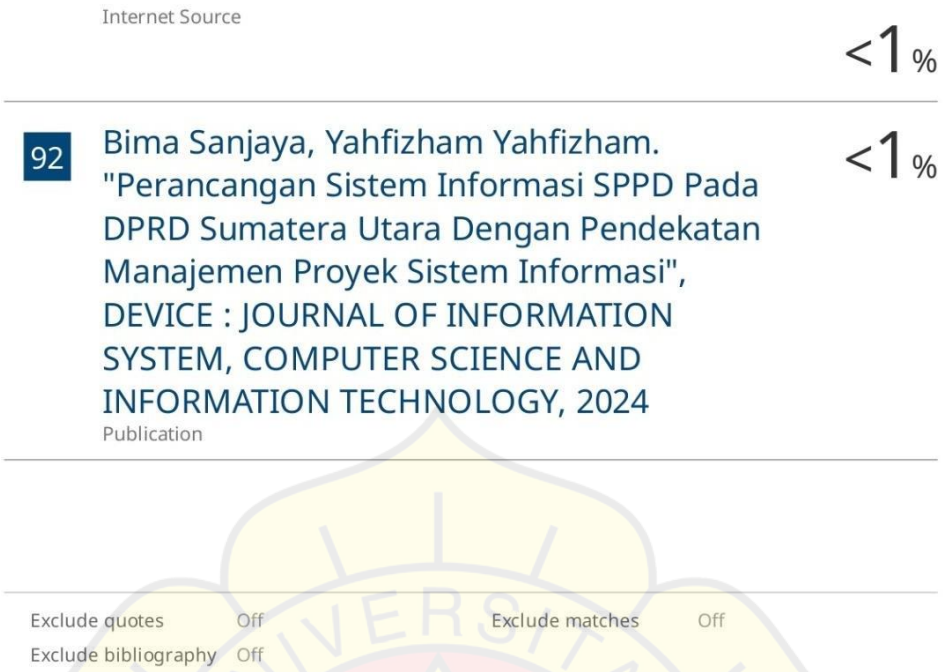

Lampiran 3 Source Code

```
## instalasi library / package yang digunakan.
import numpy as np
import pandas as pd
from pandas import DataFrame
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,
classification_report
from sklearn.model_selection import train_test_split,
cross_val_score, RandomizedSearchCV, GridSearchCV, KFold
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
# from sklearn.linear_model import LogisticRegression
# from sklearn.tree import DecisionTreeClassifier
# from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
# from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis
# from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, confusion_matrix,
classification_report

dataset = pd.read_csv("/content/balanced_training_set.csv")
```

```
from sklearn.model_selection import train_test_split,
GridSearchCV
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score
# Define features and labels
X = dataset.drop('LABEL', axis=1)
y = dataset['LABEL']

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
# Standardization
scaler = StandardScaler()
# Define the models and parameter grids for GridSearchCV
rf = RandomForestClassifier(random_state=42)
rf_param_grid = {
    'classifier__n_estimators': [100, 200, 300],
    'classifier__max_depth': [None, 10, 20, 30],
    'classifier_min_samples_split': [2, 5, 10]}
svm = SVC(random_state=42)
svm_param_grid = {
    'classifier__C': [0.1, 1, 10, 100],
    'classifier_gamma': [1, 0.1, 0.01, 0.001],
    'classifier__kernel': ['rbf']}
# Create pipelines
rf_pipeline = Pipeline([
    ('scaler', scaler),
    ('classifier', rf)])
svm_pipeline = Pipeline([
    ('scaler', scaler),
    ('classifier', svm)])
# Perform GridSearchCV
rf_grid_search = GridSearchCV(rf_pipeline, rf_param_grid,
cv=5, n_jobs=-1, verbose=1)
svm_grid_search = GridSearchCV(svm_pipeline, svm_param_grid,
cv=5, n_jobs=-1, verbose=1)
# Fit the models
rf_grid_search.fit(X_train, y_train)
svm_grid_search.fit(X_train, y_train)
# Get the best models
best_rf = rf_grid_search.best_estimator_
best_svm = svm_grid_search.best_estimator_
# Predict and evaluate
rf_predictions = best_rf.predict(X_test)
svm_predictions  =  best_svm.predict(X_test)
rf_accuracy = accuracy_score(y_test, rf_predictions)
svm_accuracy = accuracy_score(y_test, svm_predictions)
rf_accuracy, svm_accuracy
```

```
import pandas as pd
from sklearn.preprocessing import RobustScaler
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
import numpy as np

data = pd.read_csv('/content/balanced_training_set.csv')
# Menghapus outlier menggunakan IQR
def remove_outliers_iqr(df):
    Q1 = df.quantile(0.25)
    Q3 = df.quantile(0.75)
    IQR = Q3 - Q1
    return df[~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 *
IQR))).any(axis=1)]
data_cleaned = remove_outliers_iqr(data)
scaler = RobustScaler()
scaled_features =
scaler.fit_transform(data_cleaned.drop(columns=['LABEL']))
data_scaled = pd.DataFrame(scaled_features,
columns=data_cleaned.columns[:-1])
data_scaled['LABEL'] = data_cleaned['LABEL'].values
X = data_scaled.drop(columns=['LABEL'])
y = data_scaled['LABEL']
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
import numpy as np
X_train_balanced = np.random.rand(100, 5) # Contoh data
X_train
y_train_balanced = np.random.randint(0, 2, size=100) # Contoh
data y_train
X_test = np.random.rand(50, 5) # Contoh data X_test
y_test = np.random.randint(0, 2, size=50) # Contoh data
y_test

# Hyperparameter tuning untuk Random Forest
param_grid_rf = {
    'n_estimators': [10, 50, 100, 200, 300],
    'max_depth': [10, 20, 30, 40, 50, None],
    'min_samples_split': [2, 5, 10, 15],
    'min_samples_leaf': [1, 2, 4, 6]
}

grid_search_rf =
GridSearchCV(estimator=RandomForestClassifier(random_state=42)
, param_grid=param_grid_rf, cv=10, n_jobs=-1, verbose=2)
grid_search_rf.fit(X_train_balanced, y_train_balanced)
best_rf = grid_search_rf.best_estimator_
```

```
# Evaluasi model RF terbaik
y_pred_rf = best_rf.predict(X_test)
# Menggunakan fungsi untuk mendapatkan akurasi dan laporan
yang diinginkan
accuracy_rf, creport_rf = generate_report('RF')

# Output hasil RF
print(f"Akurasi RF: {accuracy_rf:.2%}")
print("Classification Report untuk RF:")
print(creport_rf)

# Hyperparameter tuning untuk SVM
param_grid_svm = {
    'C': [0.1, 1, 10, 100],
    'gamma': [1, 0.1, 0.01, 0.001],
    'kernel': ['rbf']
}

grid_search_svm = GridSearchCV(SVC(random_state=42),
param_grid=param_grid_svm, cv=10, n_jobs=-1, verbose=2)
grid_search_svm.fit(X_train_balanced, y_train_balanced)
best_svm = grid_search_svm.best_estimator_

# Evaluasi model SVM terbaik
y_pred_svm = best_svm.predict(X_test)
# Menggunakan fungsi untuk mendapatkan akurasi dan laporan
yang diinginkan
accuracy_svm, creport_svm = generate_report('SVM')

# Output hasil SVM
print(f"Akurasi SVM: {accuracy_svm:.2%}")
print("Classification Report untuk SVM:")
print(creport_svm)
import pickle
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import KFold, cross_val_score

# Menyiapkan model
models = []
models.append(('SVM', SVC(random_state=seed)))
models.append(('RF',
RandomForestClassifier(n_estimators=num_trees,
random_state=seed)))

# Evaluasi setiap model
results = []
names = []
trained_models = {} # Dictionary untuk menyimpan model yang
telah dilatih
```

```
import pickle
from sklearn.metrics import accuracy_score,
classification_report

# Muat model Random Forest dari file pkl
with open('random_forest_model.pkl', 'rb') as rf_file:
    loaded_rf_model = pickle.load(rf_file)
# Muat model SVM dari file pkl
with open('svm_model.pkl', 'rb') as svm_file:
    loaded_svm_model = pickle.load(svm_file)
# Lakukan prediksi menggunakan model yang telah dimuat
rf_predictions = loaded_rf_model.predict(x_test)
svm_predictions = loaded_svm_model.predict(x_test)
# Evaluasi performa model Random Forest
rf_accuracy = accuracy_score(y_test, rf_predictions)
print(f"Random Forest Accuracy: {rf_accuracy}")
print("Random Forest Classification Report:")
print(classification_report(y_test, rf_predictions))
# Evaluasi performa model SVM
svm_accuracy = accuracy_score(y_test, svm_predictions)
print(f"SVM Accuracy: {svm_accuracy}")
print("SVM Classification Report:")
print(classification_report(y_test, svm_predictions))
import pickle
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score,
classification_report
# Contoh: Melatih model KNN
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
# Load dataset
iris = load_iris()
X_train, X_test, y_train, y_test = train_test_split(iris.data,
iris.target, test_size=0.2, random_state=42)
# Initialize KNN classifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(X_train, y_train)
# Simpan model ke dalam file pickle
with open('knn_model.pkl', 'wb') as knn_file:
    pickle.dump(knn, knn_file)
  # Memuat model KNN dari file pickle
with open('knn_model.pkl', 'rb') as knn_file:
    loaded_knn_model = pickle.load(knn_file)
  # Contoh: Prediksi dengan model yang dimuat
y_pred = loaded_knn_model.predict(X_test)
# Evaluasi model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy:.2f}')
# Generate classification report
print('Classification Report:')
print(classification_report(y_test, y_pred))
```