning-databases/adult/adult.data"
names = ['age', 'workclass', 'fnlwgt', 'education', 'education-num', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'capi
dataset = pd.read_csv(url, names=names)
dataset = pd.get_dummies(dataset, columns=['workclass', 'education', 'marital-status', 'occupation', 'relationship', 'race', 'sex', 'nati
1
X = dataset.drop('income', axis=1)
y = dataset['income']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
model = LogisticRegression()
model.fit(X_train, y_train)
y_train_pred = model.predict(X_train)
print(classification_report(y_train, y_train_pred))
y_test_pred = model.predict(X_test)
print(classification_report(y_test, y_test_pred))
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
              precision    recall  f1-score   support

       <=50K       0.88      0.93      0.90     19778
        >50K       0.74      0.60      0.66      6270

    accuracy                           0.85     26048
   macro avg       0.81      0.76      0.78     26048
weighted avg       0.84      0.85      0.85     26048

              precision    recall  f1-score   support

       <=50K       0.88      0.94      0.91      4942
        >50K       0.75      0.61      0.67      1571

    accuracy                           0.86      6513
   macro avg       0.82      0.77      0.79      6513
weighted avg       0.85      0.86      0.85      6513
```

Question 2

Create the sparse COO representation (ref) of the adjacency matrix (no need to find the dense representation) of the following graph. Find its CSR representation using appropriate Python function. Visualize this matrix by using spy() function. Refer to the notebook of Lecture 10 live session.

Image source: Mathworks

[16]:
```
# Your code comes here
import numpy as np
from scipy.sparse import coo_matrix

# Define the adjacency matrix
adj_matrix = np.
 ↪array([[0,1,0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
 ↪[1,0,1,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
  ↪[0,1,0,1,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,[0,0,1,0,1,0,0,0,0,0, 0,
                              0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0],
                              [1,0,0,1,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 ↪0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0],
                              [1,0,0,0,0,0,1,0,0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0],
                              [0,0,0,0,0,1,0,1,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 ↪0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0],
                              [0,0,0,0,0,0,1,0,1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0],
                              [0,0,0,0,0,0,0,1,0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0],
                              [0,0,0,0,0,1,0,0,1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0],
                              [0,1,0,0,0,0,0,0,0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0],
                              [0,0,0,0,0,0,0,0,0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
 0, 0, 0, 0, 0],
```



```
                              [0,0,0,0,0,0,0,0,0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
 ↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,  ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
```

0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,1,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 1, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,␣
↪1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,1,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1,␣
↪0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 1, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 1, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,␣
↪0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,1,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0],

[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣

```
↪0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪1, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,1,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 1, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 0, 0],
                                 [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
                                 [0,0,0,0,0,0,0,1,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0],
```

```
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1],
[0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0],
```

```
                                                [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 1, 0, 1, 0],
                                                [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 1, 0, 0, 0, 0, 0,
0, 0, 1, 0, 1],
                                                [0,0,0,0,0,0,0,0,0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣
↪0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,␣ ↪0, 0, 0, 0, 0, 0, 0, 0, 1,
1, 0, 0, 1, 0]])
```

```
# Create the COO matrix
coo_matrix = coo_matrix(adj_matrix)

print(coo_matrix)
```

```
  (0, 1) 1
  (0, 4) 1
  (0, 5) 1
  (1, 0) 1
  (1, 2) 1
  (1, 10) 1
  (2, 1) 1
  (2, 3) 1
  (2, 15) 1
  (3, 2) 1
  (3, 4) 1
  (3, 20) 1
  (4, 0) 1
  (4, 3) 1
  (4, 25) 1
  (5, 0) 1
  (5, 6) 1
  (5, 9) 1
  (6, 5) 1
  (6, 7) 1
  (6, 29) 1
  (7, 6) 1
  (7, 8) 1
  (7, 41) 1
  (8, 7) 1
  : :
  (51, 52) 1
  (52, 18) 1
  (52, 51) 1
```

```
  (52, 53) 1
  (53, 30) 1
  (53, 52) 1
```



```
  (53, 54) 1
  (54, 50) 1
  (54, 53) 1
  (54, 59) 1
  (55, 34) 1
  (55, 56) 1
  (55, 59) 1
  (56, 39) 1
  (56, 55) 1
  (56, 57) 1
  (57, 44) 1
  (57, 56) 1
  (57, 58) 1
  (58, 49) 1
  (58, 57) 1
  (58, 59) 1
  (59, 54) 1
  (59, 55) 1
  (59, 58) 1
```

[17]:
```
from scipy.sparse import csr_matrix

# Convert the COO matrix to CSR format
csr_matrix = csr_matrix(coo_matrix)

print(csr_matrix)
```

```
  (0, 1) 1
  (0, 4) 1
  (0, 5) 1
  (1, 0) 1
  (1, 2) 1
  (1, 10) 1
  (2, 1) 1
  (2, 3) 1
  (2, 15) 1
  (3, 2) 1
  (3, 4) 1
  (3, 20) 1
  (4, 0) 1
  (4, 3) 1
  (4, 25) 1
```

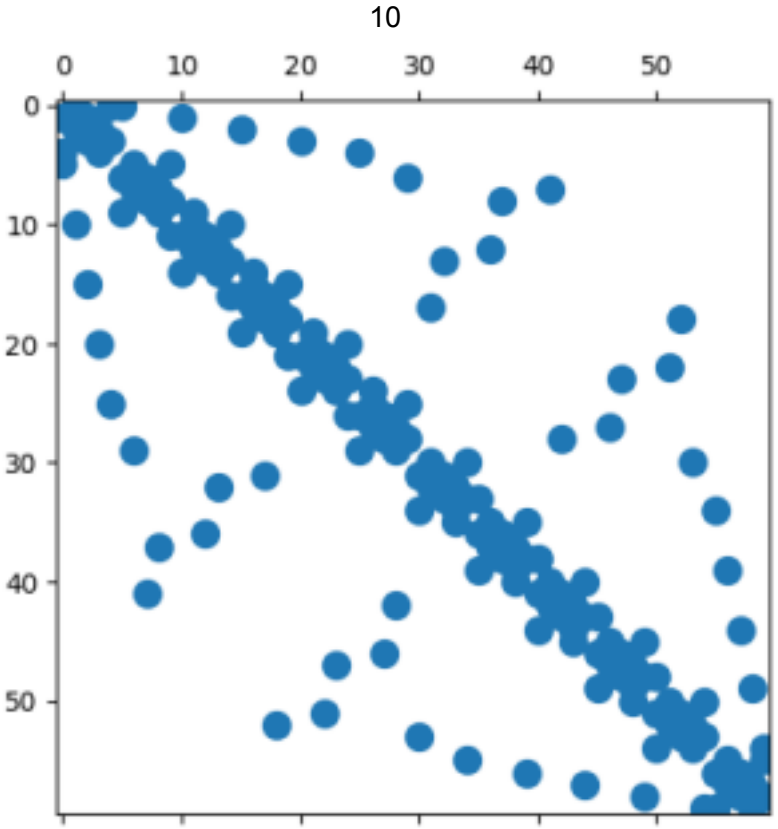
```
  (5, 0) 1
  (5, 6) 1
  (5, 9) 1
  (6, 5) 1
  (6, 7) 1
  (6, 29) 1
```



```
  (7, 6) 1
  (7, 8) 1
  (7, 41) 1
  (8, 7) 1
  : :
  (51, 52) 1
  (52, 18) 1
  (52, 51) 1
  (52, 53) 1
  (53, 30) 1
  (53, 52) 1
  (53, 54) 1
  (54, 50) 1
  (54, 53) 1
  (54, 59) 1
  (55, 34) 1
  (55, 56) 1
  (55, 59) 1
  (56, 39) 1
  (56, 55) 1
  (56, 57) 1
  (57, 44) 1
  (57, 56) 1
  (57, 58) 1
  (58, 49) 1
  (58, 57) 1
  (58, 59) 1
  (59, 54) 1
  (59, 55) 1
  (59, 58) 1
```

[20]:
```
from matplotlib import pyplot as plt

# Visualize the matrix using the spy function
plt.spy(csr_matrix, marker='o')
plt.show()
```



0.3 Question 3

The adjacency matrix of toy world-wide-web has been provided as a text file on the assignment page. Implement the page-ranking algorithm that displays the index of the 10 highest ranking web-pages. Also report on the time it takes to perform these calculations.

[15]:

In [7]:
```
import numpy as np
import time

# Read the adjacency matrix from the text file
adj_matrix = np.loadtxt("toy_www_1000.csv", delimiter=',')

# Get the number of pages
n_pages = adj_matrix.shape[0]

# Initialize the page ranks
page_ranks = np.full((n_pages,), 1/n_pages)

# Set the damping factor
damping_factor = 0.85

n = len(adj_matrix)

# Set the maximum number of iterations
max_iterations = n

# Set the convergence threshold
convergence_threshold = 1e-8

# Start the timer
start_time = time.time()

# Run the PageRank algorithm
for i in range(max_iterations):
    prev_page_ranks = page_ranks.copy()
    for j in range(n_pages):
        incoming_links = np.where(adj_matrix[:,j] == 1)[0]
        if len(incoming_links) > 0:
            link_weights = np.full((len(incoming_links),), 1/len(incoming_links))

            page_ranks[j] = np.sum(prev_page_ranks[incoming_links] *link_weights)
        else:
            page_ranks[j] = 0
    page_ranks = (1 - damping_factor) * (1/n_pages) + damping_factor *page_ranks

# Stop the timer and print the time taken
end_time = time.time()
print("Time taken:", end_time - start_time, "seconds")
# Sort the page ranks and display the top 10 pages
top_indices = np.argsort(-page_ranks)[:10]
print("Top 10 pages by PageRank:")
for i in top_indices:
    print(i+1)
```

```
---------------------------------------------------------------------------
OSError                                   Traceback (most recent call last)
/var/folders/tt/qn8g5b_15hbbhs7xxx68p9m40000gn/T/ipykernel_40830/3865403991.py in <module>
      3
      4 # Read the adjacency matrix from the text file
----> 5 adj_matrix = np.loadtxt("toy_www_1000.csv", delimiter=',')
      6
      7 # Get the number of pages

/Applications/anaconda3/lib/python3.9/site-packages/numpy/lib/npyio.py in loadtxt(fname, dtype, comments, delimiter, converters, skiprows, usecols, unpack, ndmin, encoding, max_rows, like)
   1065             fname = os_fspath(fname)
   1066         if _is_string_like(fname):
-> 1067             fh = np.lib._datasource.open(fname, 'rt', encoding=encoding)
   1068             fencoding = getattr(fh, 'encoding', 'latin1')
   1069             fh = iter(fh)
```