PRAKTIKUM

MMAI1004 – PEMBELAJARAN MESIN

PERTEMUAN IV

DOSEN PENGASUH:

Prof. Dr. Taufik Fuadi Abidin, S.Si., M.Tech

Dr. Rumaisa Kruba, S.Si

Prof. Dr. Ir. Hammam Riza, M.Sc. IPU

Praktisi Komdigi (Dr. Said Mirza Pahlevi)

ASISTEN PRAKTIKUM:

Muhammad Chaidir, S.Kom, M.Kom

Hands on Random Forest and Ensemble Learning with Multiple Classifiers

1. Introduction

In this tutorial, we will explore Random Forest Classification using the Bank Marketing Dataset from UCI. Random Forest is an ensemble learning method that combines multiple decision trees to improve classification performance.

The objectives of this session include:

- ✓ Understanding Random Forest for classification tasks.
- ✓ Preprocessing categorical and numerical data.
- ✓ Evaluating model performance using accuracy, confusion matrix, and classification report.
- ✓ Analyzing Feature Importance.
- ✓ Exploring Ensemble Learning using multiple classifiers.

2. Dataset: Bank Marketing Dataset

The dataset can be downloaded from dataset page:

https://archive.ics.uci.edu/dataset/222/bank+marketing

Features:

- ✓ Categorical: Job, marital status, education, default, housing, loan, contact, month, day_of_week, etc.
- ✓ Numerical: Age, duration, campaign, pdays, previous, etc.
- ✓ Target variable (y): Whether a client subscribed (1: Yes, 0: No)

3. Steps to Implement Decision Tree Classifier

Step 1: Load Dataset

```
import pandas as pd

# Load dataset from UCI repository
path = "../../datasets/bank-additional-full.csv"
data = pd.read_csv(path, sep=';')
```

Step 2: Data Preprocessing

Handling Missing Values

```
# Remove rows with missing values
data.dropna(inplace=True)
```

Encoding Categorical Features

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
categorical_cols = data.select_dtypes(include=['object']).columns

for col in categorical_cols:
    data[col] = label_encoder.fit_transform(data[col])
```

Splitting Features and Target Variable

```
X = data.drop(columns=['y'])  # Features
y = data['y']  # Target Variable
```

Normalizing Numerical Features

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Step 3: Train-Test Split

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3,
random_state=42)
```

Step 4: Train Decision Tree Classifier

```
from sklearn.ensemble import RandomForestClassifier

clf = RandomForestClassifier(n_estimators=100, max_depth=4, random_state=42)
clf.fit(X_train, y_train)
```

Step 5: Model Evaluation

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

y_pred = clf.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", conf_matrix)
print("Classification Report:\n", class_report)
```

Step 6: Feature Importance Analysis

```
import seaborn as sns

feature_importance = pd.DataFrame({'Feature': X.columns, 'Importance':
clf.feature_importances_})
feature_importance = feature_importance.sort_values(by='Importance',
ascending=False)
```

```
plt.figure(figsize=(10, 5))
sns.barplot(x='Importance', y='Feature', data=feature_importance)
plt.title('Feature Importance in Decision Tree')
plt.show()
```

4. Ensemble Learning with Multiple Classifiers

Ensemble learning combines multiple machine learning models to improve overall performance.

```
from sklearn.ensemble import VotingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression

# Define individual classifiers
clf1 = LogisticRegression()
clf2 = DecisionTreeClassifier(max_depth=4, random_state=42)
clf3 = RandomForestClassifier(n_estimators=100, max_depth=4,
random_state=42)

# Combine classifiers in a Voting Ensemble
ensemble = VotingClassifier(estimators=[
    ('lr', clf1),
    ('dt', clf2),
    ('rf', clf3)
], voting='hard')

ensemble.fit(X_train, y_train)
y_pred_ensemble = ensemble.predict(X_test)

# Evaluate ensemble model
accuracy_ensemble = accuracy_score(y_test, y_pred_ensemble)
print("Ensemble Accuracy:", accuracy_ensemble)
```

5. Conclusion

- ✓ We trained a Random Forest Classifier on the Bank Marketing Dataset.
- ✓ The model's performance was evaluated using Accuracy, Confusion Matrix, and Classification Report.

- ✓ We analyzed Feature Importance to understand key factors.
- ✓ We implemented Ensemble Learning using multiple classifiers.

6. Further Exploration

- ✓ Modify hyperparameters like max_depth, n_estimators, and compare results.
- ✓ Try different ensemble methods like Bagging, Boosting (e.g., AdaBoost, Gradient Boosting).
- ✓ Apply this approach to other classification datasets.