```
!wget https://www.dropbox.com/s/cpzql2jgv5f69r5/Data_AI.zip?dl=0 #
download file yang dibutuhkan

!apt install unzip
!unzip /content/Data_AI.zip?dl=0 # unzip file yang telah di download

--2022-04-08 03:42:12--
https://www.dropbox.com/s/cpzql2jgv5f69r5/Data_AI.zip?dl=0
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.18,
2620:100:601d:18::a27d:512
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.18|:443...
connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: /s/raw/cpzql2jgv5f69r5/Data_AI.zip [following]
--2022-04-08 03:42:12--
https://www.dropbox.com/s/raw/cpzql2jgv5f69r5/Data_AI.zip
Reusing existing connection to www.dropbox.com:443.
HTTP request sent, awaiting response... 302 Found
Location:
https://uc50b81237a59cdd8adafca299f8.dl.dropboxusercontent.com/cd/0/
inline/Bi9TnRxRJV82Ft2R7IRw1J_vCenyhMWzqe3RWy-6XXoh0IjqFD-
vVYmen_JDUd57rWOJXOnZa3wcyHlfgwLfL_N90MCaLFDgEAOsUkUjcMElZf8QBHT6UYVeB
e1sM1udZk_sBg44ejZ-NTGJ1_d5d_ZWluq8ePAh3NbsXodCXtTCuA/file#
[following]
--2022-04-08 03:42:12--
https://uc50b81237a59cdd8adafca299f8.dl.dropboxusercontent.com/cd/0/
inline/Bi9TnRxRJV82Ft2R7IRw1J_vCenyhMWzqe3RWy-6XXoh0IjqFD-
vVYmen_JDUd57rWOJXOnZa3wcyHlfgwLfL_N90MCaLFDgEAOsUkUjcMElZf8QBHT6UYVeB
e1sM1udZk_sBg44ejZ-NTGJ1_d5d_ZWluq8ePAh3NbsXodCXtTCuA/file
Resolving uc50b81237a59cdd8adafca299f8.dl.dropboxusercontent.com
(uc50b81237a59cdd8adafca299f8.dl.dropboxusercontent.com)...
162.125.5.15, 2620:100:601d:15::a27d:50f
Connecting to uc50b81237a59cdd8adafca299f8.dl.dropboxusercontent.com
(uc50b81237a59cdd8adafca299f8.dl.dropboxusercontent.com)|
162.125.5.15|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location:
/cd/0/inline2/Bi_S2ZUpSldubSoIGLe4bUijx_tqdvFw5oSqdueb5L7XkkQziHaxb4u2
SjQsKHv_3BhiY9PezIUKlW4o41JMfbG9s11AFfss_tMA3Vp4qXNfCGwUr8T-
C_WhvTKDi8CfnKTe6DIi1MBq5mfWLaGoEfILTT_F6mu3qkrgWCQQggAyOrBXiHKq9zKN3A
8HaSeGmHufHN6gwuBBl3MhaxipjyzgjNKLKtziYpEejdbLR2ONGIce5KNIKjOOn_r3uiLF
ecR7s0vXAyuMGsyVkjBHXdiirGAc66pIwCJLe0wAuanzJ0xdo3Ie5skJINnF1XHbnf56nD
s-XTTB-fEXk4lhxlMiRg4J9_t2EKHtci8ToLpc3hsTxijie6MHjXzohflAl2q74-
WxD2Ca_yBJTpvEiArloPfOH8XTA4X5zYDGi_TE6Q/file [following]
--2022-04-08 03:42:12--
https://uc50b81237a59cdd8adafca299f8.dl.dropboxusercontent.com/cd/0/
inline2/
Bi_S2ZUpSldubSoIGLe4bUijx_tqdvFw5oSqdueb5L7XkkQziHaxb4u2SjQsKHv_3BhiY9
PezIUKlW4o41JMfbG9s11AFfss_tMA3Vp4qXNfCGwUr8T-
C_WhvTKDi8CfnKTe6DIi1MBq5mfWLaGoEfILTT_F6mu3qkrgWCQQggAyOrBXiHKq9zKN3A
```

```
8HaSeGmHufHN6gwuBBl3MhaxipjyzgjNKLKtziYpEejdbLR2ONGIce5KNIKjOOn_r3uiLF
ecR7s0vXAyuMGsyVkjBHXdiirGAc66pIwCJLe0wAuanzJ0xdo3Ie5skJINnF1XHbnf56nD
s-XTTB-fEXk4lhxlMiRg4J9_t2EKHtci8ToLpc3hsTxijie6MHjXzohflAl2q74-
WxD2Ca_yBJTpvEiArloPfOH8XTA4X5zYDGi_TE6Q/file
Reusing existing connection to
uc50b81237a59cdd8adafca299f8.dl.dropboxusercontent.com:443.
HTTP request sent, awaiting response... 200 OK
Length: 6218 (6.1K) [application/zip]
Saving to: ‘Data_AI.zip?dl=0.1’

Data_AI.zip?dl=0.1  100%[===================>]   6.07K  --.-KB/s    in
0s

2022-04-08 03:42:13 (667 MB/s) - ‘Data_AI.zip?dl=0.1’ saved
[6218/6218]

Reading package lists... Done
Building dependency tree
Reading state information... Done
unzip is already the newest version (6.0-21ubuntu1.1).
0 upgraded, 0 newly installed, 0 to remove and 39 not upgraded.
Archive:  /content/Data_AI.zip?dl=0
replace data_decision_trees.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename:
n
replace adjacent_states.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace coastal_states.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace data.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
replace data_clustering.txt? [y]es, [n]o, [A]ll, [N]one, [r]ename: n
```

mendownload file-file yang dibutuhkan

#classifier Visualize

```
def visualize_classifier(classifier, X, y, title=''):
    # Define the minimum and maximum values for X and Y
    # that will be used in the mesh grid
    min_x, max_x = X[:, 0].min() - 1.0, X[:, 0].max() + 1.0
    min_y, max_y = X[:, 1].min() - 1.0, X[:, 1].max() + 1.0

    # Define the step size to use in plotting the mesh grid
    mesh_step_size = 0.01

    # Define the mesh grid of X and Y values
    x_vals, y_vals = np.meshgrid(np.arange(min_x, max_x,
mesh_step_size), np.arange(min_y, max_y, mesh_step_size))

    # Run the classifier on the mesh grid
    output = classifier.predict(np.c_[x_vals.ravel(), y_vals.ravel()])

    # Reshape the output array
```

```
    output = output.reshape(x_vals.shape)

    # Create a plot
    plt.figure()

    # Specify the title
    plt.title(title)

    # Choose a color scheme for the plot
    plt.pcolormesh(x_vals, y_vals, output, cmap=plt.cm.gray)

    # Overlay the training points on the plot
    plt.scatter(X[:, 0], X[:, 1], c=y, s=75, edgecolors='black',
linewidth=1, cmap=plt.cm.Paired)

    # Specify the boundaries of the plot
    plt.xlim(x_vals.min(), x_vals.max())
    plt.ylim(y_vals.min(), y_vals.max())

    # Specify the ticks on the X and Y axes
    plt.xticks((np.arange(int(X[:, 0].min() - 1), int(X[:, 0].max() +
1), 1.0)))
    plt.yticks((np.arange(int(X[:, 1].min() - 1), int(X[:, 1].max() +
1), 1.0)))

    plt.show()
```

berfungsi untuk melakukan pengklasifikasian visual

1. Analisa *logistic_regression.py*

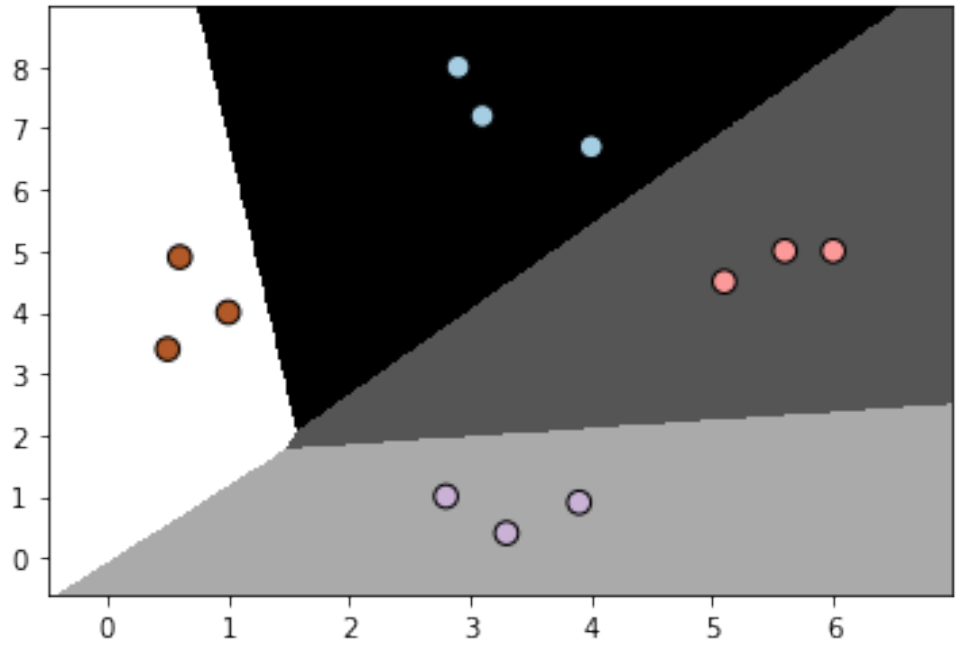
Logistic Regresion merupakan salah satu algoritma Machine Learning dan termasuk supervised learning. algoritma ini berfungsi untuk melakukan pengklasifikasian.

```
# import library
import numpy as np # library untuk oprasi vektor dan matriks
from sklearn import linear_model  # library untuk melakukan processing
dengan model linier
import matplotlib.pyplot as plt # library untuk memvisualisasikan data
```

melakukan import library yang dibutuhkan yaitu numpy, sklearn, dan matlotlib

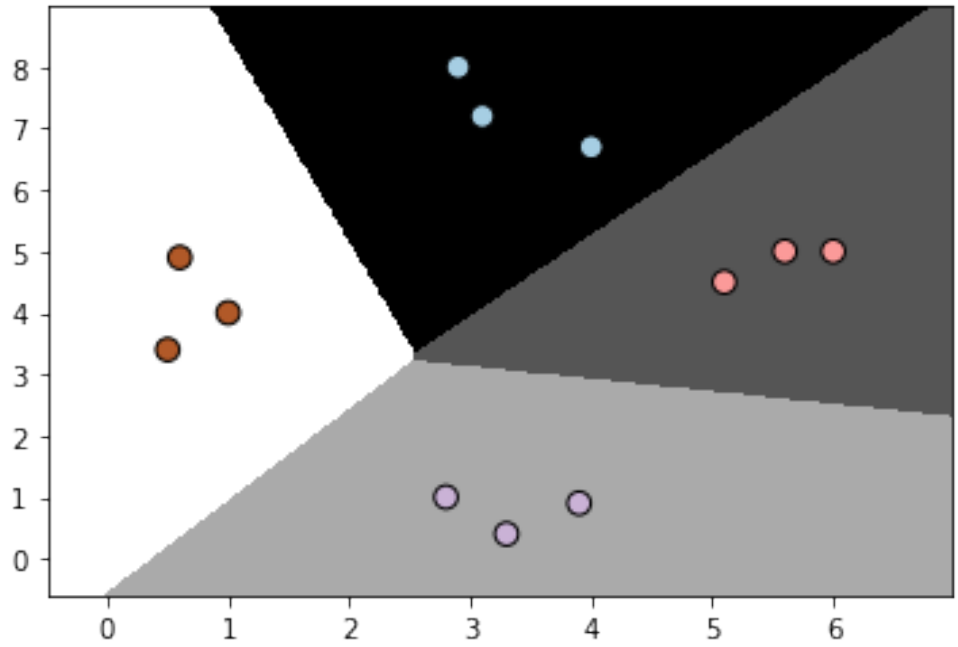
```
# Define sample input data
X = np.array([[3.1, 7.2], [4, 6.7], [2.9, 8], [5.1, 4.5], [6, 5],
[5.6, 5], [3.3, 0.4], [3.9, 0.9], [2.8, 1], [0.5, 3.4], [1, 4], [0.6,
4.9]]) # dataset x dalam bentuk array
y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3]) # dataset y dalam
bentuk array

# membuat model untuk klasifikasi regresi logistic
```

```
classifier = linear_model.LogisticRegression(solver='liblinear', C=1)

# Train the classifier
classifier.fit(X, y)

# Visualize the performance of the classifier
visualize_classifier(classifier, X, y)
```



membuat variabel x dan y sebagai dataset yang berada dalam bentuk array, kemudian dataset akan dilakukan training dengan linear_model.logisticRegression c = 1, lalu hasil train akan dilakukan visualisasi menggunakan visualize_classifier

```
# membuat model untuk klasifikasi regresi logistic
classifier = linear_model.LogisticRegression(solver='liblinear',
C=100)

# Train the classifier
classifier.fit(X, y)

# Visualize the performance of the classifier
visualize_classifier(classifier, X, y)
```



sama sepeerti sebelumnya yaitu dataset x dan y akan dilakukan training namun dengan c = 100. berdasarkan hasil diantara kedua hasil training. visualisasi data pada c = 100 terlihat lebih rapih dibandingkan visualisasi data training c = 1

2. *decission_trees.py*

Decision trees merupakan algoritma yang termasuk kedalam supervised learning. algoritma ini berfungsi mengubah data menjadi aturan-aturan keputusan. Manfaat utama dari penggunaan decision tree adalah kemampuannya untuk mem-break down proses pengambilan keputusan yang kompleks menjadi lebih simple, sehingga pengambil keputusan akan lebih menginterpretasikan solusi dari permasalahan.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
#from sklearn import cross_validation
from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split
```

melakukan import library yang dibutuhkan

```
# Load input data
input_file = 'data_decision_trees.txt'
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]
```

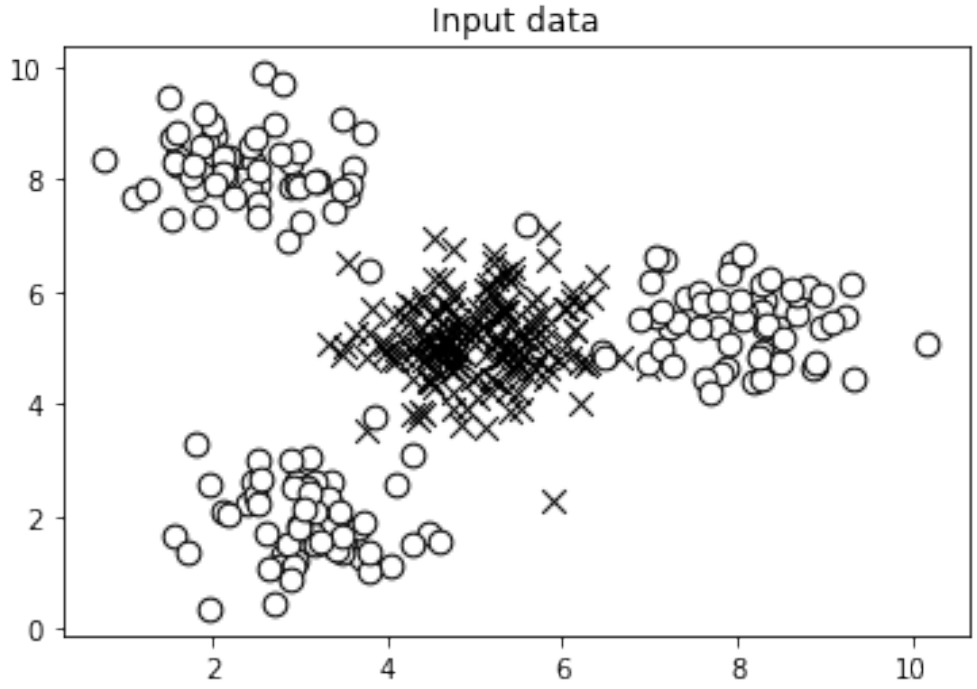
menginput file data dari data_decision_trees.txt menggunakan numpy karena data berupa array

```
# Separate input data into two classes based on labels
class_0 = np.array(X[y==0])
class_1 = np.array(X[y==1])
```

melakukan pelabelan menjadi dua class yaitu class_0 dan class_1

```
# Visualize input data
plt.figure()
plt.scatter(class_0[:, 0], class_0[:, 1], s=75, facecolors='black',
        edgecolors='black', linewidth=1, marker='x')
plt.scatter(class_1[:, 0], class_1[:, 1], s=75, facecolors='white',
        edgecolors='black', linewidth=1, marker='o')
plt.title('Input data')
```

```
Text(0.5, 1.0, 'Input data')
```

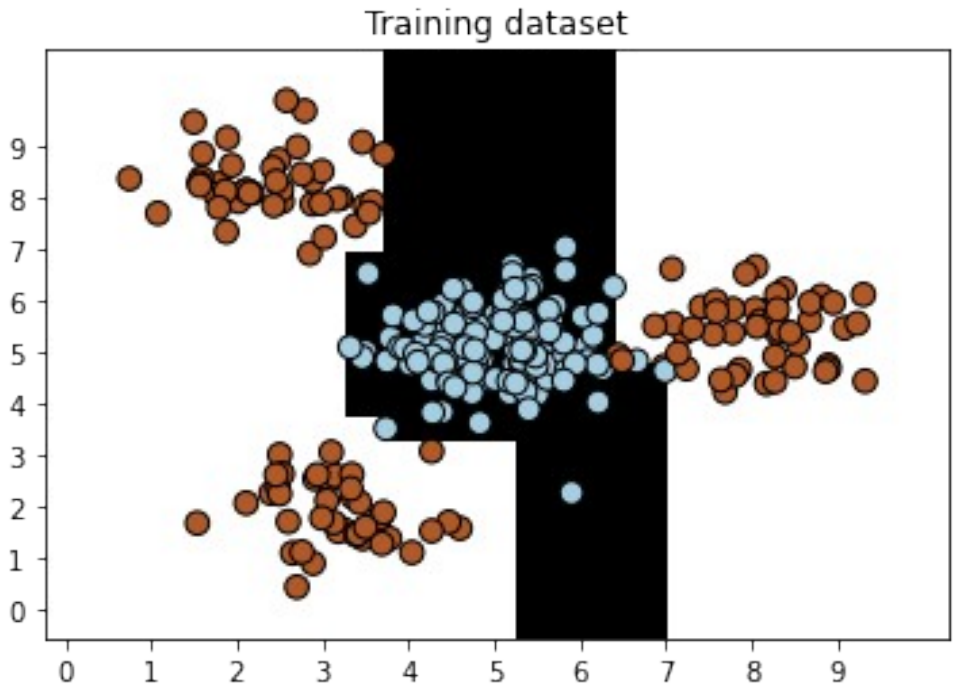


melakukan visualisasi data dengan data class_0 bertanda x dan data class_1 bertanda o

```
# Split data into training and testing datasets
X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.25, random_state=5)
```

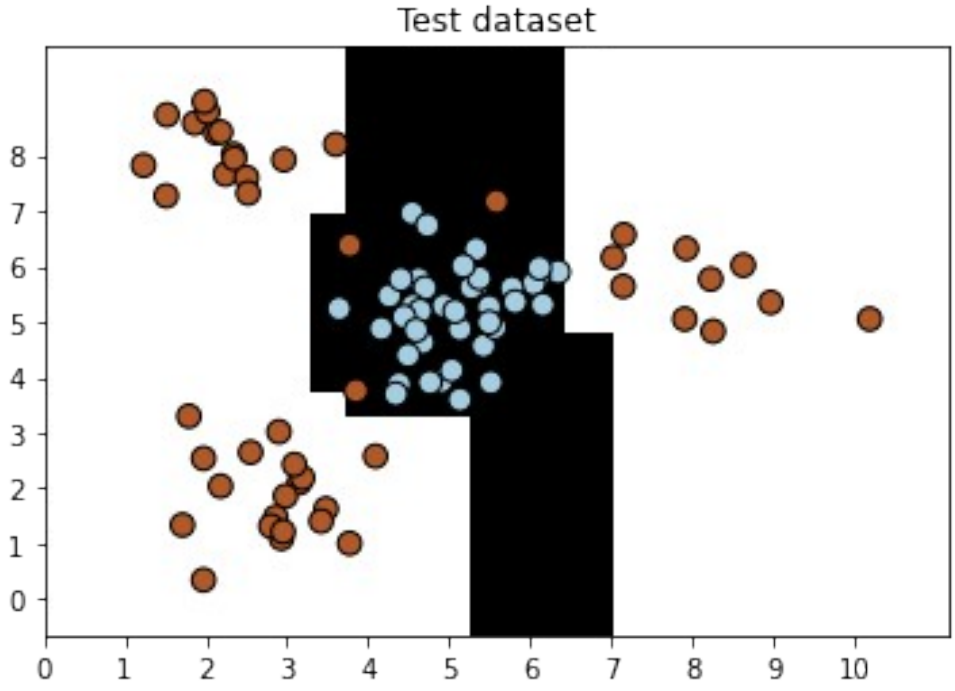
melakukan spliting data menjadi data training dan data testing

```
# Decision Trees classifier
params = {'random_state': 0, 'max_depth': 4}
classifier = DecisionTreeClassifier(**params)

classifier.fit(X_train, y_train)
visualize_classifier(classifier, X_train, y_train, 'Training dataset')
```



memvisualisasikan data training dengan menggunakan pemodelan DecisionTreeClassifier

```
y_test_pred = classifier.predict(X_test)
visualize_classifier(classifier, X_test, y_test, 'Test dataset')
```



memvisualisasikan data test sama seperti sebelumnya menggunakan pemodelan DecisionTreeClasifier

```
# Evaluate classifier performance
class_names = ['Class-0', 'Class-1']
print("\n" + "#"*40)
print("\nClassifier performance on training dataset\n")
print(classification_report(y_train, classifier.predict(X_train),
target_names=class_names))
print("#"*40 + "\n")


########################################

Classifier performance on training dataset

              precision    recall  f1-score   support

     Class-0       0.99      1.00      1.00       137
     Class-1       1.00      0.99      1.00       133

    accuracy                           1.00       270
   macro avg       1.00      1.00      1.00       270
weighted avg       1.00      1.00      1.00       270

########################################
```

mengevaluasi hasil klasifikasi data training dengan classification_report.

```
print("#"*40)
print("\nClassifier performance on test dataset\n")
print(classification_report(y_test, y_test_pred,
target_names=class_names))
print("#"*40 + "\n")
plt.show()

########################################

Classifier performance on test dataset

              precision    recall  f1-score   support

     Class-0       0.93      1.00      0.97        43
     Class-1       1.00      0.94      0.97        47

    accuracy                           0.97        90
   macro avg       0.97      0.97      0.97        90
weighted avg       0.97      0.97      0.97        90

########################################
```

mengevaluasi hasil klasifikasi data test.

berdasarkan hasil diantara kedua klaisifikasi, pebandingan antara data test dan data training memiliki keunggulan pada data training, hal ini dikarenakan pada precision yang memiliki nilai yang paling mendekati 1 akan semakin bagus.

3. Analisa *mean_shift.py*

Mean shift merupakan sebuah algoritma yang termasuk kedalam unsupervised learning. algoritma ini sering digunakan untuk clustering.

```
# load library
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth
from itertools import cycle
```

mengimport library yang dibutuhkan

```
# Load data from input file
X = np.loadtxt('data_clustering.txt', delimiter=',')
```

mengimport dataset menggunakan np.loadtxt pada data_clustering.txt

```
# Estimate the bandwidth of X
bandwidth_X = estimate_bandwidth(X, quantile=0.1, n_samples=len(X))
```

mencari nilai estimasi bandwith karena diperlukan untuk menghasilkan kepadatan clustering

```
# Cluster data with MeanShift
meanshift_model = MeanShift(bandwidth=bandwidth_X, bin_seeding=True)
meanshift_model.fit(X)
```

```
MeanShift(bandwidth=1.3044799765090382, bin_seeding=True)
```

```
# Extract the centers of clusters
cluster_centers = meanshift_model.cluster_centers_
print('\nCenters of clusters:\n', cluster_centers)
```

```
Centers of clusters:
 [[2.95568966 1.95775862]
 [7.20690909 2.20836364]
 [2.17603774 8.03283019]
 [5.97960784 8.39078431]
 [4.99466667 4.65844444]]
```

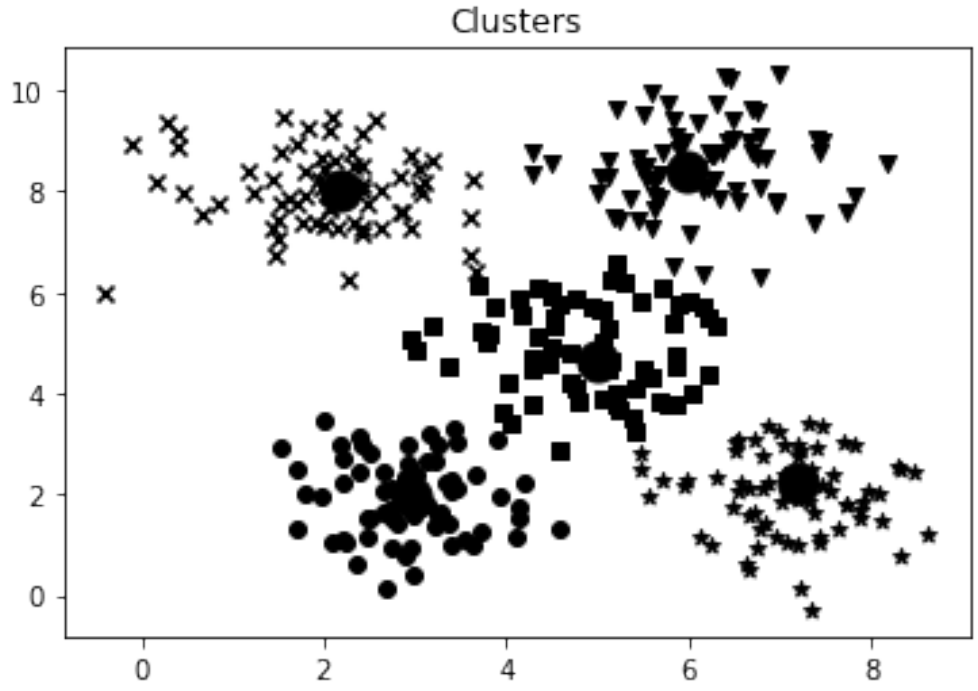
melakukan extraksi centers dari clusters menggunakan meanshift_model.cluster_centers_

```
# Estimate the number of clusters
labels = meanshift_model.labels_
```

```
num_clusters = len(np.unique(labels))
print("\nNumber of clusters in input data =", num_clusters)
```

```
Number of clusters in input data = 5
```

```
# Plot the points and cluster centers
plt.figure()
markers = 'o*xvs'
for i, marker in zip(range(num_clusters), markers):
    # Plot points that belong to the current cluster
    plt.scatter(X[labels==i, 0], X[labels==i, 1], marker=marker,
color='black')

    # Plot the cluster center
    cluster_center = cluster_centers[i]
    plt.plot(cluster_center[0], cluster_center[1], marker='o',
            markerfacecolor='black', markeredgecolor='black',
            markersize=15)

plt.title('Clusters')
plt.show()
```



ploting hasil dari point dan cluster center

4. nearest_neighbors_classifier.py

Nearest neighbor classification adalah machine learning yang bertujuan untuk memberi label objek kueri yang sebelumnya tidak terlihat sambil membedakan dua atau lebih kelas

tujuan. Sebagai pengklasifikasi, secara umum, itu membutuhkan beberapa data pelatihan dengan label yang diberikan dan, dengan demikian, adalah contoh pembelajaran yang diawasi.

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from sklearn import neighbors, datasets
```

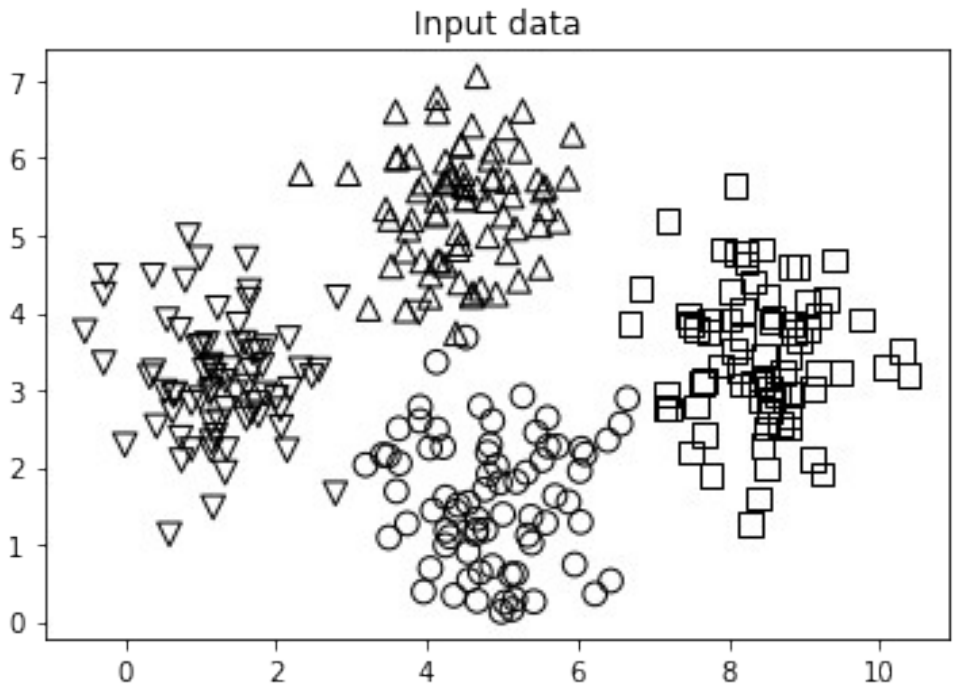
mengimport library yang dibutuhkan

```
# Load input data
input_file = 'data.txt'
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1].astype(np.int)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4:
DeprecationWarning: `np.int` is a deprecated alias for the builtin
`int`. To silence this warning, use `int` by itself. Doing this will
not modify any behavior and is safe. When replacing `np.int`, you may
wish to use e.g. `np.int64` or `np.int32` to specify the precision. If
you wish to review your current use, check the release note link for
additional information.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  after removing the cwd from sys.path.
```

mengimport dataset dari data.txt

```
# Plot input data
plt.figure()
plt.title('Input data')
marker_shapes = 'v^os'
mapper = [marker_shapes[i] for i in y]
for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
            s=75, edgecolors='black', facecolors='none')
```



memvisualisasikan data menggunakan plot

```
# Number of nearest neighbors
num_neighbors = 12
```

menentukan nilai K

```
# Step size of the visualization grid
step_size = 0.01

# Create a K Nearest Neighbours classifier model
classifier = neighbors.KNeighborsClassifier(num_neighbors,
weights='distance')
```

melakukan pemodelan dengan neigbors.kneighborsclassifier

```
# Train the K Nearest Neighbours model
classifier.fit(X, y)
```

```
KNeighborsClassifier(n_neighbors=12, weights='distance')
```

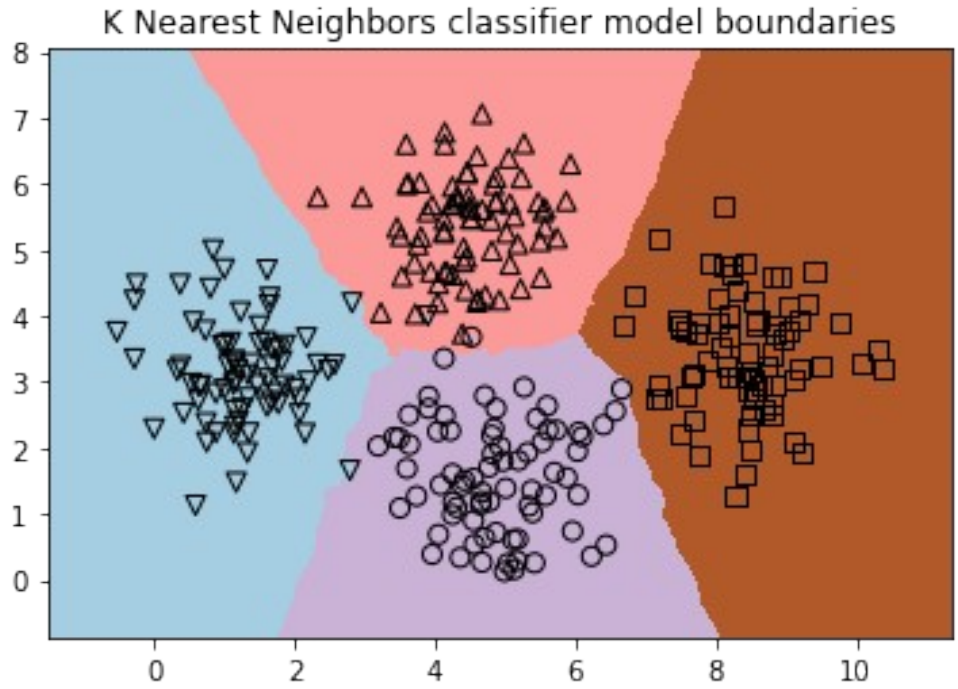
memasukan dataset ke pemodelan

```
# Create the mesh to plot the boundaries
x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
y_min, y_max = X[:, 1].min() - 1, X[:, 1].max() + 1
x_values, y_values = np.meshgrid(np.arange(x_min, x_max, step_size),
        np.arange(y_min, y_max, step_size))

# Evaluate the classifier on all the points on the grid
output = classifier.predict(np.c_[x_values.ravel(), y_values.ravel()])
```

mengevaluasi classifier pada setiap point

```
# Visualize the predicted output
output = output.reshape(x_values.shape)
plt.figure()
plt.pcolormesh(x_values, y_values, output, cmap=cm.Paired)
# Overlay the training points on the map
for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
            s=50, edgecolors='black', facecolors='none')

plt.xlim(x_values.min(), x_values.max())
plt.ylim(y_values.min(), y_values.max())
plt.title('K Nearest Neighbors classifier model boundaries')
```

```
Text(0.5, 1.0, 'K Nearest Neighbors classifier model boundaries')
```

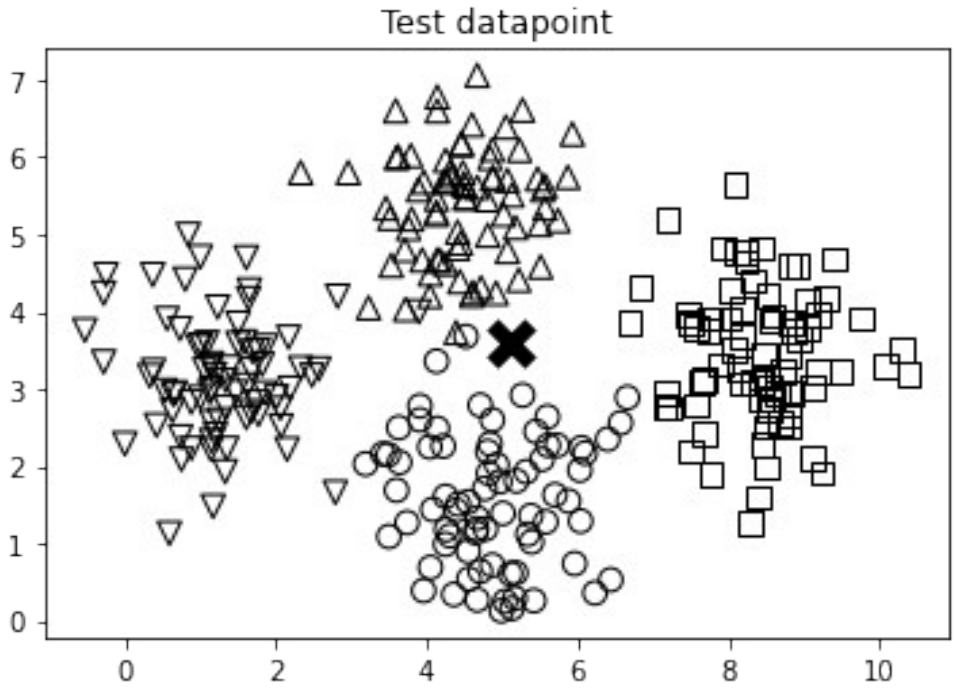


memvisualisasikan ploting hasil dari pemodelan dan batasan antar label menggunakan model KNN

```
# Test input datapoint
test_datapoint = [5.1, 3.6]
plt.figure()
plt.title('Test datapoint')
for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
            s=75, edgecolors='black', facecolors='none')
```

```
plt.scatter(test_datapoint[0], test_datapoint[1], marker='x',
        linewidth=6, s=200, facecolors='black')
```

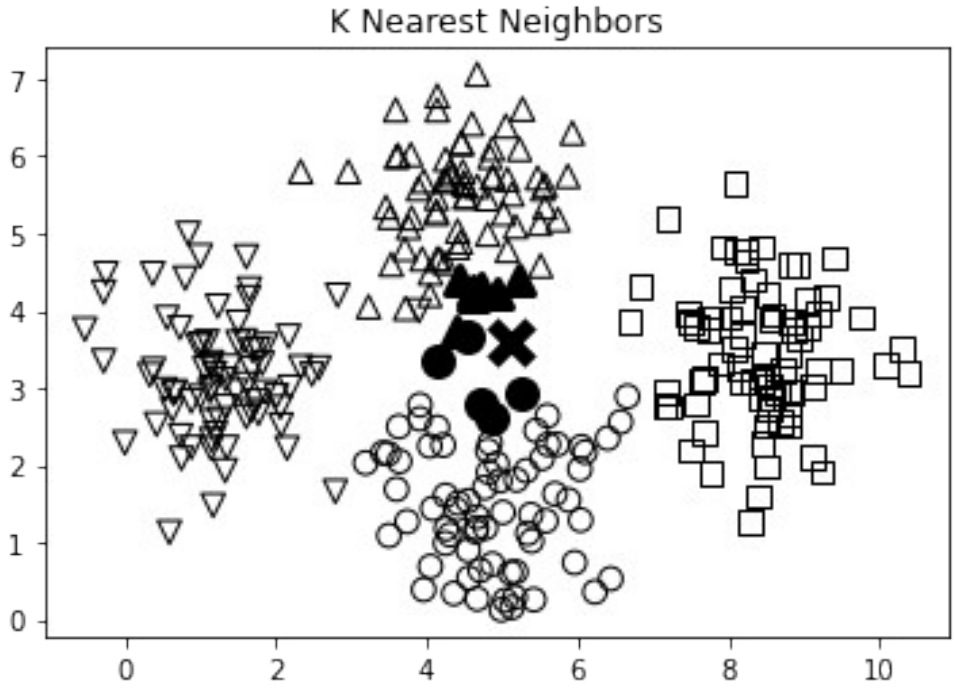
```
<matplotlib.collections.PathCollection at 0x7ff4076059d0>
```



melakukan test data point dengan marker x

```
# Extract the K nearest neighbors
_, indices = classifier.kneighbors([test_datapoint])
indices = indices.astype(np.int)[0]
# Plot k nearest neighbors
plt.figure()
plt.title('K Nearest Neighbors')

for i in indices:
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[y[i]],
            linewidth=3, s=100, facecolors='black')

plt.scatter(test_datapoint[0], test_datapoint[1], marker='x',
        linewidth=6, s=200, facecolors='black')

for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
            s=75, edgecolors='black', facecolors='none')

print("Predicted output:", classifier.predict([test_datapoint])[0])

plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3:
DeprecationWarning: `np.int` is a deprecated alias for the builtin
`int`. To silence this warning, use `int` by itself. Doing this will
not modify any behavior and is safe. When replacing `np.int`, you may
wish to use e.g. `np.int64` or `np.int32` to specify the precision. If
you wish to review your current use, check the release note link for
additional information.
Deprecated in NumPy 1.20; for more details and guidance:
https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  This is separate from the ipykernel package so we can avoid doing
imports until

Predicted output: 1
```



terdapat 12 titik yang direpresentasikan dengan bold merupakan data tetangga terdekat.

5. states.py

```
!pip install logic

Requirement already satisfied: logic in /usr/local/lib/python3.7/dist-
packages (0.2.3)
Requirement already satisfied: toolz in /usr/local/lib/python3.7/dist-
packages (from logic) (0.11.2)
Requirement already satisfied: multipledispatch in
/usr/local/lib/python3.7/dist-packages (from logic) (0.6.0)
Requirement already satisfied: unification in
/usr/local/lib/python3.7/dist-packages (from logic) (0.2.2)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-
packages (from multipledispatch->logic) (1.15.0)
```

menginstal logic untuk instalasi logpy

```
!pip install 'git+https://github.com/MHordecki/LogPy#egg=logPy'
```

```
Collecting logPy
  Cloning https://github.com/MHordecki/LogPy to /tmp/pip-install-
xjr7jo7l/logpy_ab8c8f88557f40039a6eb08e7e3ee6a7
  Running command git clone -q https://github.com/MHordecki/LogPy
/tmp/pip-install-xjr7jo7l/logpy_ab8c8f88557f40039a6eb08e7e3ee6a7
```

instalasi logpy

```
from logpy import run, fact, eq, Relation, var

adjacent = Relation()
coastal = Relation()

file_coastal = 'coastal_states.txt'
file_adjacent = 'adjacent_states.txt'
```

mengimport library logpy dan dataset coastal_states.txt, adjacent_states.tx

```
# Read the file containing the coastal states
with open(file_coastal, 'r') as f:
    line = f.read()
    coastal_states = line.split(',')
print(coastal_states)
```

```
['Washington', 'Oregon', 'California', 'Texas', 'Louisiana',
'Michigan', 'Alabama', 'Georgia', 'Florida', 'South Carolina', 'North
Carolina', 'Virgin Islands', 'Maryland', 'Delaware', 'New Jersey',
'New York', 'Connecticut', 'Rhode Island', 'Massachusetts',
'Minnesota', 'New Hampshire']
```

mambaca dataset coastal dan melakukan split

```
# Add the info to the fact base
for state in coastal_states:
    fact(coastal, state)

# Read the file containing the coastal states
with open(file_adjacent, 'r') as f:
    adjlist = [line.strip().split(',') for line in f if line and
line[0].isalpha()]
print(adjlist)
```

```
[['Alaska'], ['Alabama', 'Mississippi', 'Tennessee', 'Georgia',
'Florida'], ['Arkansas', 'Missouri', 'Tennessee', 'Mississippi',
'Louisiana', 'Texas', 'Oklahoma'], ['Arizona', 'California', 'Nevada',
'Utah', 'Colorado', 'New Mexico'], ['California', 'Oregon', 'Nevada',
'Arizona'], ['Colorado', 'Wyoming', 'Nebraska', 'Kansas', 'Oklahoma',
'New Mexico', 'Arizona', 'Utah'], ['Connecticut', 'New York',
```

'Massachusetts', 'Rhode Island'], ['District of Columbia', 'Maryland',
'Virginia'], ['Delaware', 'Maryland', 'Pennsylvania', 'New Jersey'],
['Florida', 'Alabama', 'Georgia'], ['Georgia', 'Florida', 'Alabama',
'Tennessee', 'North Carolina', 'South Carolina'], ['Hawaii'], ['Iowa',
'Minnesota', 'Wisconsin', 'Illinois', 'Missouri', 'Nebraska', 'South
Dakota'], ['Idaho', 'Montana', 'Wyoming', 'Utah', 'Nevada', 'Oregon',
'Washington'], ['Illinois', 'Indiana', 'Kentucky', 'Missouri', 'Iowa',
'Wisconsin'], ['Indiana', 'Michigan', 'Ohio', 'Kentucky', 'Illinois'],
['Kansas', 'Nebraska', 'Missouri', 'Oklahoma', 'Colorado'],
['Kentucky', 'Indiana', 'Ohio', 'West Virginia', 'Virginia',
'Tennessee', 'Missouri', 'Illinois'], ['Louisiana', 'Texas',
'Arkansas', 'Mississippi'], ['Massachusetts', 'Rhode Island',
'Connecticut', 'New York', 'New Hampshire', 'Vermont'], ['Maryland',
'Virginia', 'West Virginia', 'Pennsylvania', 'District of Columbia',
'Delaware'], ['Maine', 'New Hampshire'], ['Michigan', 'Wisconsin',
'Indiana', 'Ohio'], ['Minnesota', 'Wisconsin', 'Iowa', 'South Dakota',
'North Dakota'], ['Missouri', 'Iowa', 'Illinois', 'Kentucky',
'Tennessee', 'Arkansas', 'Oklahoma', 'Kansas', 'Nebraska'],
['Mississippi', 'Louisiana', 'Arkansas', 'Tennessee', 'Alabama'],
['Montana', 'North Dakota', 'South Dakota', 'Wyoming', 'Idaho'],
['North Carolina', 'Virginia', 'Tennessee', 'Georgia', 'South
Carolina'], ['North Dakota', 'Minnesota', 'South Dakota', 'Montana'],
['Nebraska', 'South Dakota', 'Iowa', 'Missouri', 'Kansas', 'Colorado',
'Wyoming'], ['New Hampshire', 'Vermont', 'Maine', 'Massachusetts'],
['New Jersey', 'Delaware', 'Pennsylvania', 'New York'], ['New Mexico',
'Arizona', 'Utah', 'Colorado', 'Oklahoma', 'Texas'], ['Nevada',
'Idaho', 'Utah', 'Arizona', 'California', 'Oregon'], ['New York', 'New
Jersey', 'Pennsylvania', 'Vermont', 'Massachusetts', 'Connecticut'],
['Ohio', 'Pennsylvania', 'West Virginia', 'Kentucky', 'Indiana',
'Michigan'], ['Oklahoma', 'Kansas', 'Missouri', 'Arkansas', 'Texas',
'New Mexico', 'Colorado'], ['Oregon', 'California', 'Nevada', 'Idaho',
'Washington'], ['Pennsylvania', 'New York', 'New Jersey', 'Delaware',
'Maryland', 'West Virginia', 'Ohio'], ['Rhode Island', 'Connecticut',
'Massachusetts'], ['South Carolina', 'Georgia', 'North Carolina'],
['South Dakota', 'North Dakota', 'Minnesota', 'Iowa', 'Nebraska',
'Wyoming', 'Montana'], ['Tennessee', 'Kentucky', 'Virginia', 'North
Carolina', 'Georgia', 'Alabama', 'Mississippi', 'Arkansas',
'Missouri'], ['Texas', 'New Mexico', 'Oklahoma', 'Arkansas',
'Louisiana'], ['Utah', 'Idaho', 'Wyoming', 'Colorado', 'New Mexico',
'Arizona', 'Nevada'], ['Virginia', 'North Carolina', 'Tennessee',
'Kentucky', 'West Virginia', 'Maryland', 'District of Columbia'],
['Vermont', 'New York', 'New Hampshire', 'Massachusetts'],
['Washington', 'Idaho', 'Oregon'], ['Wisconsin', 'Michigan',
'Minnesota', 'Iowa', 'Illinois'], ['West Virginia', 'Ohio',
'Pennsylvania', 'Maryland', 'Virginia', 'Kentucky'], ['Wyoming',
'Montana', 'South Dakota', 'Nebraska', 'Colorado', 'Utah', 'Idaho']]

sama seperti sebelumnya yaitu membaca dataset adjacent dan melakukan split untuk mendapatkan data terdekat pada dataset

```
# Add the info to the fact base
for L in adjlist:
    head, tail = L[0], L[1:]
    for state in tail:
        fact(adjacent, head, state)

# Initialize the variables
x = var()
y = var()
```

menginisialisasi variabel x dan y

```
# Is Nevada adjacent to Louisiana?
output = run(0, x, adjacent('Nevada', 'Louisiana'))
print('\nIs Nevada adjacent to Louisiana?:')
print('Yes' if len(output) else 'No')
```

```
Is Nevada adjacent to Louisiana?:
No
```

melakukan inisialisasi nevada dan lousiana pada variabel outpu dan melakukan percabanagan dengan jika nevada berdekatan dengan lousiana maka jawaban akan yes, namun jika jawaban selain itu maka jawaban akan no

jawaban yang dihasilkan adalah no dikarenakan nevada terdapat pada dataset adjacent state namun lousiana terdapat pada dataset coastal

```
# States adjacent to Oregon
output = run(0, x, adjacent('Oregon', x))
print('\nList of states adjacent to Oregon:')
for item in output:
    print(item)
```

```
List of states adjacent to Oregon:
California
Washington
Idaho
Nevada
```

melakukan inisialisasi oregon pada variabel output untuk menampilkan negara terdekat terdekat dari oregon

```
# States adjacent to Mississippi that are coastal
output = run(0, x, adjacent('Mississippi', x), coastal(x))
print('\nList of coastal states adjacent to Mississippi:')
for item in output:
    print(item)
```

```
List of coastal states adjacent to Mississippi:
```

```
Louisiana
Alabama
```

masih sama seperti sebelumnya untuk mencari negara terdekat pada mississippi dengan menginisialisasi mississippi ke dalam variabel output

```
# List of 'n' states that border a coastal state
n = 7
output = run(n, x, coastal(y), adjacent(x, y))
print('\nList of ' + str(n) + ' states that border a coastal state:')
for item in output:
    print(item)
```

```
List of 7 states that border a coastal state:
Ohio
Nevada
New Mexico
Pennsylvania
Maine
Rhode Island
New York
```

menginisialisasi variabel n dengan 7 yang berarti banyaknya data adalah 7, dan menampilkan data terdekat dengan jumlah data berdasarkan variabel n

```
# List of states that adjacent to the two given states
output = run(0, x, adjacent('Arkansas', x), adjacent('Kentucky', x))
print('\nList of states that are adjacent to Arkansas and Kentucky:')
for item in output:
    print(item)
```

```
List of states that are adjacent to Arkansas and Kentucky:
Tennessee
Missouri
```

menginisialisaasi arkansas dan kentucky pada variabel output untuk mendapatkan data yang terdekat diantara kedua data