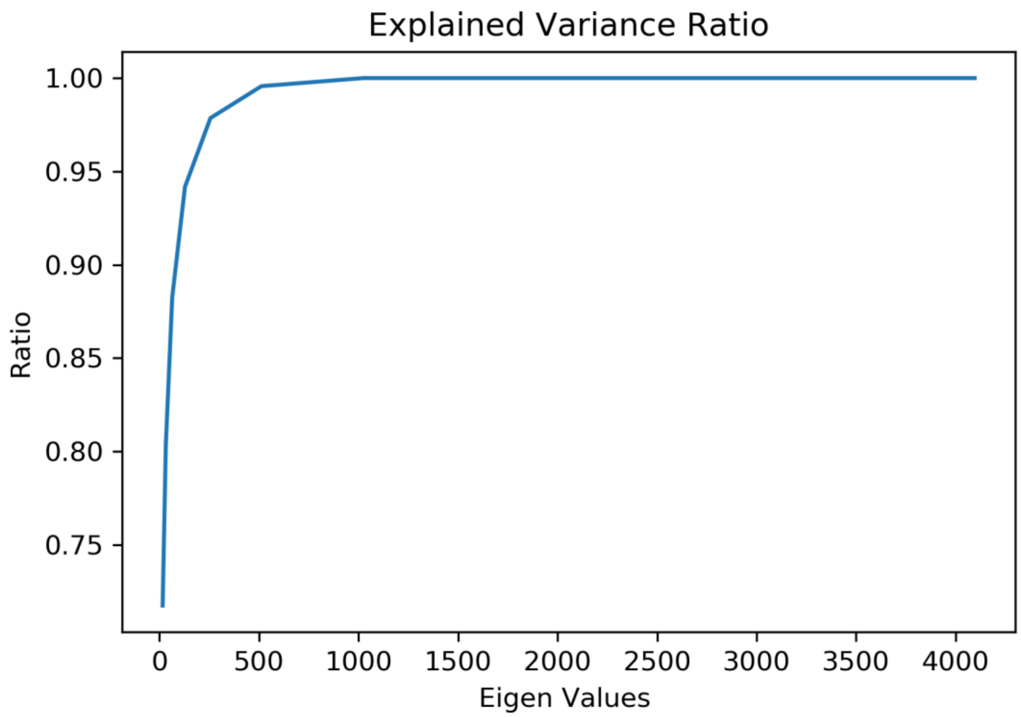
QUESTION 1

1.1

Both PCA and SVD, we need to extract the mean and calculate covariance matrix. For SVD, the covariance matrix will find k most eigen values and eigen values. For PCA, the covariance matrix will be used to get all eigenvalues and eigen vectors and eigen vectors need to be sorted according to eigen values. Since finding eigenvectors and eigen values is costly operation. SVD is more efficient due to only k most eigenvalues is found. Both PCA and SVD try to find eigen values and eigenvectors(or principal components) to determine variance.

1.2



Explained variance ratio is basically sum of first k eigenvalues / sum of all eigen values. Explained variance explains how much information is saved with given k values(k eigenvalues). In other words, ratio of information is saved by pca.

1.3

Code :

```
def recons_img(eig_vecs, mean, imgs):
    #pc scores
    reduced_imgs = np.dot(imgs - mean, eig_vecs.T)
    return np.dot(reduced_imgs, eig_vecs) + mean
```

Since Sklearn uses SVD to calculate eigenvectors and eigenvalues inverse the calculation backward to get reconstructed images.

$$\boxed{\text{PCA reconstruction} = \text{PC scores} \cdot \text{Eigenvectors}^{\top} + \text{Mean}}$$

From plots given below we can say that, when we increase the number of eigenfaces, it contains more variance directions and therefore it represent given image attributes more.


Aaron Eckhart
Aaron Guiel
Aaron Patterson
Aaron Peirsol
Aaron Peirsol
Aaron Peirsol
Aaron Peirsol
Aaron Pena
Aaron Sorkin
Aaron Sorkin
1. Eigen Face
2. Eigen Face
3. Eigen Face
4. Eigen Face
5. Eigen Face
6. Eigen Face
7. Eigen Face
8. Eigen Face
9. Eigen Face
10. Eigen Face
32 Eigenfaces Aaron Eckhart
32 Eigenfaces Aaron Guiel
32 Eigenfaces Aaron Patterson
32 Eigenfaces Aaron Peirsol
32 Eigenfaces Aaron Peirsol
32 Eigenfaces Aaron Peirsol
32 Eigenfaces Aaron Peirsol
32 Eigenfaces Aaron Pena
32 Eigenfaces Aaron Sorkin
32 Eigenfaces Aaron Sorkin
128 Eigenfaces Aaron Eckhart
128 Eigenfaces Aaron Guiel
128 Eigenfaces Aaron Patterson
128 Eigenfaces Aaron Peirsol
128 Eigenfaces Aaron Peirsol
128 Eigenfaces Aaron Peirsol
128 Eigenfaces Aaron Peirsol
128 Eigenfaces Aaron Pena
128 Eigenfaces Aaron Sorkin
128 Eigenfaces Aaron Sorkin
512 Eigenfaces Aaron Eckhart
512 Eigenfaces Aaron Guiel
512 Eigenfaces Aaron Patterson
512 Eigenfaces Aaron Peirsol
512 Eigenfaces Aaron Peirsol
512 Eigenfaces Aaron Peirsol
512 Eigenfaces Aaron Peirsol
512 Eigenfaces Aaron Pena
512 Eigenfaces Aaron Sorkin
512 Eigenfaces Aaron Sorkin

1.4

144 eigenfaces enough to reconstruct image via keeping 95% of the original dataset. Eigenfaces has dimension 144x4096 and original dataset has 1000x 4096 . The ratio is 144/1000 means 14.4% of the original data

QUESTION 2

2.1

$J_n = (y - X\beta)^T(y - X\beta)$

$J_n = (y^T - \beta^T X^T)(y - X\beta)$

$J_n = (y^T y - y^T X\beta - \beta^T X^T y - \beta^T X^T X\beta)$

Since $y^T X\beta$ and $\beta^T X^T y$ is scalar and therefore equal we could write :

$J_n = (y^T y - 2\beta^T X^T y - \beta^T X^T X\beta)$

To minimize β vector the derivative should be equal to 0.

$\partial J_n / \partial\beta = 0$

$\partial J_n / \partial\beta = 0 - 2X^T y + 2X^T X\beta$

$0 = - 2X^T y + 2X^T X\beta$

$X^T y = X^T X\beta$

If X is full column rank means that $X^T X$ is positive definite and therefore invertible

$\beta = (X^T X)^{-1} X^T y$

2.2

Since $X^T X$ has rank of 8 which indicates full rank means that $X^T X$ is positive definite and therefore invertible.

Code :

```
np.linalg.matrix_rank(np.dot(data['train_X'].T, data['train_X']))
```

2.3

Weights(bias(w0) at the end)

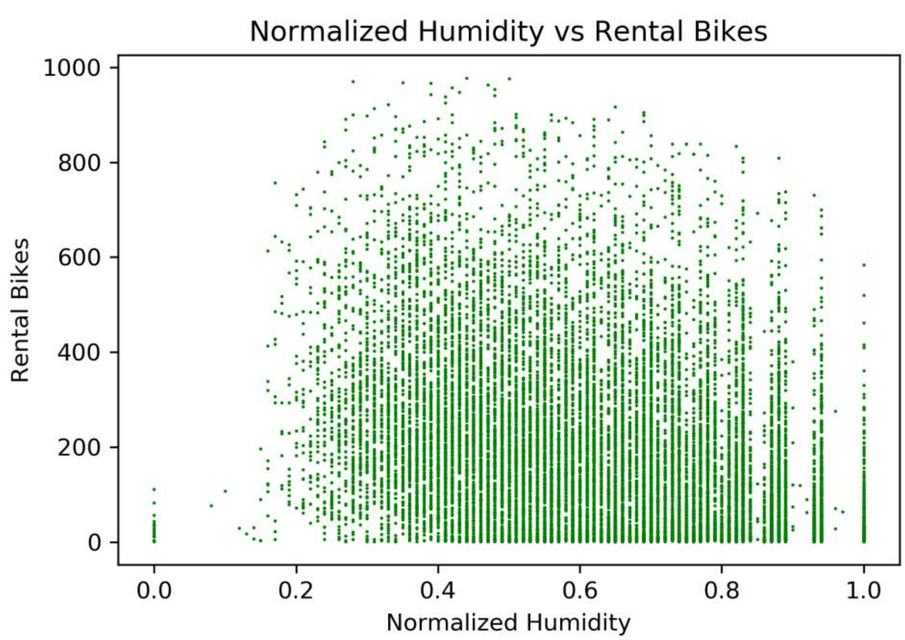
```
array([[  -0.47656134],
       [   7.00054171],
       [   1.37221717],
       [  -4.24724733],
       [  68.4775968 ],
       [ 286.19995694],
       [-180.5873212 ],
       [  24.55935004],
       [  35.52032936]])

Train mse result : 18204.227976890259
Test mse result : 40379.067822401645

Since residuals that we got from train data is more than test mse result,
overfitting may be occurred. Or since we trained on train dataset linear r
egression the residuals should be smaller than test since it learns from t
rain.
```

2.4

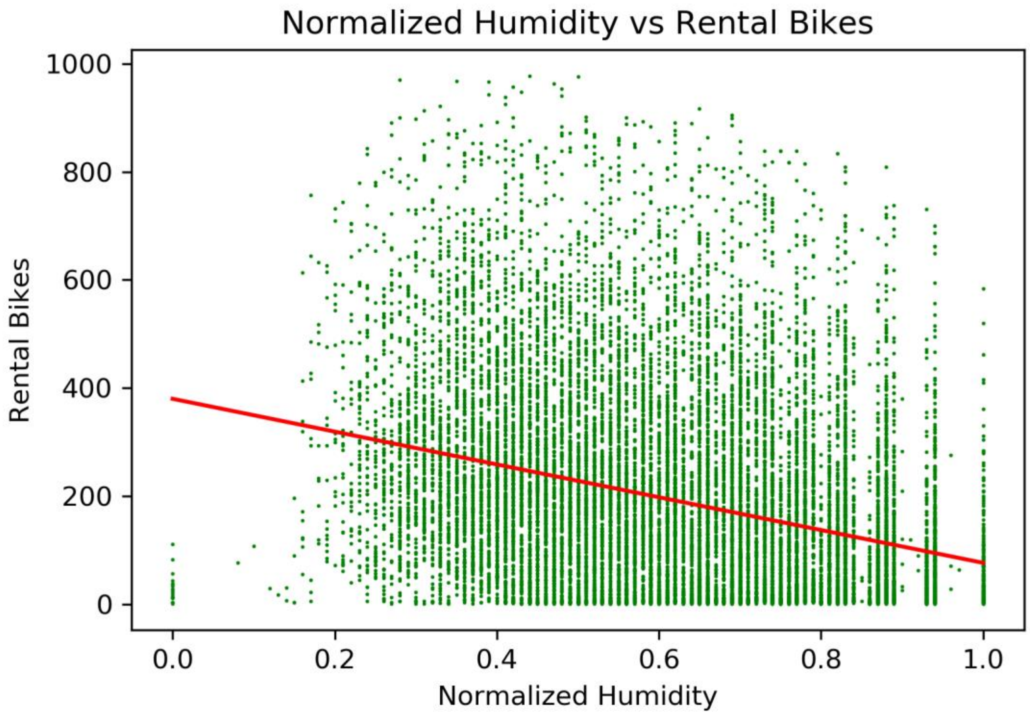
In regression results, if the correlation coefficient is negative, it provides statistical evidence of a negative relationship between the variables. The increase in the first variable will cause the decrease in the second variable. We'll go back to have a look at literature review to possible explanations and rationales. If the magnitude of the coefficient is bigger than others, it will effect predicted value more since we are finding predicted value with weighted sum.



As could be seen given plot, increase in the normalized humidity affect the number of rental bikes negatively.

2.5

Y = `-303.55891962 * X + 379.870785`



From linear regression line we can see that normalized humidity increase is affecting number of rental bikes negatively and our assumption is hold in 2.4.

QUESTION 3

3.1

I choose 0.1 as learning rate because is accuracy (0.71) and f1 (0.78) score is the highest among of all other options for learning rate(I choose f1 score as my metrics since it is more reliable than accuracy)

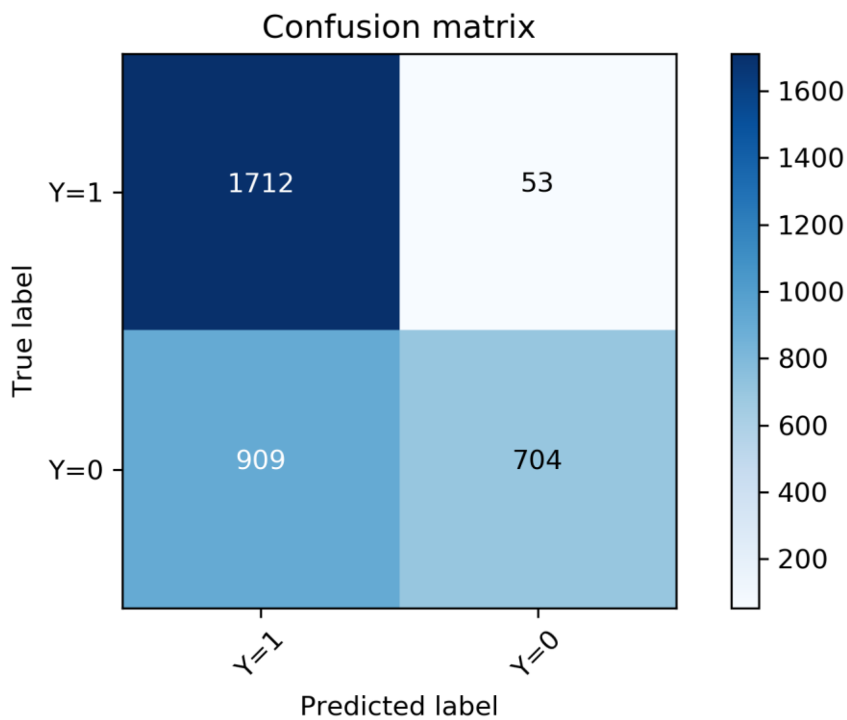
Macro Average

'F1': 0.687379290839091,
'F2': 0.72864489340264071,
'FDR': 0.29678725792033211,
'FPR': 0.20841370154785779,
'NPV': 0.70321274207966789,
'Precision': 0.70321274207966789,

'Recall': 0.79158629845214223

<u>Macro Average</u>

```
'F1': 0.71521610420367077,
'F2': 0.71521610420367077,
'FDR': 0.28478389579632918,
'FPR': 0.28478389579632918,
'NPV': 0.71521610420367077,
'Precision': 0.71521610420367077,
'Recall': 0.71521610420367077
```



Accuracy : (1712+704)/3378 = 0.7152161042

3.2

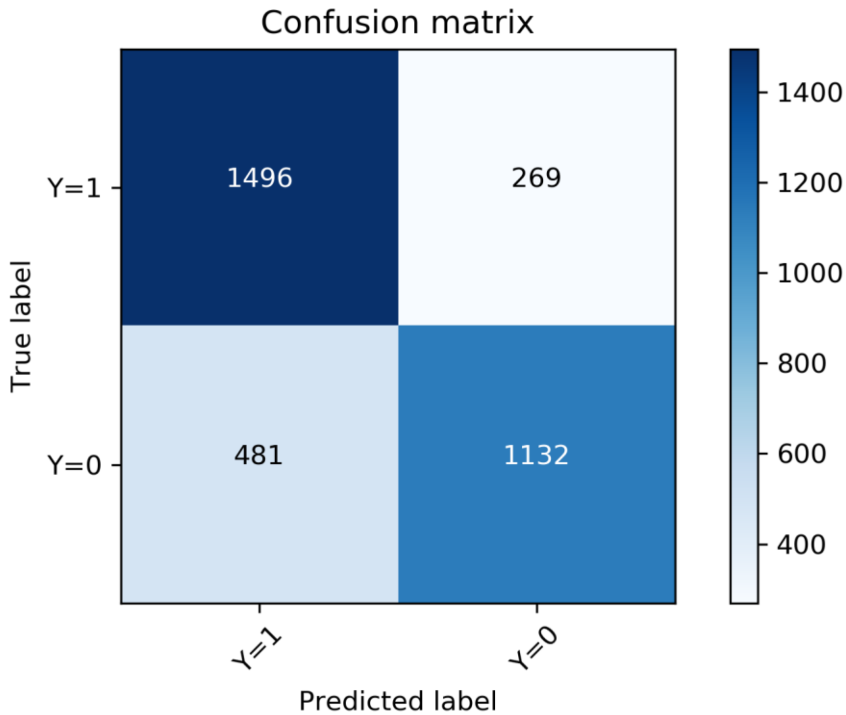
Learning Rate : 0.1

<u>Macro Average</u>

```
'F1': 0.77536683433838249,
'F2': 0.7787729275219879,
'FDR': 0.22530501994242952,
'FPR': 0.2176518181788642,
'NPV': 0.77469498005757043,
'Precision': 0.77469498005757043,
'Recall': 0.78234818182113575
```

Micro Average

```
'F1': 0.77797513321491996,
'F2': 0.77797513321492007,
'FDR': 0.22202486678507993,
'FPR': 0.22202486678507993,
'NPV': 0.77797513321492007,
'Precision': 0.77797513321492007,
'Recall': 0.77797513321492007
```



Accuracy : (1496+1132)/3378 = 0.77797513321

3.3

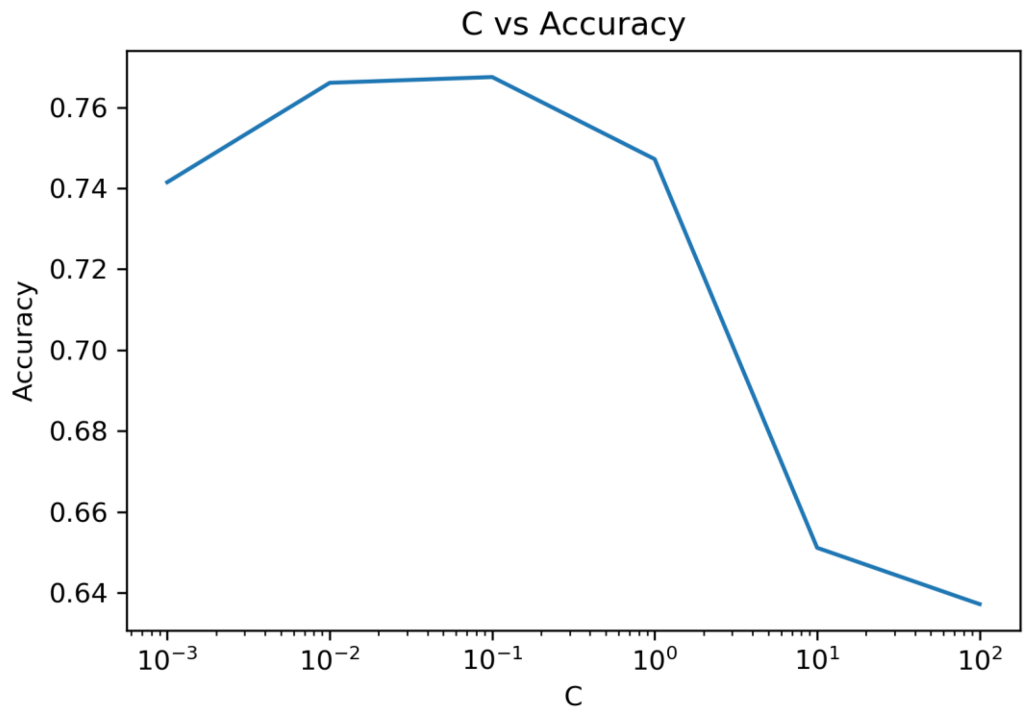
NO COMMENT

QUESTION 4

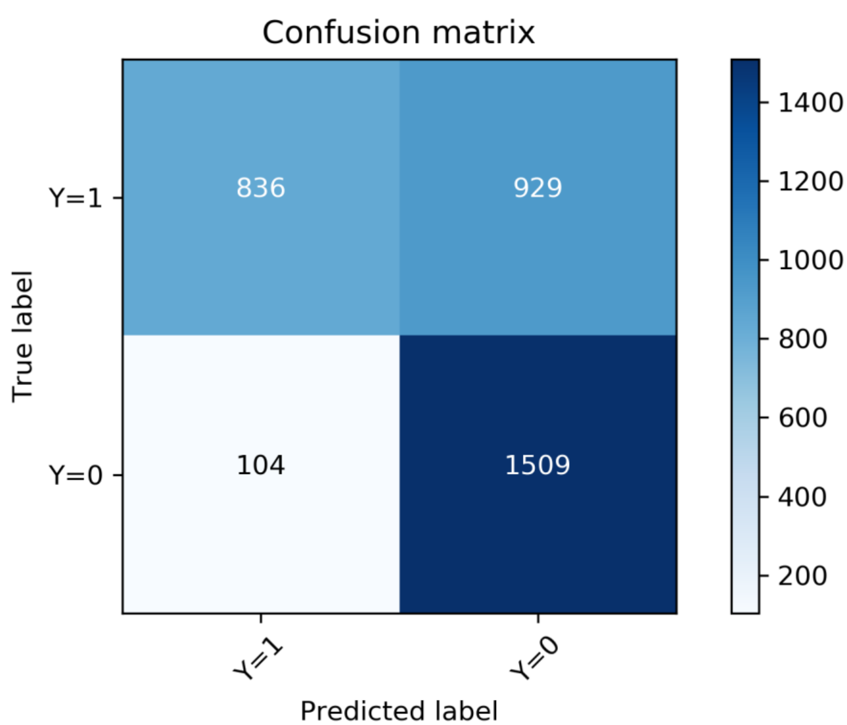
4.1

C : 0.1

Test Acc : 0.69419775014801655



I choose C : 0.1 Since it gives the best mean result 10 fold cross validation.



<u>**Macro Average**</u>

```
 'F1': 0.68155791842547253,
```

```
 'F2': 0.71022071016636379,
 'FDR': 0.29541087024863494,
 'FPR': 0.24584416944478382,
 'NPV': 0.70458912975136512,
 'Precision': 0.70458912975136512,
 'Recall': 0.75415583055521607
```

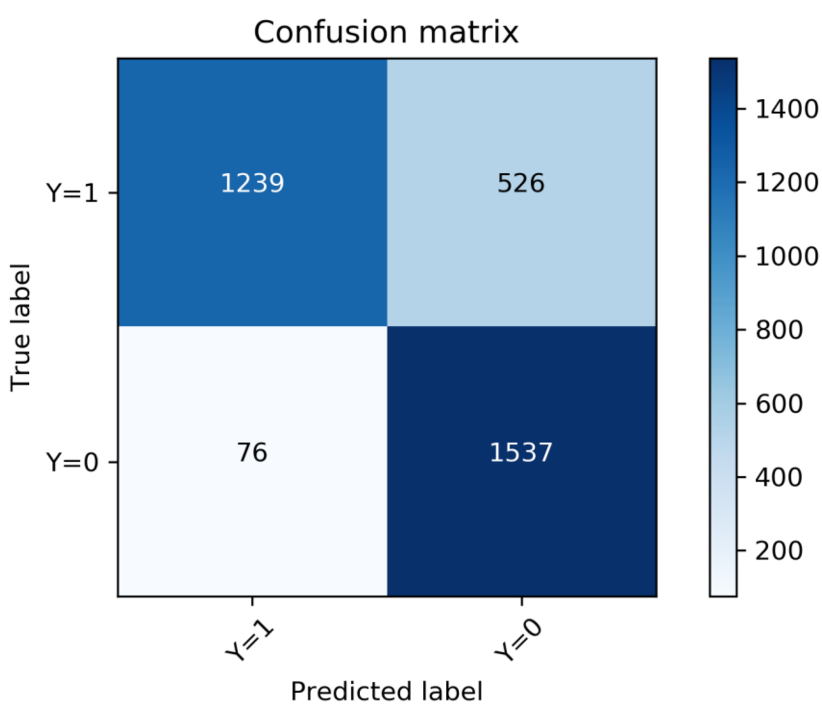
Micro Average

```
'F1': 0.69419775014801655,
 'F2': 0.69419775014801666,
 'FDR': 0.3058022498519834,
 'FPR': 0.3058022498519834,
 'NPV': 0.69419775014801655,
 'Precision': 0.69419775014801655,
 'Recall': 0.69419775014801655
```

4.2

In the question, we should train hard margin that C should be very high numbers. I choose C as 1000, C values that are bigger than that value may improve accuracy, however, overfitting and time complexity make the tradeoff, since time complexity is bigger than quadratic time complexity.

Gamma = 0.0625
Test Accuracy : 0.82178804026050922



Macro Average

```
'F1': 0.82039024631516466,
'F2': 0.83043362980305302,
'FDR': 0.17256708506838031,
'FPR': 0.15638158464637678,
'NPV': 0.82743291493161975,
'Precision': 0.82743291493161975,
'Recall': 0.84361841535362325
```

Micro Average

```
'F1': 0.82178804026050922,
'F2': 0.82178804026050922,
'FDR': 0.17821195973949083,
'FPR': 0.17821195973949083,
'NPV': 0.82178804026050922,
'Precision': 0.82178804026050922,
'Recall': 0.82178804026050922
```