Rachit Jaiswal

2018404

Q1.

a). PCA compresses a lot of features such that it does not lose the quality of data by finding the major patterns in data for dimensionality reduction where the correlation between new features is zero.
PCA first calculates the covariance matrix X of data points.
Then it calculates eigenvectors and corresponding eigenvalues.
Sort the eigenvectors according to their eigenvalues in decreasing order.
Choose first k eigenvectors and that will be the new k dimensions.
Then the original n-dimensional data points transformed into k dimensions.

b).

Singular value decomposition reduces a rank R matrix to a rank K matrix, by approximating R unique vectors as a linear combination of K unique vectors.
It decomposes the matrix into 3 matrices

$A=U\ sigma_k\ V^T_k$ where U= left singular values, sigma =singular values, V^T=right singular values
Eigenvectors are sorted in descending order where first k<n values are selected for k components.
Eigenvectors of a matrix are gives the variance of data.

For reducing dimension,
Each row can be written as the sum of k singular vectors vectors
$R_i=c_1 \cdot SV_1 + c_2 \cdot SV_2 + \ldots. \ c_k \cdot SV_k$, where k most influential vectors are taken for reducing rank r matrix to rank k matrix .

For highly correlated data, $sigma_i$ will be small.

c).
t-SNE is dimension reduction technique which projects data to lower-dimensional space to make it easier to analyse.
It retains non-linear variance which is the drawback for PCA.

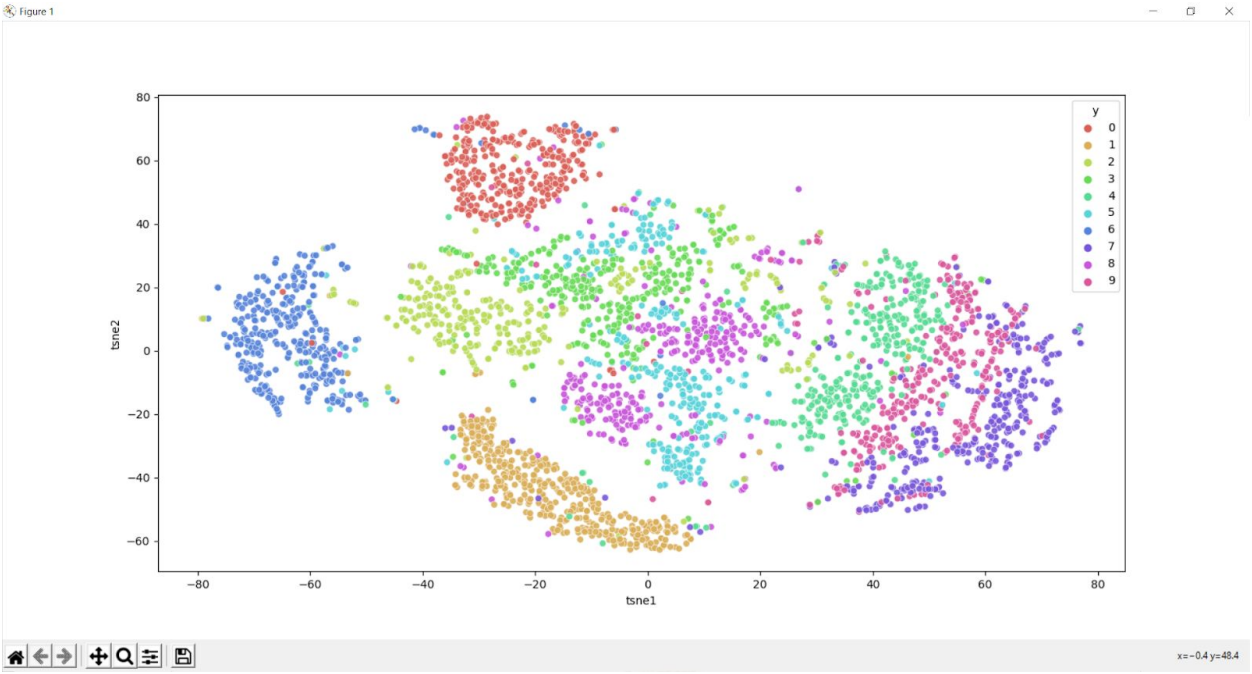
It generates a probability distribution where similar objects have a high probability on higher dimensions, then also on lower dimensions till the divergence is minimized.
t-SNE shrinks widespread data and expands densely packed data.

d). Class frequency for train and test split is distributed in equal proportions by class after stratification.

```
1    395
6    353
7    345
3    339
4    333
8    328
0    320
5    319
9    314
2    314
Name: 0, dtype: int64
1    99
6    88
7    86
3    85
4    83
8    82
0    80
9    79
5    79
2    79
Name: 0, dtype: int64
```

e).

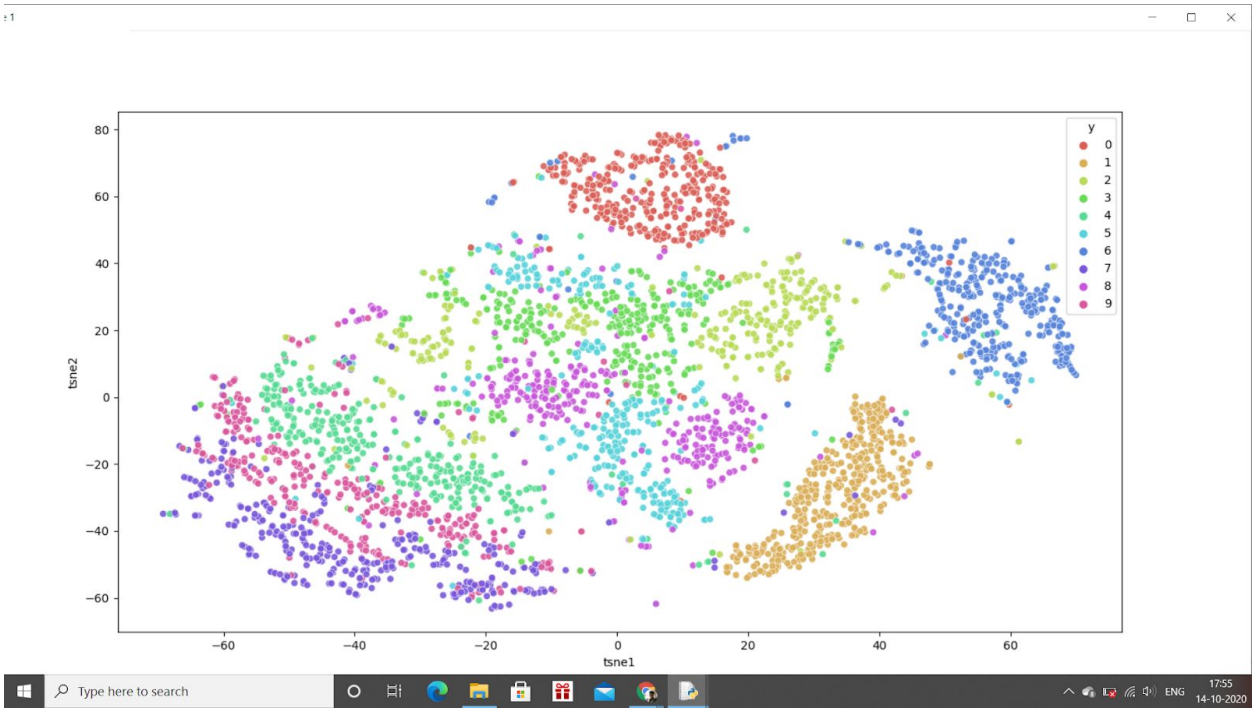
```
n components 170
Accuracy 0.8690476190476191
```



Clusters are well spread and some features can be distinguished with each other where clusters are grouped and isolated from others by looking at this graph of reduced two components of t-SNE after PCA.

f).

```
n components 170
Accuracy 0.8654761904761905
```



Clusters are well spread and many features can be distinguished with each other where clusters are grouped and isolated from others by looking at this graph of reduced two components of t-SNE after SVD.

g.
The accuracy obtained for PCA at 0.95 variations having 170 components is 0.869.
The accuracy obtained for SVD having 170 components is 0.866

Accuracy for PCA is higher than SVD.
SVD may be affected by outliers which does not happen to incase of PCA.
They are both giving almost same clusters of classes after t-SNE
Also, we can specify the variance we want to conserve for PCA, which does not happen in SVD, so we have to specify dimensions we want in SVD without knowing how much variance will be lost with it.

Q2.

```
bias -0.11134623035032314
variance 1.0813605627238308e-23
MSE 141.925529973539
141.91313199052578
```

Q3.

a).

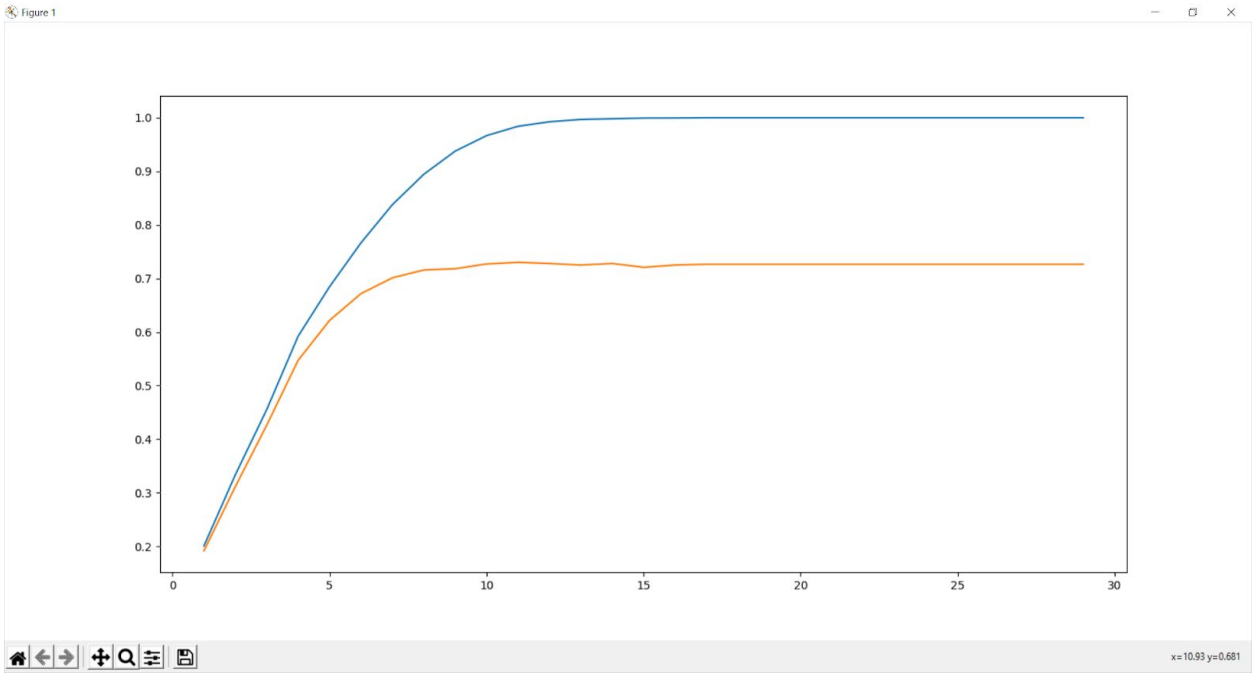
Dataset A

```
optimal depth 11
```

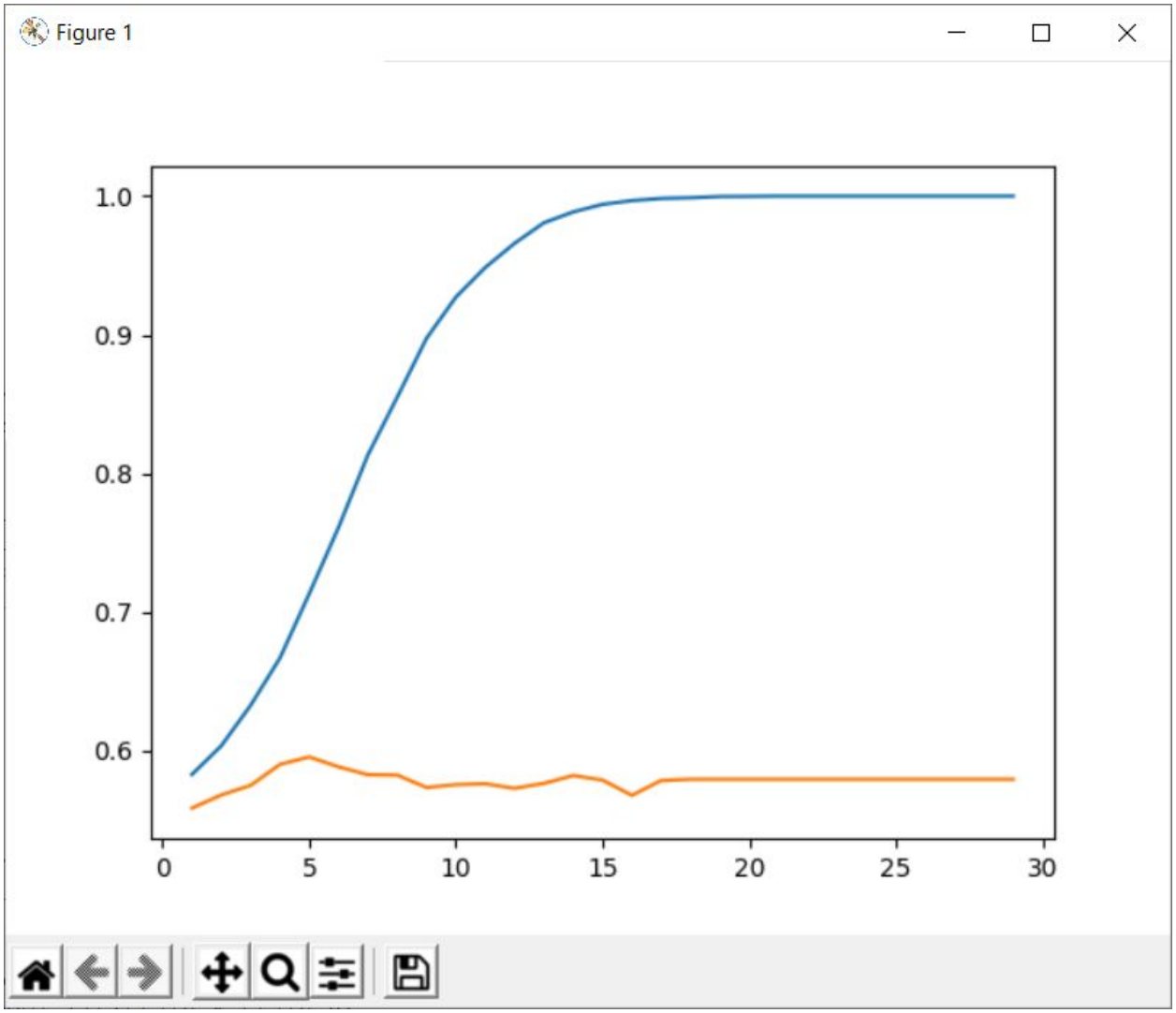
Dataset B

```
optimal depth 5
```

b). Dataset A



Dataset B



In both the graphs, trainset gets overfitted with accuracy reaching 100% after some specific depth whereas the accuracy of the test set decreases by a few decimals and become constant after specific depth because of overfitting.
So, inshort accuracy increases upto specific depth after which it decreases when model gets overfitted due to increase in depth.

c).

```
import pickle

def savemodel(modelx,filename):
    pickle.dump(modelx, open(filename, 'wb'))


def loadresult(filename,test_x,test_y):
    loadx_model = pickle.load(open(filename, 'rb'))
```

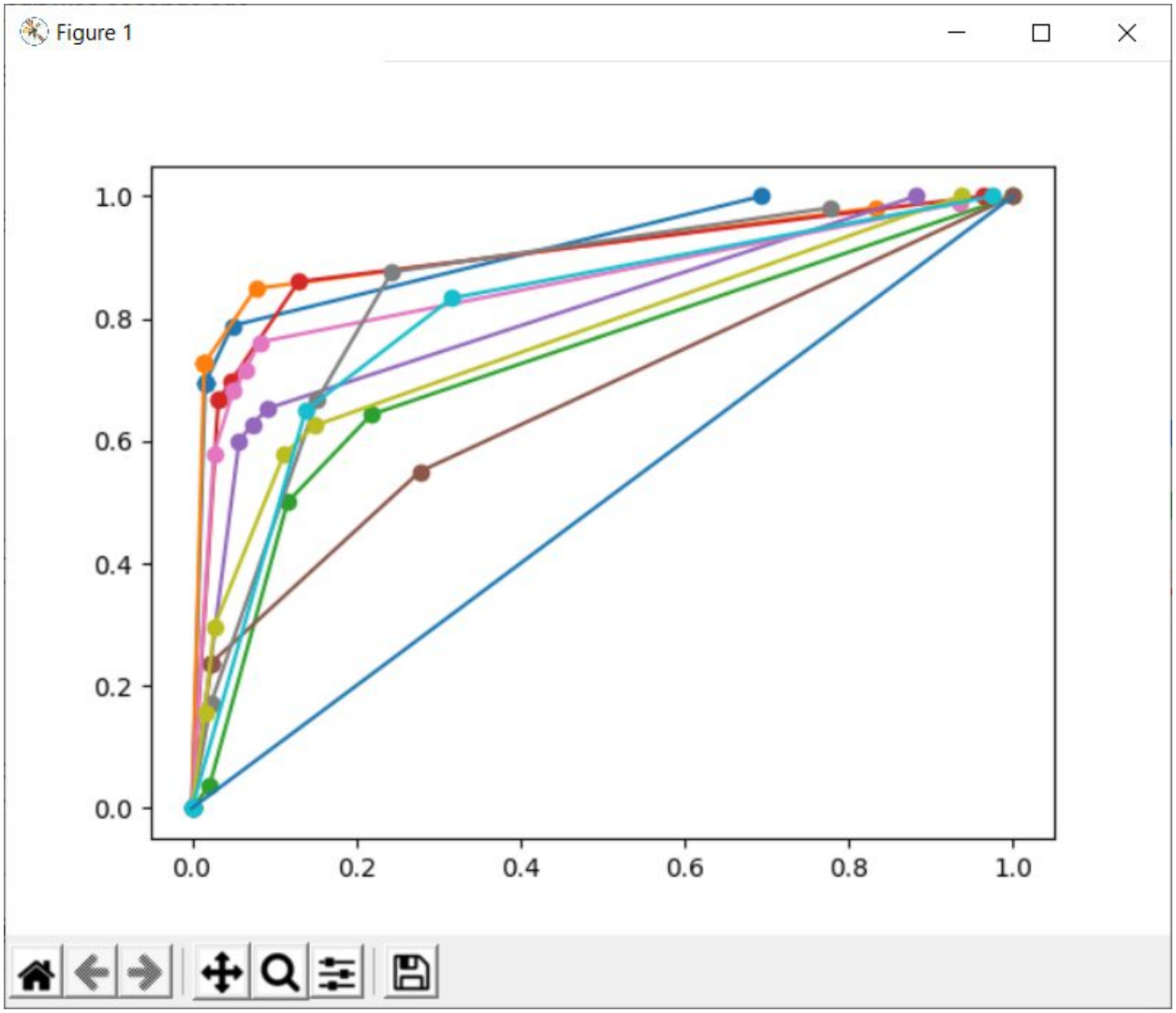
```
    result = loadx_model.score(test_x, test_y)
    print(result)
```

```
tree_model.max_depth=grid.bestdepth
tree_model.fit(train_x,train_y)
savemodel(tree_model,"abc")
loadresult("abc",test_x,test_y)
```

d).
Dataset A
at maxa_depth=4

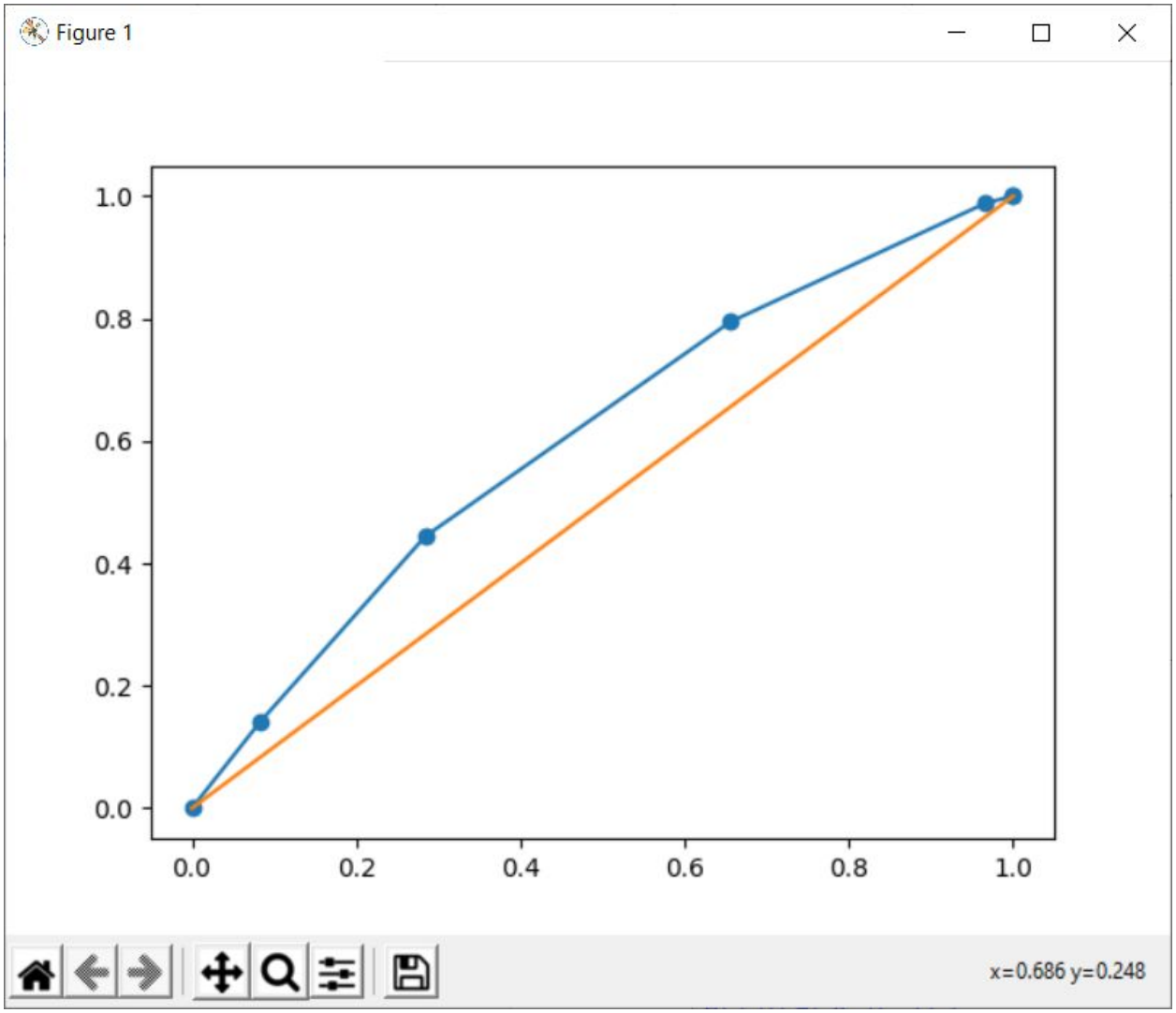
```
Confusion Matrix
[[71  0  2  2  2  7  1  3  7  1]
 [ 0 94  0  0  2  2  1  1  1  1]
 [ 3  2 59  4  0  6  4  1  1  0]
 [ 0  1  2 66  0  6  0  1  2  3]
 [ 1  3  1  0 56  0  1  5  2  5]
 [ 2  1  7  9  4 50  5  4  5  2]
 [ 2  0  1  0  1  5 74  0  2  0]
 [ 1  2  6  4  1  2  1 81  2  7]
 [ 4  3  5  4  1  1  1  2 37  1]
 [ 1  0  1  4  8  1  0  7  5 40]]
Accuracy 0.7476190476190476
Recall, 0.7362940937884035
Precision 0.7383752435694315
F1 Score 0.7361092803022641

Micro
Accuracy for micro 0.7476190476190476
Recall for micro 0.7476190476190476
Precision for micro 0.7476190476190476
F1 Score micro 0.7476190476190476
```



Dataset B
at max_depth=4

```
Confusion Matrix
[[277 222]
 [135 206]]
Accuracy 0.575
Recall, 0.576819254151166
Precision 0.5796078961441946
F1 Score 0.5719418350203195
```



,

Q4.
For dataset A

```
85.47619047619047
From sklearn 85.47619047619047
```

,

For dataset B

```
58.214285714285715
From sklearn 58.214285714285715
```