

NAMA : WAHYU HANAFAI

NIM : 3332190073

KELAS : A

UTS KECERDASAN BUATAN

*Note: saya menjalankan program menggunakan Google Colaboratory.

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
from sklearn import linear_model

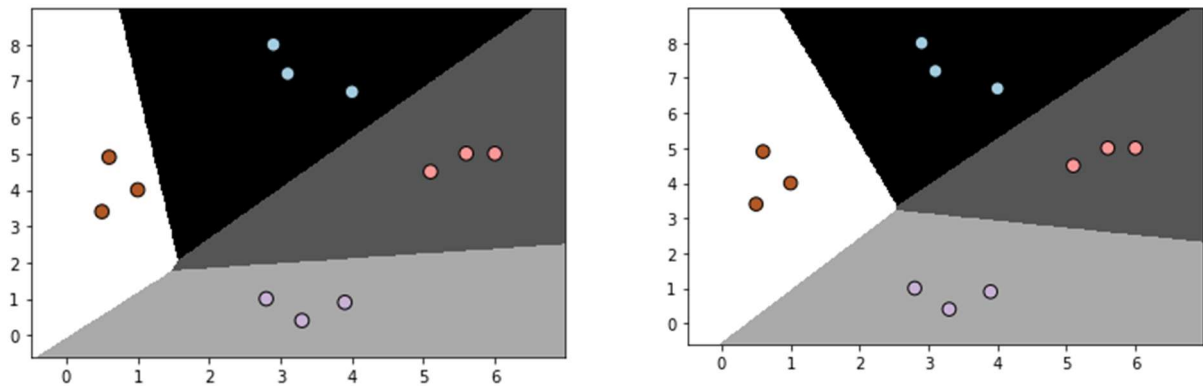
# Define sample input data
X = np.array([[3.1, 7.2], [4, 6.7], [2.9, 8], [5.1, 4.5], [6, 5], [5.6, 5],
, [3.3, 0.4], [3.9, 0.9], [2.8, 1], [0.5, 3.4], [1, 4], [0.6, 4.9]])
y = np.array([0, 0, 0, 1, 1, 1, 2, 2, 2, 3, 3, 3])

# 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)
```

Pada listing code *logistic_regression.py* harus mengimport library terlebih dahulu seperti numpy, matplotlib, dan file dari sklearn linear model, dan untuk merunning file ini harus run terlebih dahulu file *utilities.py*, setelah itu dari file *utilities.py* mendefinisikan fungsi visual classifier yang dimana untuk mendefinisikan nilai minimum dan maksimal untuk variabel X dan Y .



Gambar 1. $C = 1$ dan $C = 100$

Pada gambar 1 diatas hasil merunning program *logistic_regression.py* variabel x dan y tersebar beberapa segmen dengan nilai $C = 1$ dan $C = 100$, C disini adalah kurva yang membatasi data data di segmen putih, hitam, abu tua, dan abu muda, perbatasan area atau segmen data pada $C = 1$ lebih sempit kondisi ini dinamakan underfit yang dimana kondisi underfit tersebut adalah sebuah data yang bisa menyerupai data lainnya yang seharusnya tidak sama, dan juga pada $C = 100$ pembagian per segmen sangat merata disebut kondisi overfit yang dimana tidak ada kesalahan dari model data yang diambil.

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
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report

from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
```

```

# Load input data from google drive
input_file = '/content/drive/My Drive/Data UTS AI/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')

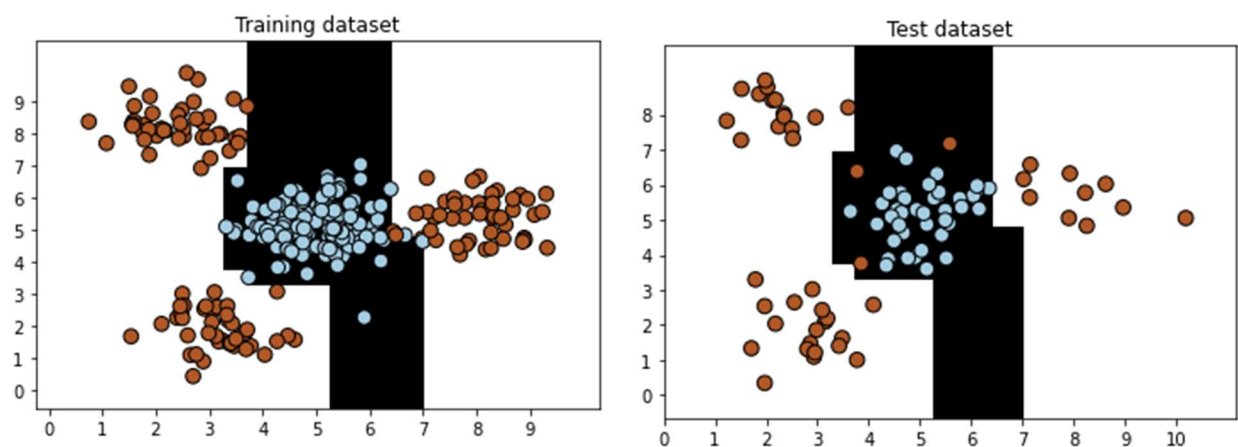
# Evaluate classifier performance
class_names = ['Class-0', 'Class-1']
print("\n" + "#" * 40)
print("\nClassifier performance on training dataset\n")
print(classification_report(y_train, classifier.predict(X_train), target_n
ames=class_names))
print("#" * 40 + "\n")

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 disini adalah pemilihan input berdasarkan data output yang diinginkan atau kita bisa samakan dengan logika statement kondisional (if else). Pada listing code *decision_trees.py* harus mengimport library terlebih dahulu seperti numpy, matplotlib, classification report, DecisionTreeClassifier, dan train_test_split dan juga untuk merunning file ini harus run terlebih dahulu file *utilities.py*, setelah itu dari file *utilities.py* mendefinisikan fungsi visual classifier yang dimana untuk mendefinisikan nilai minimum dan maksimal untuk variabel X dan Y, dan juga meng-load file *data_decision_trees.txt* sebagai input agar program dapat dijalankan.



Gambar 2.1 Training dataset dan Test data set

Pada gambar diatas adalah hasil dari program *decision_trees.py* yang kita jalankan, bentuk bintik biru disini adalah data yang presisi atau akurat sedangkan coklat itu adalah data yang kurang akurat.

| Classifier performance on training dataset | | | | | Classifier performance on test dataset | | | | |
|--|-----------|--------|----------|---------|--|-----------|--------|----------|---------|
| | precision | recall | f1-score | support | | precision | recall | f1-score | support |
| Class-0 | 0.99 | 1.00 | 1.00 | 137 | Class-0 | 0.93 | 1.00 | 0.97 | 43 |
| Class-1 | 1.00 | 0.99 | 1.00 | 133 | Class-1 | 1.00 | 0.94 | 0.97 | 47 |
| accuracy | | | 1.00 | 270 | accuracy | | | 0.97 | 90 |
| macro avg | 1.00 | 1.00 | 1.00 | 270 | macro avg | 0.97 | 0.97 | 0.97 | 90 |
| weighted avg | 1.00 | 1.00 | 1.00 | 270 | weighted avg | 0.97 | 0.97 | 0.97 | 90 |

Gambar 2.2 data dari masing masing classifier training dataset dan test dataset

Pada gambar diatas adalah hasil classifier dari kedua data tersebut, berbeda dari precision, recall, F1- score, dan support. Data training sangat akurat tetapi pada data test akurasi nya menurun.

- Precision disini adalah ketepatan data menempati plot hitam yang seharusnya.
- Recall disini adalah banyaknya data yang dipanggil kembali.
- F1-score disini adalah nilai dari harmonic mean.
- Support disini adalah banyaknya data.

Adapun rumus mencari precision, recall, dan F1-score yaitu:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - Score = \frac{Precision * Recall}{Precision + Recall}$$

Keterangan:

- TP = True positif
- FN = False Negatif
- FP = False Positif

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('/content/drive/My Drive/Data UTS AI/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')

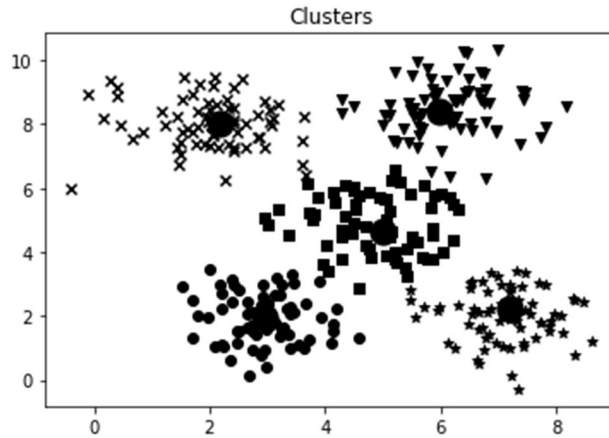
    # Plot the cluster center
    cluster_center = cluster_centers[i]
    plt.plot(cluster_center[0], cluster_center[1], marker='o',
             markerfacecolor='black', markeredgecolor='black',
             markersize=15)

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

```

means shift disini adalah pergeseran rata rata atau bisa disebut juga pengelompokkan data yang sama. Pada listing code *mean_shift.py* harus mengimport library terlebih dahulu seperti numpy, matplotlib, MeanShift, estimate_bandwidth dan cycle, dan juga untuk merunning file ini harus run terlebih dahulu file *utilities.py*, setelah itu dari file *utilities.py* mendefinisikan fungsi visual classifier yang dimana untuk mendefinisikan nilai minimum dan maksimal untuk

variabel X dan Y, dan juga meng-load file data_clustering.txt sebagai input agar program dapat dijalankan.



Gambar 3. Data Clustering

Dapat dilihat diatas pengelompokkan data berdasarkan kesamaan, data x, data segitiga, data lingkaran, dan data bintang masing masing dikelompokkan berdasarkan kedekatan informasinya, semakin mendekati lingkaran hitam semakin mirip data yang dikelompokkannya.

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 = '/content/drive/My Drive/Data UTS AI/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')

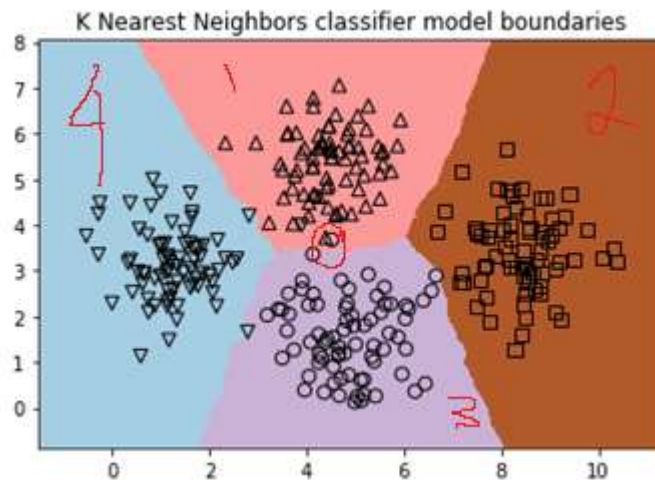
for i in range(X.shape[0]):
    plt.scatter(X[i, 0], X[i, 1], marker=mapper[i],
                s=75, edgecolors='black', facecolors='none')

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

plt.show()

```

Nearest neighbour disini adalah pengelompokkan data yang sama tetapi bisa menyerupai pada data lainnya. Pada listing code *nearest_neighbors_classifier.py* harus mengimport library terlebih dahulu seperti numpy, matplotlib, neighbours dan datasets, dan juga untuk merunning file ini harus run terlebih dahulu file *utilities.py*, setelah itu dari file *utilities.py* mendefinisikan fungsi visual classifier yang dimana untuk mendefinisikan nilai minimum dan maksimal untuk variabel X dan Y, dan juga meng-load file *data_clustering.txt* sebagai input agar program dapat dijalankan.



Gambar 4. K Nearest Neighbor Classifier

Pada gambar diatas pengelompokkan data yang sama, tetapi lihat pada tengah yang dilingkari terdapat data ke-3 tercampur pada segmen data ke-1, hal ini terjadi karena model memprediksi data yang ditengah ini bisa terbaca sebagai data ke-1 dan juga bisa terbaca sebagai data ke-3. Data ini ditentukan oleh data kedekatan dengan data tetangga lainnya.

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 = '/content/drive/My Drive/Data UTS AI/coastal_states.txt'
file_adjacent = '/content/drive/My Drive/Data UTS AI/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)

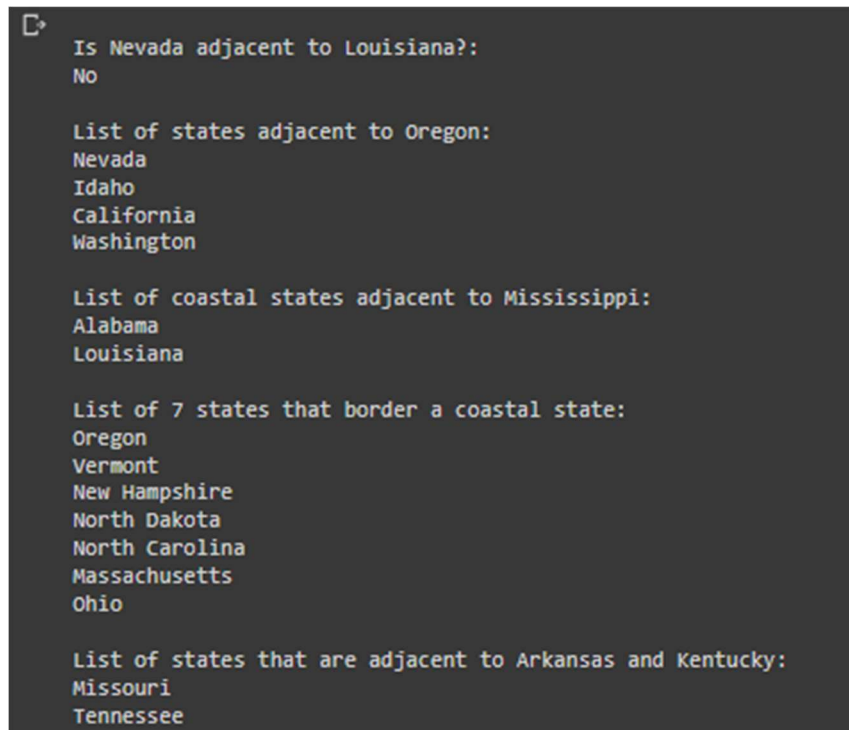
# States adjacent to Mississippi that are coastal
output = run(0, x, adjacent('Mississippi', x), coastal(x))
print('\nList of coastal states adjacent to Mississippi:')
for item in output:
    print(item)

# 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)

```

Algoritma dari *states.py* disini adalah menentukan negara apa saja yang berdekatan dan juga negara yang mempunyai pantai, Pada listing code *states.py* harus mengimport library terlebih dahulu seperti *logpy*, *run*, *fact*, *eq*, *Relation*, *var*, dan juga meng-load file *coastal_states.txt* dan *adjacent_states.txt* sebagai input agar program dapat dijalankan.



```
Is Nevada adjacent to Louisiana?:  
No  
  
List of states adjacent to Oregon:  
Nevada  
Idaho  
California  
Washington  
  
List of coastal states adjacent to Mississippi:  
Alabama  
Louisiana  
  
List of 7 states that border a coastal state:  
Oregon  
Vermont  
New Hampshire  
North Dakota  
North Carolina  
Massachusetts  
Ohio  
  
List of states that are adjacent to Arkansas and Kentucky:  
Missouri  
Tennessee
```

Gambar 5. Hasil program *states.py*

Pada gambar diatas adalah hasil dari program *states.py* yang dijalankan, terlihat yang pertama apakah negara Nevada berdekatan dengan negara Louisiana? Jawaban nya tidak, hal ini berdasarkan data yang ada pada *adjacent_state.txt* negara Nevada ini tidak berdekatan dengan Louisiana, program ini memanggil jawaban dari file data input *adjacent_state.txt* maupun *coastal_states.txt* , dan hanya menampilkan jawaban yang sesuai dari kedua data tersebut.