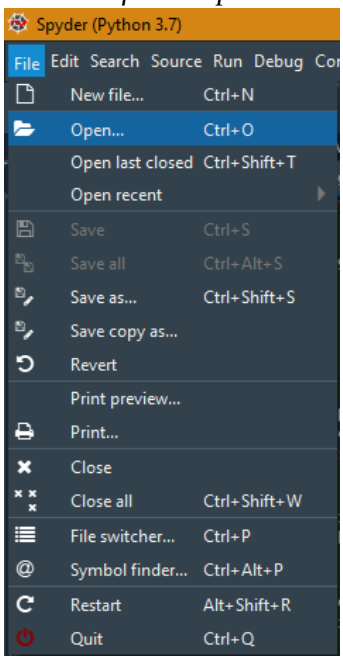
Irza Ramira Putra – 1810511100
Quina Alifa – 1810511115

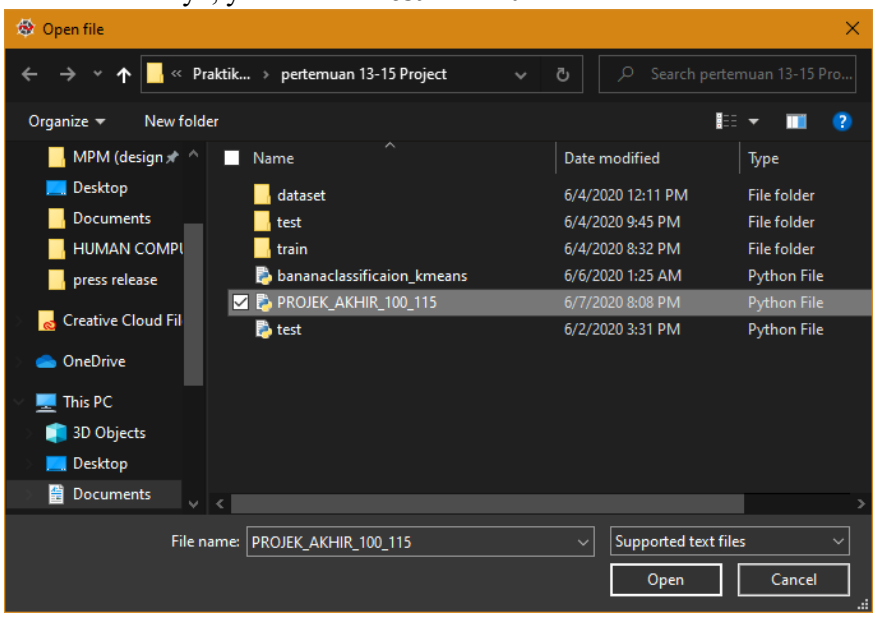
TUTORIAL MENJALANKAN PROGRAM
UAS PCD – KELAS D

---

1. Buka Spyder.
2. Lalu klik *file* > *Open*

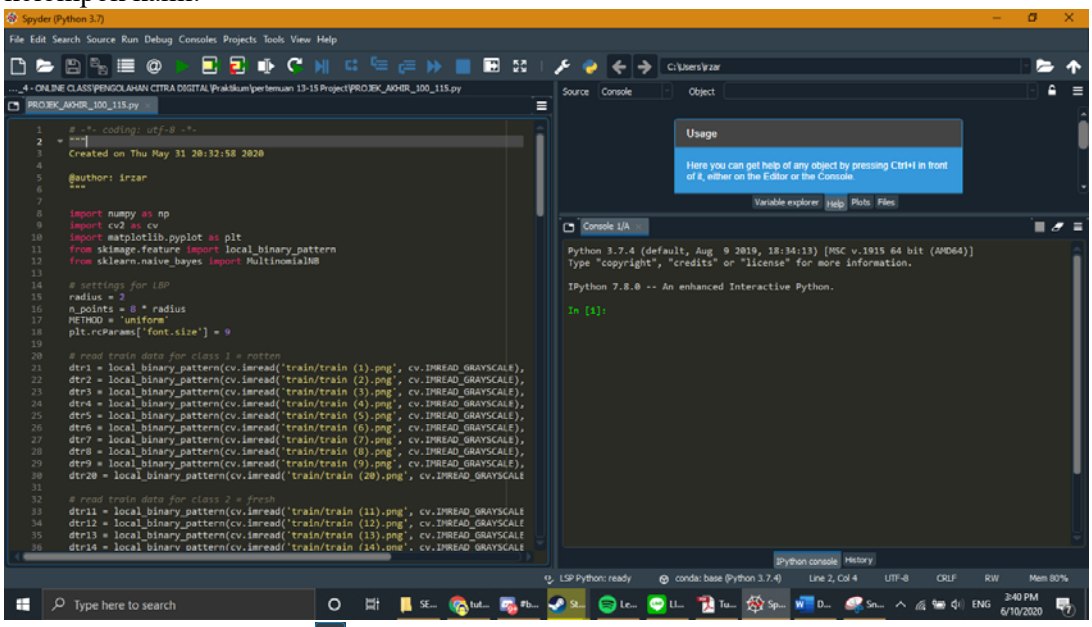


3. Pilih file *PROJEK_AKHIR_100_115.py* . Pastikan codingan kami sudah satu folder dengan folder datasetnya, yaitu folder *Test* dan *Train*



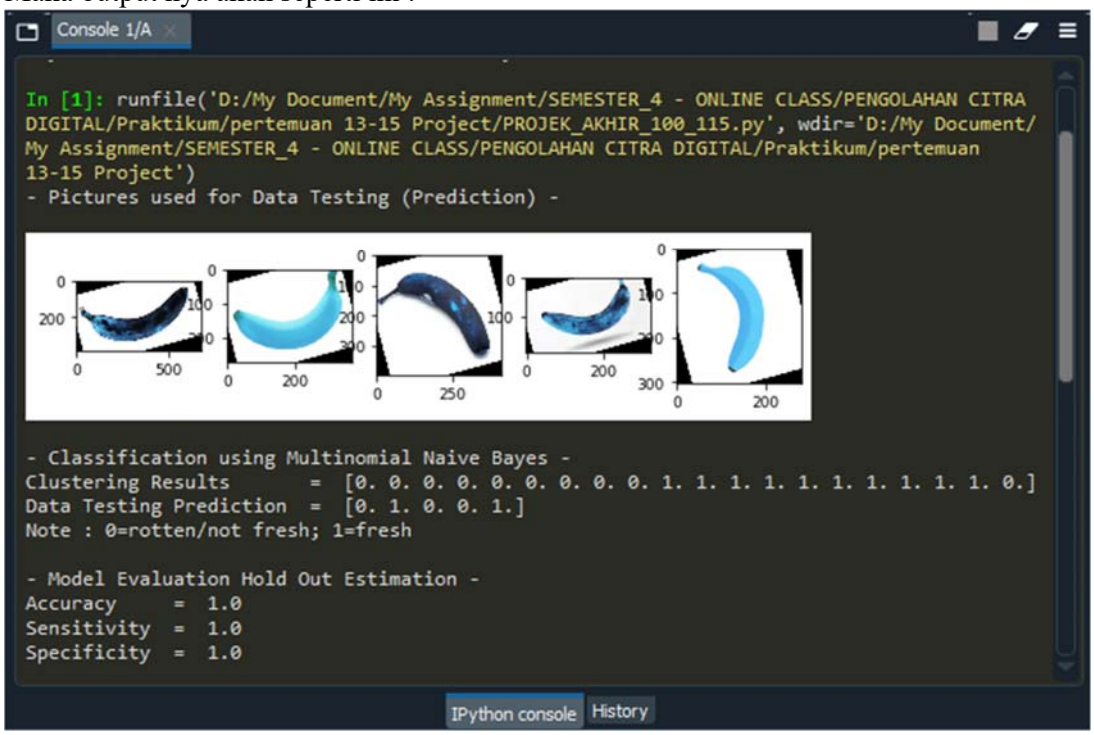
Lalu klik *Open*

4. Setelah terbuka file pythonnya, tampilan spyder akan terlihat seperti ini dengan codingan kelompok kami.



Maka bisa di klik tombol  (*Run File*) atau bisa menekan tombol) atau bisa menekan tombol *F5*.

5. Maka output nya akan seperti ini :



Tutorial Program :

1. Menyiapkan library yang dibutuhkan

```
import numpy as np
import cv2 as cv
import matplotlib.pyplot as plt
from skimage.feature import local_binary_pattern
from sklearn.naive_bayes import MultinomialNB
```

2. Membaca data file citra dan langsung melakukan ekstraksi dengan LBP

```
dtr1 = local_binary_pattern(cv.imread('train/train (1).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr2 = local_binary_pattern(cv.imread('train/train (2).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr3 = local_binary_pattern(cv.imread('train/train (3).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr4 = local_binary_pattern(cv.imread('train/train (4).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr5 = local_binary_pattern(cv.imread('train/train (5).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr6 = local_binary_pattern(cv.imread('train/train (6).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr7 = local_binary_pattern(cv.imread('train/train (7).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr8 = local_binary_pattern(cv.imread('train/train (8).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr9 = local_binary_pattern(cv.imread('train/train (9).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr20 = local_binary_pattern(cv.imread('train/train (20).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
```

```
dtr11 = local_binary_pattern(cv.imread('train/train (11).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr12 = local_binary_pattern(cv.imread('train/train (12).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr13 = local_binary_pattern(cv.imread('train/train (13).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr14 = local_binary_pattern(cv.imread('train/train (14).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr15 = local_binary_pattern(cv.imread('train/train (15).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr16 = local_binary_pattern(cv.imread('train/train (16).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr17 = local_binary_pattern(cv.imread('train/train (17).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr18 = local_binary_pattern(cv.imread('train/train (18).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr19 = local_binary_pattern(cv.imread('train/train (19).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtr10 = local_binary_pattern(cv.imread('train/train (10).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
```

3. Mendapatkan Histogram dari ekstraksi LBP tadi

```
# getting Histogram from LBP
dtrlbp1,bins = np.histogram(dtr1.ravel(),256,[0,256])
dtrlbp2,bins = np.histogram(dtr2.ravel(),256,[0,256])
dtrlbp3,bins = np.histogram(dtr3.ravel(),256,[0,256])
dtrlbp4,bins = np.histogram(dtr4.ravel(),256,[0,256])
dtrlbp5,bins = np.histogram(dtr5.ravel(),256,[0,256])
dtrlbp6,bins = np.histogram(dtr6.ravel(),256,[0,256])
dtrlbp7,bins = np.histogram(dtr7.ravel(),256,[0,256])
dtrlbp8,bins = np.histogram(dtr8.ravel(),256,[0,256])
dtrlbp9,bins = np.histogram(dtr9.ravel(),256,[0,256])
dtrlbp10,bins = np.histogram(dtr10.ravel(),256,[0,256])
dtrlbp11,bins = np.histogram(dtr11.ravel(),256,[0,256])
dtrlbp12,bins = np.histogram(dtr12.ravel(),256,[0,256])
dtrlbp13,bins = np.histogram(dtr13.ravel(),256,[0,256])
dtrlbp14,bins = np.histogram(dtr14.ravel(),256,[0,256])
dtrlbp15,bins = np.histogram(dtr15.ravel(),256,[0,256])
dtrlbp16,bins = np.histogram(dtr16.ravel(),256,[0,256])
dtrlbp17,bins = np.histogram(dtr17.ravel(),256,[0,256])
dtrlbp18,bins = np.histogram(dtr18.ravel(),256,[0,256])
dtrlbp19,bins = np.histogram(dtr19.ravel(),256,[0,256])
dtrlbp20,bins = np.histogram(dtr20.ravel(),256,[0,256])
```

4. Mengubah vector ke matriks dan mentranspose kan histogram tadi

```
dtrlbp1 = np.transpose(dtrlbp1[0:18,np.newaxis])
dtrlbp2 = np.transpose(dtrlbp2[0:18,np.newaxis])
dtrlbp3 = np.transpose(dtrlbp3[0:18,np.newaxis])
dtrlbp4 = np.transpose(dtrlbp4[0:18,np.newaxis])
dtrlbp5 = np.transpose(dtrlbp5[0:18,np.newaxis])
dtrlbp6 = np.transpose(dtrlbp6[0:18,np.newaxis])
dtrlbp7 = np.transpose(dtrlbp7[0:18,np.newaxis])
dtrlbp8 = np.transpose(dtrlbp8[0:18,np.newaxis])
dtrlbp9 = np.transpose(dtrlbp9[0:18,np.newaxis])
dtrlbp10 = np.transpose(dtrlbp10[0:18,np.newaxis])
dtrlbp11 = np.transpose(dtrlbp11[0:18,np.newaxis])
dtrlbp12 = np.transpose(dtrlbp12[0:18,np.newaxis])
dtrlbp13 = np.transpose(dtrlbp13[0:18,np.newaxis])
dtrlbp14 = np.transpose(dtrlbp14[0:18,np.newaxis])
dtrlbp15 = np.transpose(dtrlbp15[0:18,np.newaxis])
dtrlbp16 = np.transpose(dtrlbp16[0:18,np.newaxis])
dtrlbp17 = np.transpose(dtrlbp17[0:18,np.newaxis])
dtrlbp18 = np.transpose(dtrlbp18[0:18,np.newaxis])
dtrlbp19 = np.transpose(dtrlbp19[0:18,np.newaxis])
dtrlbp20 = np.transpose(dtrlbp20[0:18,np.newaxis])
```

5. Menggabungkan semua data yang sudah ditranspose menjadi satu untuk data *training*.

```
traindata = np.concatenate((dtrlbp1, dtrlbp2, dtrlbp3, dtrlbp4, dtrlbp5,
                            dtrlbp6, dtrlbp7, dtrlbp8, dtrlbp9, dtrlbp10,
                            dtrlbp11, dtrlbp12, dtrlbp13, dtrlbp14, dtrlbp15,
                            dtrlbp16, dtrlbp17, dtrlbp18, dtrlbp19, dtrlbp20),
                           axis=0).astype(np.float32)
```

6. Membuat label asli dari data *training* untuk klasifikasi menggunakan *Naïve Bayes* nanti.

```
label = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0]).astype(np.float32)
```

Dengan keterangan 0 untuk pisang yang busuk, 1 untuk pisang yang segar

7. Membuat data *test* (dengan metode sama seperti no.2-4)

```
dtst1 = local_binary_pattern(cv.imread('test/test (1).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtstlbp1,bins = np.histogram(dtst1.ravel(),256,[0,256])
dtstlbp1 = np.transpose(dtstlbp1[0:18,np.newaxis])
dtst2 = local_binary_pattern(cv.imread('test/test (2).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtstlbp2,bins = np.histogram(dtst2.ravel(),256,[0,256])
dtstlbp2 = np.transpose(dtstlbp2[0:18,np.newaxis])
dtst3 = local_binary_pattern(cv.imread('test/test (3).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtstlbp3,bins = np.histogram(dtst3.ravel(),256,[0,256])
dtstlbp3 = np.transpose(dtstlbp3[0:18,np.newaxis])
dtst4 = local_binary_pattern(cv.imread('test/test (4).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtstlbp4,bins = np.histogram(dtst4.ravel(),256,[0,256])
dtstlbp4 = np.transpose(dtstlbp4[0:18,np.newaxis])
dtst5 = local_binary_pattern(cv.imread('test/test (5).png', cv.IMREAD_GRAYSCALE), n_points, radius, METHOD)
dtstlbp5,bins = np.histogram(dtst5.ravel(),256,[0,256])
dtstlbp5 = np.transpose(dtstlbp5[0:18,np.newaxis])
```

8. Menggabungkan semua data yang sudah ditranspose menjadi satu untuk data *testing*.

```
testdata = np.concatenate((dtstlbp1, dtstlbp2, dtstlbp3, dtstlbp4, dtstlbp5),
                          axis=0).astype(np.float32)
```

9. Menunjukkan Citra *Test* yang digunakan pada console

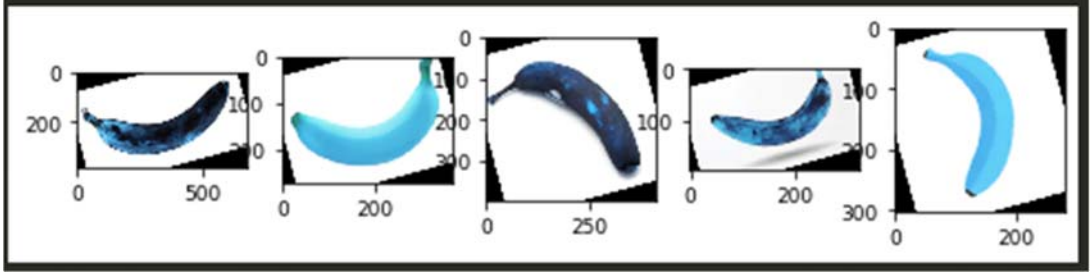
```
pic = ['',cv.imread('test/test (1).png', cv.IMREAD_COLOR),
        cv.imread('test/test (2).png', cv.IMREAD_COLOR),
        cv.imread('test/test (3).png', cv.IMREAD_COLOR),
        cv.imread('test/test (4).png', cv.IMREAD_COLOR),
        cv.imread('test/test (5).png', cv.IMREAD_COLOR)]

fig=plt.figure(figsize=(8, 8))
columns = 5
rows = 1

print('- Pictures used for Data Testing (Prediction) -')
for i in range(1, 6):
    fig.add_subplot(rows, columns, i)
    plt.imshow(pic[i])
plt.show()
print()
```

Menghasilkan Output :

```
- Pictures used for Data Testing (Prediction) -
```



10. Melakukan klasifikasi dengan *Naïve Bayes*

```
nb = MultinomialNB()
nb.fit(traindata, label)
result = nb.predict(traindata)
predict = nb.predict(testdata)
print('- Classification using Multinomial Naive Bayes -')
print("Clustering Results       = ", result)
print("Data Testing Prediction  = ", predict)
print('Note : 0=rotten/not fresh; 1=fresh')
print()
```

Menghasilkan Output :

```
- Classification using Multinomial Naive Bayes -
Clustering Results       =  [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 0.]
Data Testing Prediction  =  [0. 1. 0. 0. 1.]
Note : 0=rotten/not fresh; 1=fresh
```

11. Membuat fungsi *Confussion Matrix*

```
149  def Conf_matrix(y_actual, y_pred):
150      TP = 0
151      FP = 0
152      TN = 0
153      FN = 0
154
155      for i in range(len(y_pred)):
156          if y_actual[i]==y_pred[i]==1:
157              TP += 1
158          if y_pred[i]==1 and y_actual[i] !=y_pred[i]:
159              FP += 1
160          if y_actual[i]==y_pred[i]==0:
161              TN += 1
162          if y_pred[i]==0 and y_actual[i]!= y_pred[i]:
163              FN += 1
164      return (TP, FN, TN, FP)
```

12. Melakukan Evaluasi Model dengan *Hold Out Estimation*

```
167  TP, FN, TN, FP = Conf_matrix(label, result)
168
169  print('- Model Evaluation Hold Out Estimation -')
170  print('Accuracy     = ', (TP+TN)/(TP+TN+FP+FN))
171  print('Sensitivity  = ', TP/(TP+FN))
172  print('Specificity  = ', TN/(TN+FP))
```

Menghasilkan Output :

```
- Model Evaluation Hold Out Estimation -
Accuracy     =  1.0
Sensitivity  =  1.0
Specificity  =  1.0
```