

UTS KECERDASAN BUATAN AI

Nama : Muhammad Sofyan Abiyyu

NIM : 3332200060

UTS Kecerdasan Buatan B

Link Github : [silvercb5/UTS_Kecerdasan_Buatan \(github.com\)](https://github.com/silvercb5/UTS_Kecerdasan_Buatan)

Link YouTube:

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])

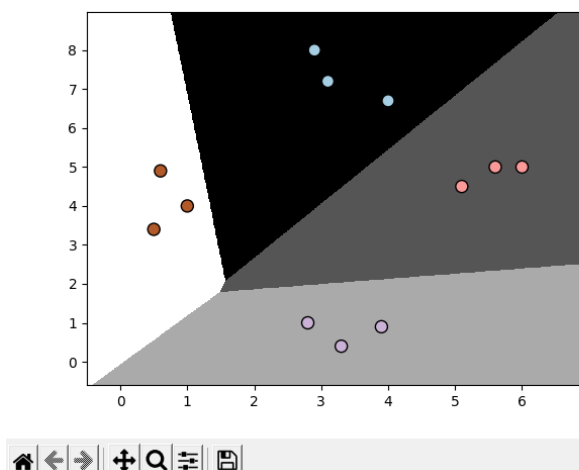
# Create the logistic regression classifier
classifier =
linear_model.LogisticRegression(solver='liblinear', C=1)
#classifier =
linear_model.LogisticRegression(solver='liblinear', C=100)

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

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

```

Figure 1



Logistic Regression merupakan sebuah model yang digunakan untuk melakukan prediksi apakah sesuatu bernilai benar atau salah (0 atau 1). Apabila

pada *linear regression* garis yang terbentuk adalah garis lurus, tetapi pada *logistic regression* garis yang dibentuk mirip dengan huruf “S” antara titik 0 sampai 1. *Logistic regression* akan mengukur hubungan antara variabel target (yang ingin diprediksi) dan variabel input (fitur yang digunakan) dengan fungsi logistik. *Probabilitas* akan dihitung menggunakan fungsi *sigmoid* untuk mengubah nilai-nilai tadi menjadi 0 atau 1. Setelah kita melakukan kode pada program dapat diketahui bahwa program tersebut melakukan suatu plot data acak pada program berdasarkan areanya pada program ini kita melakukan regresi logistik. Pada gambar diatas dapat dilihat bahwa sistem regresi tersebut memiliki 4 buah data yang dimasukan dalam berbagai border dengan warna yang berbeda-beda. Setiap data memiliki 3 buah data lagi yang terdapat pada segmentasi-segmentasi warna pada program yang ada. Dalam menunjukan ketajaman segmentasi, pada program logistic regression dipengaruhi oleh nilai kurvanya. Semakin besar nilai kurvanya, maka akan semakin bagus titik temu segmentasi logistic regression-nya. Sebaliknya, ketika nilai kurvanya dikecilkan maka yang akan terjadi adalah titik temu kurvanya akan semakin memburuk.

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']

```

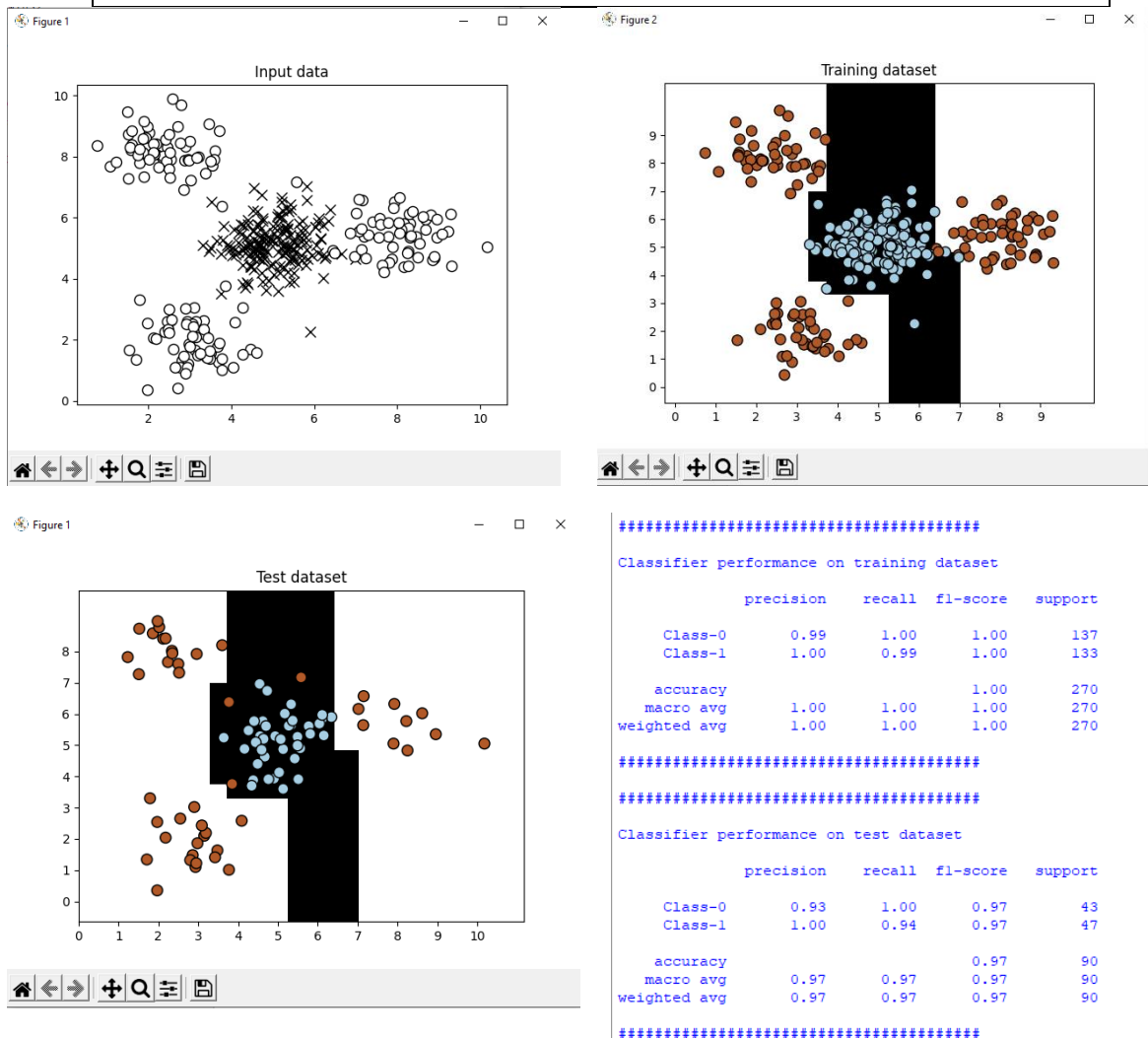
```

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 tree merupakan suatu metode yang membuat suatu keputusan yang benar berdasarkan data yang ada pada sebuah program yang telah dijalankan

dengan mempelajari aturan penentuan kategori berdasarkan fitur-fitur yang dimiliki oleh data. Decision tree dibedakan menjadi 2 jenis berdasarkan kategorinya, yaitu classification tree dan regression tree. Classification tree memiliki kategori berupa data diskrit berhingga, sedangkan regression tree memiliki kategori berupa data diskrit berhingga atau data kontinu. Biasanya pada decision tree kita menggunakan library tambahan pada python yaitu scikit-learn pada library python machine learning. Berdasarkan percobaan diatas kita menggunakan decision tree yang menggunakan metode if (jika) dengan menggunakan metode pohon. Pada decision tree terdapat precision, recall, f1-score, dan support. Support merupakan jumlah data yang digunakan pada program, f1-score merupakan harmonic means, recall merupakan banyaknya data yang dipanggil kembali, dan precision merupakan ketepatan menempati plot yang seharusnya. Dilihat pada performansi klasifikasinya pada training test terlihat pada f1-score memiliki nilai 1 yang mengartikan bahwa nilainya memiliki nilai yang baik. Sedangkan, pada saat test data test memiliki nilai 0.97 yang menandakan bahwa nilainya memiliki nilai yang baik karena pada setiap percobaan tidak ada percobaan yang benar-benar 100%. Data yang digunakan pada saat training test dengan test sangat berbeda.

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)

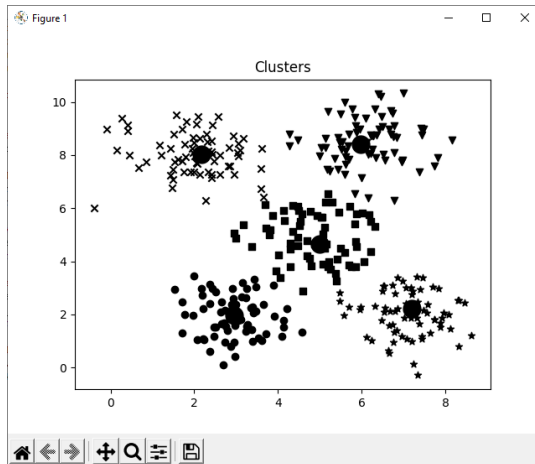
# Extract the centers of clusters
cluster_centers = meanshift_model.cluster_centers_
print('\nCenters of clusters:\n', cluster_centers)

# 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()
```

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

Mean_shift merupakan algoritma pengelompokan kuat lainnya yang digunakan dalam unsupervised belajar. Tidak seperti pengelompokan K-means, mean shift tidak membuat asumsi apapun, maka mean shift adalah algoritma non parametric. Algoritma ini pada dasarnya menetapkan titik data ke cluster secara iteratif dengan menggeser menunjuk ke arah kepadatan tertinggi dari titik data yaitu centroid. Berdasarkan program yang telah dijalankan didapati bahwa program ini bertujuan untuk mengklasifikasikan data yang ada pada program pada program juga didapati bahwa terdapat 5 buah data. Bentuk data tersebut dibentuk dalam berbagai bentuk seperti segitiga, silang, bintang, lingkaran, dan kotak. Setiap data memiliki region-region-nya sendiri yang dimana pusat dari setiap region tersebut akan berbentuk lingkaran tepat ditengah region-nya. Fungsinya adalah untuk mengelompokkan suatu data berdasarkan kemiripan datanya.

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
from sklearn.neighbors import NearestNeighbors

# Input data
X = np.array([[2.1, 1.3], [1.3, 3.2], [2.9, 2.5], [2.7, 5.4],
              [3.8, 0.9],
```

```

        [7.3, 2.1], [4.2, 6.5], [3.8, 3.7], [2.5, 4.1], [3.4,
1.9],
        [5.7, 3.5], [6.1, 4.3], [5.1, 2.2], [6.2, 1.1]])

# Number of nearest neighbors
k = 5

# Test datapoint
test_datapoint = [4.3, 2.7]

# Plot input data
plt.figure()
plt.title('Input data')
plt.scatter(X[:,0], X[:,1], marker='o', s=75, color='black')

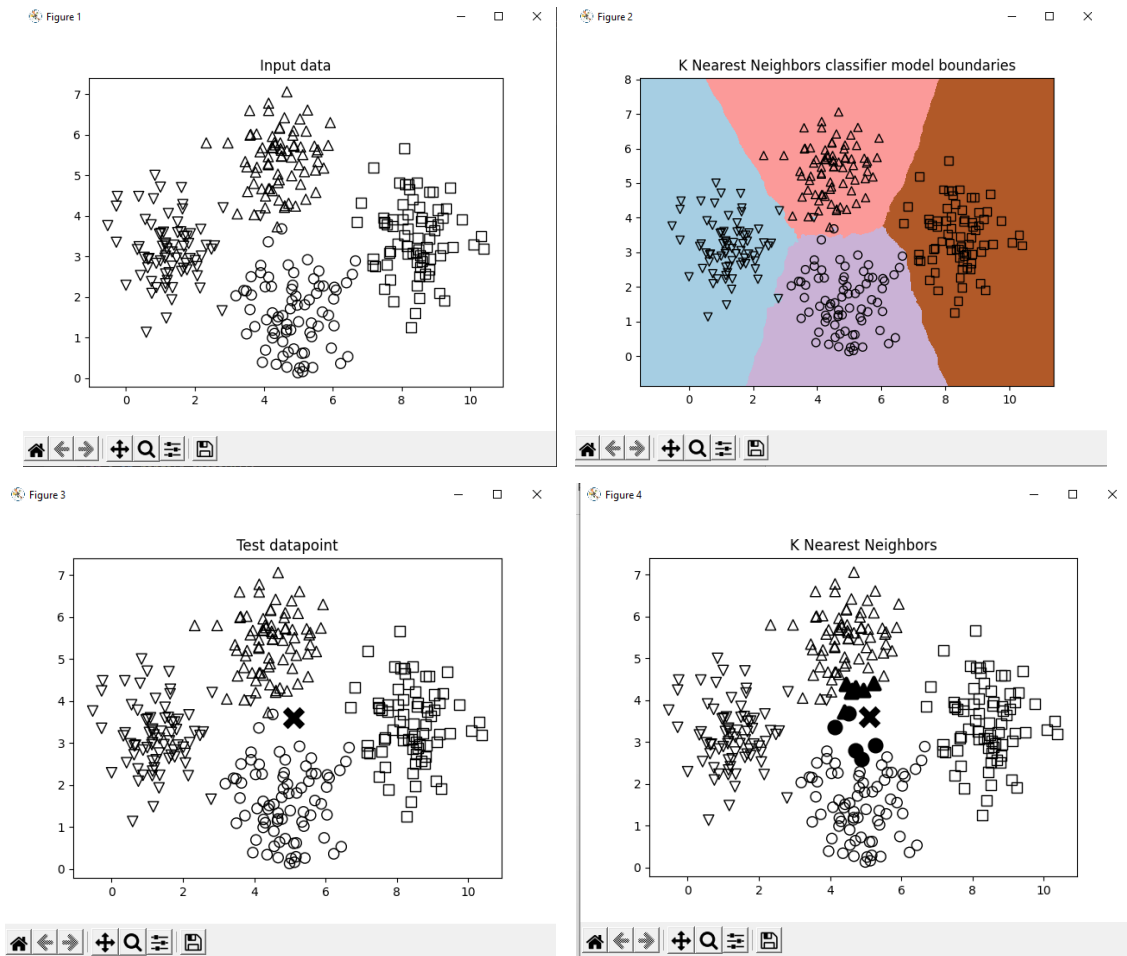
# Build K Nearest Neighbors model
knn_model = NearestNeighbors(n_neighbors=k,
algorithm='ball_tree').fit(X)
distances, indices = knn_model.kneighbors([test_datapoint])

# Print the 'k' nearest neighbors
print("\nK Nearest Neighbors:")
for rank, index in enumerate(indices[0][:k], start=1):
    print(str(rank) + " ==>", X[index])

# Visualize the nearest neighbors along with the test
datapoint
plt.figure()
plt.title('Nearest neighbors')
plt.scatter(X[:, 0], X[:, 1], marker='o', s=75, color='k')
plt.scatter(X[indices[0][:k][:, 0], X[indices[0][:k][:, 1],
marker='o', s=250, color='k', facecolors='none')
plt.scatter(test_datapoint[0], test_datapoint[1],
marker='x', s=75, color='k')

plt.show()

```



K-Nearest Neighbor (KNN) adalah suatu metode yang menggunakan algoritma supervised dimana hasil dari query instance yang baru diklasifikasi berdasarkan mayoritas dari kategori pada KNN. Tujuan dari algoritma ini adalah mengklasifikasikan obyek baru berdasarkan atribut dan training sample. Classifier tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Diberikan titik query, akan ditemukan sejumlah k obyek atau (titik training) yang paling dekat dengan titik query. Klasifikasi menggunakan voting terbanyak diantara klasifikasi dari k obyek.. algoritma KNN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari query instance yang baru. Berdasarkan program yang dilakukan dapat diketahui bahwa program tersebut membuat klasifikasi bagian-bagian program yang ada dengan region-region yang ada berdasarkan tetangga terdekatnya. Program tersebut memiliki data dengan menggunakan klasifikasi tetangganya. Suatu data dapat memiliki nilai region dan data yang berbeda, kasus tersebut terjadi pada bentuk region pink dan region ungu dimana pada region pink terdapat data lingkaran. Hal ini menyebabkan data lingkaran tersebut dapat dinamakan data dari region pink dan data dari region ungu. Algoritma program tersebut dilihat pada kedekatan tetangganya, label, dan juga region-nya.

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)
```

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

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

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

```
List of 7 states that border a coastal state:  
Georgia  
Massachusetts  
Arizona  
Nevada  
Vermont  
Pennsylvania  
Idaho
```

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

Berdasarkan program yang telah dijalankan, didapati bahwa program tersebut merupakan program yang membentuk relasi antara program dan data yang ada pada suatu program. Pada program yang telah dilakukan data yang telah dipersiapkan akan dilakukan relasi program dengan bantuan data tersebut. Data tersebut dibandingkan antara data `coastal_states` dengan `adjacent_states`.