D. KHLOUD AL JALLAD

S. SABRIN ALHRIRI

US_COVID19_DAILY

index

Dataset description

Context

Data is obtained from COVID-19 Tracking project and NYTimes. Sincere thanks to them for making it available to the public.

Coronaviruses are a large family of viruses which may cause illness in animals or humans. In humans, several coronaviruses are known to cause respiratory infections ranging from the common cold to more severe diseases such as Middle East Respiratory Syndrome (MERS) and Severe Acute Respiratory Syndrome (SARS). The most recently discovered coronavirus causes coronavirus disease COVID-19 - World Health Organization

The number of new cases are increasing day by day around the world. This dataset has information from 50 US states and the District of Columbia at daily level.

us_states_covid19_daily.csv

Main Content : us_states_covid19_daily.csv

This dataset has number of tests conducted in each state at daily level. Column descriptions are

date - date of observation

state - US state 2 digit code

positive - number of tests with positive results

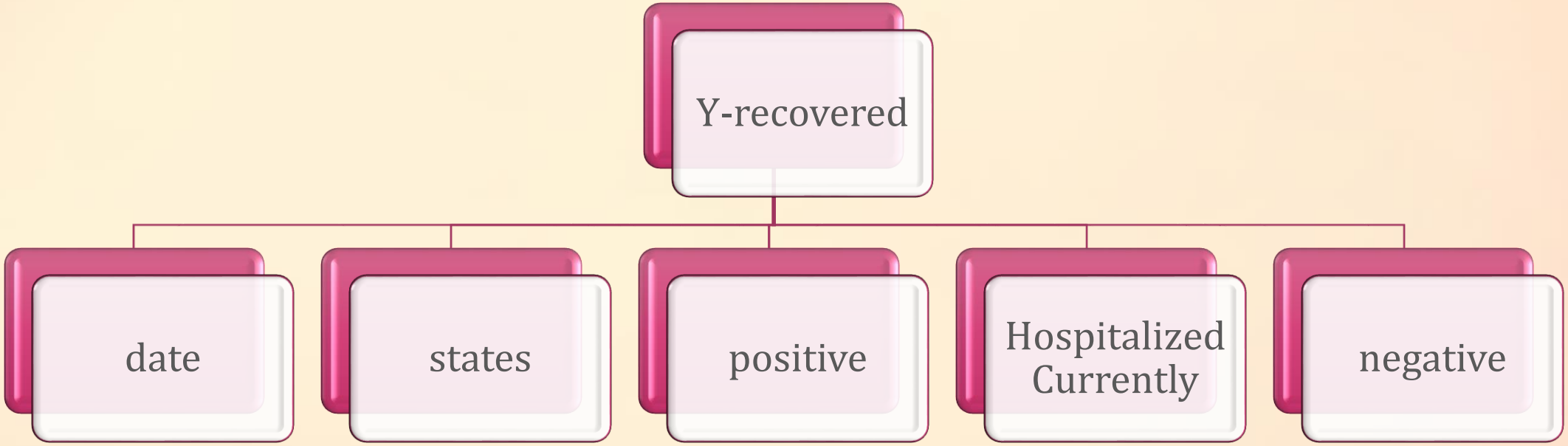
negative - number of tests with negative results

pending - number of test with pending results

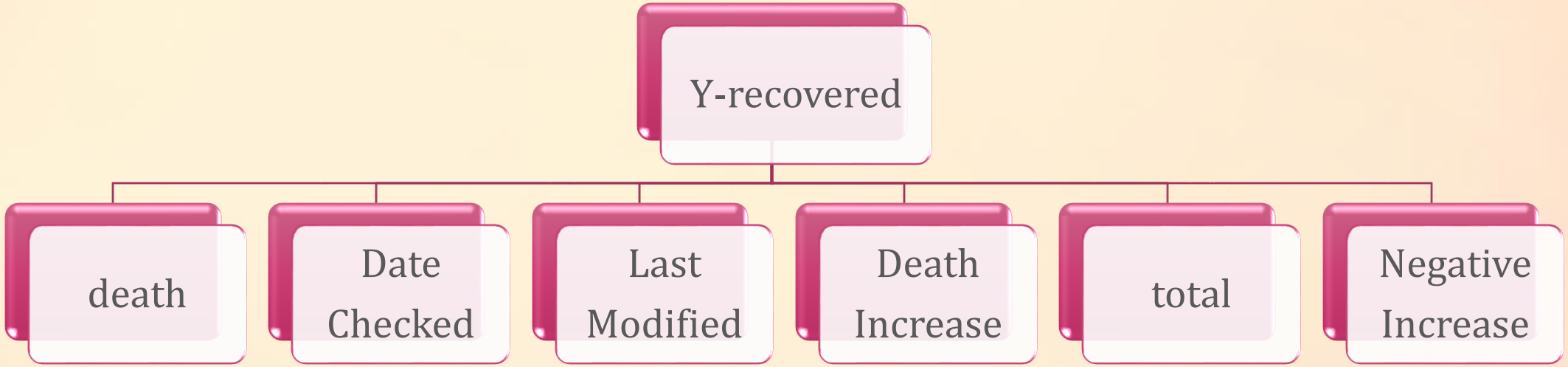
death - number of deaths

total - total number of tests

feature (column).



feature (column).



STEPS WORK

1. Libraries.
2. Show Data.
3. Clean DATA 'missing value‘ .
4. PROCESSING “CONVERT TO NUMBER ONLY ”.
5. PLOT DATA .
6. Split data & work on

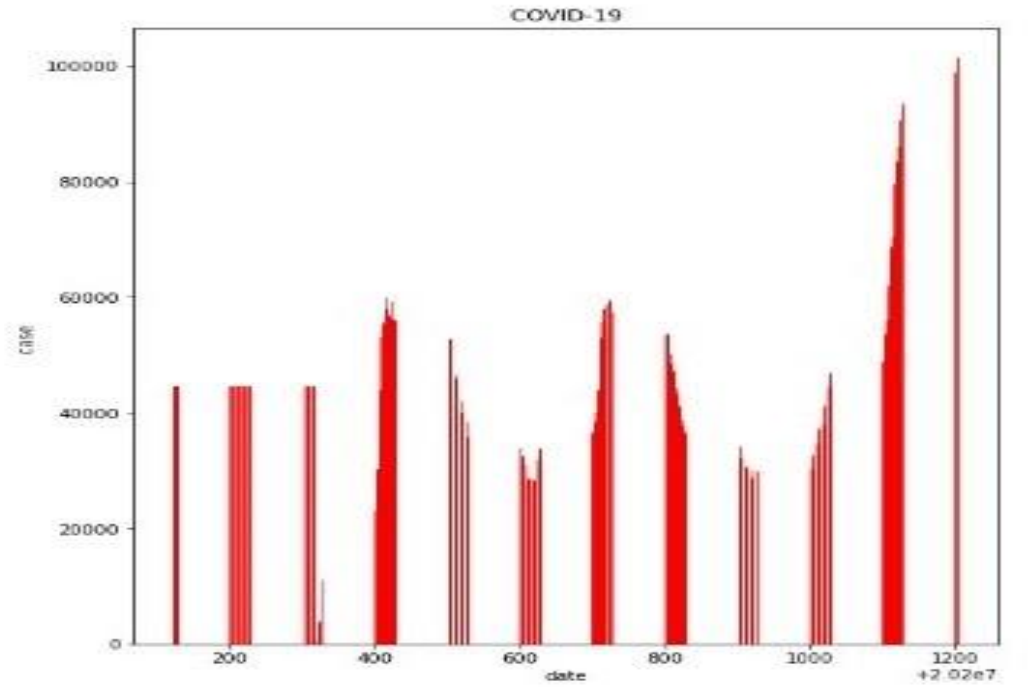
Split data & work on

- 1 – spilt data to train and test (80%- 20%).
- 2-work on data : all Algorithms :

1. SVM.
2. CLUSTERING.
3. Metrics with(D_tree, SVM ,)
4. Decision Tree.
5. Kneighbors Classifier
6. Random Forest Classifier

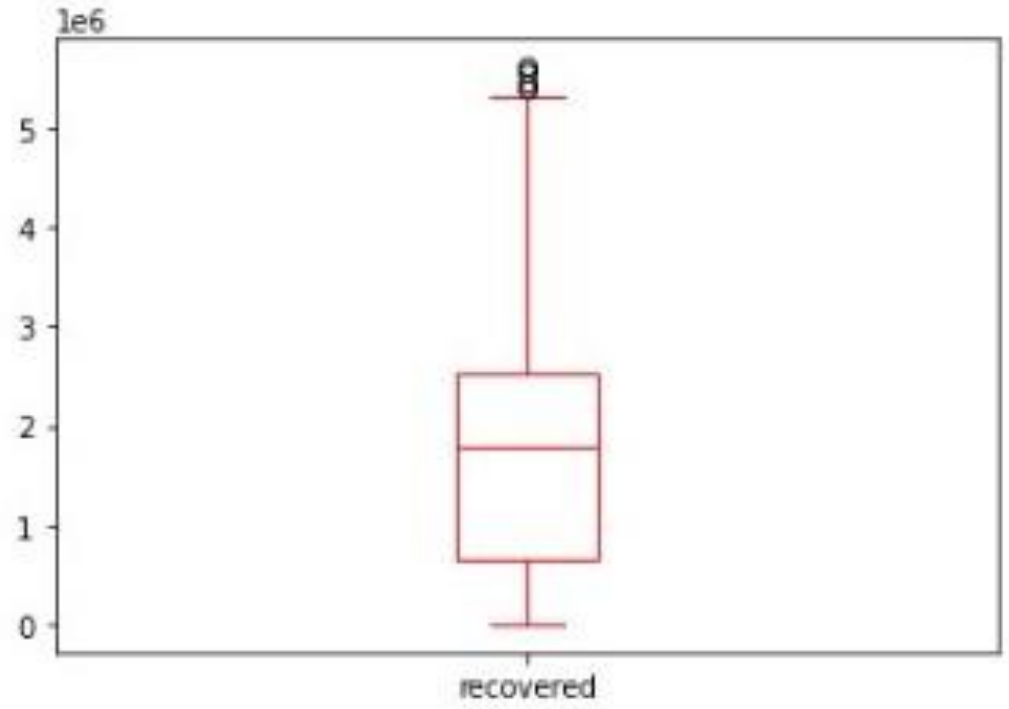
PLOT Data :D

```
In [39]: ## state + recovered
         plt.figure(figsize=(8, 9))
         plt.bar(df['date'], df['hospitalizedCurrently'], color='red')
         plt.title(' COVID-19 ')
         plt.xlabel('date')
         plt.ylabel('case ')

Out[39]: Text(0, 0.5, 'case ')
```



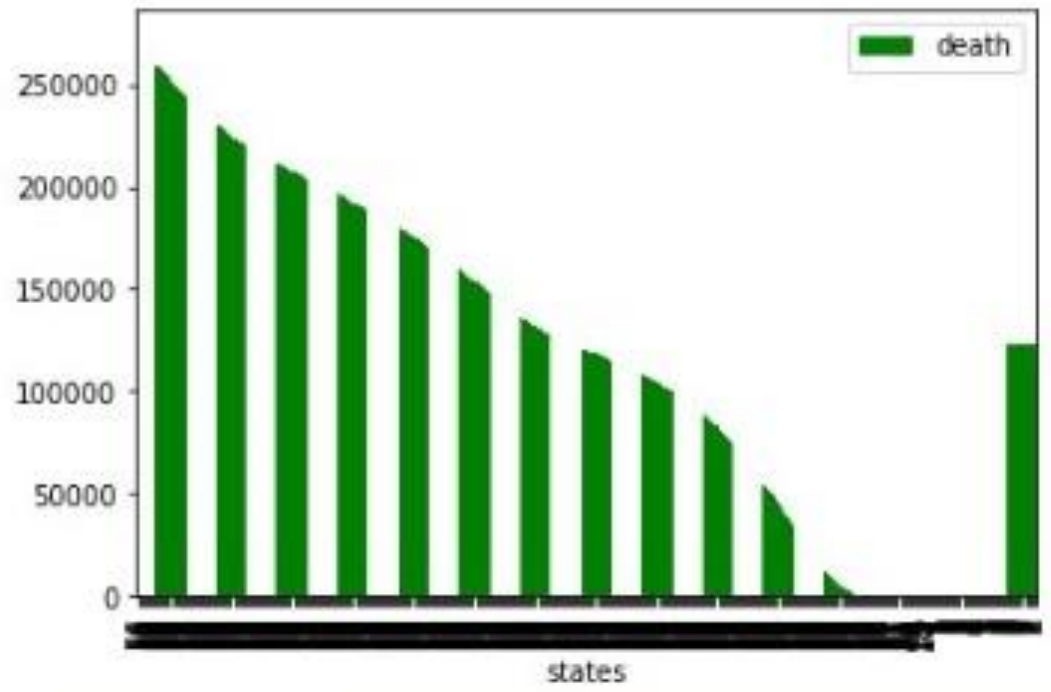
```
[133]: ### the state with code 56 :
       df.plot('states' , 'recovered' ,kind='box',color ='red')
```

```
Out[133]: <matplotlib.axes._subplots.AxesSubplot at 0x1f41aa3ad60>
```



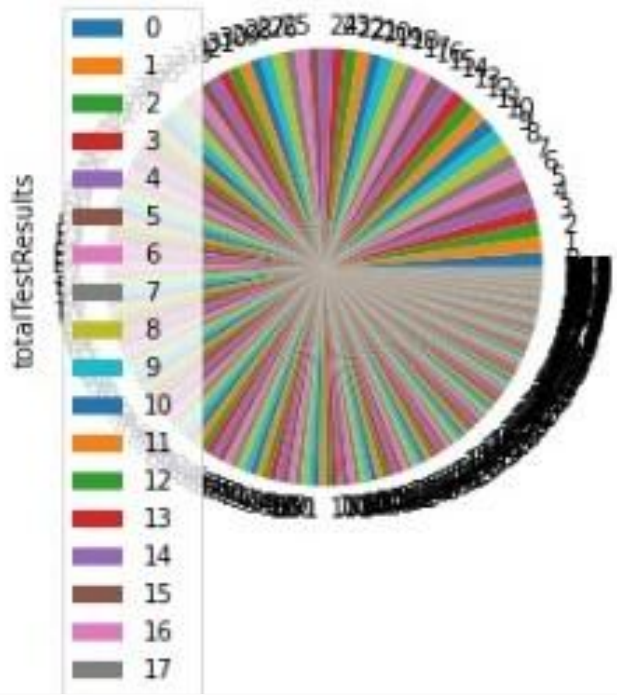
```
In [42]: ### the state with code 56 :
         df.plot('states' , 'death' ,kind='bar',color ='green')
```

```
Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x21e06d49580>
```



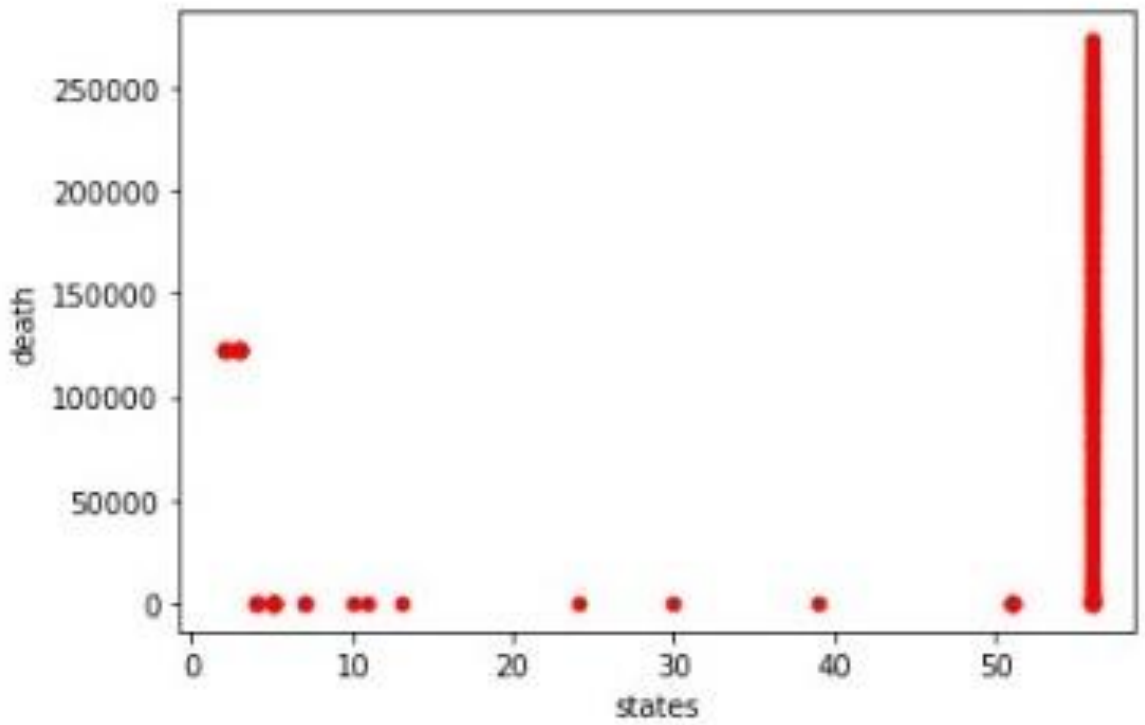
```
[41]: ###https://www.kaggle.com/philrowe/cheatsheet-70-ggplot-charts-a67fec
       # dad to find resulte-.-

      df.plot('states' , 'totalTestResults' ,kind='pie')
```

```
Out[41]: <matplotlib.axes._subplots.AxesSubplot at 0x21e063aa880>
```



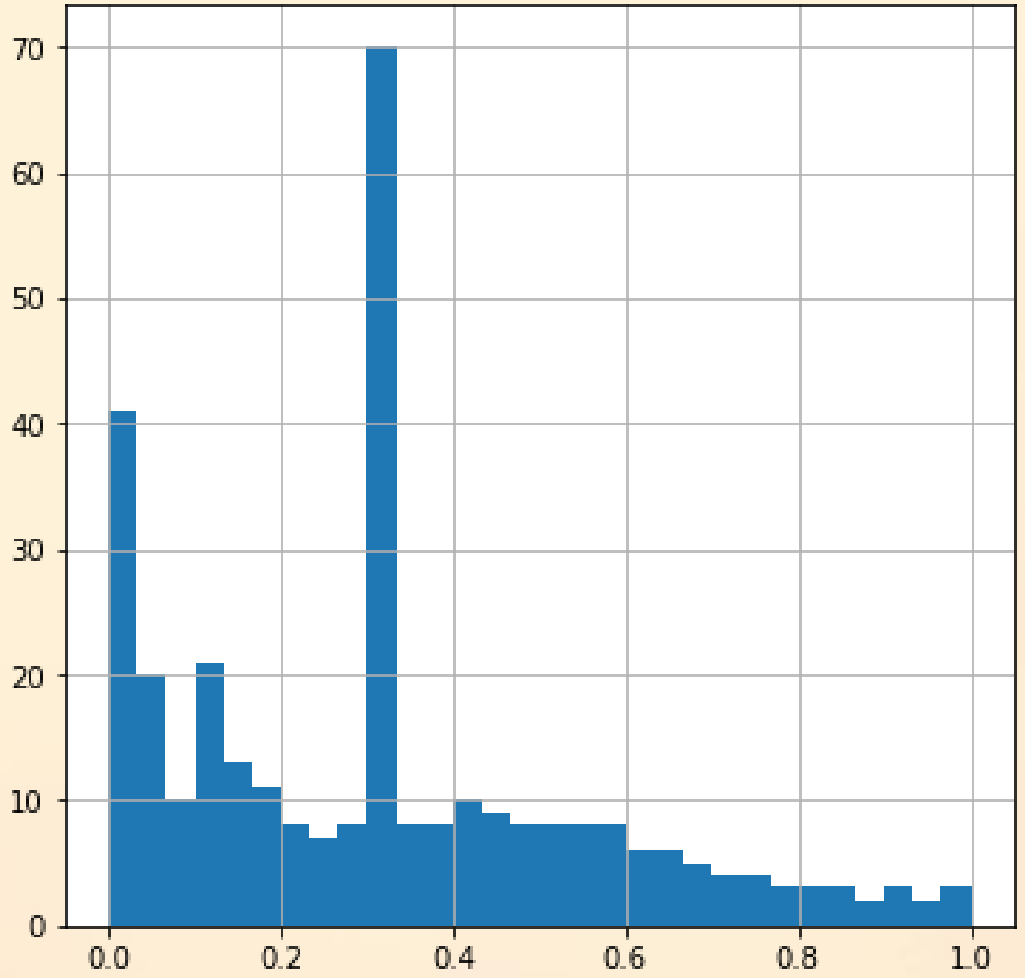
In [131]:
```
### the state with code 56 :
df.plot('states' , 'death' ,kind='scatter',color ='red')
```

Out[131]: <matplotlib.axes._subplots.AxesSubplot at 0x1f41e9e2ac0>



In [43]:
```
#correlation map
import seaborn as sns # visualization tool
f,ax = plt.subplots(figsize=(18, 18))
sns.heatmap(df.corr(), annot=True, linewidths=.5, fmt= '.1f',ax=ax)
plt.show()
```

Split data & work on : “reg”
SPLIT TO 5 CLASSES”classfition”



RBF-kernel svc

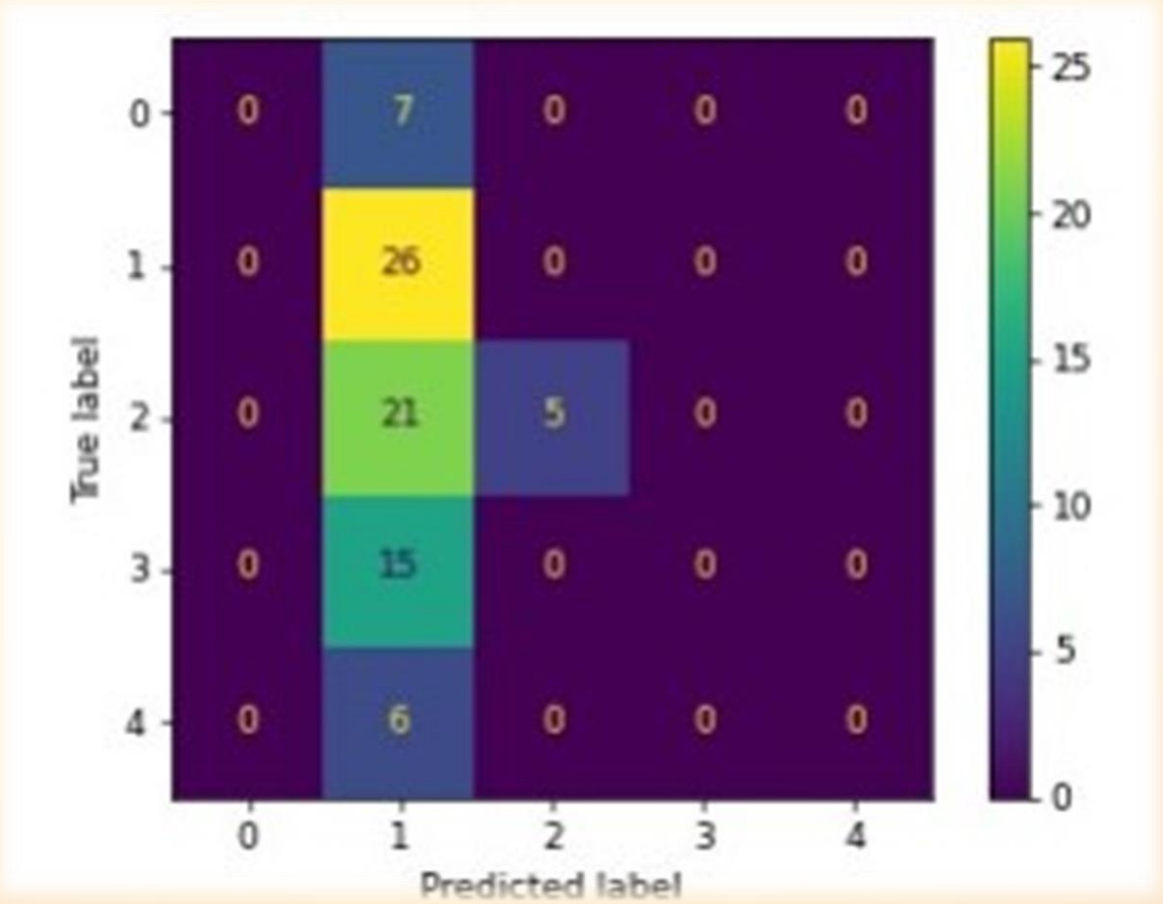
- **1-The first "result ": under fitting**

```
Accuracy of RBF-kernel SVC on training set: 0.38
Accuracy of RBF-kernel SVC on test set: 0.33
Out[45]:
SVC(C=0.001, gamma=1)
```

- **2-The second "over fitting ":**

```
 Accuracy of RBF-kernel SVC on training set: 1.00
Accuracy of RBF-kernel SVC on test set: 0.39
Out[43]:
SVC(C=1, gamma=0.5)
```

Metrics : the data un balance
class 2 take the large amount from data



accuracy_score, precision_score, recall_score, f1_score
bad training

f1_score

```
1-
Micro-averaged f1 = 0.39 (treat
instances equally)
Macro-averaged f1 = 0.17 (treat
classes equally)
```

precision–recall score

```
2-Micro-averaged precision = 0.39
(treat instances equally) Macro-
averaged precision = 0.27 (treat
classes equally)

3-Micro-averaged recall = 0.39
(treat instances equally) Macro-
averaged recall = 0.24 (treat
classes equally)
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	7
1	0.35	1.00	0.51	26
2	1.00	0.19	0.32	26
3	0.00	0.00	0.00	15
4	0.00	0.00	0.00	6
accuracy			0.39	80
macro avg	0.27	0.24	0.17	80
weighted avg	0.44	0.39	0.27	80

tree& confusion_matrix :
1-the first : max_depth = 5 , “result”: over fitting

In [66]:
```
from sklearn.tree import DecisionTreeClassifier
Xt_train, Xt_test, yt_train, yt_test = train_test_split(X_data, y_data,test_size=0.2, random_state = 0)
dt = DecisionTreeClassifier(max_depth=5).fit(Xt_train, yt_train)
tree_predicted = dt.predict(Xt_test)
confusion = confusion_matrix(yt_test, tree_predicted)
print('Accuracy of D_t on test set: {:.2f}'
     .format(dt.score(Xt_test, yt_test)))
print('Accuracy of D_t on test set: {:.2f}'
     .format(dt.score(Xt_train, yt_train)))

print('Decision tree classifier (max_depth = 2)\n', confusion)
```

```
Accuracy of D_t on test set: 1.00
Accuracy of D_t on test set: 1.00
Decision tree classifier (max_depth = 2)
 [[ 6  0  0  0  0]
 [ 0 21  0  0  0]
 [ 0  0 19  0  0]
 [ 0  0  0 14  0]
 [ 0  0  0  0  4]]
```

D_tree& confusion_matrix :
2-the second: max_depth = 2 , “result”: good training

```
from sklearn.tree import DecisionTreeClassifier
Xt_train, Xt_test, yt_train, yt_test = train_test_split(X_data, y_data,test_size=0.2, random_state = 0)
dt = DecisionTreeClassifier(max_depth=3).fit(Xt_train, yt_train)
tree_predicted = dt.predict(Xt_test)
confusion = confusion_matrix(yt_test, tree_predicted)
print('Accuracy of D_t on test set: {:.2f}'
     .format(dt.score(Xt_test, yt_test)))
print('Accuracy of D_t on test set: {:.2f}'
     .format(dt.score(Xt_train, yt_train)))

print('Decision tree classifier (max_depth = 2)\n', confusion)
```

```
Accuracy of D_t on test set: 0.91
Accuracy of D_t on test set: 0.96
Decision tree classifier (max_depth = 2)
 [[ 0  0  0  0  6]
 [ 0 21  0  0  0]
 [ 0  0 19  0  0]
 [ 0  0  0 14  0]
 [ 0  0  0  0  4]]
```

svc & D_tree & confusion_matrix good training

```
Accuracy of RBF-kernel SVC on test set: 0.98
Accuracy of RBF-kernel SVC on test set: 1.00
Decision tree classifier (max_depth = 2)
 [[ 5  0  0  0  1]
 [ 0 21  0  0  0]
 [ 0  0 19  0  0]
 [ 0  0  0 14  0]
 [ 0  0  0  0  4]]
```

Kneighbors Classifier
Goog training

```
In [102]: from sklearn.neighbors import KNeighborsClassifier

          model = KNeighborsClassifier(n_neighbors=3)
          X_train, X_test, y_train, y_test = train_test_split(X_data, np.ravel(y_data,order='C'), random_state = 3)
          model.fit(X_train, y_train)
          print('Accuracy of  KNeighborsClassifier classifier on training set: {:.2f}'
               .format(model.score(X_train, y_train)))
          print('Accuracy of  KNeighborsClassifier classifier on test set: {:.2f}'
               .format(model.score(X_test, y_test)))

          Accuracy of  KNeighborsClassifier classifier on training set: 0.90
          Accuracy of  KNeighborsClassifier classifier on test set: 0.79
```

Random Forest Classifier
Very Good training :D

In [105]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification
clf3 = RandomForestClassifier(max_depth=2, random_state=0)
clf3.fit(X_data,y_data)
print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf3.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf3.score(X_test, y_test)))
```

```
Accuracy of Decision Tree classifier on training set: 0.96
Accuracy of Decision Tree classifier on test set: 0.91
```

Decision Tree Classifier

In [104]:
```
from sklearn.tree import DecisionTreeClassifier
clf2 = DecisionTreeClassifier(max_depth = 30).fit(X_train, y_train)

print('Accuracy of Decision Tree classifier on training set: {:.2f}'
     .format(clf2.score(X_train, y_train)))
print('Accuracy of Decision Tree classifier on test set: {:.2f}'
     .format(clf2.score(X_test, y_test)))
```

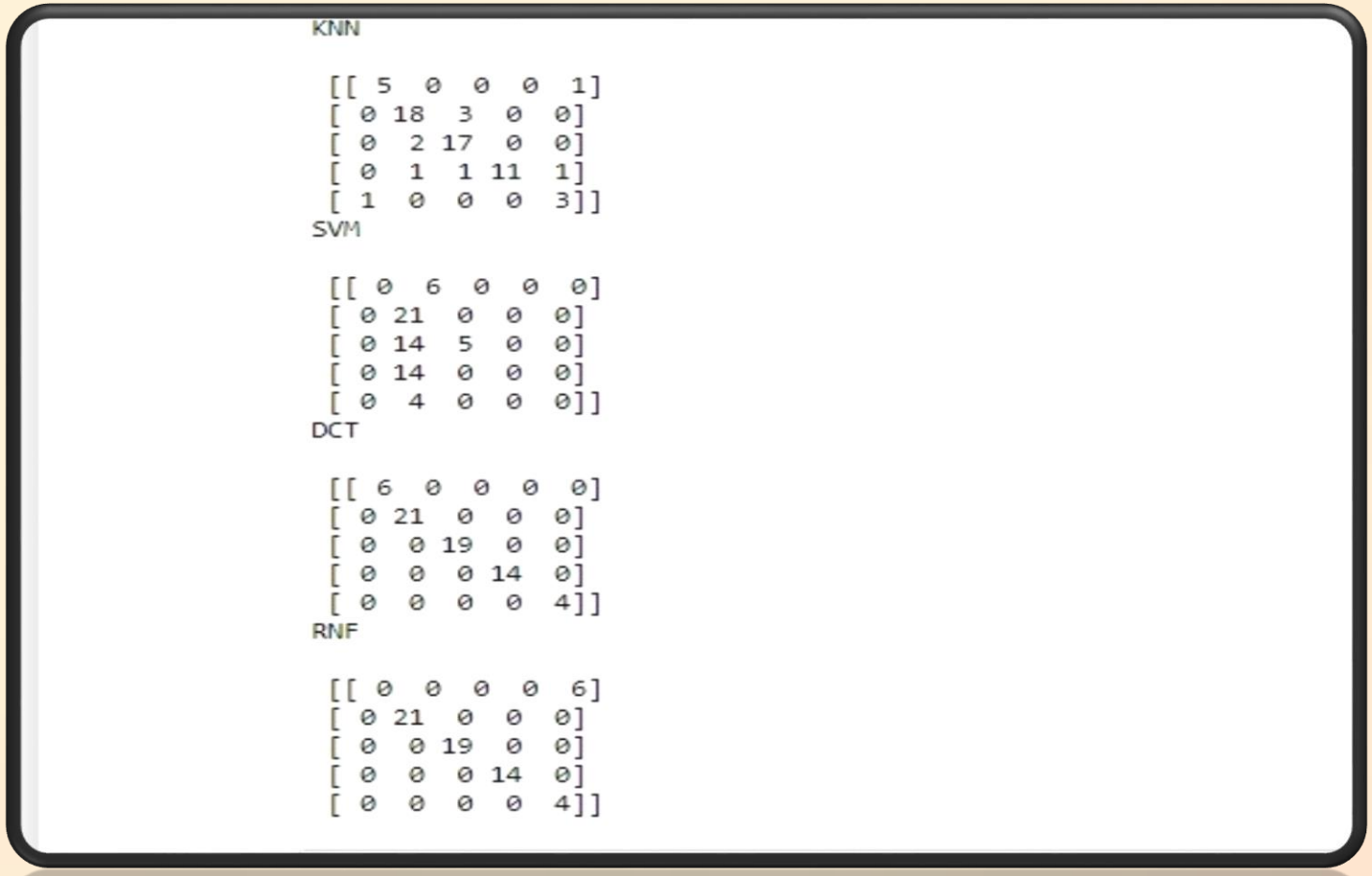
```
Accuracy of Decision Tree classifier on training set: 1.00
Accuracy of Decision Tree classifier on test set: 1.00
```

RBF-kernel SVC

In [101]:
```
from sklearn.model_selection import train_test_split
from sklearn.datasets import make_classification, make_blobs
np.set_printoptions(precision=2)
from sklearn.svm import SVC
#from adspy_shared_utilities import plot_class_regions_for_classifier_subplot
X_train, X_test, y_train, y_test = train_test_split(X_data, np.ravel(y_data,order='C'),test_size = 0.2,random_state = 0)
clf = SVC( C=0.001,gamma=0.1).fit(X_train, y_train)
print('Accuracy of RBF-kernel SVC on training set: {:.2f}'
     .format(clf.score(X_train, y_train)))
print('Accuracy of RBF-kernel SVC on test set: {:.2f}'
     .format(clf.score(X_test, y_test)))
```

```
Accuracy of RBF-kernel SVC on training set: 0.37
Accuracy of RBF-kernel SVC on test set: 0.33
```

The best “ DCT & RNF“



Svm &knn baaaaaaad

A1	0.83	0.83	0.83	6
A15	0.86	0.86	0.86	21
A7	0.81	0.89	0.85	19
B2	1.00	0.79	0.88	14
B3	0.60	0.75	0.67	4
C11	0.00	0.00	0.00	0
C6	0.00	0.00	0.00	0
D14	0.00	0.00	0.00	0
D4	0.00	0.00	0.00	0
E13	0.00	0.00	0.00	0
E18	0.00	0.00	0.00	0
E5	0.00	0.00	0.00	0
micro avg	0.84	0.84	0.84	64
macro avg	0.34	0.34	0.34	64
weighted avg	0.86	0.84	0.85	64

SVM

	precision	recall	f1-score	support
A1	0.00	0.00	0.00	6
A15	0.36	1.00	0.53	21
A7	1.00	0.26	0.42	19
B2	0.00	0.00	0.00	14
B3	0.00	0.00	0.00	4
C11	0.00	0.00	0.00	0
C6	0.00	0.00	0.00	0
D14	0.00	0.00	0.00	0
D4	0.00	0.00	0.00	0
E13	0.00	0.00	0.00	0
E18	0.00	0.00	0.00	0
E5	0.00	0.00	0.00	0
micro avg	0.41	0.41	0.41	64
macro avg	0.11	0.11	0.08	64
weighted avg	0.41	0.41	0.30	64

Random Forest

DCT & RNF

Random Forest

	precision	recall	f1-score	support
A1	0.00	0.00	0.00	6
A15	1.00	1.00	1.00	21
A7	1.00	1.00	1.00	19
B2	1.00	1.00	1.00	14
B3	0.40	1.00	0.57	4
C11	0.00	0.00	0.00	0
C6	0.00	0.00	0.00	0
D14	0.00	0.00	0.00	0
D4	0.00	0.00	0.00	0
E13	0.00	0.00	0.00	0
E18	0.00	0.00	0.00	0
E5	0.00	0.00	0.00	0
micro avg	0.91	0.91	0.91	64
macro avg	0.28	0.33	0.30	64
weighted avg	0.87	0.91	0.88	64

Decision tree

	precision	recall	f1-score	support
A1	1.00	1.00	1.00	6
A15	1.00	1.00	1.00	21
A7	1.00	1.00	1.00	19
B2	1.00	1.00	1.00	14
B3	1.00	1.00	1.00	4
C11	0.00	0.00	0.00	0
C6	0.00	0.00	0.00	0
D14	0.00	0.00	0.00	0
D4	0.00	0.00	0.00	0
E13	0.00	0.00	0.00	0
E18	0.00	0.00	0.00	0
E5	0.00	0.00	0.00	0
micro avg	1.00	1.00	1.00	64
macro avg	0.42	0.42	0.42	64
weighted avg	1.00	1.00	1.00	64

RNF & DCT:BAD “unbalance Data”
THE RNF BETTER THAN DCT

ON RNF

Macro : IN THE NUM_CLASSES

THE RESULT : THE DATA TRAINING ONLY IN CLASS “2 ”.

```
micro avg 0.91 0.91 0.91 64
macro avg 0.28 0.33 0.30 64
weighted avg 0.87 0.91 0.88 64
```

On DCT

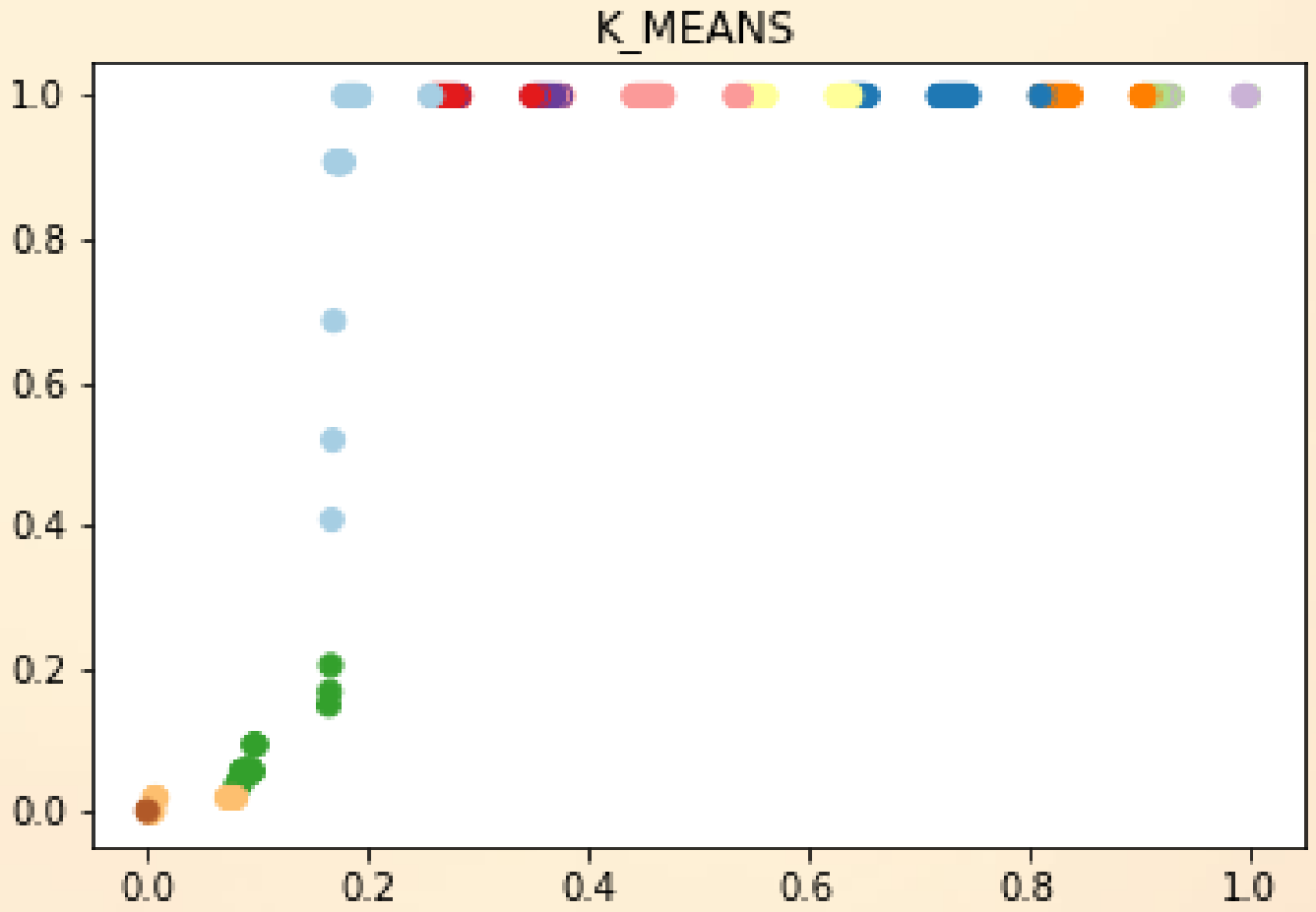
Macro : IN THE NUM_CLASSES

THE RESULT : THE DATA TRAINING ONLY IN CLASS “2

```
micro avg 1.00 1.00 1.00 64
macro avg 0.42 0.42 0.42 64
weighted avg 1.00 1.00 1.00 64
```

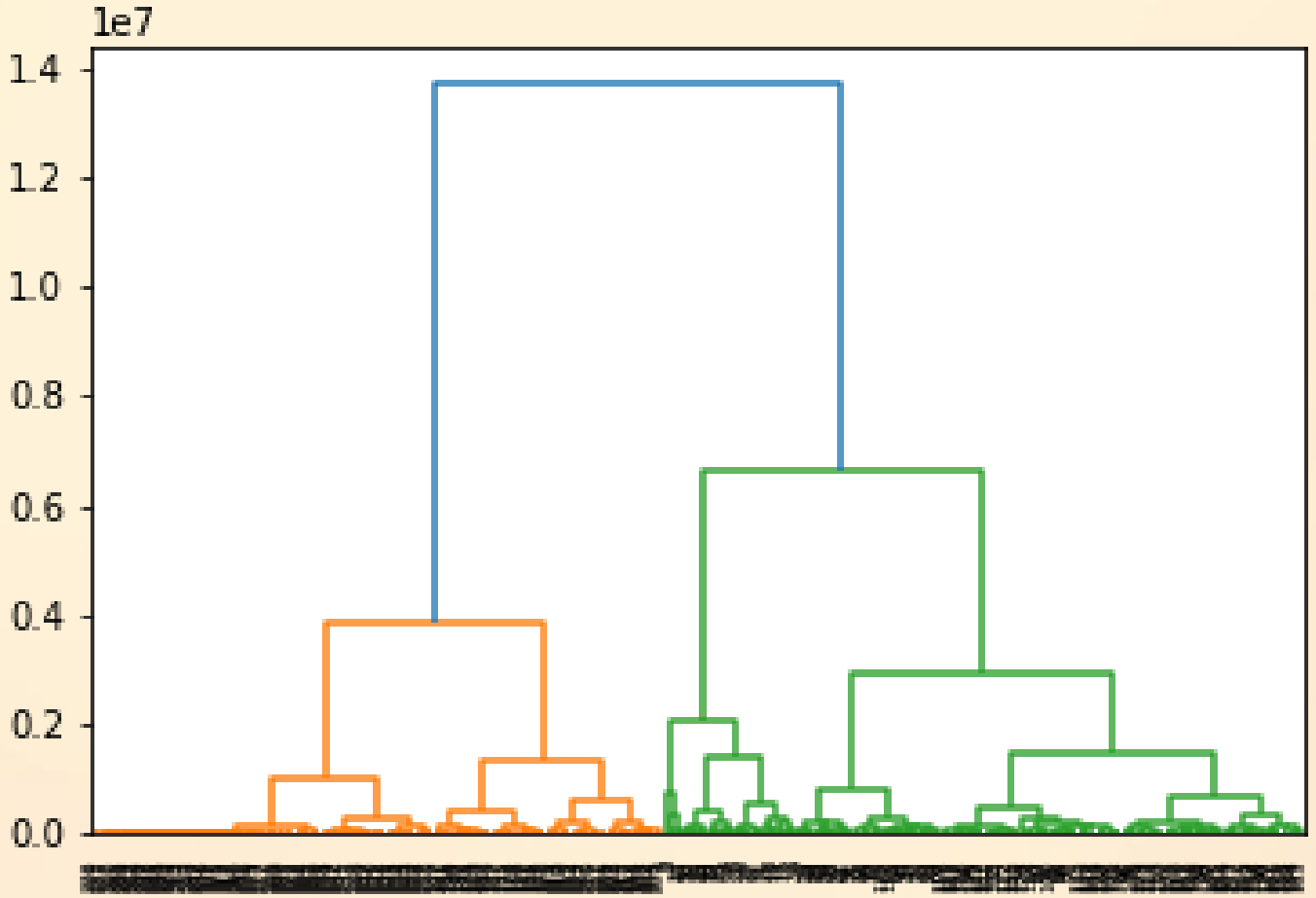
Clustering

K_mean : MinMaxScaler & fit

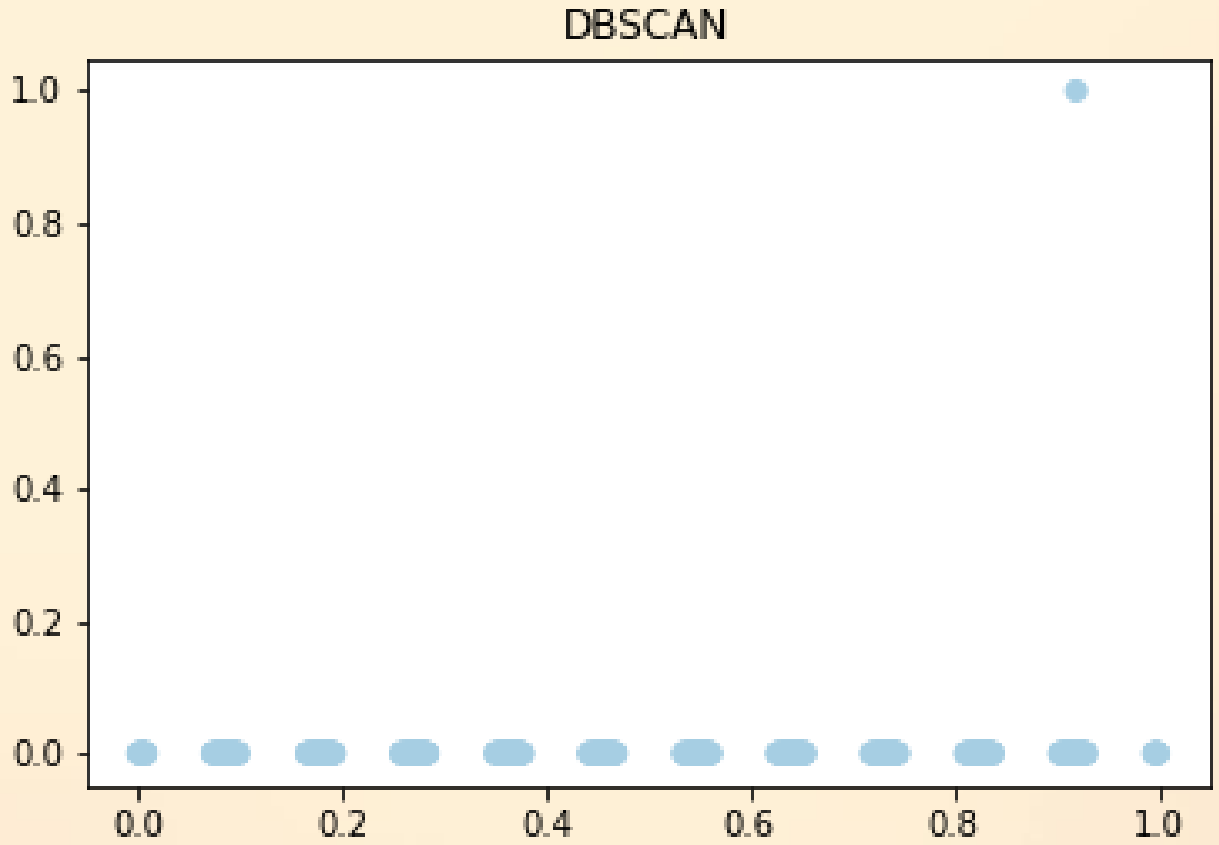


Clustering

Agglomerative : fit_predict &



from sklearn.cluster import DBSCAN



Principal Component Analysis (PCA)
from sklearn.decomposition import PCA

In [159]:
```
# PCA
pca = PCA()
df_pca = pca.fit_transform(X=X)
print('Accuracy of PCA: {:.2f}'
      .format(clf2.score(X_train, y_train)))
# Store as dataframe and print
df_pca = pd.DataFrame(df_pca)
print(df_pca.shape)
df_pca.head()
```

```
Accuracy of PCA: 0.41
(320, 24)
```

Out[159]:

	0	1	2	3	4	5	6	7	8	9	
0	1.872740e+08	4.127329e+06	1.032620e+06	524301.361942	-10936.161209	33038.914171	9231.481522	-11564.379637	258.831430	1373.285866	-3121.69
1	1.852650e+08	3.971102e+06	1.215987e+06	-99482.180690	-47575.124708	-44667.751313	-1815.301834	-30133.470684	-7241.625631	-1411.949042	-3975.69
2	1.826080e+08	3.802185e+06	9.940708e+05	238974.444709	-25236.290767	-55201.930339	-9943.145401	750.335937	8910.317517	15495.679081	-1571.85
3	1.803650e+08	3.605638e+06	8.844867e+05	222262.604188	-21959.027017	-56102.655058	-10645.242199	-3791.260258	5086.391631	7685.457079	-1669.57
4	1.781541e+08	3.428972e+06	6.276949e+05	589399.115471	5397.624867	-20138.730233	-6867.096658	19412.519861	13766.318470	14172.452767	-245.96