# UJIAN TENGAH SEMESTER

## Kecerdasan Buatan

Nama : Paulus Miki Febrian

Nim : 3332190052

Mendownload materi yang diperlukan

```
!wget --no-check-
certificate 'https://docs.google.com/uc?export=download&id=1zlJwPKAsKhl
GlNu6Cn7XhKt_IUFH87YF' -O data_decision_trees.txt
!wget --no-check-
certificate 'https://docs.google.com/uc?export=download&id=1oNqpwqQarcu
IavyZFHYRAtxBnShDOPIo' -O data_clustering.txt
!wget --no-check-
certificate 'https://docs.google.com/uc?export=download&id=13FHXPHZK3yR
Hhr0OMqz6YTM70HGxEeMa' -O data.txt
!wget --no-check-
certificate 'https://docs.google.com/uc?export=download&id=13iqaBwvhTty
D7IX-5v5pSnAJQ0ouxGFT' -O coastal_states.txt
!wget --no-check-
certificate 'https://docs.google.com/uc?export=download&id=1AnbKOMXh_3G
B2Y9jTGkmF4bDXfpbgRAr' -O adjacent_states.txt
```

Classifier Visualizer

```
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', linewi
dth=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()
```

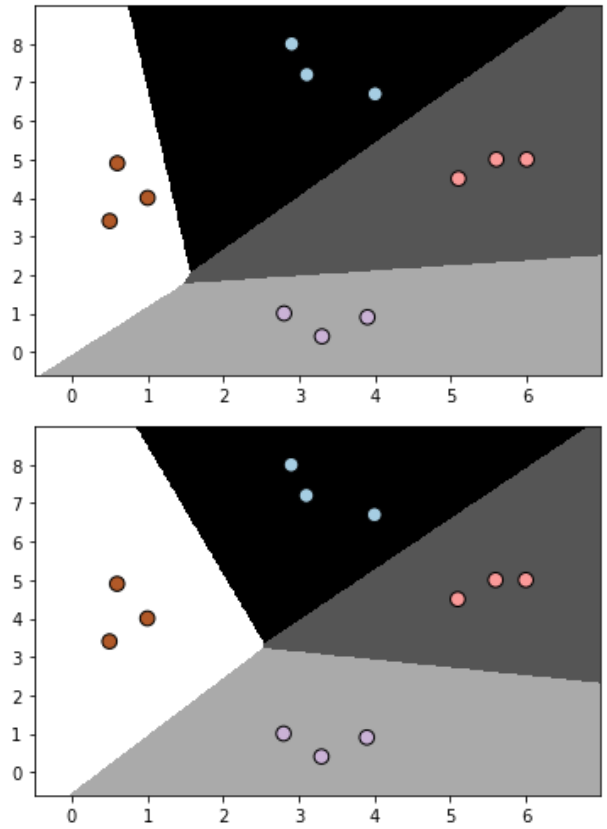
Soal

1. Analisa algoritma untuk *logistic_regression.py*. Dan analisa algoritmanya dan jalankan di komputer anda.

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

    #from utilities import visualize_classifier

    # 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
    classifier1 = linear_model.LogisticRegression(solver='liblinear',
    C=1)
    classifier2 = linear_model.LogisticRegression(solver='liblinear',
    C=100)

    # Train the classifier
    classifier1.fit(X, y)
```

```
classifier2.fit(X, y)
# Visualize the performance of the classifier
visualize_classifier(classifier1, X, y)
visualize_classifier(classifier2, X, y)
```

Hasilnya :




Gambar di atas merupakan grafik dari proses klasifikasi. Titik-titik di atas adalah variabel dependent dan area yang mengitari titik tersebut merupakan klasifikasi yang membedakan data-data tersebut.

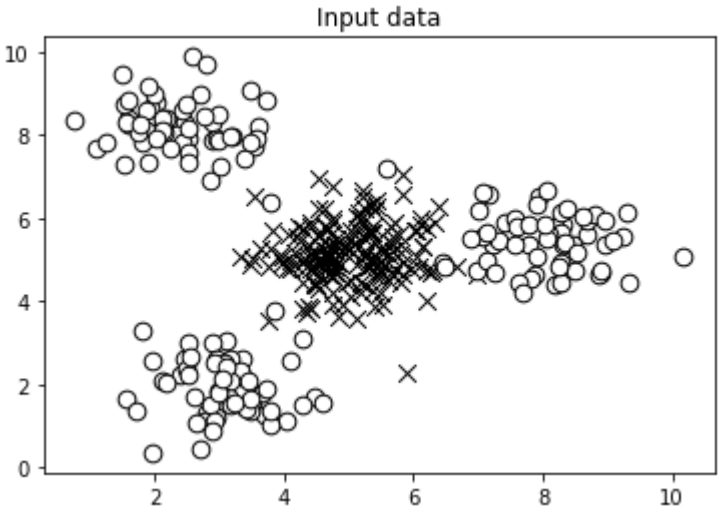
2. Analisa algoritma untuk *decision_trees.py*. Dan analisa algoritmanya dan jalankan di komputer anda.

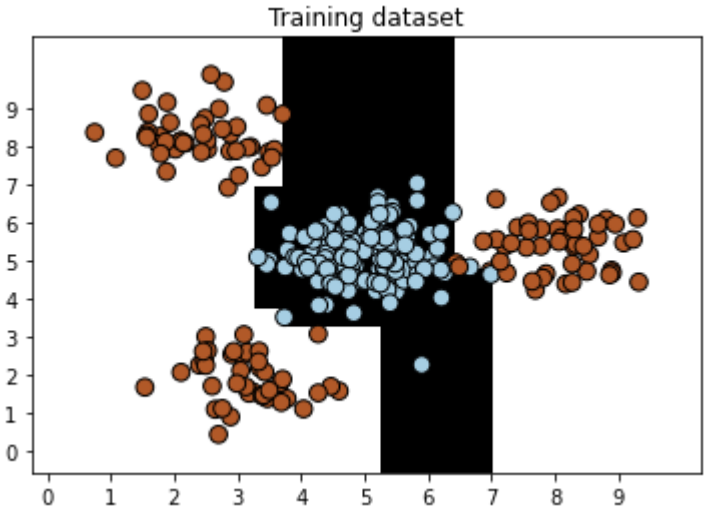
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
#from sklearn import cross_validation
from sklearn.tree import DecisionTreeClassifier

from sklearn.model_selection import train_test_split

#from utilities import visualize_classifier
```

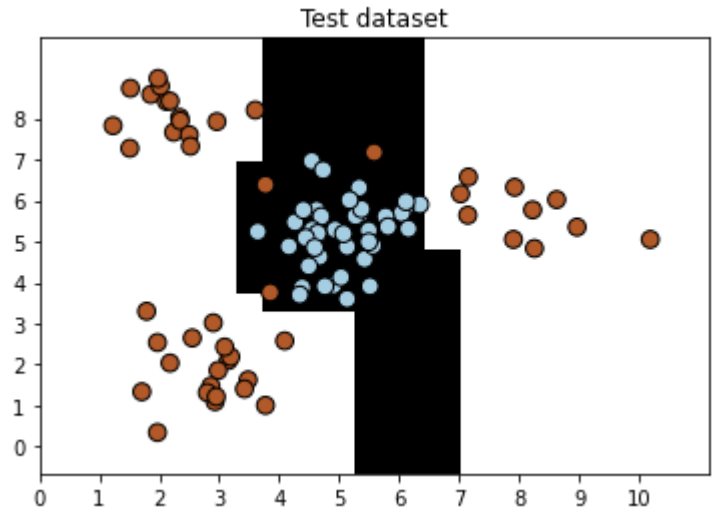
```
# Load input data
input_file = 'data_decision_trees.txt'
data = np.loadtxt(input_file, delimiter=',')
X, y = data[:, :-1], data[:, -1]

# 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='bl
ack',
edgecolors='black', linewidth=1, marker='x')
plt.scatter(class_1[:, 0], class_1[:, 1], s=75, facecolors='wh
ite',
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 d
ataset')

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_trai
n), 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()
```

Hasil:






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

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

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

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

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

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

3. Analisa algoritma untuk *mean_shift.py*. Dan analisa algoritmanya dan jalankan di komputer anda.

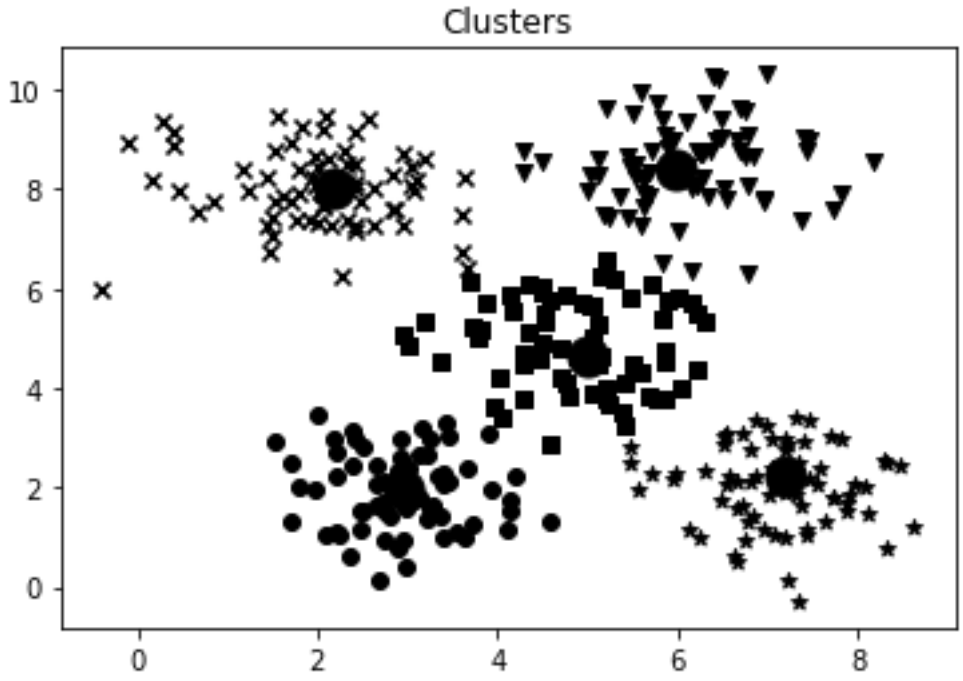
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import MeanShift, estimate_bandwidth
from itertools import cycle
```

```
    # 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=le
    n(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=marke
    r, 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()
```

Hasil :

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

Number of clusters in input data = 5




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

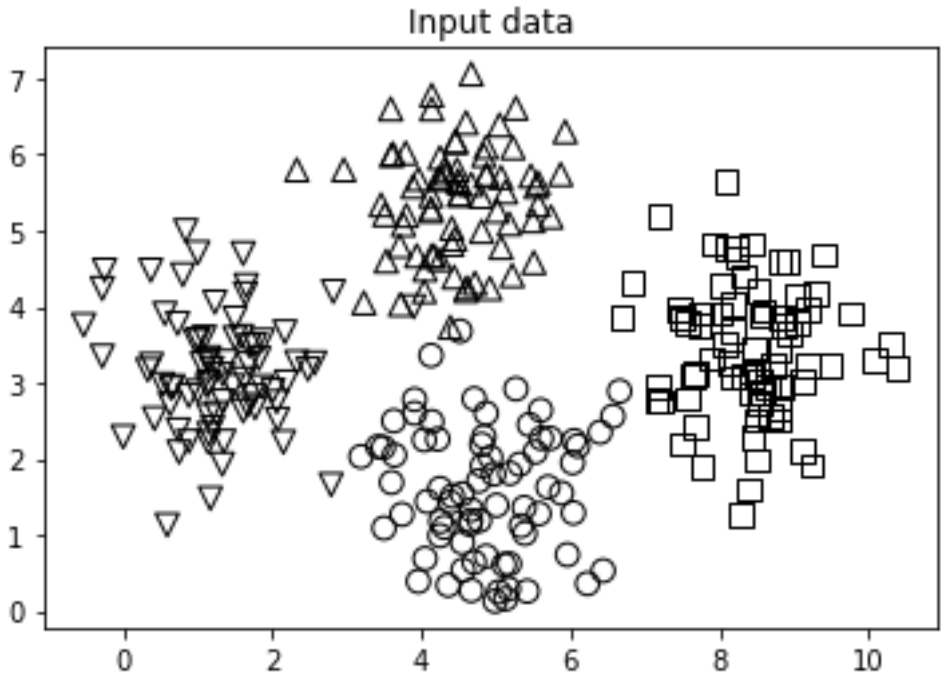
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from sklearn import neighbors, datasets

# 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
```

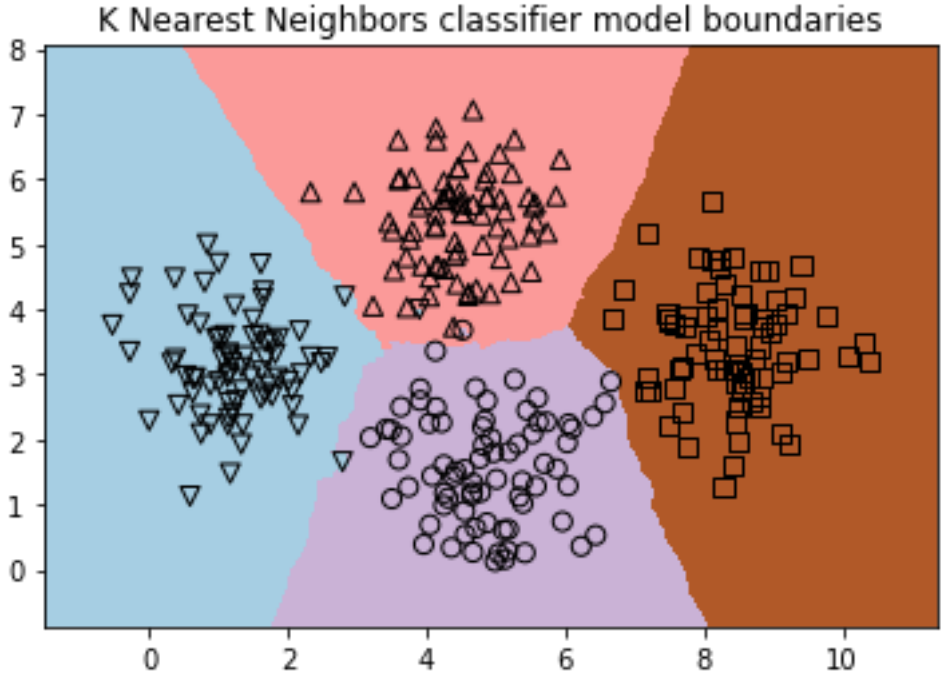
```
    classifier = neighbors.KNeighborsClassifier(num_neighbors, wei
    ghts='distance')

    # 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.r
    avel()])

    # 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()
```

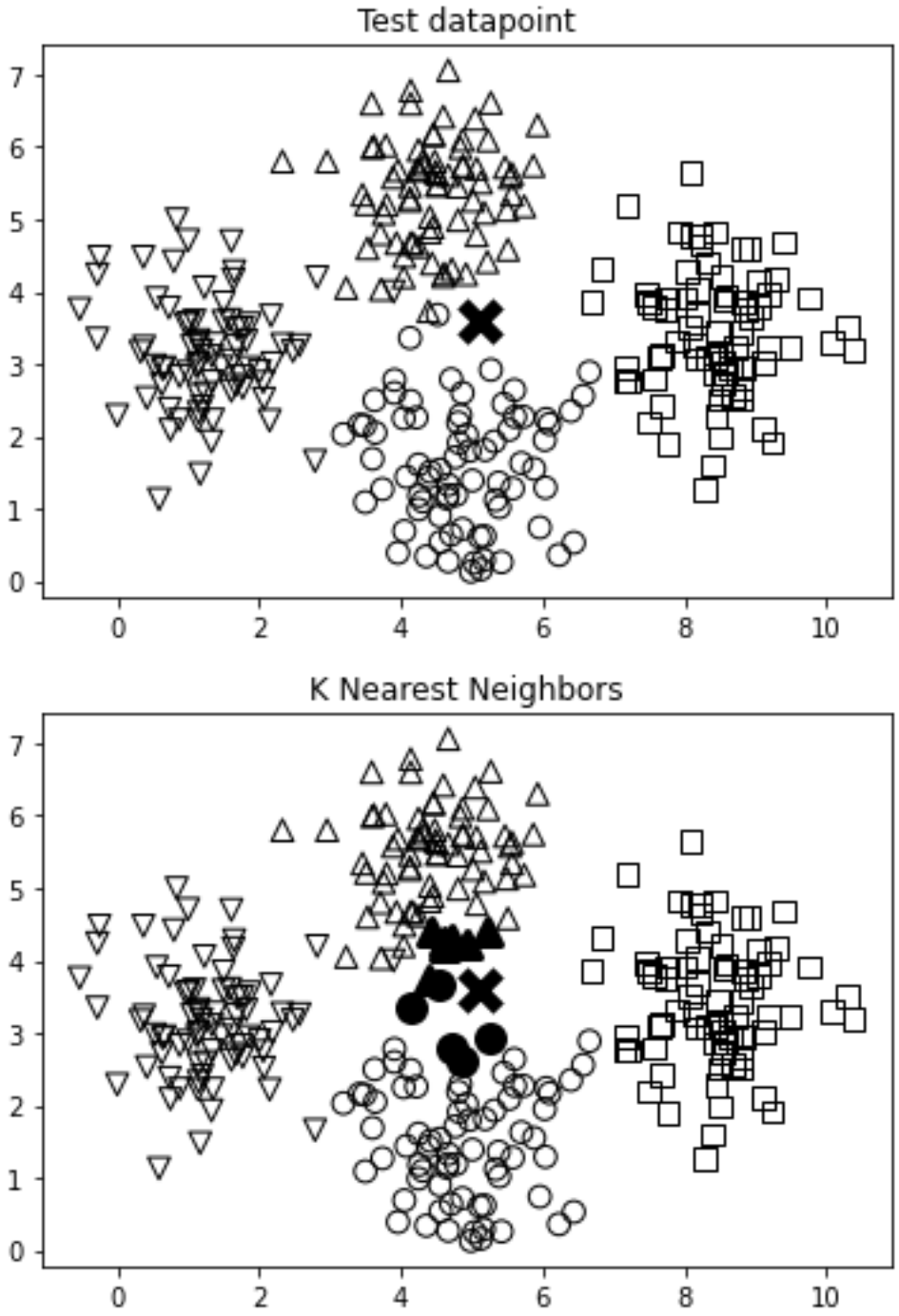
```
    plt.title('K Nearest Neighbors')

    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()
```

Hasil :

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:9:
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
  if __name__ == '__main__':
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:68:
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
Predicted output: 1
```


Input data
0
1
2
3
4
5
6
7
0
2
4
6
8
10


K Nearest Neighbors classifier model boundaries
0
1
2
3
4
5
6
7
8
0
2
4
6
8
10




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

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

```
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 stat
e:')
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 Kentuck
y:')
for item in output:
    print(item)
```

Hasil

```
Is Nevada adjacent to Louisiana?:
No

List of states adjacent to Oregon:
Idaho
Washington
Nevada
California

List of coastal states adjacent to Mississippi:
Louisiana
Alabama

List of 7 states that border a coastal state:
Maryland
Indiana
Texas
California
New Jersey
North Carolina
Wisconsin

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