Mini Project

DIGITS CLASSIFICATION USING RRANDOM FOREST AND Decision Tree

By: Dian Oktarisa

26 Maret 2025

Digital Skill Fair Data Science 37.0

Hello I'm Dian Oktarisa

II am a Physics graduate from Andalas University with a focus on instrumentation. My interest in data processing, sensor systems, and experimental analysis has driven me to delve into Data Science.

https://github.com/Dianokta23

www.linkedin.com/in/dian-oktarisa

dianoktarisa23@gmail.com

MACHINE LEARNING CLASSIFICATION ALGORITMA

Random Forest

LRandom Forest is a machine learning method based on ensemble learning, which combines multiple decision trees to improve prediction accuracy and reduce overfitting. This method works by creating many Decision Trees randomly and aggregating their predictions.

DecisionTree

A Decision Tree is a machine learning method used to create a predictive model in the form of a branching tree structure. This model works by splitting data based on certain features in a process that mimics human decision-making.

Import Library

Library berisi kumpulan fungsi dan prosedur yang telah dibuat sebelumnya, sehingga programmer tidak perlu menulis ulang kode dari nol dan memastikan pemerosesan data berrjalan dengan baik

Import Library

```
[ ] import numpy as np
    import pandas as pd
    from sklearn import datasets
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn import datasets
    from sklearn.model_selection import train_test_split
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
    from sklearn.preprocessing import StandardScaler
```

Persiapan Data

Read Dataset

```
digits = datasets.load_digits()

X = digits.data     # inputan untuk machine learning
y = digits.target   # output yang dinginkan dari machine learning

# Mengonversi data fitur dan target menjadi DataFrame
df_X = pd.DataFrame(X, columns=digits.feature_names)
df_y = pd.Series(y, name='target')
```

Data Digits

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_0	pixel_1_1	...	pixel_6_6	pixel_6_7	pixel_7_0	pixel_7_1
0	0.0	0.0	5.0	13.0	9.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	12.0	13.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	4.0	15.0	12.0	0.0	0.0	0.0	0.0	...	5.0	0.0	0.0	0.0
3	0.0	0.0	7.0	15.0	13.0	1.0	0.0	0.0	0.0	8.0	...	9.0	0.0	0.0	0.0
4	0.0	0.0	0.0	1.0	11.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1792	0.0	0.0	4.0	10.0	13.0	6.0	0.0	0.0	0.0	1.0	...	4.0	0.0	0.0	0.0
1793	0.0	0.0	6.0	16.0	13.0	11.0	1.0	0.0	0.0	0.0	...	1.0	0.0	0.0	0.0
1794	0.0	0.0	1.0	11.0	15.0	1.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1795	0.0	0.0	2.0	10.0	7.0	0.0	0.0	0.0	0.0	0.0	...	2.0	0.0	0.0	0.0
1796	0.0	0.0	10.0	14.0	8.0	1.0	0.0	0.0	0.0	2.0	...	8.0	0.0	0.0	1.0

1797 rows × 64 columns

Nilai Deskripsi Data

nands + Code + Text Connect

```
[ ] df.describe()
```

	pixel_0_0	pixel_0_1	pixel_0_2	pixel_0_3	pixel_0_4	pixel_0_5	pixel_0_6	pixel_0_7	pixel_1_0	pixel_1_1	...	pixel_6_7	pixel_7_0	
count	1797.0	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	1797.000000	...	1797.000000	1797.000000	1
mean	0.0	0.303840	5.204786	11.835838	11.848080	5.781859	1.362270	0.129661	0.005565	1.993879	...	0.206455	0.000556	
std	0.0	0.907192	4.754826	4.248842	4.287388	5.666418	3.325775	1.037383	0.094222	3.196160	...	0.984401	0.023590	
min	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	
25%	0.0	0.000000	1.000000	10.000000	10.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	
50%	0.0	0.000000	4.000000	13.000000	13.000000	4.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	
75%	0.0	0.000000	9.000000	15.000000	15.000000	11.000000	0.000000	0.000000	0.000000	3.000000	...	0.000000	0.000000	
max	0.0	8.000000	16.000000	16.000000	16.000000	16.000000	16.000000	15.000000	2.000000	16.000000	...	13.000000	1.000000	

8 rows × 65 columns

Split Data

untuk memecah data menjadi beberapa bagian berdasarkan pemisah tertentu.

Split Data

```
[ ] # Membagi data menjadi train dan test
    X_train, X_test, y_train, y_test = train_test_split(df_X, df_y, test_size=0.2, random_state=42)
```

Random Forest

Train the Model Random Forest

Train the Model Random Forest digunakan untuk melatih model Random Forest agar dapat membuat prediksi berdasarkan data yang diberikan.

```
# membuat dan melatih model Random Forest
model = RandomForestClassifier()
model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

Predict and Evaluate

Predict and Evaluate

```
# 4. Memprediksi dan mengevaluasi
y_pred = model.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print("Laporan Klasifikasi:")
print(f"Akurasi: {accuracy * 100:.2f}%")
```

```
Laporan Klasifikasi:
Akurasi: 98.06%
```

Predict and Evaluate digunakan untuk mengukur seberapa baik model melakukan klasifikasi berdasarkan data uji. model random forest berhasil memprediksi data uji dengan akurasi 98%.

Decision Tree

Train the Model Decision Tree

melatih model agar bisa mengenali pola dalam data dan membuat keputusan berdasarkan input baru.

Train the Model

```
[26] # membuat dan melatih model Decision Tree
     model = DecisionTreeClassifier()
     model.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

Predict and Evaluate

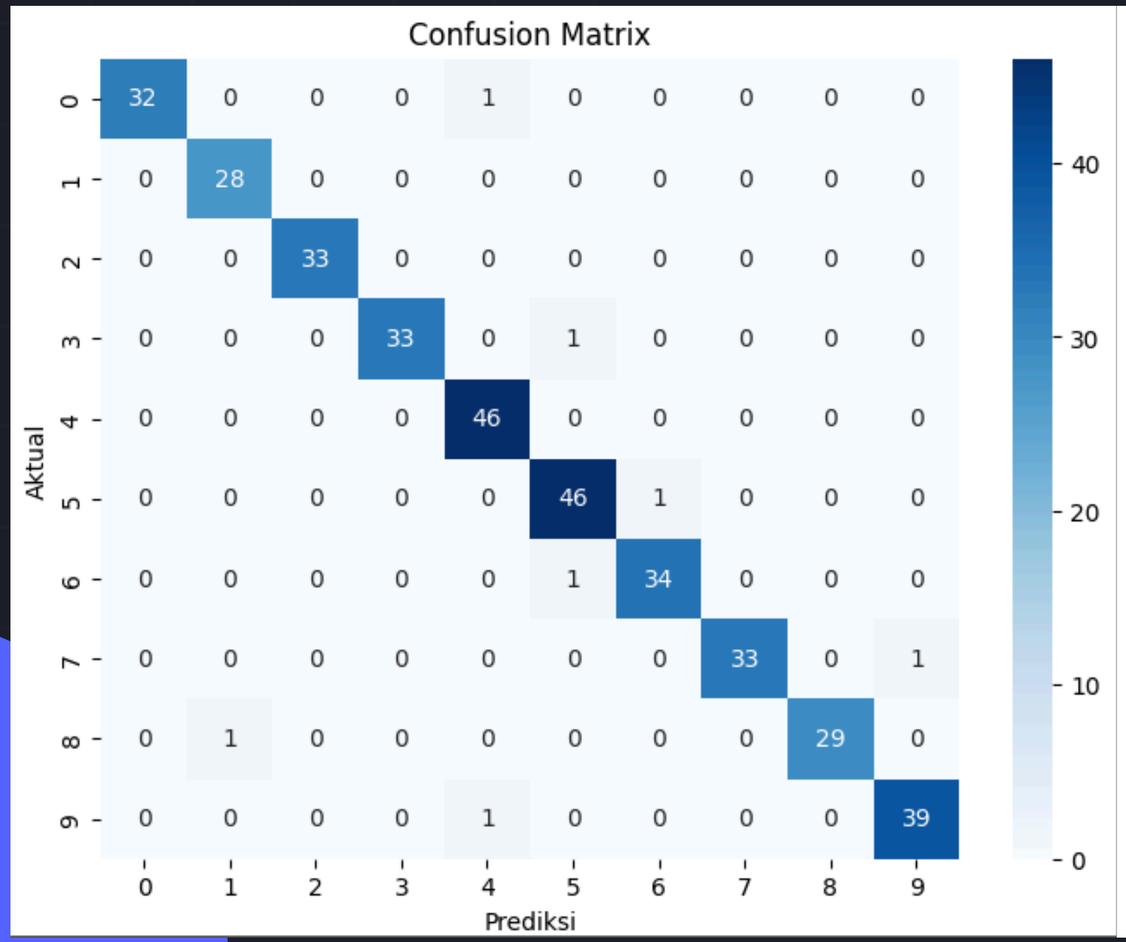
Predict and Evaluate

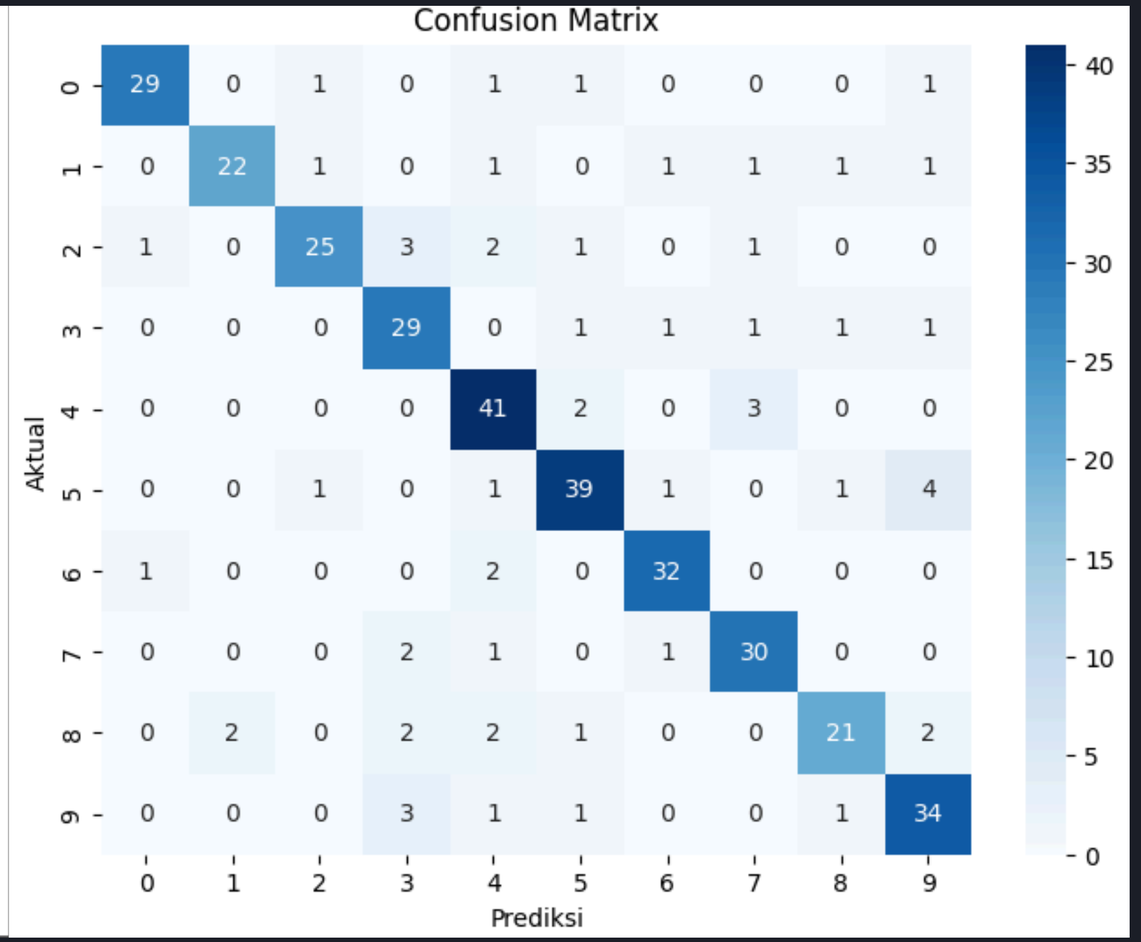
```
[27] # 4. Memprediksi dan mengevaluasi
     y_pred = model.predict(X_test)

     accuracy = accuracy_score(y_test, y_pred)

     print("Laporan Klasifikasi:")
     print(f"Akurasi: {accuracy * 100:.2f}%")
```

```
Laporan Klasifikasi:
Akurasi: 83.89%
```

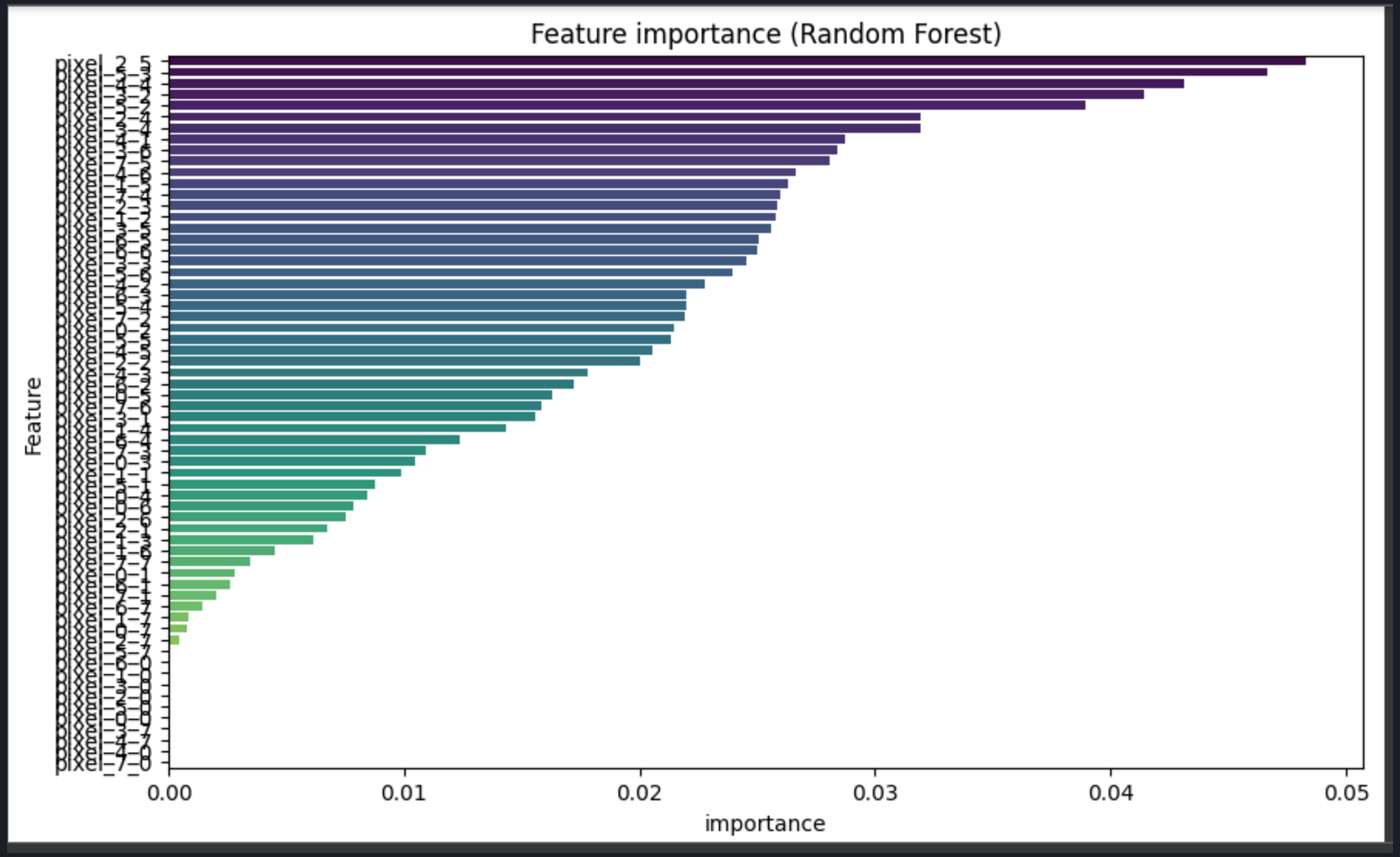
Predict and Evaluate digunakan untuk mengukur seberapa baik model melakukan klasifikasi berdasarkan data uji. model Decision Tree berhasil memprediksi data uji dengan akurasi 83.89%.

Perbandingan Confusiom Matrix Random Forest dan Desicius Tree

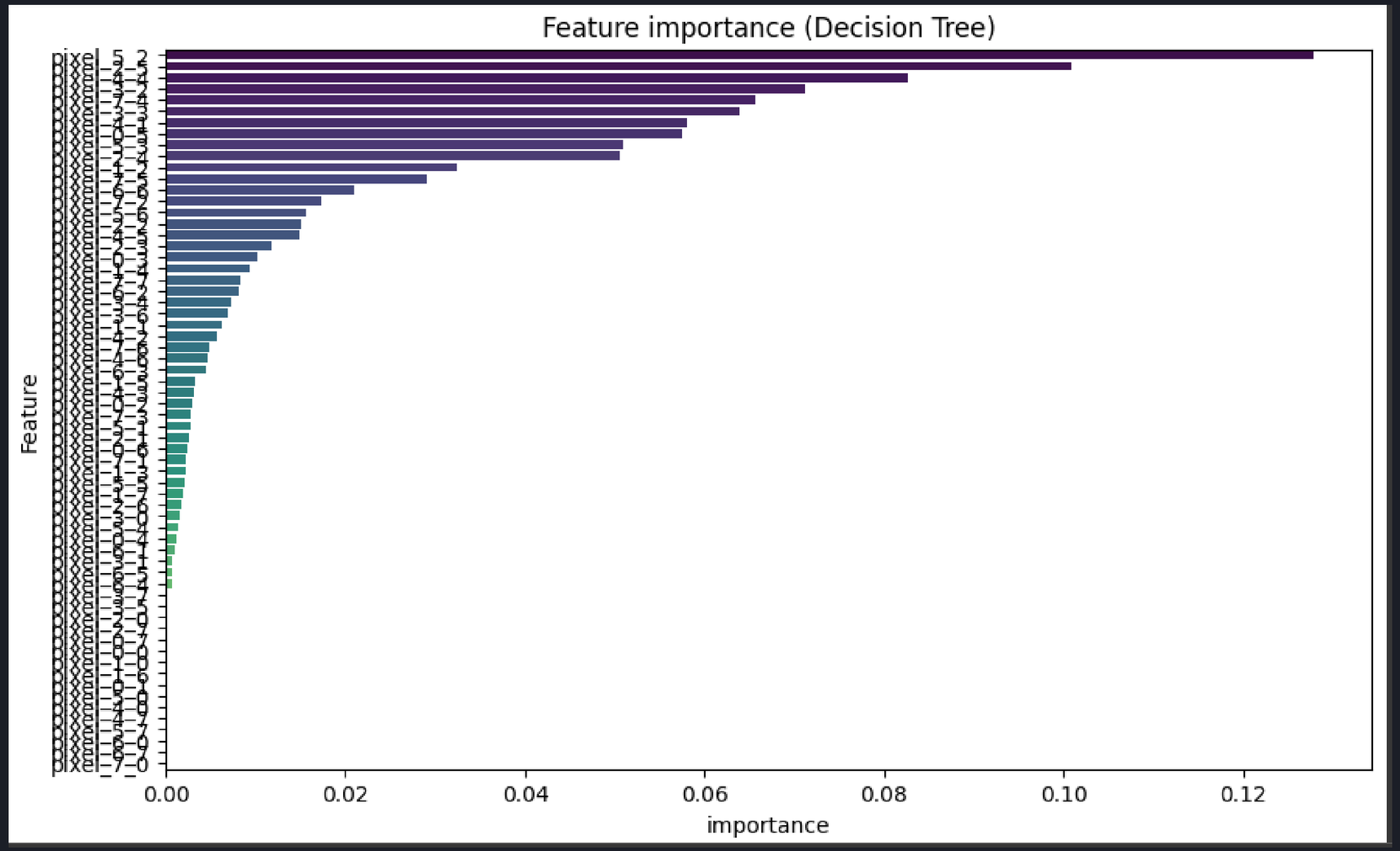




Feature Importance Random Forest



Feature Importance Decision Tree



Thank You