

Klasifikasi Jamur Berdasarkan Kelayakan Konsumsi Menggunakan *Support Vector Machine (SVM)*



Nurul Ummah


2108108010014

Metadata



Data yang digunakan dalam analisis ini merupakan data sekunder yang bersumber dari situs UCI Machine Learning dengan nama **Mushrooms**. Dataset ini mencakup deskripsi sampel hipotetis yang berhubungan dengan 23 spesies jamur insang dalam Famili Agaricus dan Lepiota. Data karakteristik jamur dari dataset Mushrooms memiliki 8124 baris dan 23 kolom yang terdiri atas 22 fitur dan 1 label. Label terdiri dari 2 nilai, yaitu dapat dikonsumsi (e) dan beracun (p).


Terdapat 22 variabel feature/atribut, berikut nama variabelnya: bentuk tudung, permukaan tudung, warna tudung, bau memar, perlekatan insang, jarak insang, ukuran insang, warna insang, bentuk tangkai, akar tangkai, permukaan-tangkai-cincin-atas, tangkai-permukaan-bawah-lingkar, warna-tangkai-lingkar-atas, warna-batang-bawah lingkar, tipe kerudung, warna kerudung, nomor cincin, tipe cincin, warna cetakan spora, populasi dan habitat.

Metadata

 UC Irvine
Machine Learning
Repository

DatasetsContribute DatasetAbout Us

Search datasets...   Login



Mushroom

Donated on 4/26/1987

From Audobon Society Field Guide; mushrooms described in terms of physical characteristics; classification: poisonous or edible


Dataset Characteristics	Subject Area	Associated Tasks
Multivariate	Biology	Classification
Feature Type	# Instances	# Features
Categorical	8124	22

Dataset Information


Additional Information


This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended. This latter class was combined with the poisonous one. The Guide clearly states that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be" for Poisonous Oak and Ivy.

DOWNLOAD

 IMPORT IN PYTHON

CITE

 77 citations

 131937 views


Keywords

ecology

DOI

10.24432/C5959T

License



Dataset (<https://archive.ics.uci.edu/ml/datasets/mushroom>)

Analysis Method

Support Vector Machine (SVM) adalah algoritma supervised learning yang kuat namun fleksibel yang digunakan baik untuk klasifikasi maupun regresi. SVM memiliki cara implementasi yang unik dibandingkan dengan algoritma pembelajaran mesin lainnya. Model SVM pada dasarnya adalah representasi dari kelas yang berbeda dalam hyperplane di ruang multidimensi. Hyperplane akan digenerate secara iteratif oleh SVM sehingga error dapat diminimalkan.

Dalam prakteknya, algoritma SVM diimplementasikan dengan kernel yang mengubah ruang input data menjadi bentuk yang dibutuhkan. SVM menggunakan teknik yang disebut trik kernel di mana kernel mengambil ruang masukan berdimensi rendah dan mengubahnya menjadi ruang berdimensi lebih tinggi. Dengan kata sederhana, kernel mengubah masalah yang tidak dapat dipisahkan menjadi masalah yang dapat dipisahkan dengan menambahkan lebih banyak dimensi ke dalamnya. Contoh kernel SVM antara lain kernel linier, radial basis function (RBF), polynomial, dan lain-lain.

Table of contents

01

Data processing

Data cleaning, data transformation, etc

02

Exploratory Data Analysis (EDA)

Describe, visualization, etc

03

Data preparation

Splitting data training and data testing, label encoding, etc

04

Modelling dan Evaluation

Modelling, evaluation using confusion matrix and classification report, etc



01

Data Processing

Data cleaning, data transformation, etc

Data Type

nurul.dtypes

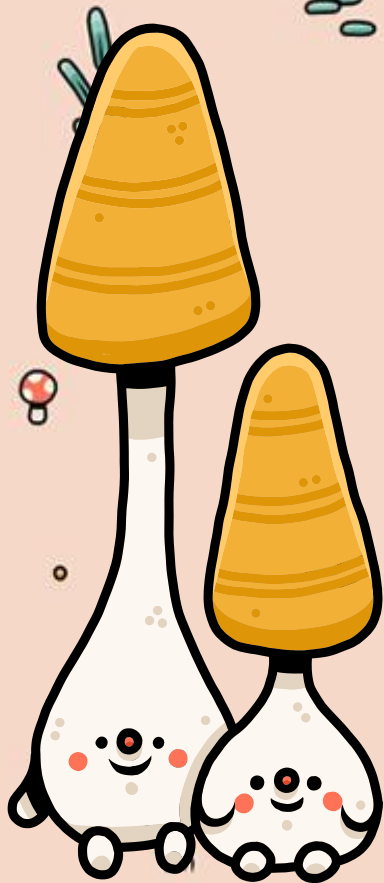
class	object
cap-shape	object
cap-surface	object
cap-color	object
bruises	object
odor	object
gill-attachment	object
gill-spacing	object
gill-size	object
gill-color	object
stalk-shape	object
stalk-root	object
stalk-surface-above-ring	object
stalk-surface-below-ring	object
stalk-color-above-ring	object
stalk-color-below-ring	object
veil-type	object
veil-color	object
ring-number	object
ring-type	object
spore-print-color	object
population	object
habitat	object
dtype:	object

```
# mengubah data object menjadi integer  
labelencoder=LabelEncoder()  
for column in nurul.columns:  
    nurul[column] = labelencoder.fit_transform(nurul[column])
```

nurul.dtypes

class	int32
cap-shape	int32
cap-surface	int32
cap-color	int32
bruises	int32
odor	int32
gill-attachment	int32
gill-spacing	int32
gill-size	int32
gill-color	int32
stalk-shape	int32
stalk-root	int32
stalk-surface-above-ring	int32
stalk-surface-below-ring	int32
stalk-color-above-ring	int32
stalk-color-below-ring	int32
veil-type	int32
veil-color	int32
ring-number	int32
ring-type	int32
spore-print-color	int32
population	int32
habitat	int32
dtype:	object

Missing dan Duplicate Data



```
nurul.isnull().sum()
```

class	0
cap-shape	0
cap-surface	0
cap-color	0
bruises	0
odor	0
gill-attachment	0
gill-spacing	0
gill-size	0
gill-color	0
stalk-shape	0
stalk-root	0
stalk-surface-above-ring	0
stalk-surface-below-ring	0
stalk-color-above-ring	0
stalk-color-below-ring	0
veil-type	0
veil-color	0
ring-number	0
ring-type	0
spore-print-color	0
population	0
habitat	0
dtype: int64	

```
# memeriksa duplikat data, jika ada  
nurul[nurul.duplicated()].count()
```

class	0
cap-shape	0
cap-surface	0
cap-color	0
bruises	0
odor	0
gill-attachment	0
gill-spacing	0
gill-size	0
gill-color	0
stalk-shape	0
stalk-root	0
stalk-surface-above-ring	0
stalk-surface-below-ring	0
stalk-color-above-ring	0
stalk-color-below-ring	0
veil-type	0
veil-color	0
ring-number	0
ring-type	0
spore-print-color	0
population	0
habitat	0
dtype: int64	



02

Explaratory Data Analysis

Describe, visualization, etc

Describe

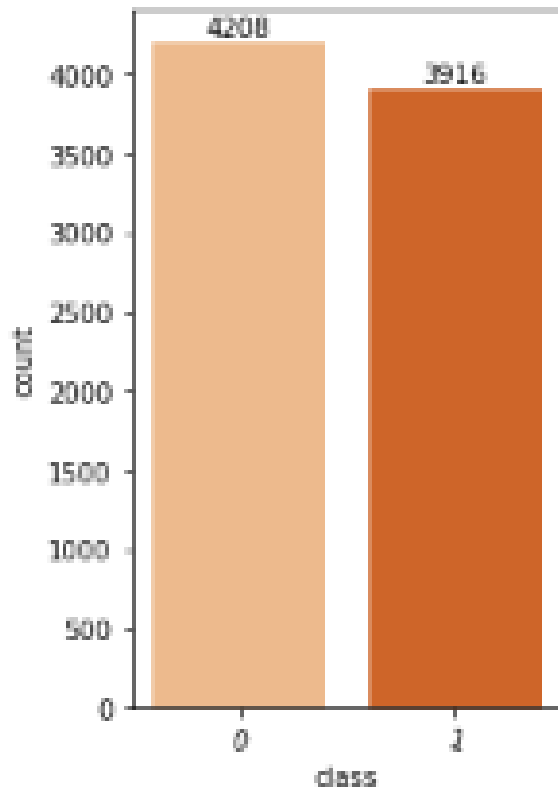
```
# deskriptif dari data  
nurul.describe().T
```

	count	mean	std	min	25%	50%	75%	max
class	8124.0	0.482029	0.499708	0.0	0.0	0.0	1.0	1.0
cap-shape	8124.0	3.348104	1.604329	0.0	2.0	3.0	5.0	5.0
cap-surface	8124.0	1.827671	1.229873	0.0	0.0	2.0	3.0	3.0
cap-color	8124.0	4.504677	2.545821	0.0	3.0	4.0	8.0	9.0
bruise	8124.0	0.415559	0.492848	0.0	0.0	0.0	1.0	1.0
odor	8124.0	4.144756	2.103729	0.0	2.0	5.0	5.0	8.0
gill-attachment	8124.0	0.974151	0.158695	0.0	1.0	1.0	1.0	1.0
gill-spacing	8124.0	0.161497	0.368011	0.0	0.0	0.0	0.0	1.0
gill-size	8124.0	0.309207	0.462195	0.0	0.0	0.0	1.0	1.0
gill-color	8124.0	4.810684	3.540359	0.0	2.0	5.0	7.0	11.0
stalk-shape	8124.0	0.567208	0.495493	0.0	0.0	1.0	1.0	1.0
stalk-root	8124.0	1.109798	1.061106	0.0	0.0	1.0	1.0	4.0
stalk-surface-above-ring	8124.0	1.575086	0.621459	0.0	1.0	2.0	2.0	3.0
stalk-surface-below-ring	8124.0	1.603644	0.675974	0.0	1.0	2.0	2.0	3.0
stalk-color-above-ring	8124.0	5.816347	1.901747	0.0	6.0	7.0	7.0	8.0
stalk-color-below-ring	8124.0	5.794682	1.907291	0.0	6.0	7.0	7.0	8.0
veil-color	8124.0	1.965534	0.242869	0.0	2.0	2.0	2.0	3.0
ring-number	8124.0	1.069424	0.271064	0.0	1.0	1.0	1.0	2.0
ring-type	8124.0	2.291974	1.801672	0.0	0.0	2.0	4.0	4.0
spore-print-color	8124.0	3.598750	2.382663	0.0	2.0	3.0	7.0	8.0
population	8124.0	3.644018	1.252082	0.0	3.0	4.0	4.0	5.0
habitat	8124.0	1.508616	1.719975	0.0	0.0	1.0	2.0	6.0

Untuk statistik deskriptif dapat dilihat label/class memiliki 2 nilai, yaitu 0 (dapat dikonsumsi) dan 1 (beracun) dan untuk feature/atribut yang semuanya merupakan tipe data object yang diubah kedalam numerik dimana setiap atribut memiliki karakteristik nilainya tersendiri. Sebagai contoh pada di dalam atribut **gill-color** terdapat maksimal 11 nilai. Deskriptif data di atas ditampilkan setelah data di-processing, yaitu setelah data diubah menjadi numerik yang sebelumnya bertipe kategorik.

Kategori dalam Class

```
labeled_barplot(nurul, "class")
```

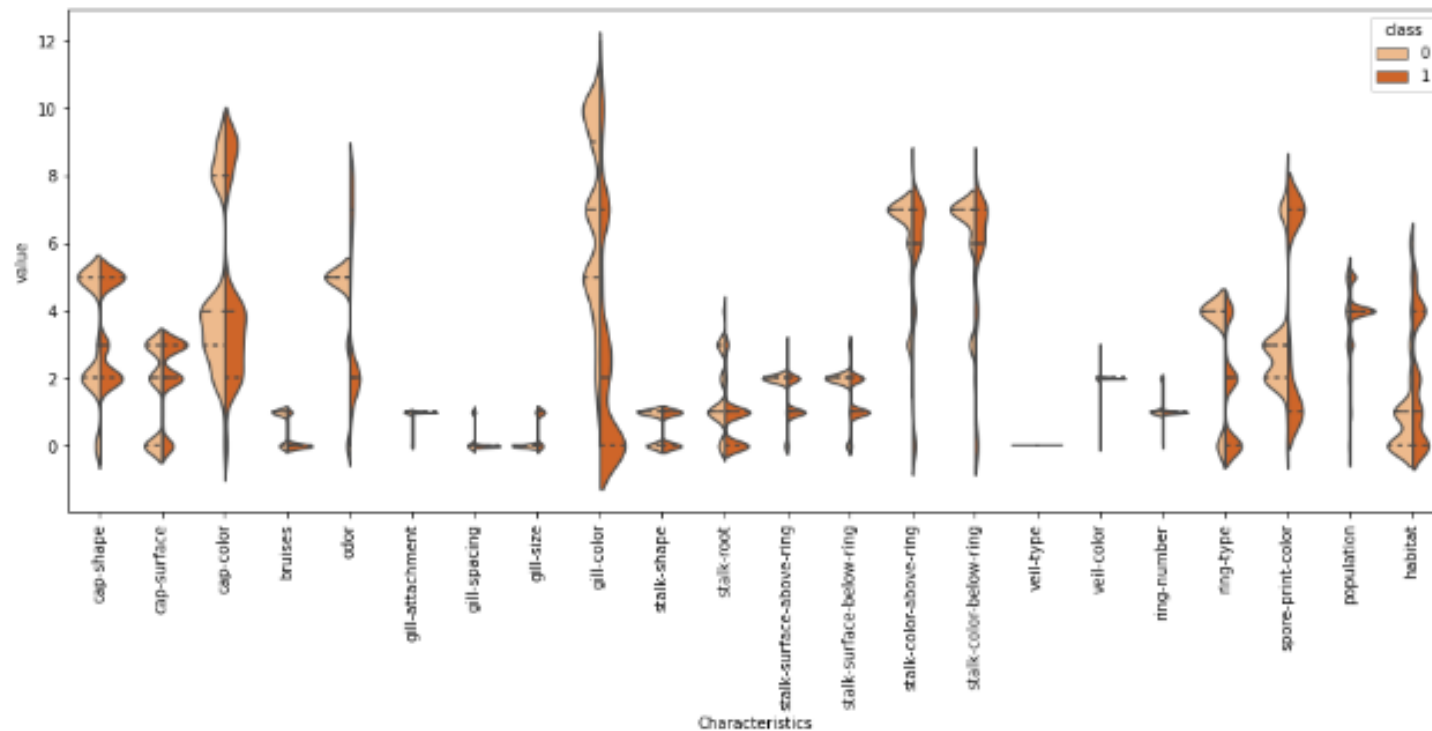


Dapat dilihat bahwa pada kelas 0 (dapat dikonsumsi) terdapat 4208 observasi dan kelas 1 (beracun) terdapat 3916 observasi.



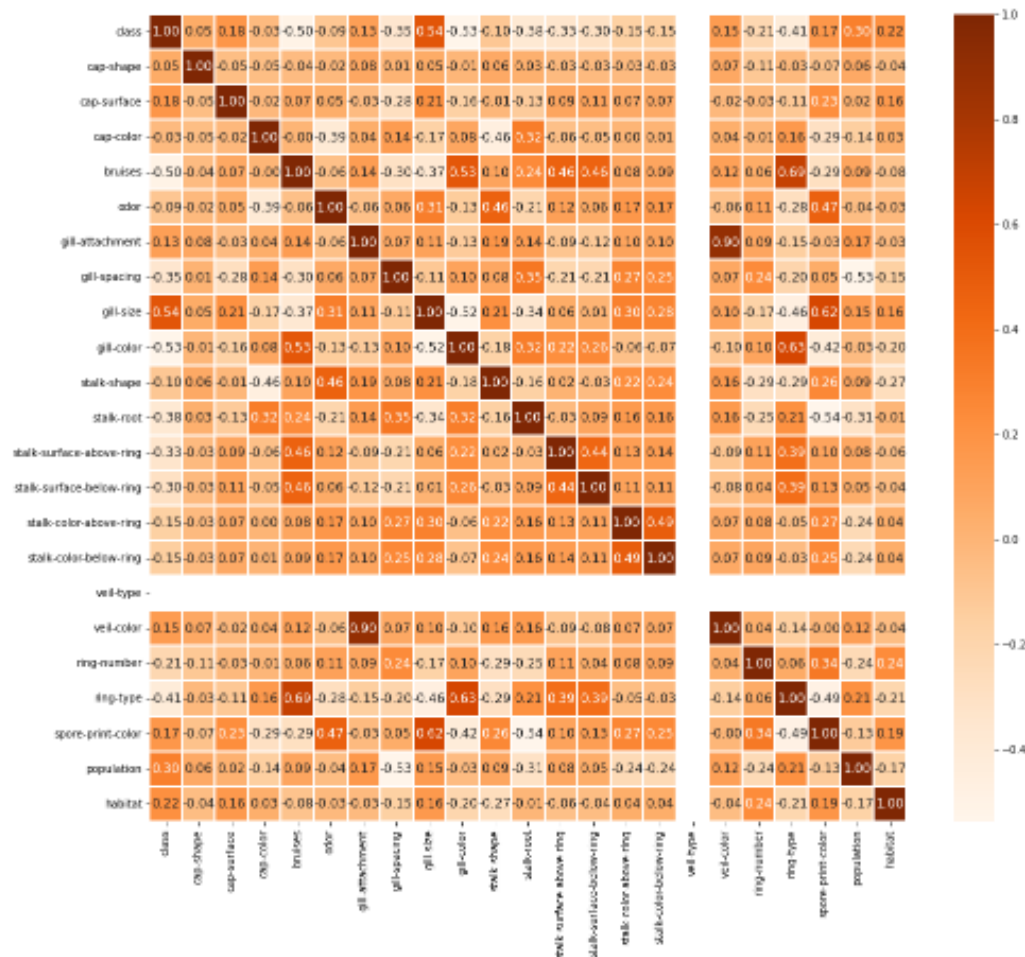
Violin Plot

```
# violin plot
nurul_violin = pd.melt(nurul, "class", var_name="Characteristics")
fig, ax = plt.subplots(figsize=(16,6))
plot_nurul = sns.violinplot(ax = ax, x="Characteristics", y="value", hue="class", split = True,
                             data=nurul_violin, inner = 'quartile', palette = 'Oranges')
nurul_selain_class = nurul.drop(["class"],axis = 1)
plot_nurul.set_xticklabels(rotation = 90, labels = list(nurul_selain_class.columns));
```



Heatmap

```
# plot untuk melihat korelasi antar feature
plt.figure(figsize=(16,14))
sns.heatmap(murul.corr(),linewidths=.2,cmap="Oranges",fct=".2F",annot=True, annot_kws={"size": 12})
plt.xticks(rotation=0);
```



Dapat dilihat bahwa atribut **gill-attachment** dan **veil-color** memiliki korelasi yang sangat kuat, yaitu sebesar 90%. Antara atribut **ring-type** dengan **bruises** memiliki korelasi yang sedang, yaitu sebesar 69% dan antara atribut **gill-color** dengan **ring-type** memiliki korelasi yang sedang juga, yaitu sebesar 63%.

03

Data Preparation

Splitting data training and data testing, label encoding, etc

Data Preparation



Features/Atribut

Semua variabel selain **class**



Label

Variabel **class**



Data Training

80% dari dataset
6519



Data Testing

20% dari dataset
1605



Label Encoding dan Standarisasi Data

```
lb = LabelEncoder()  
lb.fit(y_train)  
  
LabelEncoder()
```

Label Encoding adalah salah satu teknik encoding yang dapat digunakan pada dataset dengan variabel target berupa kelas atau kategori. Teknik ini digunakan untuk mengkonversi nilai teks menjadi nilai numerik, di mana setiap kelas diberi label numerik yang berbeda.

```
# standarisasi data  
scaler = StandardScaler()  
scaler.fit(X_train)  
  
X_train = scaler.transform(X_train)  
X_test = scaler.transform(X_test)
```

Syntax di atas menunjukkan bahwa data distandarisasi menggunakan fungsi `StandardScaler()` untuk memastikan bahwa skala dari setiap features atau variabel pada data training dan data testing setara atau seragam sebelum dilakukan training model.

The background of the slide features a repeating pattern of stylized green plants with three leaves and red mushrooms with white spots, set against a light orange background. A large orange square is positioned in the upper right area, containing the number 04.

04

Modelling dan Evaluasi

Modelling, evaluation using confusion matrix and classification report, etc

Model SVM Kernel Linier

```
#KERNEL LINEAR
model = svm.SVC(kernel='linear')
#train model pake data train
model.fit(X_train, y_train)
# predict data set pada data yang sudah ditraining
y_pred = model.predict(X_test)
print(y_pred)
```

```
[1 1 0 ... 0 0 1]
```

	precision	recall	f1-score	support
0	1.00	0.98	0.99	842
1	0.98	0.99	0.99	783
accuracy			0.99	1625
macro avg	0.99	0.99	0.99	1625
weighted avg	0.99	0.99	0.99	1625

	Hasil Prediksi	
Data Aktual	0	1
0	826	16
1	4	779

Model SVM dengan kernel linier memiliki rata-rata nilai *precision* sebesar 99%, rata-rata nilai *recall* sebesar 99% dan rata-rata *F1-Score* sebesar 99%. Secara keseluruhan, model ini memberikan akurasi sebesar 99%.

Model SVM Kernel RBF

```
# model svm rbf
model2 = svm.SVC(kernel='rbf')
#train model pake data train
model2.fit(X_train, y_train)
# predict data set pakai model sudah ditraining
y_pred2 = model2.predict(X_test)
print(y_pred2)
```

```
[1 1 0 ... 0 0 1]
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	842
1	1.00	1.00	1.00	783
accuracy			1.00	1625
macro avg	1.00	1.00	1.00	1625
weighted avg	1.00	1.00	1.00	1625

	Hasil Prediksi	
Data Aktual	0	1
0	842	0
1	0	763

Model SVM dengan kernel RBF memiliki rata-rata nilai *precision* sebesar 100%, rata-rata nilai *recall* sebesar 100% dan rata-rata *F1-Score* sebesar 100%. Secara keseluruhan, model ini memberikan akurasi sebesar 100%.

Model SVM Kernel Polinomial

```
# model svm polinomial
model3 = svm.SVC(kernel = 'poly')
#train model pake data train
model3.fit(X_train, y_train)
# predict data set pakai model sudah ditraining
y_pred3 = model3.predict(X_test)
print(y_pred3)
```

```
[1 1 0 ... 0 0 1]
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	842
1	1.00	1.00	1.00	783
accuracy			1.00	1625
macro avg	1.00	1.00	1.00	1625
weighted avg	1.00	1.00	1.00	1625

	Hasil Prediksi	
Data Aktual	0	1
0	842	0
1	0	763

Model SVM dengan kernel linier memiliki rata-rata nilai *precision* sebesar 100%, rata-rata nilai *recall* sebesar 100% dan rata-rata *F1-Score* sebesar 100%. Secara keseluruhan, model ini memberikan akurasi sebesar 100%.

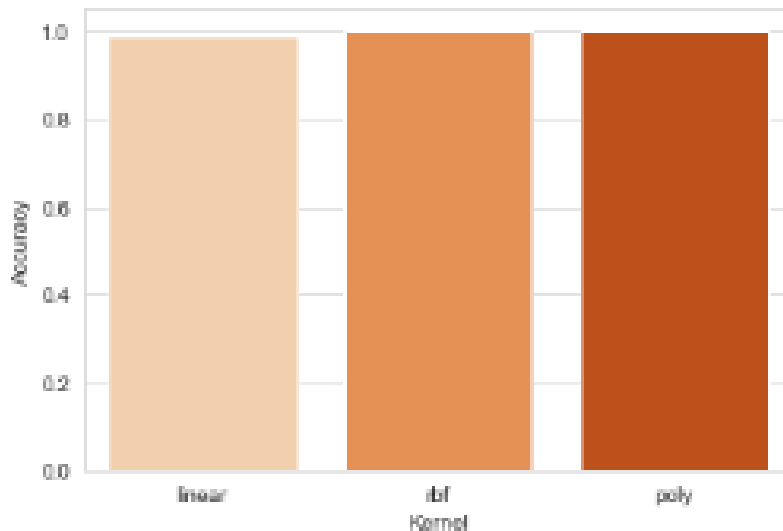
Perbandingan Akurasi Model

```
#Visualisasi Perbandingan Performa
kernels = ['linear', 'rbf', 'poly']
accuracy = [0.99, 1.00, 1.0]

#create barplot
sns.set_style("whitegrid")
sns.barplot(x=kernels, y=accuracy, palette='Oranges')

#set Labels
plt.xlabel("Kernel")
plt.ylabel("Accuracy")

#show plot
plt.show()
```



Dapat dilihat bahwa bahwa ketiga model SVM memberikan keakuratan yang hampir sama, yaitu sebesar 99% untuk kernel linier, 100% untuk kernel RBF dan 100% untuk kernel polinomial. Model SVM dengan kernel RBF dan polinomial hanya sedikit lebih akurat dibandingkan model SVM dengan kernel linier. Jadi, untuk memprediksi kelas pada *dataset* Mushrooms dapat digunakan model klasifikasi SVM dengan kernel linier, RBF atau polinomial.

THANKS!

