Wine Recognition Classification: A Comparative Study Using Logistic Regression, KNN, and SVM

BY ERIKA YOSEPIN SINAMBELA

Our Agenda

- About Me
- About The Project
- Tools
- Machine Learning
- Conclusion

About Me

I am a 6th-semester Chemistry student passionate about learning new things, particularly in Data Science. I aim to combine my scientific background with modern technologies, aspiring to become a researcher and contribute to groundbreaking discoveries that solve real-world challenges.

STUDIO SHODWE

About The Project

"Create a simple machine learning (classification) program using a datasets that has already been provided by scikit-learn"

Tools

pandas

Import Library & Loaf Dataset

```python
import pandas as pd

# Load dataset
wine = load_wine()
df = pd.DataFrame(wine.data, columns=wine.feature_names)
df['target'] = wine.target

# Simpan ke CSV
df.to_csv('wine_dataset.csv', index=False)
print("Dataset berhasil disimpan sebagai wine_dataset.csv")
```

```python
url = "https://github.com/ErikaYosepinS/Wine-Dataset/raw/refs/heads/main/wine_dataset.csv"

# Load dataset
data = pd.read_csv(url)

# Tampilkan 5 baris pertama
print(data.head())

# Cek informasi dataset
print("\nInfo Dataset:\n")
print(data.info())
```

Data Processing

	alcohol	malic_acid	ash	alcalinity_of_ash	magnesium	total_phenols	\
0	14.23	1.71	2.43	15.6	127.0	2.80	
1	13.20	1.78	2.14	11.2	100.0	2.65	
2	13.16	2.36	2.67	18.6	101.0	2.80	
3	14.37	1.95	2.50	16.8	113.0	3.85	
4	13.24	2.59	2.87	21.0	118.0	2.80	

	flavanoids	nonflavanoid_phenols	proanthocyanins	color_intensity	hue	\
0	3.06	0.28	2.29	5.64	1.04	
1	2.76	0.26	1.28	4.38	1.05	
2	3.24	0.30	2.81	5.68	1.03	
3	3.49	0.24	2.18	7.80	0.86	
4	2.69	0.39	1.82	4.32	1.04	

	od280/od315_of_diluted_wines	proline	target
0	3.92	1065.0	0
1	3.40	1050.0	0
2	3.17	1185.0	0
3	3.45	1480.0	0
4	2.93	735.0	0

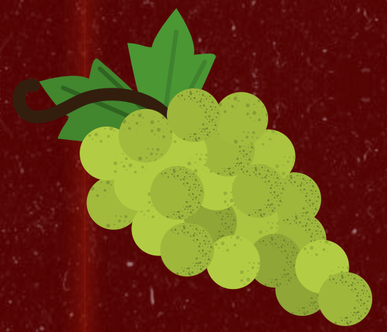
Info Dataset:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 178 entries, 0 to 177
Data columns (total 14 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   alcohol                       178 non-null    float64
 1   malic_acid                    178 non-null    float64
 2   ash                           178 non-null    float64
 3   alcalinity_of_ash             178 non-null    float64
 4   magnesium                     178 non-null    float64
 5   total_phenols                 178 non-null    float64
 6   flavanoids                    178 non-null    float64
 7   nonflavanoid_phenols          178 non-null    float64
 8   proanthocyanins               178 non-null    float64
 9   color_intensity               178 non-null    float64
 10  hue                           178 non-null    float64
 11  od280/od315_of_diluted_wines  178 non-null    float64
 12  proline                       178 non-null    float64
 13  target                        178 non-null    int64
dtypes: float64(13), int64(1)
memory usage: 19.6 KB
None
```

Data Preparation & Training Data

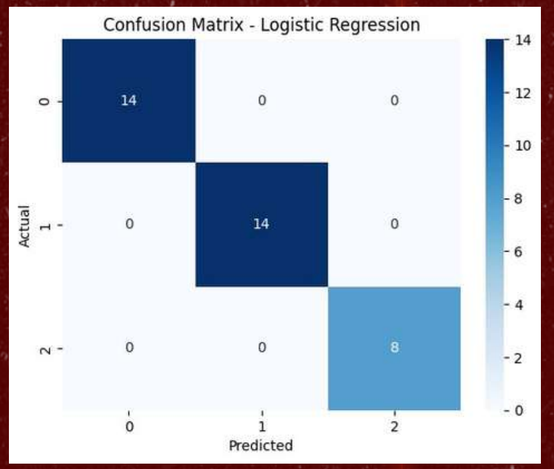
```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Pisahkan fitur (X) dan target (y)
X = data.drop(columns=['target'])  # Semua kecuali target
y = data['target']  # Kolom target

# Split dataset menjadi training (80%) dan testing (20%)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardisasi fitur
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

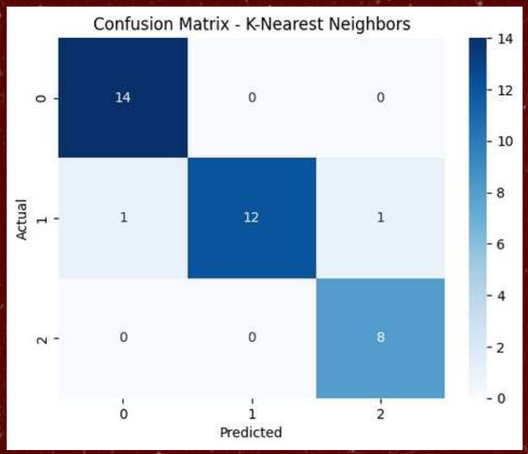


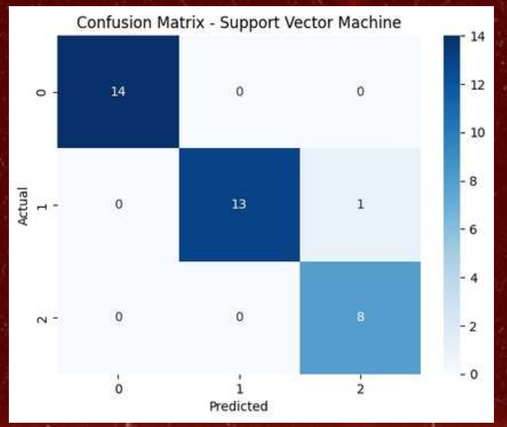
```python
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC

# Model Logistic Regression
log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)

# Model KNN (dengan k=5)
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Model SVM
svm = SVC(kernel='linear')  # Kernel linear agar hasilnya bisa dibandingkan dengan Logistic Regression
svm.fit(X_train, y_train)

print("Ketiga model berhasil dilatih!")
```

Evaluate Models

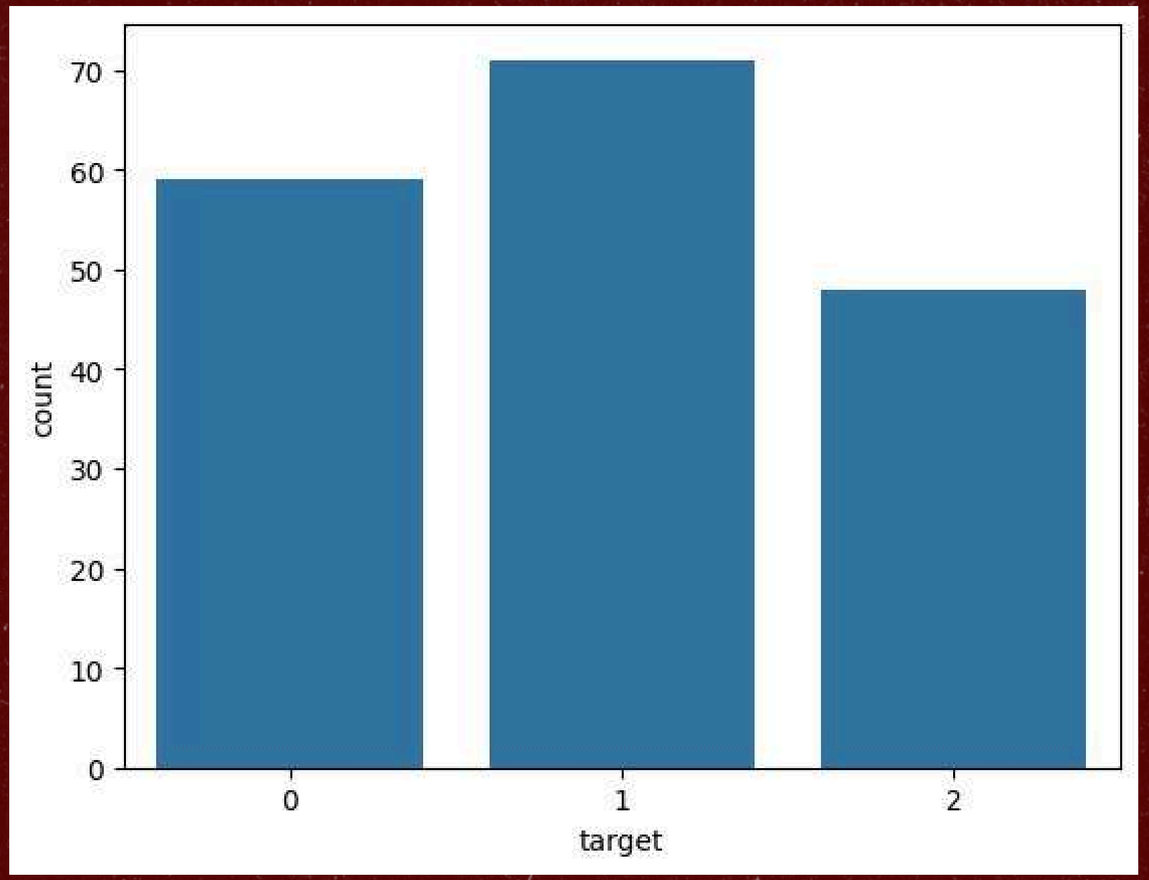
```python
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import seaborn as sns
import matplotlib.pyplot as plt

# Prediksi
y_pred_log_reg = log_reg.predict(X_test)
y_pred_knn = knn.predict(X_test)
y_pred_svm = svm.predict(X_test)

# Evaluasi fungsi
def evaluate_model(model_name, y_test, y_pred):
    print(f"\n=== {model_name} ===")
    print("\nClassification Report:\n", classification_report(y_test, y_pred))
    print("\nConfusion Matrix:")
    cm = confusion_matrix(y_test, y_pred)
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
    plt.title(f'Confusion Matrix - {model_name}')
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.show()
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Model Accuracy: {accuracy:.2f}")

# Evaluasi setiap model
evaluate_model("Logistic Regression", y_test, y_pred_log_reg)
evaluate_model("K-Nearest Neighbors", y_test, y_pred_knn)
```

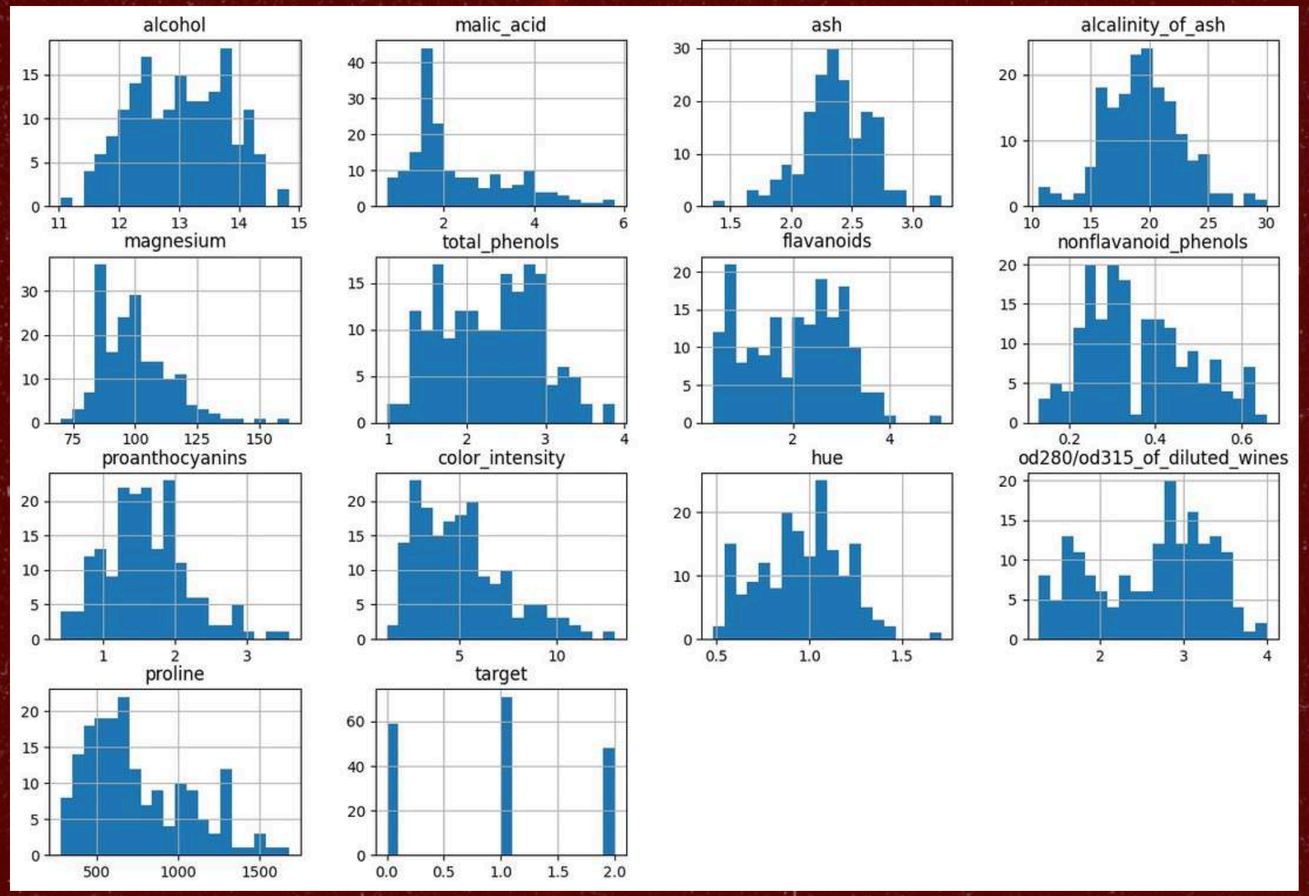




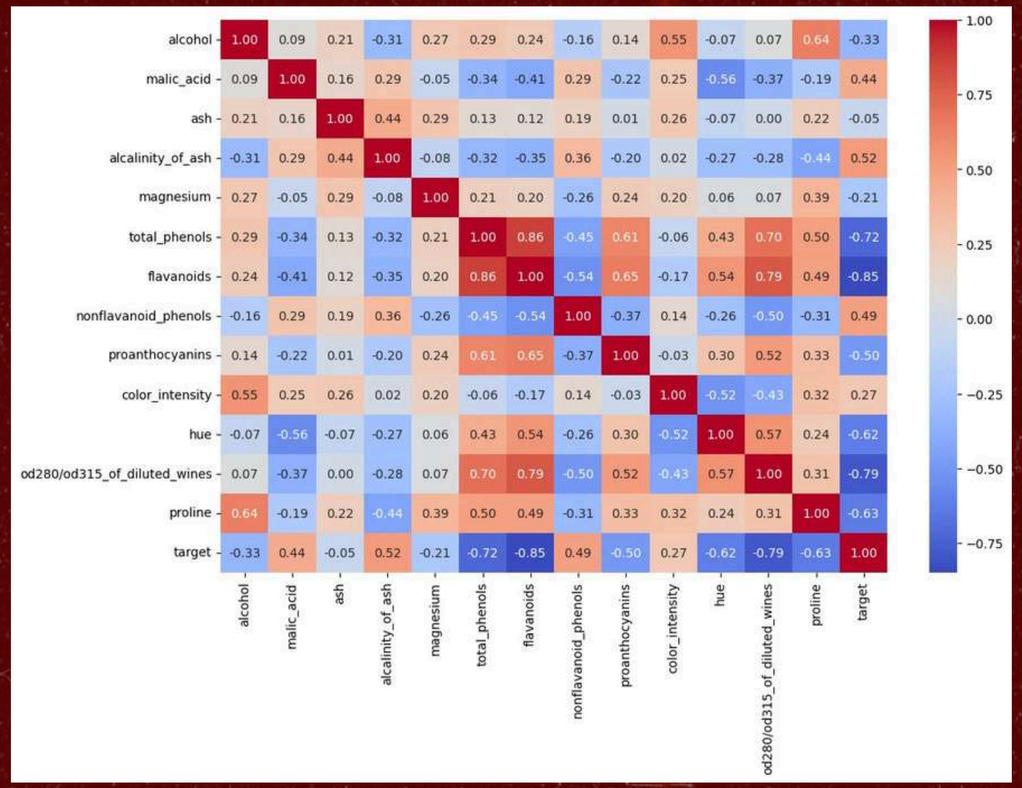


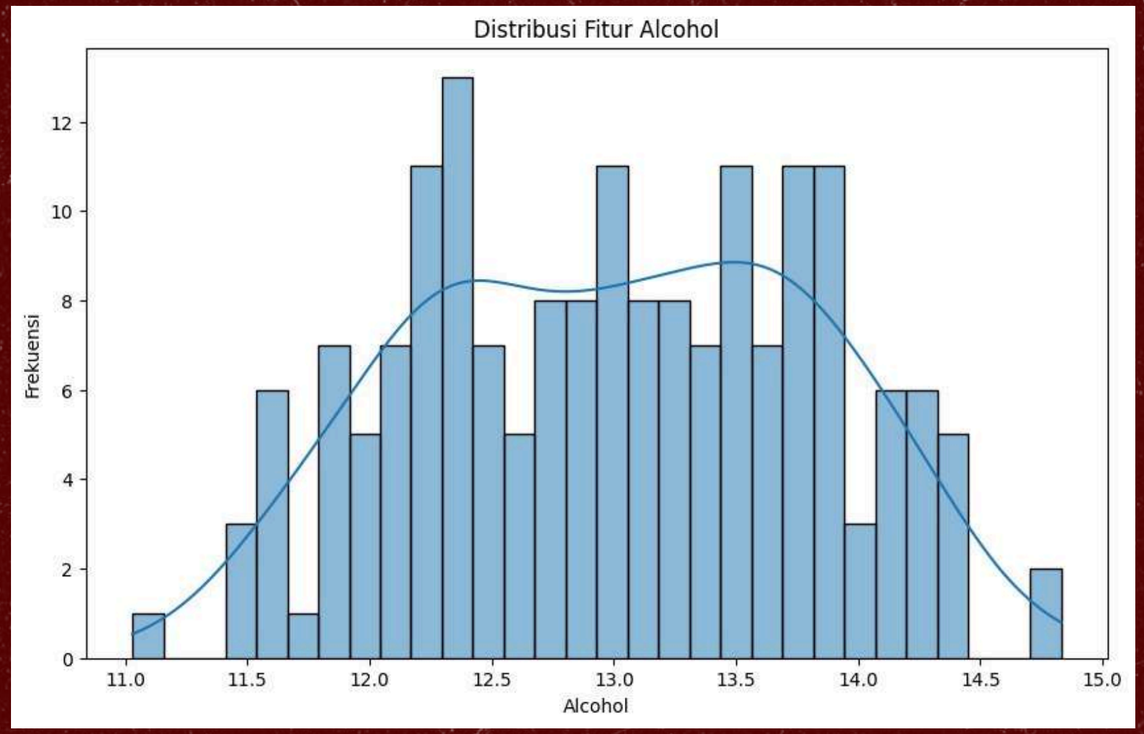
Exploratory Data Analysis (EDA)





EDA & Data Visualization





Conclusion

```
# Calculate accuracy for each model
log_reg_accuracy = accuracy_score(y_test, log_reg_pred)
knn_accuracy = accuracy_score(y_test, knn_pred)
svm_accuracy = accuracy_score(y_test, svm_pred)

# Print out the accuracies
print(f"Logistic Regression Accuracy: {log_reg_accuracy:.4f}")
print(f"K-Nearest Neighbors Accuracy: {knn_accuracy:.4f}")
print(f"Support Vector Machine Accuracy: {svm_accuracy:.4f}")

# Find the best model
accuracies = {'Logistic Regression': log_reg_accuracy, 'K-Nearest Neighbors':
best_model = max(accuracies, key=accuracies.get)
print(f"The best model is: {best_model} with an accuracy of {accuracies[best_m
```

```
Logistic Regression Accuracy: 1.0000
K-Nearest Neighbors Accuracy: 0.7407
Support Vector Machine Accuracy: 0.7593
The best model is: Logistic Regression with an accuracy of 1.0000
```

The conclusion from the results above shows that Logistic Regression achieves a perfect accuracy of 1.0000 on the tested dataset, making it the best model in this experiment. Meanwhile, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) provide lower accuracies of 0.7407 and 0.7593, respectively. This indicates that Logistic Regression is more effective in modeling this data, while KNN and SVM are less optimal in terms of accuracy.


Thank You
FOR YOUR ATTENTION