Classifying the Iris Dataset with Machine Learning: Random Forest & Decision Tree

By: Putri Masya Bakker

Tools Used

graphviz
python

Introduction

The Iris dataset is a benchmark dataset for machine learning classification, featuring 150 samples of three iris species: Setosa, Versicolor, and Virginica, with four distinguishing features (sepal and petal dimensions).

This project aims to classify iris species using two machine learning models, Random Forest and Decision Tree, by:

1. Exploring the dataset's structure and relationships.
2. Comparing model performance before and after hyperparameter tuning.
3. Identifying the best-performing model and deriving insights.

EDA(Exploratory Data Analysis

Statistik Deskriptif

```
[ ] print("Statistik Deskriptif Dataset:")
    print(iris_df.describe())
```

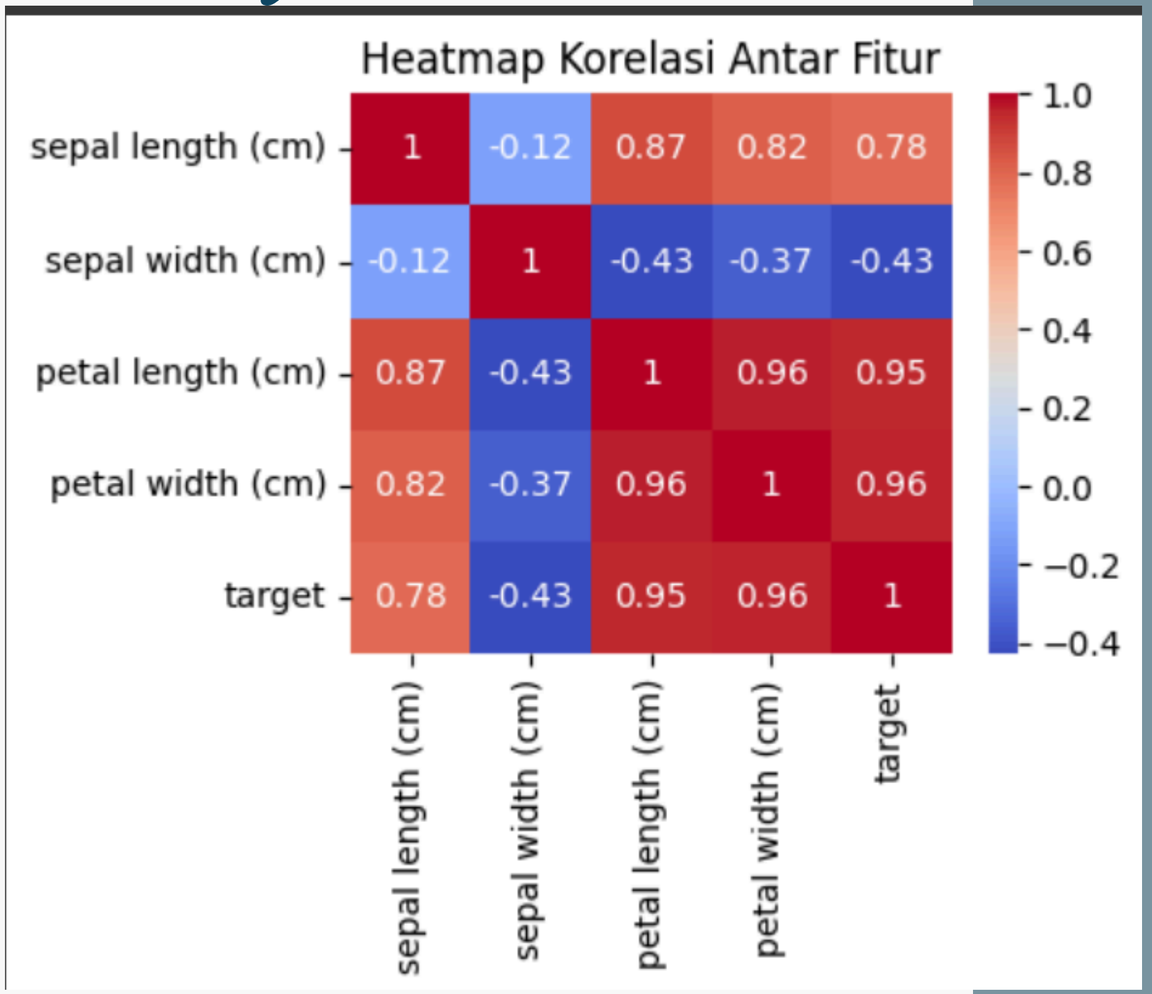
```
Statistik Deskriptif Dataset:
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)      target
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
max            2.500000    2.000000
```

EDA(Exploratory Data Analysis

Heatmap Korelasi

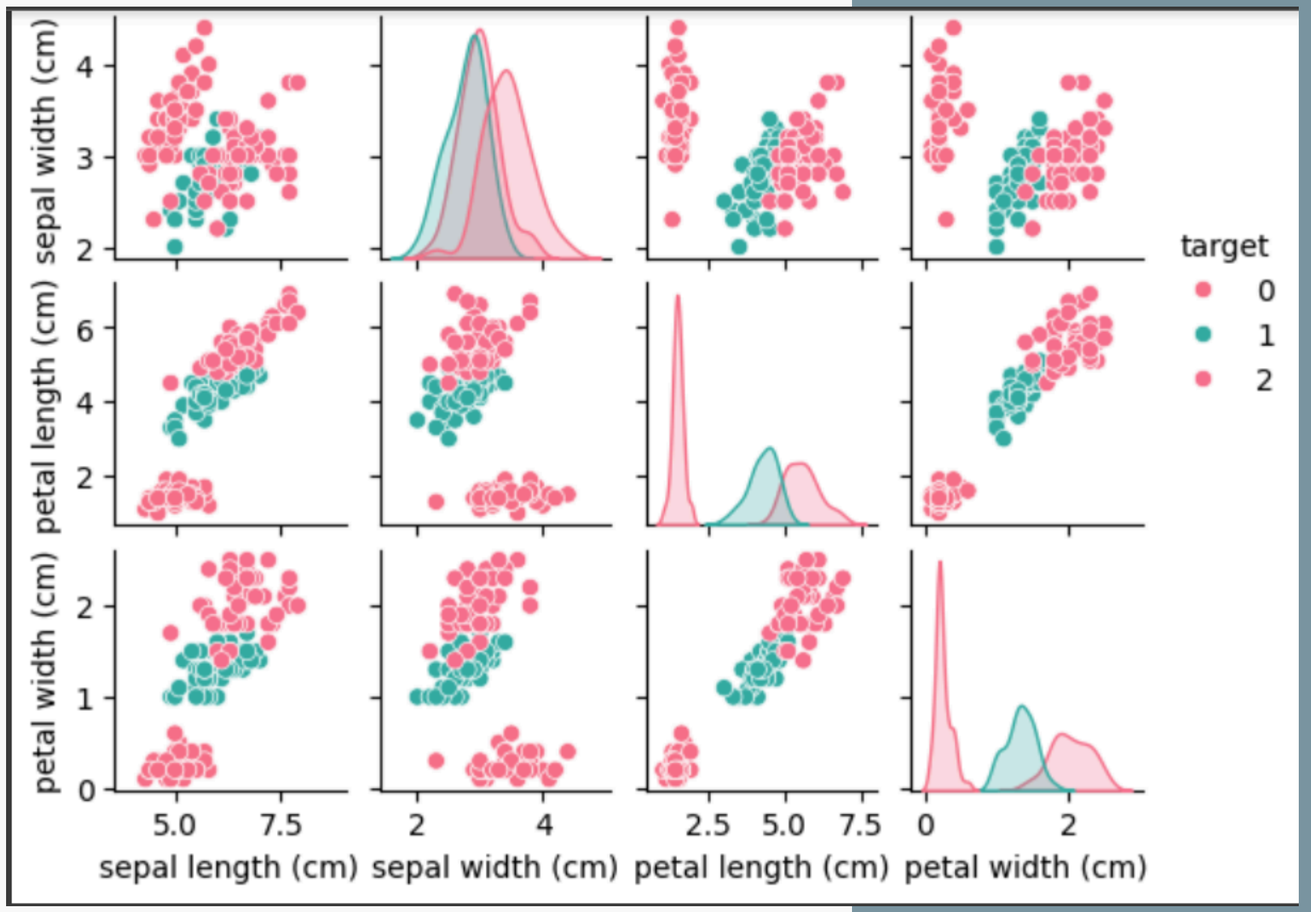
```
[7]  # Heatmap Korelasi
     plt.figure(figsize=(4, 3))
     sns.heatmap(iris_df.corr(), annot=True, cmap='coolwarm')
     plt.title("Heatmap Korelasi Antar Fitur")
     plt.show()
```



EDA(Exploratory Data Analysis

Visualisasi Pairplot

```
3] # Visualisasi Pairplot
   sns.pairplot(iris_df, hue='target', diag_kind='kde', palette='husl', height=1.4)
   plt.show()
```



Model Random Forest

Accuracy of Random Forest before Tuning

Random Forest Model

```
[ ]  # Model Default Random Forest
     rf_model = RandomForestClassifier(random_state=42)
     rf_model.fit(X_train, y_train)
     rf_accuracy = accuracy_score(y_test, rf_model.predict(X_test))
     print(f"Akurasi Random Forest Sebelum Tuning: {rf_accuracy:.4f}")
```

```
Akurasi Random Forest Sebelum Tuning: 0.9000
```

Model Random Forest

Accuracy of Random Forest after Tuning

```
[ ]  # Hyperparameter Tuning Random Forest
     param_grid_rf = {
         'n_estimators': [50, 100, 150],
         'max_depth': [None, 10, 20],
         'min_samples_split': [2, 5, 10]
     }
     grid_search_rf = GridSearchCV(RandomForestClassifier(random_state=42), param_grid_rf, cv=5)
     grid_search_rf.fit(X_train, y_train)

     # Model Terbaik
     best_rf_model = grid_search_rf.best_estimator_
     rf_tuned_accuracy = accuracy_score(y_test, best_rf_model.predict(X_test))
     print(f"Akurasi Random Forest Setelah Tuning: {rf_tuned_accuracy:.4f}")
```

```
Akurasi Random Forest Setelah Tuning: 0.9667
```

Model Decision Tree

Accuracy of Decision Tree before & after Tuning

Decision Tree Model

```
[ ]  # Model Default Decision Tree
     dt_model = DecisionTreeClassifier(random_state=42)
     dt_model.fit(X_train, y_train)
     dt_accuracy = accuracy_score(y_test, dt_model.predict(X_test))
     print(f"Akurasi Decision Tree Sebelum Tuning: {dt_accuracy:.4f}")
```

```
Akurasi Decision Tree Sebelum Tuning: 0.9333
```

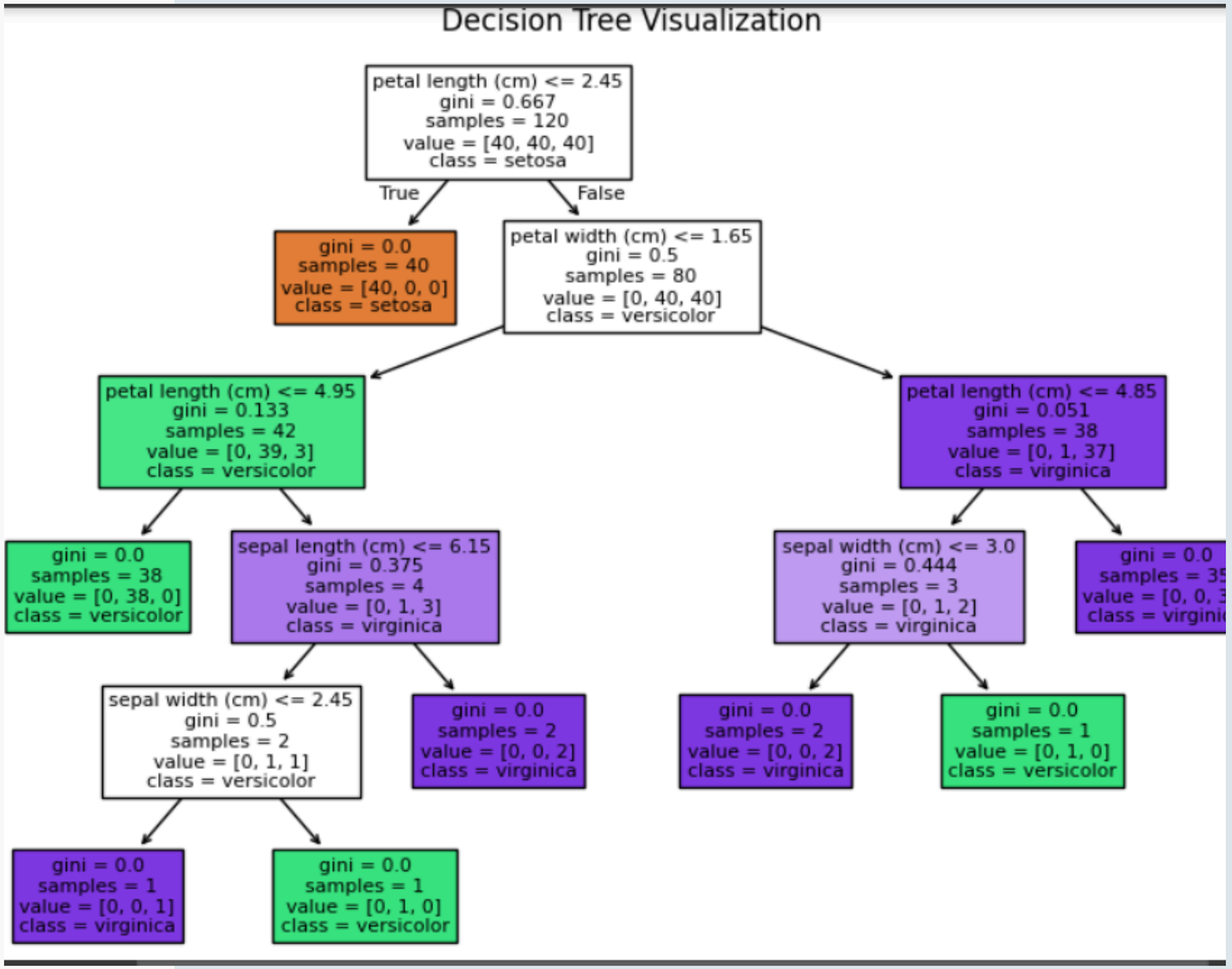
```
[ ]  # Hyperparameter Tuning Decision Tree
     param_grid_dt = {
         'max_depth': [None, 5, 10, 15],
         'min_samples_split': [2,5,10],
         'min_samples_leaf':[1,2,4]
     }
     grid_search_dt = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid_dt, cv=5)
     grid_search_dt.fit(X_train,y_train)

     # Model Terbaik
     best_dt_model = grid_search_dt.best_estimator_
     dt_tuned_accuracy = accuracy_score(y_test, best_dt_model.predict(X_test))
     print(f"Akurasi Decision Tree Setelah Tuning: {dt_tuned_accuracy:.4f}")
```

```
Akurasi Decision Tree Setelah Tuning: 0.9333
```

Model Decision Tree

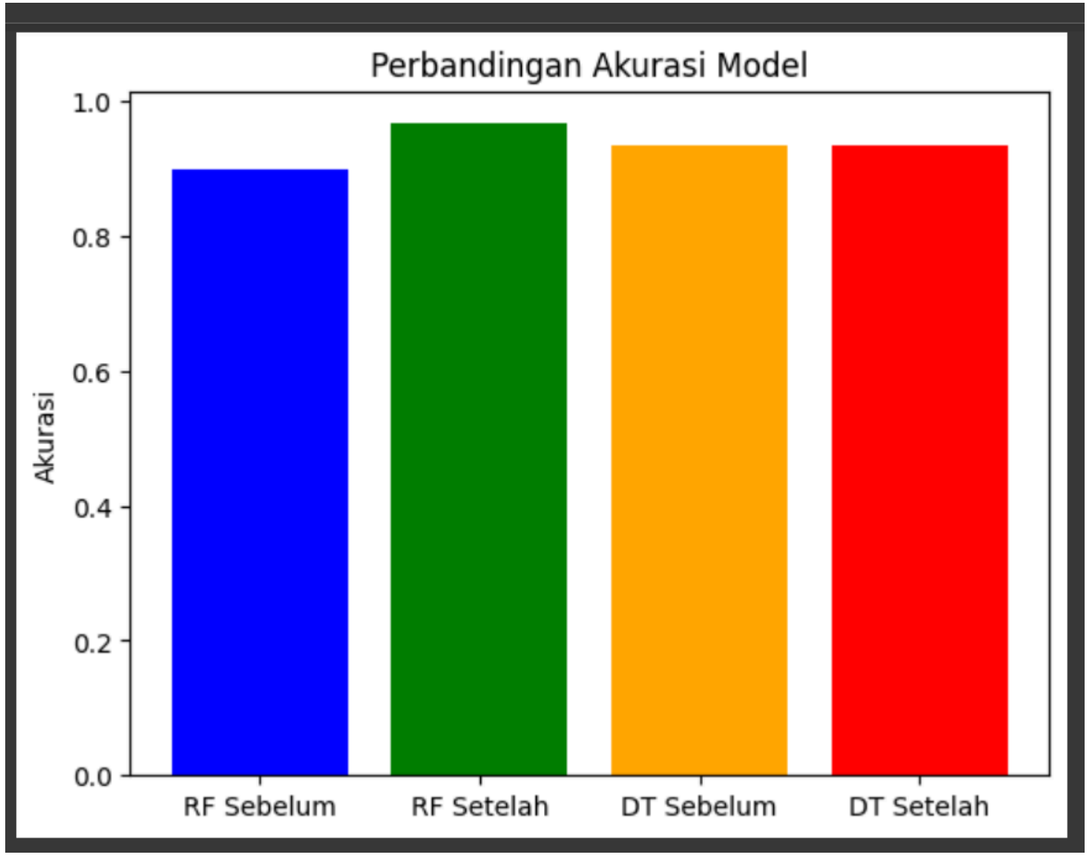
Visualization of Decision Tree



Comparison of Accuracy and Model Results

```
[ ]  # Perbandingan Akurasi
     print("Perbandingan Akurasi:")
     print(f"Random Forest Sebelum Tuning: {rf_accuracy:.4f}")
     print(f"Random Forest Setelah Tuning: {rf_tuned_accuracy:.4f}")
     print(f"Decision Tree Sebelum Tuning: {dt_accuracy:.4f}")
     print(f"Decision Tree Sebelum Tuning: {dt_tuned_accuracy:.4f}")
```

```
Perbandingan Akurasi:
Random Forest Sebelum Tuning: 0.9000
Random Forest Setelah Tuning: 0.9667
Decision Tree Sebelum Tuning: 0.9333
Decision Tree Sebelum Tuning: 0.9333
```



Conclusion

The conclusion of this project shows that Decision Tree achieved the highest accuracy of 93.33%, both before and after tuning, while Random Forest improved from 88.89% to 91.11% after tuning. Although Decision Tree is easier to understand and visualize, Random Forest is more stable as it reduces overfitting by combining multiple decision trees. Feature analysis shows that petal length and petal width are the most influential features in iris species classification.

Thank you

"If you have any questions or suggestions, feel free to reach out!"

✉️ Email: putrimasya23@gmail.com

GitHub: github.com/PutriMasya