by Azzah M

KLASIFIKASI KESEGARAN IKAN MENGGUNAKAN MACHINE LEARNING

Machine Learning Project
June 2025

Isi Konten

Project Overview

Kualitas ikan segar merupakan faktor penting dalam industri perikanan karena berkaitan langsung dengan kepuasan konsumen dan nilai jual produk. Untuk membantu proses identifikasi kualitas ikan secara lebih efisien, **proyek ini bertujuan membangun model klasifikasi untuk mengidentifikasi kondisi ikan (segar atau busuk) berdasarkan data sensor aroma, serta membandingkan performa beberapa algoritma machine learning.**

Dataset yang digunakan berisi 80 data ikan dengan 24 fitur numerik yang diperoleh dari dosen sebagai bagian dari pembelajaran klasifikasi praktis. **Model dikembangkan menggunakan algoritma SVM, Random Forest, KNN, dan Decision Tree** dengan evaluasi performa menggunakan confusion matrix dan akurasi untuk menentukan algoritma terbaik.

Proyek ini diharapkan menjadi langkah awal automasi klasifikasi kualitas ikan sehingga pelaku usaha perikanan dapat menjaga kualitas produk dengan lebih cepat dan akurat.

Tools dan Dataset

TOOLS

- Jupyter Lab
- Python
- Library pandas, numpy, matplotlib, seaborn, dan scikit-learn.

DATA SOURCE

- **Jumlah data:** 80 entry ikan dengan label segar dan busuk.
- **Fitur:** 24 fitur numerik dari 8 sensor aroma.
- **Asal dataset:** Dataset bersifat internal yang diperoleh dari dosen untuk keperluan pembelajaran klasifikasi kondisi ikan pada mata kuliah Machine Learning.

Preprocessing

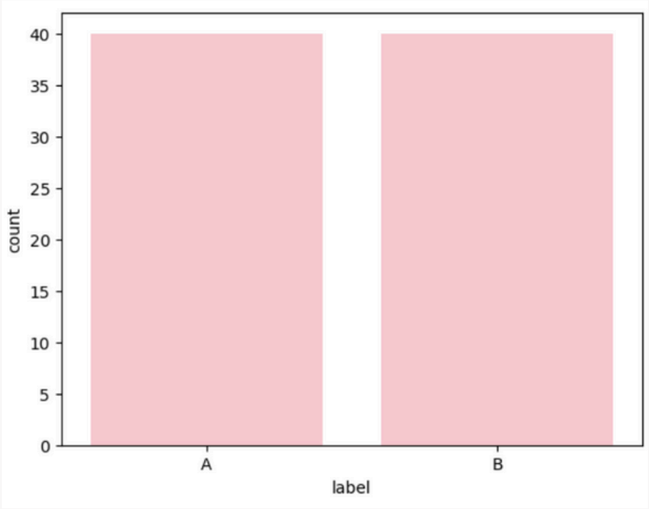
- Membuat dataframe dari file csv dengan `df = pd.read_csv('fitur_30s.csv')`
- Eksplorasi awal dengan `df.head()` untuk melihat isi dataset.
- EDA awal menggunakan `df.describe()` untuk melihat ringkasan statistik dataset.

  Tahap ini menghasilkan informasi:
  - Dataset berisi **80 data ikan dengan 24 fitur** aroma dari 8 sensor.
  - Setiap sensor memiliki 3 fitur:
    - **fn_mean**: Rata-rata respons sensor selama periode pengukuran yang memberikan informasi terkait kestabilan dan intensitas aroma selama waktu pengukuran.
    - **fn_max**: Nilai tertinggi respons sensor yang mendeteksi lonjakan aroma busuk selama periode pengukuran.
    - **fn_auc**: Luas area di bawah kurva respons sensor yang menggambarkan akumulasi intensitas aroma selama waktu pengukuran.

Preprocessing

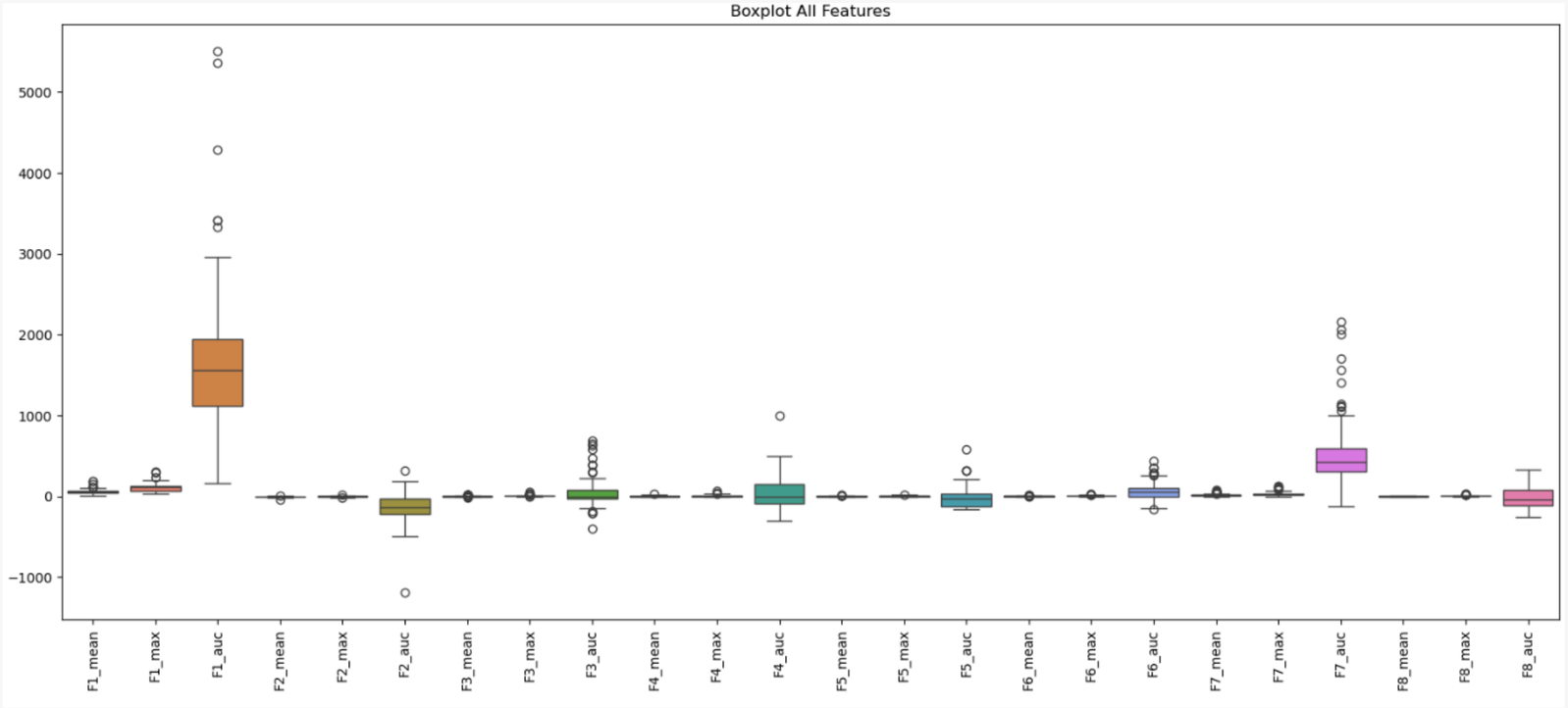
- Nilai antar fitur tidak seragam (rentang kecil hingga ribuan).
- Normalisasi diperlukan agar model tidak bias terhadap fitur berskala besar.
- Standar deviasi bervariasi yang menunjukkan persebaran data yang beragam.
- Terdapat indikasi outlier di beberapa fitur misalnya pada F7_auc yang memiliki mean 548 dan max 2152.
- Terdapat fitur dengan nilai negatif yang perlu diperhatikan.

- EDA lanjutan dengan:
  - Melihat distribusi kelas target. Dapat diketahui bahwa **distribusi kelas A dan B seimbang.**



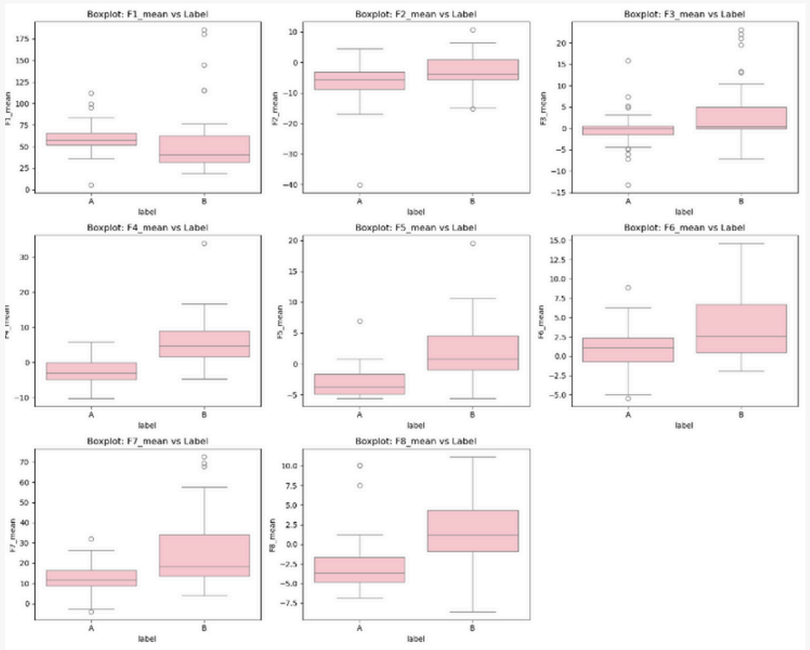
Preprocessing

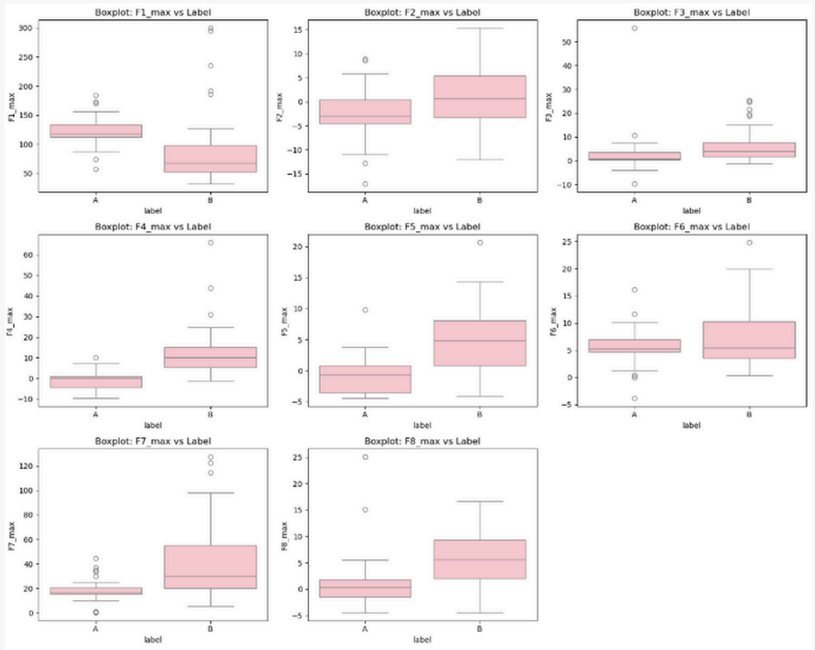
- Distribusi fitur

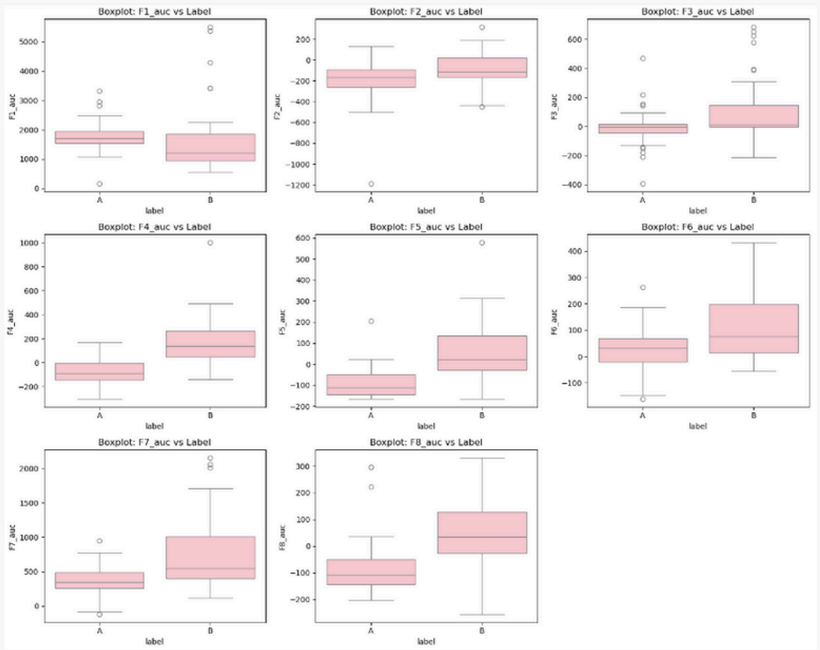


Preprocessing

- Distribusi fitur terhadap label



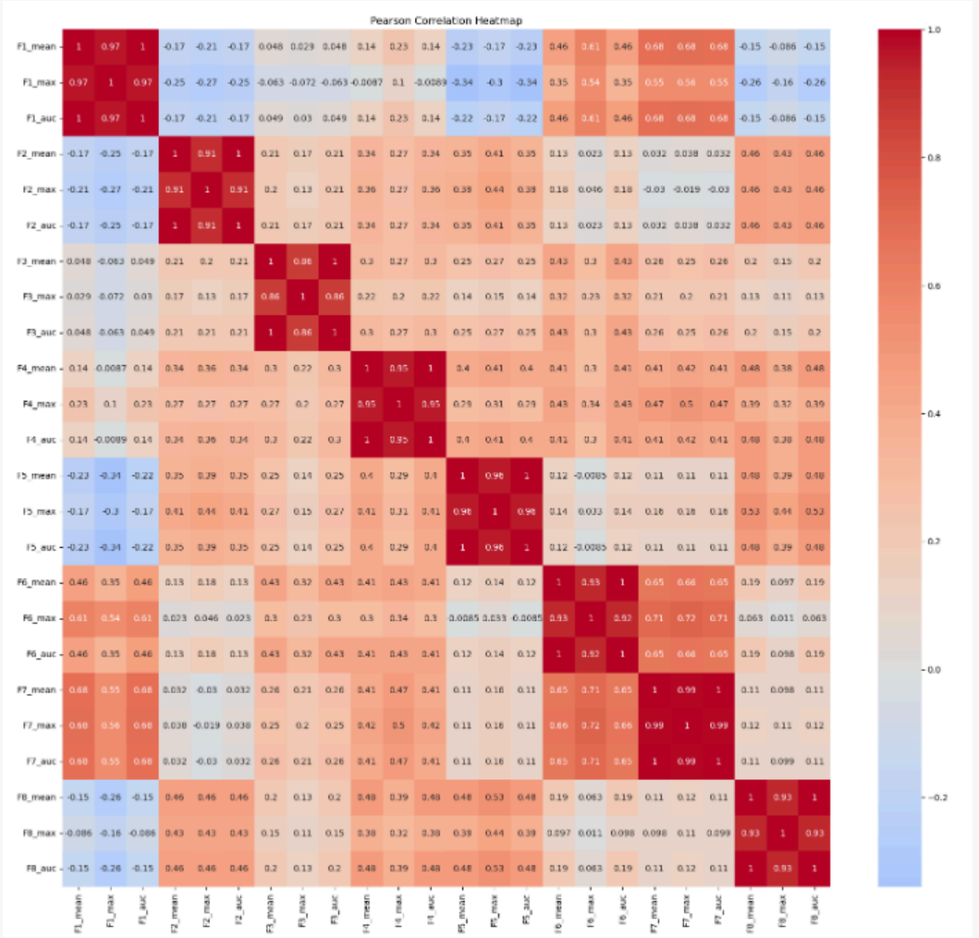




Beberapa fitur seperti *f2_mean, f6_max, dan f2_auc* **kurang informatif** untuk klasifikasi karena memiliki distribusi yang mirip antar kelas, median yang mirip, serta overlap yang tinggi, meskipun *f2_auc* memiliki outlier pada salah satu label yang bisa membantu klasifikasi pada nilai ekstrem. Fitur-fitur tersebut sebaiknya tidak digunakan atau dikombinasikan dengan fitur lain yang lebih informatif jika model memiliki mekanisme seleksi fitur internal seperti tree-based.

Preprocessing

- Korelasi fitur.
  Didapat bahwa korelasi antar fitur dalam masing-masing sensor memiliki nilai tinggi yang berarti data sensor stabil dan tidak noise.



Preprocessing

- Data cleaning, yaitu cek missing value dan duplicate value.

```
[13]:
df[df.duplicated()]
```

[13]:

	F1_mean	F1_max	F1_auc	F2_mean	F2_max	F2_auc

0 rows × 25 columns

dan *df.isnull().sum()*

- Normalisasi dilakukan di pipeline model untuk menghindari data leakege.
- Seleksi fitur juga dilakukan di pipeline karena beberapa model memiliki mekanisme internal secara otomatis untuk mengidentifikasi fitur yang paling relevan.

Modeling

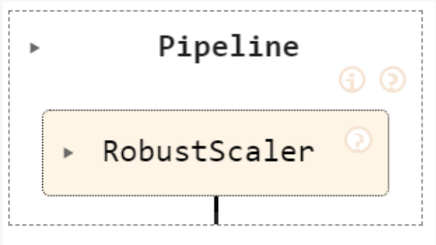
DATASET SPLITTING

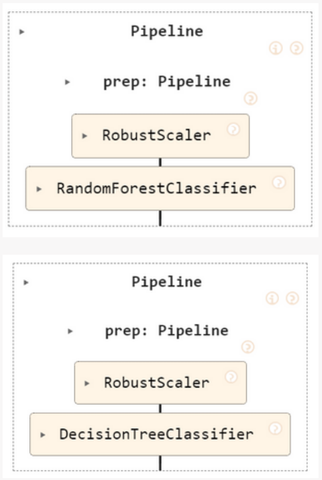
```
X = df.drop(columns='label')
y = df['label']
le = LabelEncoder()
y_encoded = le.fit_transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, shuffle=True, random_state=42)
X_train.shape, X_test.shape
```

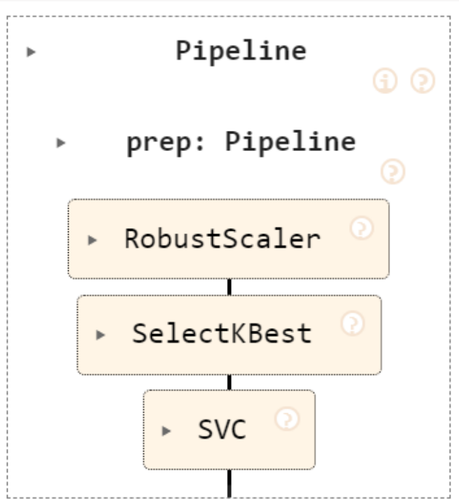
- Target/y diencoding dari string ke numerik dengan LabelEncoder() agar dapat diproses oleh model klasifikasi.
- Dataset splitting dengan rasio 80:20 yang menghasilkan data train (64,24) dan data test (16,24).

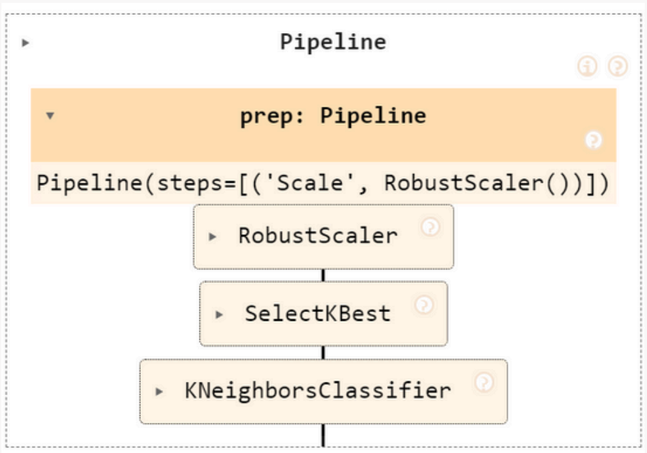
Modeling

PIPELINE MODELING









- Pemodelan dilakukan menggunakan pipeline agar proses lebih terstruktur dan menghindari data leakage.
- Pipeline terdiri dari pipeline preprocessing berisi scaling dan pipeline modeling.
- Seleksi fitur tambahan tidak diletakkan dalam pipeline preprocessing karena disesuaikan dengan kebutuhan algoritma yang digunakan.

Modeling

RANDOM FOREST MODEL

```
rf_model = Pipeline([
    ('prep', preprocessor),
    ('model', RandomForestClassifier(
    n_estimators=50,
    max_depth=4,
    min_samples_split=3,
    min_samples_leaf=2,
    random_state=42
    )),
])
```

- *max_depth = 4* digunakan untuk membatasi kedalaman tree.
- *min_samples_split = 3* dan *min_samples_leaf = 2* mencegah node tidak terlalu kecil.
- *n_estimator = 50* digunakan agar hasil prediksi stabil.

Modeling

DECISION TREE MODEL

```
dt_model = Pipeline([
    ('prep', preprocessor),
    ('model', DecisionTreeClassifier(
    criterion='entropy',
    max_depth=3,
    min_samples_split=5,
    min_samples_leaf=3,
    random_state=42)),
])
```

- *criterion='entropy'* dipilih dibanding *'gini'* untuk memanfaatkan pendekatan teori informasi dalam pemisahan node. Entropy mengukur ketidakpastian data sehingga split dipilih berdasarkan informasi yang diperoleh, yang diharapkan membantu pembagian data lebih informatif pada dataset kecil
- *max_depth=3, min_samples_split=5, dan min_samples_leaf=3* digunakan agar pohon tetap sederhana dan general.

Modeling

SVM DAN KNN MODEL

```
svm_model = Pipeline([
    ('prep', preprocessor),
    ('select', SelectKBest(score_func=f_classif, k=10)),
    ('model', SVC()),
])
```

```
knn_model = Pipeline([
    ("prep",preprocessor),
    ('select', SelectKBest(score_func=f_classif, k=10)),
    ('model', KNeighborsClassifier(n_neighbors=5)),
])
```

- *SelectKBest(score_func=f_classif, k=10)* digunakan karena SVM dan KNN tidak memiliki built-in feature selection. Seleksi fitur ini membantu meningkatkan performa model dengan memilih 10 fitur terbaik berdasarkan ANOVA F-value pada data klasifikasi.
- *n_neighbors = 5* menggunakan 5 tetangga terdekat untuk voting klasifikasi.

Modeling

TRAINING

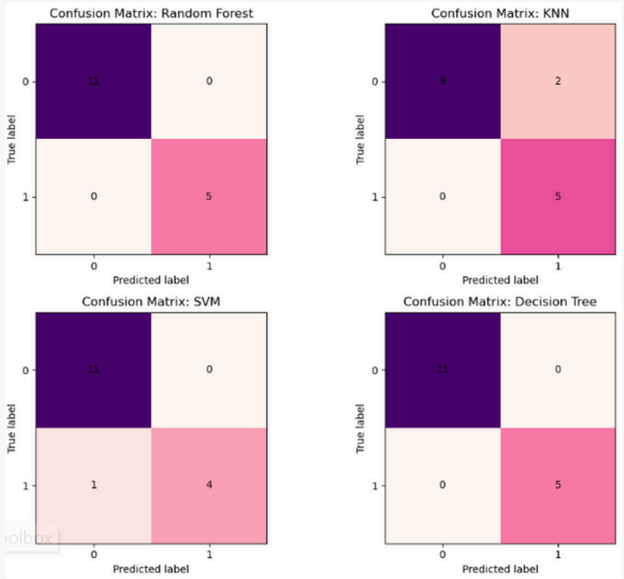
- Cross-validation dengan k=3 digunakan selama training untuk mengevaluasi stabilitas model pada data train sebelum dilakukan testing pada data test.

MODEL/ CROSS VALIDATION	RANDOM FOREST	DECISSION TREE	SVM	KNN
AKURASI	0.9062	0.8903	0.8593	0.8283
PRECISION	0.9091	0.8907	0.8592	0.8271
RECALL	0.9019	0.8886	0.8561	0.8271
F1 SCORE	0.9040	0.98890	0.8571	0.8265

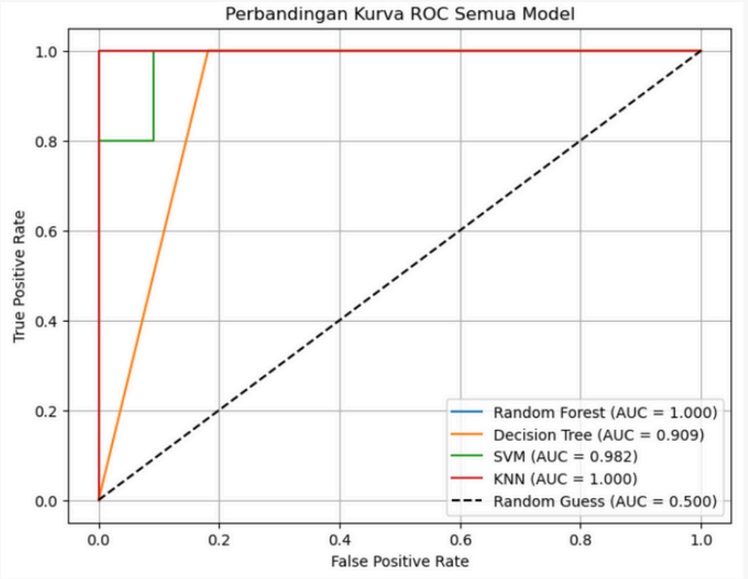
Hasil CV menunjukkan Random Forest memiliki performa stabil dan unggul dibanding model lain pada dataset ini.

Result

TESTING RESULTS



MODEL/ EVALUATION MATRIX	RANDOM FOREST	DECISSION TREE	SVM	KNN
AKURASI	1.0000	0.8570	0.9375	1.0000
PRECISION	1.0000	0.8571	0.9583	1.0000
RECALL	1.0000	0.9091	0.9000	1.0000
F1 SCORE	1.0000	0.8667	0.9227	1.0000



Hasil menunjukkan **Random Forest dan KNN bekerja sangat baik pada data uji dengan akurasi sempurna.**

Result

FEATURE IMPORTANCE RESULTS

```
======================================================
Hasil Analisis Feature Importance (Decision Tree)
======================================================
F4_max                      | Importance: 0.4468
F1_auc                      | Importance: 0.3430
F3_max                      | Importance: 0.1333
F4_auc                      | Importance: 0.0769
F2_mean                     | Importance: 0.0000
F2_max                      | Importance: 0.0000
F1_max                      | Importance: 0.0000
F1_mean                     | Importance: 0.0000
F3_mean                     | Importance: 0.0000
```

```
================================================
Hasil Seleksi Fitur SVM (SelectKBest + ANOVA F-test)
================================================
F4_mean                | Skor F-test: 28.1683
F4_auc                 | Skor F-test: 28.1638
F5_max                 | Skor F-test: 27.8823
F4_max                 | Skor F-test: 24.3613
F5_mean                | Skor F-test: 23.4852
F5_auc                 | Skor F-test: 23.4615
F7_max                 | Skor F-test: 15.3863
F7_auc                 | Skor F-test: 13.2930
F7_mean                | Skor F-test: 13.2836
F8_auc                 | Skor F-test: 11.5962
================================================
```

```
======================================================
Hasil Analisis Feature Importance (Random Forest)
======================================================
F4_max                      | Importance: 0.1183
F1_max                      | Importance: 0.0774
F1_mean                     | Importance: 0.0750
F1_auc                      | Importance: 0.0749
F5_mean                     | Importance: 0.0733
F7_max                      | Importance: 0.0728
F5_max                      | Importance: 0.0719
F8_auc                      | Importance: 0.0702
F4_auc                      | Importance: 0.0683
```

- Random Forest: *F4_max, F1_max, F1_mean* menjadi fitur paling berpengaruh.
- Decision Tree: F4_max, F1_auc dominan pada pemisahan data.
- SVM & KNN: Menggunakan SelectKBest, *F4_mean, F4_auc, F5_max* menjadi top 3 fitur terbaik.
- **Analisis feature importance menunjukkan bahwa sensor F4 dan F1 paling berpengaruh dalam klasifikasi**.

Conclusion

- **Random forest** menunjukkan performa terbaik pada dataset 40 sampel × 24 fitur dengan akurasi sempurna di data uji dan stabil pada cross-validation. Model ini menunjukkan kemampuannya menangkap pola dengan baik. Model ini juga memberikan feature importance untuk interpretasi, namun kompleksitasnya memerlukan validasi lebih lanjut agar tidak overfitting.
- **Decision tree** meraih akurasi 0.88 dengan recall tinggi pada data uji yang cukup baik untuk dataset ini. Model memanfaatkan fitur dominan seperti F4_max, namun keterbatasannya adalah hanya mengandalkan sedikit fitur sehingga performanya masih di bawah random forest.
- **SVM** memberikan akurasi tinggi dan stabil pada cross-validation pada dataset. Model ini cocok menangani data berdimensi sedang. Namun, SVM tidak menyediakan interpretasi fitur secara langsung dan memerlukan tuning parameter.
- **KNN** menghasilkan akurasi sempurna pada data uji tetapi rata-rata cross-validation lebih rendah (0.83) yang menunjukkan sensitivitas terhadap dataset kecil. Model ini sederhana, namun tidak menyediakan interpretasi fitur untuk analisis lebih lanjut.

TERIMA KASIH!

Machine Learning Project

by Azzah M