DAFTAR PUSTAKA

[1] F. Rahmah, H. Fitriyah And I. Arwani, "Sistem Klasifikasi Aktivitas Manusia Menggunakan Sensor Accelerometer Dan Gyroscope Dengan Metode K-Nearest Neighbor Berbasis Arduino," *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer,* 2018.

[2] M. S. Chen, J. Han And P. S. Yu, "Data Mining: An Overview From Database Perspective," *Ieee Transactions On Knowledge And Data Engijeering,* Vol. 8, 1996.

[3] S. Kaghyan And H. Sarukhanyan , "Activity Recognition Using K-Nearest Neighbor Algorithm On Smartphone With Tri-Axial Accelerometer," *International Journal "Information Models And Analyses,* Vol. 1, 2012.

[4] S. Putra, I. Muslim And M. I. Zul, "Identifikasi Aktivitas Manusia Di Dalam Ruangan Menggunakan IP Camera Dengan Metode Template Matching Dan Algoritma Klasifikasi," *Researchgate,* 2016.

[5] M. Zubair, K. Song And C. Yoon, "Human Activity Recognition Using Wearable Accelerometer Sensors," *IEEE International Conference On Consumer Electronics-Asia (ICCE-Asia),* 2016.

[6] A. Setiyandi, "Pengenalan Aktivitas Pergerakan Manusia Berbasis Sensor Accelerometer Dan Gyroscope Menggunakan Algoritma K-Means," *Politeknik Negeri Batam,* Pp. 1-31, 22 Juni 2017.

[7] T. V. Gestel, J. A. K. Suykens, G. Lanckriet, A. Lambrechts, B. D. Moor And J. Vandewalle, "Multiclass LS-Svms: Moderated Outputs And Coding-Decoding Schemes," *Neural Processing Letters,* P. 45–58, February 2002.

[8] L. Zhou, K. K. Lai And L. Yu, "Least Squares Support Vector Machines Ensemble Models For Credit Scoring," *Sciencedirect,* Vol. Volume 37, No. Issue 1, Pp. 127-133, 2010.

[9] L. Breiman, "Bagging Predictors," *Department Of Statistics University Of California At Berkeley,* September 1994.

[10] H. Gjoreski And M. Gams, "Accelerometer Data Preparation For Activity Recognition," *Researchgate,* 2011.

[11] C. R. Sari, "Teknik Data Mining Menggunakan Classification Dalam Sistem Penunjang Keputusan Peminatan SMA Negeri 1 Polewali," *Indonesian Journal On Networking And Security,* 2016.

[12] R. Purba, "Data Mining : Masa Lalu, Sekarang Dan Masa Mendatang," 2012.

[13] K. Sembiring, "Tutorial SVM Bahasa Indonesia," Bandung, Teknik Informatika ITB , 2007.

[14] C.-. W. Hsu And C. -J. Lin, "A Comparison Of Methods For Multi-Class Support Vector Machines," In *National Taiwan University*, Taiwan.

[15] D. Sinta, "Metode Ensemble K-Nearest Neighbor Untuk Prediksi Harga Beras Di Indonesia," *Sekolah Pascasarjana Institut Pertanian Bogor,* 2015.

[16] C. P. Lim And W. Y. Goh, "The Application Of An Ensemble Of Boosted Elman Networks To Time Series Prediction: A Benchmark Study," *International Journal Of Computational Intelligence,* Vol. 3, 2007.

[17] J. Han, M. Kamber And J. Pie, "Data Mining : Concepts And Techiques," 2012.

[18] A. Fikri, "Penerapan Data Mining Untuk Mengetahui Tingkat Kekuatan Beton Yang Dihasilkan Dengan Metode Estimasi Menggunakan Linear Regression," *Universitas Dian Nuswantoro,* 2009.

LAMPIRAN

Lampiran I Program pengambilan data pada *smarthphone*

```
import android.content.Context
import android.hardware.Sensor
import android.hardware.SensorEvent
import android.hardware.SensorEventListener
import android.hardware.SensorManager
import android.support.v7.app.AppCompatActivity
import android.os.Bundle
import kotlinx.android.synthetic.main.activity_main.*

class MainActivity : AppCompatActivity(), SensorEventListener {

    override fun onSensorChanged(event: SensorEvent?) {

        Thread.sleep(40)
        if (event!!.sensor.type == Sensor.TYPE_ACCELEROMETER) {
            aclX.text = "${event.values[0]}"
            aclY.text = "${event.values[1]}"
            aclZ.text = "${event.values[2]}"
        } else {
            gyroX.text = "${event.values[0]}"
            gyroY.text = "${event.values[1]}"
            gyroZ.text = "${event.values[2]}"
        }
    }

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```
        val mSensorManager = getSystemService(Context.SENSOR_SERVICE) as
SensorManager
        val aSensor: Sensor? =
mSensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER)
        val gSensor: Sensor? =
mSensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE)



        aSensor?.also { sensor ->
            mSensorManager.registerListener(this, sensor,
SensorManager.SENSOR_DELAY_NORMAL)
        }

        gSensor?.also { sensor ->
            mSensorManager.registerListener(this, sensor,
SensorManager.SENSOR_DELAY_NORMAL)
        }
    }
}
```

Lampiran II Program SVM

```
#load libraries
import pandas
from pandas.tools.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import cross_validation
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC

#load dataset
url = "3 Class.csv"
names = ['A1', 'A2', 'A3', 'A4', 'A5', 'A6','class']

dataset = pandas.read_csv(url, names=names)

#shape
print(dataset.shape)

#head
print(dataset.head(35))

#descriptions
print(dataset.describe())

#distribusi class
print(dataset.groupby('class').size())

#box dan whisker plot
```

```
dataset = dataset.convert_objects(['A1', 'A2', 'A3', 'A4', 'A5', 'A6','class'],
convert_numeric=True)
dataset.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False)
plt.show()

#histogram
dataset.hist()
plt.show()

#scatter plot matrix
scatter_matrix(dataset)
plt.show()

#split-out validasi dataset
array = dataset.values
X = array[:,0:6]
Y = array[:,6]
validation_size = 0.30
seed = 7
X_train, X_validation, Y_train, Y_validation = cross_validation.train_test_split(X, Y,
test_size=validation_size, random_state=seed)

# Test options and evaluation metric
num_folds = 10
num_instances = len(X_train)
seed = 7
scoring = 'accuracy'

# Spot Check Algorithms/klasifikasi
models = []
```

```
models.append(('SVM' , SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0,
shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None,
verbose=False, max_iter=-1, decision_function_shape=None, random_state=None)))

# evaluate each model in turn
results = []
names = []
for name, model in models:
        kfold = cross_validation.KFold(n=num_instances, n_folds=num_folds,
random_state=seed)
        cv_results = cross_validation.cross_val_score(model, X_train, Y_train, cv=kfold,
scoring=scoring)
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)

# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()

# Make predictions on validation dataset
svm=SVC()
svm.fit(X_train, Y_train)
predictions=svm.predict(X_validation)

print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
```

```
print(classification_report(Y_validation, predictions))
```

Lampiran III Program *Ensemble*

```
#load libraries
import pandas
from pandas.tools.plotting import scatter_matrix
import matplotlib.pyplot as plt
from sklearn import cross_validation
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier

url = "3 Class.csv"
names = ['A1', 'A2', 'A3', 'A4', 'A5', 'A6','class']

dataset = pandas.read_csv(url, names=names)

#shape
print(dataset.shape)

#head
print(dataset.head(35))

#descriptions
print(dataset.describe())

#distribusi class
print(dataset.groupby('class').size())
```

```
#box dan whisker plot
dataset = dataset.convert_objects(['A1', 'A2', 'A3', 'A4', 'A5', 'A6','class'],
convert_numeric=True)
dataset.plot(kind='box', subplots=True, layout=(3,3), sharex=False, sharey=False)
plt.show()

#histogram
dataset.hist()
plt.show()

#scatter plot matrix
scatter_matrix(dataset)
plt.show()

#split-out validasi dataset
array = dataset.values
X = array[:,0:6] #jumlah kolom
Y = array[:,6] #class
validation_size = 0.30
seed = 7 #cross validasi
X_train, X_validation, Y_train, Y_validation = cross_validation.train_test_split(X, Y,
test_size=validation_size, random_state=seed)

# Test options and evaluation metric
num_folds = 10
num_instances = len(X_train)
seed = 7
scoring = 'accuracy'
 #pemanggilan svm
Enslr=SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0, shrinking=True,
probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False,
max_iter=-1, decision_function_shape=None, random_state=seed)
```

```
# Spot Check Algorithms/klasifikasi
models = []
#models.append(('SVM' , SVC(C=1.0, kernel='rbf', degree=3, gamma='auto', coef0=0.0,
shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None,
verbose=False, max_iter=-1, decision_function_shape=None, random_state=None)))
#ensemble svm dari penggabungan svm dan bagging
models.append(('Bagging' , BaggingClassifier(base_estimator = Enslr, n_estimators=10,
max_samples=0.8, max_features=0.8, bootstrap=True, bootstrap_features=False,
oob_score=False, warm_start=False, n_jobs=1, random_state=seed, verbose=0)))

# evaluate each model in turn
results = []
names = []
for name, model in models:
        kfold = cross_validation.KFold(n=num_instances, n_folds=num_folds,
random_state=seed)
        cv_results = cross_validation.cross_val_score(model, X_train, Y_train, cv=kfold,
scoring=scoring)
        results.append(cv_results)
        names.append(name)
        msg = "%s: %f (%f)" % (name, cv_results.mean(), cv_results.std())
        print(msg)

# Compare Algorithms
fig = plt.figure()
fig.suptitle('Algorithm Comparison')
ax = fig.add_subplot(111)
plt.boxplot(results)
ax.set_xticklabels(names)
plt.show()
# Make predictions on validation dataset
```

```
#svm=SVC()
#svm.fit(X_train, Y_train)
#predictions=svm.predict(X_validation)

#untuk menghasilkan model sama accuracy
Bagging = BaggingClassifier()
Bagging.fit(X_train, Y_train)
predictions = Bagging.predict(X_validation)

print(accuracy_score(Y_validation, predictions))
print(confusion_matrix(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```