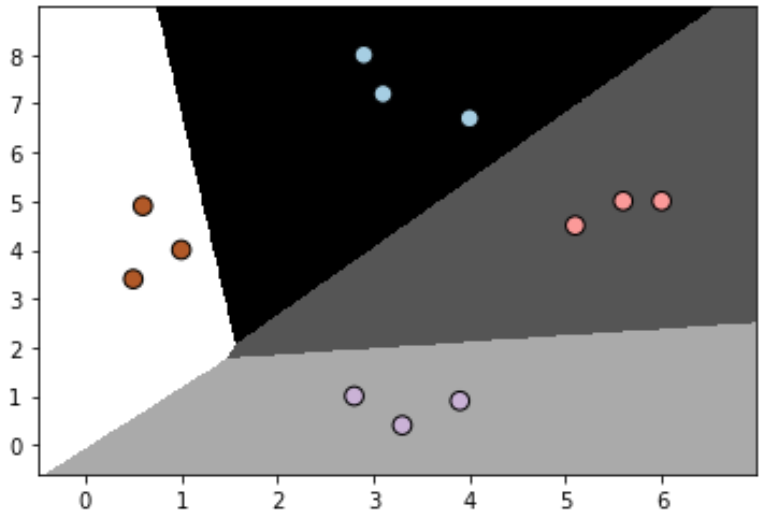
Nama : Rahmat Atoullah Gumilang Al Bantani
NIM : 3332200109
Kelas : Kecerdasan Buatan A

Link Github : https://github.com/rahmatalbantani/UTSAI/tree/master
Link YouTube : https://youtu.be/PmnsCnldZp0

1. Analisa algoritma untuk `logistic_regression.py`. Dan analisa algoritmanya dan jalankan di komputer anda. (Untuk Chapter 2).
   Jawab :

```
import numpy as np
import matplotlib.pyplot as plt

def visualize_classifier(classifier, X, y):
    # 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()

    # 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()
```

```
import numpy as np
from sklearn import linear_model
import matplotlib.pyplot as plt



# 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]])
y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])

# Create the logistic regression classifier
classifier = linear_model.LogisticRegression(solver='liblinear', C=1)
#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)
```



`logistic_regression.py` merupakan algoritma klasifikasi Machine Learning yang digunakan untuk memprediksi kategori probabilitas variabel dependen. Dalam logistic regression, variabel dependen adalah variabel biner yang berisi data berkode 1 (True, berhasil, dst) atau 0 (False, gagal, dst). Dengan kata lain, model regresi logistik memprediksi P(Y=1) sebagai fungsi dari X.

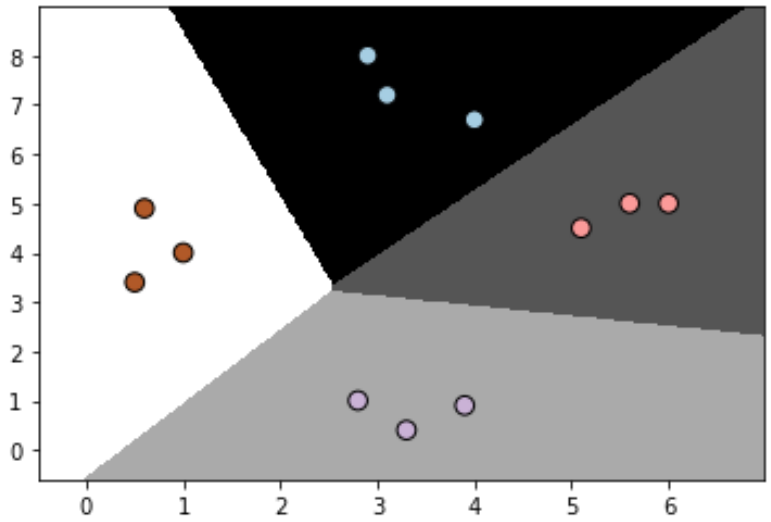
sederhananya logistic regression adalah metode analisis statistik untuk memprediksi hasil biner, seperti Yes or No, berdasarkan pengamatan sebelumnya terhadap kumpulan data. Model logistic regression memprediksi variabel data dependen dengan menganalisis hubungan antara satu atau lebih variabel independen yang ada.

Pada gambar diatas terlihat beberapa input data yang dikelompokan berdasarkan kedekatannya, sebagai contoh 3 input data warna oren dikelompokan dengan latar belakang putih. Dengan mewarnai latar belakang kelompok data tersebut menjadi warna putih itu sudah termasuk classifier,pengelompokan di sini tidak diketahui penyebabnya karena yang digunakan

adalah data *dummy*. Namun secara sederhananya di sini terjadi pengelompokan warna yang sama dengan menandai pemberian latar belakang. Kurva perpotongan antar data dipengaruhi oleh code berikut

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

Nilai C pada listing code diatas berpengaruh terhadap perpotongan kurva, semakin besar nilai C -nya makan akan semakin presisi dan smooth pula perpotongannya, hal ini dapat dibuktikan ketika nilai C-nya diganti 100,berikut adalah tampilannya



Maka terbukti dengan mengganti nilai C menjadi lebih besar,kurva akan memiliki perpotongan lebih presisi,smooth dan akurat.

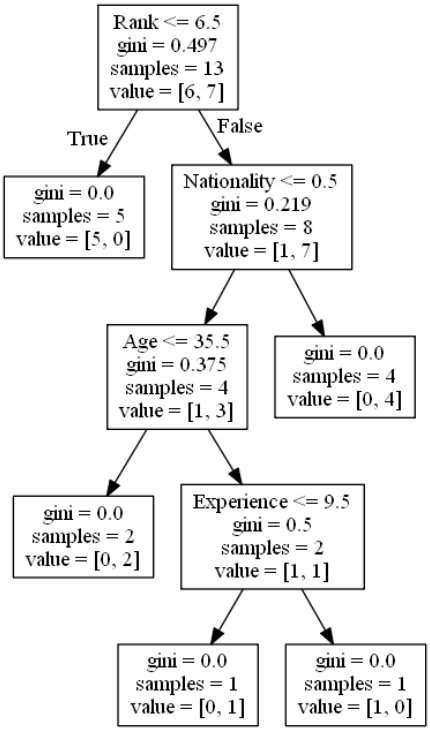
2. Analisa algoritma untuk *decision_trees.py*. Dan analisa algoritmanya dan jalankan di komputer anda. (Untuk Chapter 3)
   Jawab :

```
import numpy as np
import matplotlib.pyplot as plt

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
```

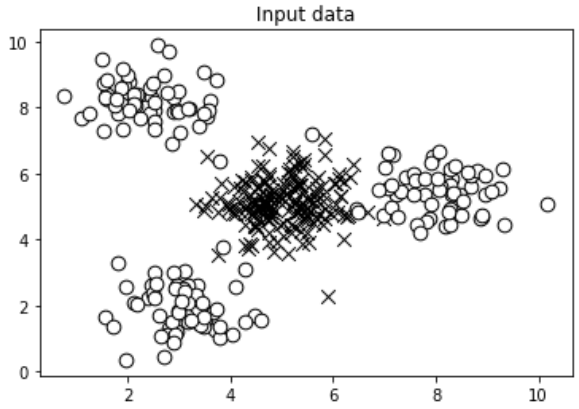
```
    output = classifier.predict(np.c_[x_vals.ravel(), y_vals.ravel()])

    # 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()



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



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

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

# 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)

# 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')

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

# 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")

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()
```

decision_trees.py adalah sebuah seleksi data dengan menggunakan metode layaknya pohon atau bercabang, seleksi *decision* dilakukan berdasarkan pertimbangan yang telah dibuat sebelumnya pada data *decision* ,bentuk visualisasi pada data *decision* diibarat seperti dibawah ini



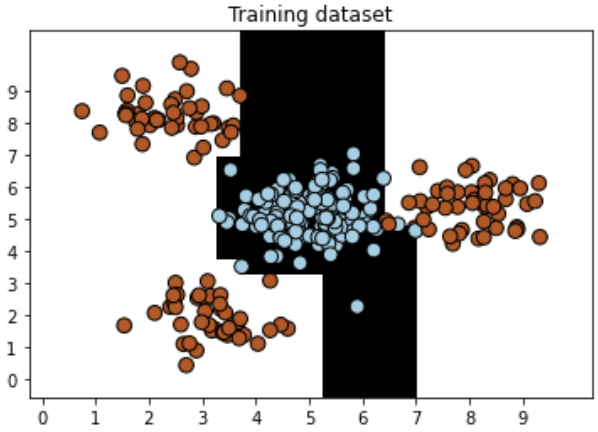
Terlihat diatas terdapat banyak percabangan terhap true dan false decision, langsung saja kita masuk ke Analisa bagian data input :



Dapat dilihat pada bagian input dibagi menjadi dua bentuk yaitu X dan O , dimana O diibaratkan sebagai *True* dan X adalah *false.* Setelah data input masuk dan melalui *decision_trees.py* maka akan menghasilkan dua output yaitu training data set dan test data set, training data set menggunakan data yang benar benar sesuai prediksi sehingga precision sempurna seperti data dibawah ini :

```
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
```

Data diatas menghasilkan gambar seperti berikut :



Terlihat bahwa data dibagi menjadi 2 bagian true dan 2 bagian prediksi berdasarkan tabel True False berikut :

Actual (Classes)	Predicted (Clusters)	
	Positive	Negative
Positive	TP	FN
Negative	FP	TN

Sehingga data yang didapat berdasarkan gambar adalah sebagai berikut

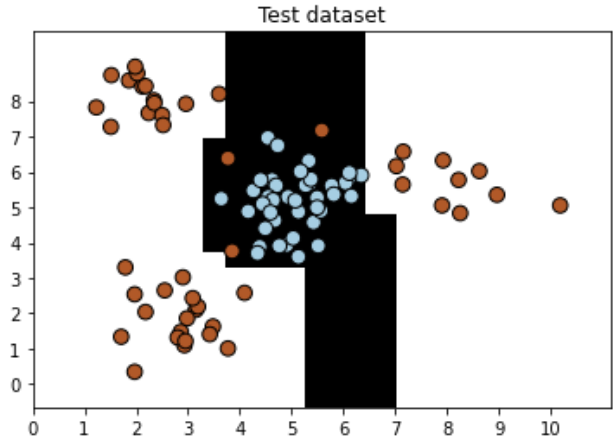
- Latar putih merupakan prediction cluster oren
- Latar hitam merupakan prediction cluster biru
- Latar hitam merupakan cluster negative
- Latar putih merupakan cluster positive

Berdasarkan dua pertanyaan diatas maka di dapat tabel True/False sebagai berikut :

Actual	Predicted	
	Positive	Negative
Oren di latar putih	Oren di latar putih	Oren di latar hitam
Biru di latar hitam	Biru di latar putih	Biru di latar hitam

Berdasarkan tabel diatas maka dapat dinyatakan :

- Warna oren berada di latar putih maka disebut sebagai True Positve
- Warna oren berada di latar hitam maka disebut sebagai False Negatif
- Warna biru berada di latar hitam maka disebut sebagai True negative
- Warna biru berada di latar putih maka disebut sebagai False positive

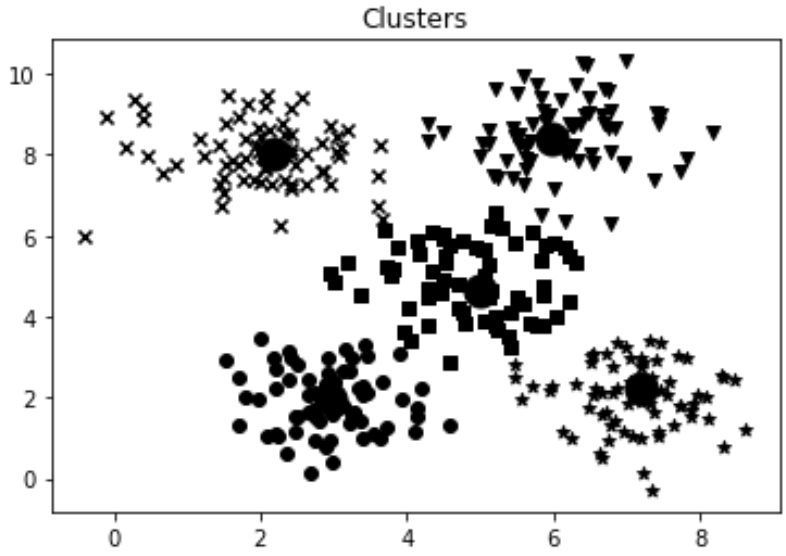


Hal yang terjadi pada *training dataset*, terjadi pula pada *test dataset* perbedaanya adalah pada test *dataset* lebih menuju ke data real sehingga memiliki nilai precision yang lebih kecil bila dibandingkan training data set yang sangat sempurna,dapat dilihat pada gambar diatas bterdapat 3 buah data oren yang seharusnya di latar putih malah ke latar hitam kondisi ini disebut *false negative*.

3. Analisa algoritma untuk *mean_shift.py.* Dan analisa algoritmanya dan jalankan di komputer anda. (untuk Chapter 4).
   Jawab :

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

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

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

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

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

# 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()
```

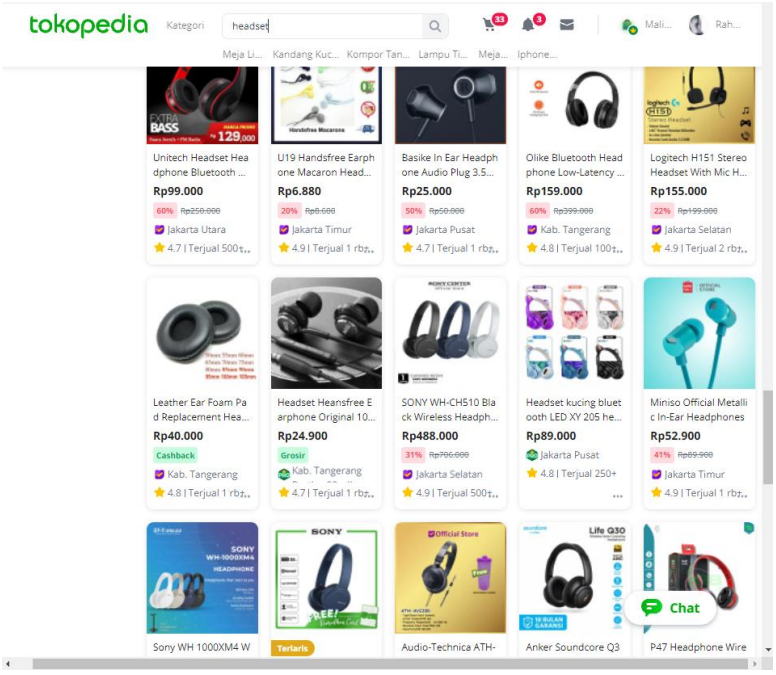


mean_shift.py adalah sebuah proses untuk melakukan cluster atau pengelompokan berdasarkan variable yang dibuat oleh developer, berbeda dengan cluster sebelumnya, pada cluster *mean_shift.py* kita dapat menentukan variable tersendiri.seperti contoh diatas, kita misalkan salah satu symbol seperti berikut :

- Symbol X dinyatakan sebagai data yang mirip,sejenis atau sekelompok dengan data induk tengahnya yaitu “headset”. Maka data X di sekitar induk adalah data yang mirip dengan induknya yaitu “headset”,semakin jauh jarak X dari induk nya maka tingkat kemiripannya dengan “headset” akan semakin kecil.

Begitu dengan symbol bulat,bintang dan segitiganya.

Dalam kehidupan sehari-hari hal ini biasa digunakan untuk searching. Contoh kita searching di Marketplace dimana variable “induk” adalah variable searching yang kita buat, asumsikan kita mencari headset di Tokopedia maka kata “headset” adalah induk yang kita buat kemudian produk yang tersedia adalah symbol X tersebut.



Dapat dilihat pada gambar diatas, variable searching adalah headset,dan produk yang muncul adalah symbol X, Adapun *cover headset* itu merupakan data yang kemiripannya jauh dari induknya.

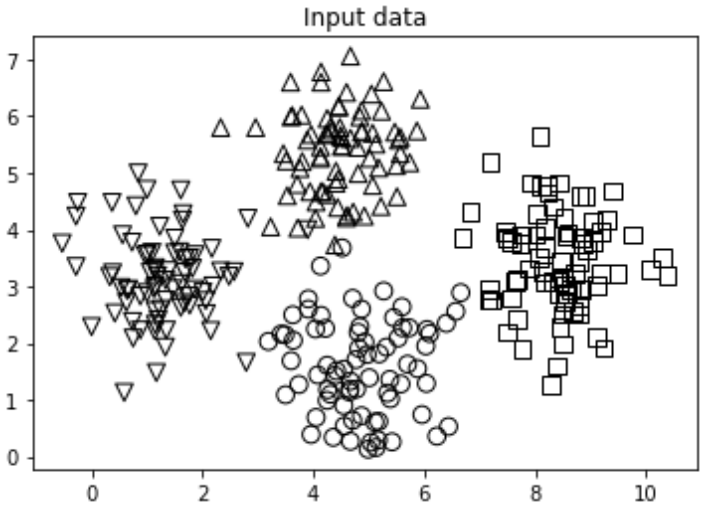
4. Analisa algoritma untuk *nearest_neighbors_classifier.py*. Dan analisa algoritmanya dan jalankan di komputer anda (untuk Chapter 5)

   Jawab :

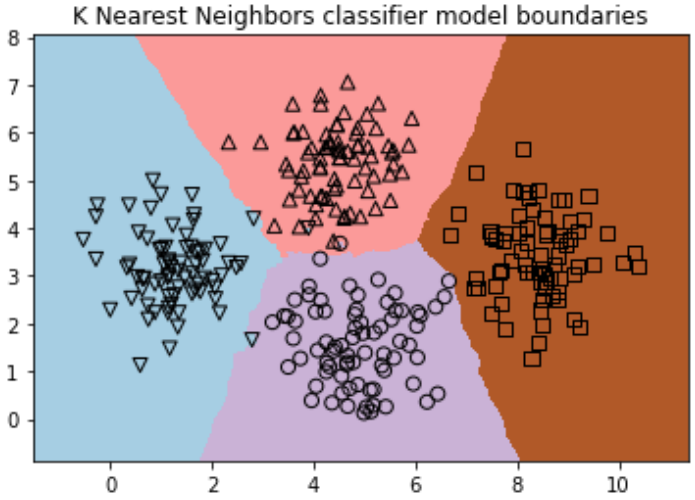
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from sklearn import neighbors, datasets

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

# 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')

# Number of nearest neighbors
num_neighbors = 12

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

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

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

# 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()])

# 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')

# 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')

# 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()
```



Pada awalnya bentuk input data di definisikan sebagai berikut kemudian menggunakan *nearest_neighbors_classifier.py* akan di klasifikasikan berdasarkan kemiripan datanya menjadi seperti berikut.



Berdasarkan analisa yang saya dapat, *nearest_neighbors_classifier.py* merupakan metode klasifikasi berdasarkan kesamaan antar sifat datanya namun variable ditentukan berdasarkan sistem, berbeda dengan *mean_shift.py* yang cluster variabelnya kita buat, pada *nearest_neighbors_classifier.py* pengelompokan data didasarkan oleh kemiripan data yang proses seleksinya dibuat oleh developer. Hal seperti ini biasanya digunakan untuk pengelompokan secara random yang ditentukan oleh sistem itu sendiri. Sebagai contoh kebiasaan “orang dewasa yang merokok” dan “orang dewasa yang tidak merokok” kedua

variable tersebut dibuat oleh sistem misal berdasarkan perilaku dan bau bekas rokoknya kemudian proses oleh machine learning dan di spesifikasikan oleh system kemudian system akan membuat variable sendiri dan mengelompokan data pada dua variable tersebut. Hal ini dibuktikan dengan bentuk gambar data di atas.

Hasil pengelompokan diatas tidak memiliki induk atau bentuk bulan diantara kemiripan data, hal ini disebabkan oleh variable yang ditentukan sendiri oleh system machine learning.

5. Analisa algoritma untuk *states.py.* Dan analisa algoritmanya dan jalankan di komputer anda (untuk Chapter 6)

   Jawab :

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

adjacent = Relation()
coastal = Relation()

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

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

# 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()

# 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')

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

```
# 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 '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 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)
```

Hasil Output Listing Code

```
Is Nevada adjacent to Louisiana?:
No

List of states adjacent to Oregon:
Nevada
California
Idaho
Washington

List of coastal states adjacent to Mississippi:
Louisiana
Alabama

List of 7 states that border a coastal state:
Delaware
New Jersey
New Hampshire
Connecticut
Alabama
West Virginia
Massachusetts

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

states.py adalah sebuah listing proses yang membuat aturan berdasarkan relasi antar data, variable relasi tersebut telah di *define* melalui `Relation()` yang di import dari library *logpy*, seperti yang dicontohkan pada hasil output diatas. Mari kita pisahkan output menjadi beberapa part

Part I

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

Berdasarkan hasil output diatas memberi arti bahwa list negara yang berdekatan,bertetangga, atau bersebelahan dengan Oregon, dan diliat hasil negara yang tercantum adalah Nevada,California,Idaho dan Washington. Hal ini dapat terjadi sesuai dengan data sheet dari listing code `file_adjacent = 'adjacent_states.txt'` mari kita lihat tabel dari adjacent_states.txt,dikarenkan data adjacent terlalu banyak,mari kita cuplik beberapa datanya berdasarkan kebutuhan seperti berikut ini :

Tabel Cuplik Data Adjacent

State Name	Bordering States
Oklahoma	Arkansas, Colorado, Kansas, Missouri, New Mexico, Texas
Oregon	California, Idaho, Nevada, Washington
Pennsylvania	Delaware, Maryland, New Jersey, New York, Ohio, West Virginia

Dapat dilihat pada data diatas negara yang bersebelahan dengan Oregon adalah California, Idaho, Nevada, Washington sehingga yang tampil pada data output adalah seperti Output awal diatas.

Part II

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

Terlihat pada hasil output diatas memberi arti list negara berkategori coastal states yang berdekatan dengan Mississippi. Berdasarkan pernyataan tersebut maka disini terjadi dua seleksi yaitu seleksi negara yang berdekatan dengan Mississippi, kemudian hasil data outputnya akan Kembali diseleksi dengan yang termasuk negara coastal. Untuk lebih jelasnya mari kita lihat tabel data sheet coastal dan adjacent yang telah di cuplik untuk kebutuhan analisis.

State Name	Bordering States
Minnesota	Iowa, Michigan (water border), North Dakota, South Dakota, Wisconsin
Mississippi	Alabama, Arkanssas, Louisiana, Tennessee
Missouri	Arkansas, Illinois, Iowa, Kansas, Kentucky, Nebraska, Oklahoma, Tennessee

Terlihat pada tabel diatas, negara yang berdekatan dengan Mississippi adalah Alabama, Arkanssas, Louisiana dan Tennessee. Namun karena yang diminta adalah negara coastal maka kita perlu data coastal terlebih dahulu seperti dibawah ini

Coastal State

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

Dapat dilihat pada tabel diatas ternyata dari 4 negara yang dekat dengan Mississipi,hanya 2 negara yang berkategori coastal maka outputnya adalah Louisiana dan Alabama.

Part III

```
List of 7 states that border a coastal state:
Delaware
New Jersey
New Hampshire
Connecticut
Alabama
West Virginia
Massachusetts
```

Kemudian selanjutnya adalah list 7 negara adjacent yang membatasi dengan coastal state dengan arti coastal state yang dengan coastal state itu sendiri.

Alabama	California	Connecticut	Delaware	Florida	Georgia	Lousiana
Massachusetts	Maryland	Michigan	Minnesota	North Carolina	New Hampshire	New York
New Jersey	Oregon	Rhode Island	South Carolina	Texas	Virgin Islands	Washington

Dapat dilihat pada tabel berikut ini yang telah dicuplik :

State Name	Bordering States
Delaware	Maryland, New Jersey, Pennsylvania
New Jersey	Delaware, New York, Pennsylvania
New Hampshire	Maine, Massachusetts, Vermont

Connecticut	Massachusetts, New York, Rhode Island
Alabama	Florida, Georgia, Mississippi, Tennessee
West Virginia	Kentucky, Maryland, Ohio, Pennsylvania, Virginia
Massachusetts	Connecticut, New Hampshire, New York, Rhode Island, Vermont

Sebagai contoh yaitu **Alabama** , Alabama merupakan coastal state dan dapat dilihat alabama memiliki border atau berdekatan dengan coastal state juga yaitu Florida dan Georgia sehingga Alabama termasuk 7 negara yang saling berdekatan dengan coastal state. Begitupun dengan coastal state yang lainnya.

Part IV

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

Terlihat pada tabel diatas, yang berarti list negara yang dekat dengan Arkansas dan Kentucky,dikarenakan kata penghubungnya adalah “and” maka yang dimaksud adalah irisan negara yang berdekatan dengan Arkansas dan Kentucky. Untuk lebih jelasnya mari kita cuplik tabel data adjacent.

State Name	Bordering States
Arkansas	Louisiana, Mississippi, Missouri, Oklahoma, Tennessee, Texas
Kentucky	Illinois, Indiana, Missouri, Ohio, Tennessee, Virginia, West Virginia

Dapat dilihat pada tabel diatas, terlihat bahwa negara yang berdekatan dengan Arkansas dan Kentucky adalah Missouri dan Tennesse dikarenakan kedua negara tersebut berdekatan dengan Arkansas maupun Kentucky.