# LAPORAN UTS PENGENALAN POLA


**Disusun untuk Memenuhi UTS**
**Mata Kuliah Peminatan Pengenalan Pola yang**
**Diampu oleh Dr. Eng. Adi Wibowo, S.Si., M.Kom.**

**OLEH :**

**Nama : Wafiq Afifah**
**NIM : 24060119130088**

**PROGRAM STUDI S-1 INFORMATIKA**
**FAKULTAS SAINS DAN MATEMATIKA**
**UNIVERSITAS DIPONEGORO**
**SEMARANG**
**2022**

**UTS Pengenalan pola Semester Genap 2021/2022**

Menggunakan dataset Alzheimer Disease MRI (Magnetic Resonance Imaging) yang terdapat pada https://www.kaggle.com/datasets/sachinkumar413/alzheimer-mri-dataset?resource=download

1. Persiapan

   Mengimport semua library yang dibutuhkan. Mengupload dataset ke drive, kemudian memindahkan direktori ke lokasi dataset serta melakukan 'Mount Drive' agar dapat diakses.

2. Menetapkan 50 citra MRI dari setiap kelas (Total 200 citra MRI)

   Setelah berhasil mengakses dataset di dalam folder drive, simpan semua informasi seperti path, file name, dan kelas kedalam array dan iterasi 50 agar dapat mengakses 50 dataset per kelas.

```
[ ]  # Pindah Direktori
     os.chdir('/content/drive/MyDrive/UTS Pengenalan Pola/')
     mypath='Dataset/'
```

```
[ ]  # Load dataset dan membaginya menjadi 50 data per kelas
     file_name = []
     tag = []
     full_path = []
     for path, subdirs, files in os.walk(mypath):
         for i, name in zip(range(50), files):
             full_path.append(os.path.join(path, name))
             tag.append(path.split('/')[-1])
             file_name.append(name)
```

Untuk melihat apakah setiap kelas sudah berasil diambil 50 dataset, lakukan groupby berdasarkan kelas terhadap dataframe dari dataset. Terlihat setiap kelas sudah terdiri dari 50 dataset.

```
df = pd.DataFrame({"path":full_path,'file_name':file_name,"Class":tag})
df.groupby(['Class']).size()
```

```
Class
Mild_Demented         50
Moderate_Demented     50
Non_Demented          50
Very_Mild_Demented    50
dtype: int64
```

3. Melakukan ekstraksi fitur dengan **GLCM**.

   Sebelum melakukan ekstraksi fitur dengan GLCM, baca data image dan ubah menjadi gray lalu masukkan kedalam array images.

```
# Melakukan ekstraksi fitur dengan GLCM
from skimage.feature import greycomatrix, greycoprops

images = []
for path in df.path:
    img = cv2.imread(os.path.join(path))
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    images.append(gray)
```

Setelah itu lakukan ekstraksi fitur dengan GLCM seperti dibawah ini. Ekstraksi fitur GLCM merupakan suatu teknik untuk mendapatkan nilai statistik orde ke-2 dengan menghitung probabilitas hubungan kedekatan antara dua buah piksel pada jarak d dan sudut teta tertentu. Untuk property atau fitur yang akan dipakai yaitu, dissimilarity, correlation, homogeneity, contrast, ASM, dan energy, untuk sudut atau angles menggunakan 0, 45, 90, 135 derjat sehingga menghasilkan 24 fitur ditambah 1 kolom label yang merupakan kelas dataset

```
def calc_glcm_all(img, label, props, dists=[5], agls=[0, np.pi/4,
                  np.pi/2, 3*np.pi/4], lvl=256, sym=True, norm=True):

    glcm = greycomatrix(img,
                        distances=dists,
                        angles=agls,
                        levels=lvl,
                        symmetric=sym,
                        normed=norm)
    feature = []
    glcm_props = [propery for name in props for propery in greycoprops(glcm, name)[0]]
    for item in glcm_props:
            feature.append(item)
    feature.append(label)

    return feature

properties = ['dissimilarity', 'correlation', 'homogeneity', 'contrast', 'ASM', 'energy']

glcm_all = []
for img, label in zip(images, df.Class):
    glcm_all.append(
            calc_glcm_all(img,
                          label,
                          props=properties)
                         )

columns = []
angles = ['0', '45', '90','135']
for name in properties :
    for ang in angles:
        columns.append(name + "_" + ang)

columns.append("label")
```

Setelah berhasil ekstraksi fitur, atur array glcm_all dan columns kedalam dataframe df_glcm agar lebih mudah untuk dilihat dalam bentuk tabel, dan sebagai opsi lain bisa d convert ke file csv dan lainnya.

```
df_glcm = pd.DataFrame(glcm_all,
                       columns = columns)

# Menyimpan ke csv
# glcm_df.to_csv("glcm_coffee_dataset.csv")

df_glcm.head()
```

Berikut penampakan tabel df_glcm.

| | dissimilarity_0 | dissimilarity_45 | dissimilarity_90 | dissimilarity_135 | correlation_0 | correlation_45 | correlation_90 | correlation_135 | homogeneity_ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 33.068725 | 33.883325 | 26.870681 | 34.606009 | 0.782609 | 0.777254 | 0.839407 | 0.770230 | 0.44566 |
| 1 | 32.094385 | 33.326353 | 30.672637 | 33.163632 | 0.802155 | 0.789030 | 0.813993 | 0.788378 | 0.39337 |
| 2 | 34.725737 | 36.655957 | 31.686865 | 36.341441 | 0.757293 | 0.735447 | 0.788476 | 0.741897 | 0.39460 |
| 3 | 32.644563 | 34.139828 | 29.586255 | 34.566467 | 0.754007 | 0.743171 | 0.785877 | 0.736890 | 0.43283 |
| 4 | 34.194487 | 36.608351 | 31.428100 | 35.184053 | 0.746210 | 0.720853 | 0.774349 | 0.742174 | 0.39329 |

5 rows × 25 columns

4. Encoding dataset

   Karena kolom label pada dataset berupa string, maka dilakukan encoding agar label dapat diolah dalam bentuk integer.

```
# Encoding dataset
from sklearn.preprocessing import LabelEncoder

dataset = df_glcm.copy()
dataset.head()

encoder = LabelEncoder()

for c in dataset.columns[1:]:
    if(dataset[c].dtype=='object'):
        dataset[c] = encoder.fit_transform(dataset[c])
    else:
        dataset[c] = dataset[c]

dataset.head()
```

5. Memisahkan setiap kelas/label menjadi 40 citra MRI untuk data latih dan 10 citra MRI lainnya untuk data uji. (Total terdapat 160 citra MRI sebagai data latih dan 40 citra MRI sebagai data uji).

   Deklarasikan variabel X untuk menampung dataset dengan kolom selain label, dan y untuk menampung dataset dengan kolom label, kemudian lakukan split data dengan ukuran tes 0.2 dari 200 dataset yang ada, kemudian atur random_state=0 dan stratify=y agar setiap kelas mendapat jumlah data uji dan data latih dalam jumlah yang sama sesuai permintaan soal.

```
[ ]  # Split dataset
     from sklearn.model_selection import train_test_split

     X= dataset.drop(['label'], axis = 1)
     y= dataset['label']
     X_train, X_test, y_train, y_test = train_test_split(
         X, y, test_size=0.20, random_state=0, stratify=y)
```

Untuk melihat apakah benar setiap kelas memiliki data uji dan data latih yang sama banyak, lakukan seperti dibawah ini.

```
     df_train = pd.DataFrame({'Class':y_train})
     df_test = pd.DataFrame({'Class':y_test})

     print('Total pembagian data train dan data uji :')
     print('train size', len(df_train))
     print('test size', len(df_test))
     print('\nJumlah data train per kelas :')
     print(df_train.groupby(['Class']).size(),'\n')
     print('\nJumlah data tes per kelas :')
     print(df_test.groupby(['Class']).size(),'\n')
```

Berikut hasilnya

```
     Total pembagian data train dan data uji
     train size 160
     test size 40

     Jumlah data train per kelas
     Class
     0    40
     1    40
     2    40
     3    40
     dtype: int64


     Jumlah data tes per kelas
     Class
     0    10
     1    10
     2    10
     3    10
     dtype: int64
```

6. Klasifier yang digunakan adalah **Random Forest** untuk melakukan tahapan pelatihan dataset

```
[99] # Klasifikasi menggunakan Random Forest
     parameters = {'bootstrap': True,
                   'min_samples_leaf': 2,
                   'n_estimators': 20,
                   'min_samples_split': 5,
                   'max_features': 'sqrt',
                   'max_depth': None,
                   'max_leaf_nodes': None}
```

```
[100] RF_model = RandomForestClassifier(**parameters)
```

```
[101] RF_model.fit(X_train, y_train)

      RandomForestClassifier(max_features='sqrt', min_samples_leaf=2,
                             min_samples_split=5, n_estimators=20)
```

7. Menguji model dengan data uji yang telah didapat sebelumnya
   Maka didapat akurasi Random Forest sekitar 0.5

```
[102] RF_predictions = RF_model.predict(X_test)
```

```
score = accuracy_score(y_test ,RF_predictions)
print('Accuracy Random Forest Model:',score)

Accuracy Random Forest Model: 0.5
```

8. Menentukan confusion matrix yang berukuran 4x4 serta menghitung akurasi, precision, recall, dan f1 score.
   Pertama-tama import accuracy_score, confusion_matrix, dan classification_report pada sklearn.metrics, dan lakukan seperti dibawah ini.

```
[104] from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Make predictions on validation dataset
rf = RandomForestClassifier()
rf.fit(X_train, y_train)
predictions = rf.predict(X_test)
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

Berikut hasil yang didapat, terdapat accuracy score, kemudian confusion matrix, dan terkahir hasil precision, recall, f1-score.

```
0.5
[[4 2 0 4]
 [2 6 2 0]
 [1 1 5 3]
 [1 1 3 5]]
              precision    recall  f1-score   support

           0       0.50      0.40      0.44        10
           1       0.60      0.60      0.60        10
           2       0.50      0.50      0.50        10
           3       0.42      0.50      0.45        10

    accuracy                           0.50        40
   macro avg       0.50      0.50      0.50        40
weighted avg       0.50      0.50      0.50        40
```

9. Membandingkan penggunaan fitur atau classifier yang berbeda sehingga diperoleh confusion matriks dan performa yang berbeda pula.

   Perbandingan yang akan digunakan yaitu classifier, dimana membandingkan Random Forest dengan K-Nearest Neighbors dan SVM. Imput terlebih dahulu library yang akan digunakan oleh K-NN dan SVM, setelah itu atur Random Forest, K-NN, dan SVM kedalam sebuah array moduls, agar dapat dieksekusi sekaligus. Lakukan pengujian accuracy dan didapat hasil seperti dibawah ini.

```
[189] from sklearn.neighbors import KNeighborsClassifier
      from sklearn.svm import SVC
      from sklearn import model_selection
```

```
[190] # Spot Check Algorithms
      models = []
      models.append(('KNN', KNeighborsClassifier()))
      models.append(('RF', RandomForestClassifier()))
      models.append(('SVM', SVC()))
```

```
# evaluate each model in turn
results = []
names = []
for name, model in models:
    kfold = model_selection.KFold(n_splits=10, random_state=7, shuffle=True)
    cv_results = model_selection.cross_val_score(model, X_train, y_train,
    cv = kfold, scoring='accuracy')
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f" % (name, cv_results.mean())
    print(msg)
```

(parameter) n_splits: Literal[10]

```
KNN: 0.406250
RF: 0.537500
SVM: 0.231250
```

Didapat akurasi RF atau Random Forest lebih baik daripada K-NN dan SVM

Berikut confusion matriks perhitungan precision, recall, f1-score, dan accuracy untuk K-NN dan SVM

## Confusion Matriks KNN

```
knn = KNeighborsClassifier()
knn.fit(X_train, y_train)
predictions = rf.predict(X_test)
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
0.5
[[4 2 0 4]
 [2 6 2 0]
 [1 1 5 3]
 [1 1 3 5]]
              precision    recall  f1-score   support

           0       0.50      0.40      0.44        10
           1       0.60      0.60      0.60        10
           2       0.50      0.50      0.50        10
           3       0.42      0.50      0.45        10

    accuracy                           0.50        40
   macro avg       0.50      0.50      0.50        40
weighted avg       0.50      0.50      0.50        40
```

## Confusion Matriks SVM

```
svm = SVC()
rf.fit(X_train, y_train)
predictions = rf.predict(X_test)
print(accuracy_score(y_test, predictions))
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

```
0.475
[[4 1 1 4]
 [1 5 3 1]
 [1 1 5 3]
 [1 0 4 5]]
              precision    recall  f1-score   support

           0       0.57      0.40      0.47        10
           1       0.71      0.50      0.59        10
           2       0.38      0.50      0.43        10
           3       0.38      0.50      0.43        10

    accuracy                           0.48        40
   macro avg       0.51      0.47      0.48        40
weighted avg       0.51      0.47      0.48        40
```

10. Ensamble Learning

Install dan import terlebih dahulu vestack yang merupakan salah satu library untuk ensamble learning model stacking.

```
[134] # Install and Import Dependencies
      !pip install vecstack
      from vecstack import stacking
```

Setelah itu inisialisasikan model yang akan dikenakan stacking, dan eksekusi bersama dengan X_train, y_train dan X_test, beserta beberapa parameter pendukung lainnya. Kemudian fit kan model yang digunakan dengan hasil stacking S_train dan y_train, dan lakukan prediksi, maka didapat hasil seperti di bawah ini.

```
model_1 = KNeighborsClassifier()
model_2 = RandomForestClassifier()
model_3 = SVC()

# putting all base model objects in one list
all_models = [model_1, model_2, model_3]
print('Number of model:',len(all_models))

t0 = time()

# Compute stacking features
S_train, S_test = stacking(all_models, X_train, y_train, X_test, n_folds = 4,
                metric=accuracy_score,shuffle = True, random_state = 7, verbose = 1)

# Initialize 2-nd level model
model = model_1

# Fit 2-nd level model
model = model.fit(S_train, y_train)

# Predict
y_test_pred = model.predict(S_test)

train_test_time = time() - t0
```

```
# Predict
y_test_pred = model.predict(S_test)

train_test_time = time() - t0
```

```
Number of model: 3
task:         [regression]
metric:       [accuracy_score]
mode:         [oof_pred_bag]
n_models:     [3]

model  0:     [KNeighborsClassifier]
    ----
    MEAN:     [0.38125000] + [0.03697550]
    FULL:     [0.38125000]

model  1:     [RandomForestClassifier]
    ----
    MEAN:     [0.46875000] + [0.05115845]
    FULL:     [0.46875000]

model  2:     [SVC]
    ----
    MEAN:     [0.20000000] + [0.03952847]
    FULL:     [0.20000000]
```

Selanjutnya dapat dihitung accuracy dari ensamble learning model stacking ini beserta train dan tes time nya.

```
[186] ## Print vecstack result
      accuracy = accuracy_score(y_test, y_test_pred)*100
      print("Stacking accuracy : {0:.2f}%".format(accuracy))
      print("train and test time: {0:.2f}s".format(train_test_time))
```

```
Stacking accuracy : 32.50%
train and test time: 0.72s
```

11. Perhatikan dataset, tentukan persamaan regresi dan hitung SSE nya.

| No. | X | y | X*y | X^2 | y^2 |
|---|---|---|---|---|---|
| 1 | 5 | 85 | 425 | 25 | 7225 |
| 2 | 4 | 103 | 412 | 16 | 10609 |
| 3 | 6 | 70 | 420 | 36 | 4900 |
| 4 | 5 | 82 | 410 | 25 | 6724 |
| 5 | 5 | 89 | 445 | 25 | 7921 |
| 6 | 5 | 98 | 490 | 25 | 9604 |
| 7 | 6 | 66 | 396 | 36 | 4356 |
| 8 | 6 | 95 | 570 | 36 | 9025 |
| 9 | 2 | 169 | 338 | 4 | 28561 |
| 10 | 7 | 70 | 490 | 49 | 4900 |
| 11 | 7 | 48 | 336 | 49 | 2304 |
| Total | 58 | 975 | 4732 | 326 | 96129 |

a = 195.4684685
b = -20.26126126
Persamaan = 195.4685-20.2613x

Didapat persamaan regresinya yaitu y`=195.4685-20.2613x

| No. | y` | (y-y`)^2 |
|---|---|---|
| 1 | 94.16216216 | 83.94521549 |
| 2 | 114.4234234 | 130.4946027 |
| 3 | 73.9009009 | 15.21702784 |
| 4 | 94.16216216 | 147.9181885 |
| 5 | 94.16216216 | 26.64791819 |
| 6 | 94.16216216 | 14.72899927 |
| 7 | 73.9009009 | 62.42423505 |
| 8 | 73.9009009 | 445.1719828 |
| 9 | 154.9459459 | 197.5164354 |
| 10 | 53.63963964 | 267.6613911 |
| 11 | 53.63963964 | 31.80553526 |
| Total | | 1423.531532 |

Maka didapat hasil SSE nya 1423.531532