In [1]:
```
# import libraries

import pandas as pd
import numpy as np
import sklearn as sk
import matplotlib as mlb
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# import dataset

zoo = 'D:/Setelah Wisuda/Digitalent Kominfo/FGA - Data Scientist/Assignment/Challenge 2/1. Zoo Animal Classification - Ka
zoo = pd.read_csv(zoo)

classes = 'D:/Setelah Wisuda\Digitalent Kominfo/FGA - Data Scientist/Assignment/Challenge 2/1. Zoo Animal Classification
classes = pd.read_csv(classes)

zhead = zoo.head()
chead = classes.head()

display(zhead)
display(chead)
```

	animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsiz
0	aardvark	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	
1	antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	
2	bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	
3	bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	
4	boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	

	Class_Number	Number_Of_Animal_Species_In_Class	Class_Type	Animal_Names
0	1	41	Mammal	aardvark, antelope, bear, boar, buffalo, calf,...
1	2	20	Bird	chicken, crow, dove, duck, flamingo, gull, haw...
2	3	5	Reptile	pitviper, seasnake, slowworm, tortoise, tuatara
3	4	13	Fish	bass, carp, catfish, chub, dogfish, haddock, h...
4	5	4	Amphibian	frog, frog, newt, toad

In [3]:
```
# Data Cleaning (check missing/invalid values, eliminating duplictae rows, formatting properly)
zDC1 = zoo.dtypes
zDC2 = zoo.info()
zDC3 = zoo.describe (include='all')
cDC1 = classes.dtypes
cDC2 = classes.info()
cDC3 = classes.describe (include='all')

# menunjukkan bahwa data zoo dan classes sudah clean karena tidak ada missing value dan formatnya sudah benar
display(zDC1)
display(zDC2)
display(zDC3)
display(cDC1)
display(cDC2)
display(cDC3)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101 entries, 0 to 100
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   animal_name  101 non-null    object
 1   hair         101 non-null    int64 
 2   feathers     101 non-null    int64 
 3   eggs         101 non-null    int64 
 4   milk         101 non-null    int64 
 5   airborne     101 non-null    int64 
 6   aquatic      101 non-null    int64 
 7   predator     101 non-null    int64 
 8   toothed      101 non-null    int64 
 9   backbone     101 non-null    int64 
 10  breathes     101 non-null    int64 
 11  venomous     101 non-null    int64 
 12  fins         101 non-null    int64 
```

In [4]:
```
# unique value tiap feature yang akan digunakan
print('unique value pada feature hair adalah:', zoo.hair.unique())
print('unique value pada feature feathers adalah:', zoo.feathers.unique())
print('unique value pada feature eggs adalah:', zoo.eggs.unique())
print('unique value pada feature milk adalah:', zoo.milk.unique())
print('unique value pada feature airbone adalah:', zoo.airborne.unique())
print('unique value pada feature aquatic adalah:', zoo.aquatic.unique())
print('unique value pada feature predator adalah:', zoo.predator.unique())
print('unique value pada feature toothed adalah:', zoo.toothed.unique())
print('unique value pada feature backbone adalah:', zoo.backbone.unique())
print('unique value pada feature breathes adalah:', zoo.breathes.unique())
print('unique value pada feature venomous adalah:', zoo.venomous.unique())
print('unique value pada feature fins adalah:', zoo.fins.unique())
print('unique value pada feature legs adalah:', zoo.legs.unique())
print('unique value pada feature domestic adalah:', zoo.domestic.unique())
print('unique value pada feature catsize adalah:', zoo.catsize.unique())

# unique value tiap feature untuk info tambahan
print('unique value pada feature Class_Number adalah:', classes.Class_Number.unique())
print('unique value pada feature Number_Of_Animal_Species_In_Class adalah:', classes.Number_Of_Animal_Species_In_Class.u
print('unique value pada feature Class_Type adalah:', classes.Class_Type.unique())
```

```
unique value pada feature hair adalah: [1 0]
unique value pada feature feathers adalah: [0 1]
unique value pada feature eggs adalah: [0 1]
unique value pada feature milk adalah: [1 0]
unique value pada feature airbone adalah: [0 1]
unique value pada feature aquatic adalah: [0 1]
unique value pada feature predator adalah: [1 0]
unique value pada feature toothed adalah: [1 0]
unique value pada feature backbone adalah: [1 0]
unique value pada feature breathes adalah: [1 0]
unique value pada feature venomous adalah: [0 1]
unique value pada feature fins adalah: [0 1]
unique value pada feature legs adalah: [4 0 2 6 8 5]
unique value pada feature domestic adalah: [0 1]
unique value pada feature catsize adalah: [1 0]
unique value pada feature Class_Number adalah: [1 2 3 4 5 6 7]
unique value pada feature Number_Of_Animal_Species_In_Class adalah: [41 20  5 13  4  8 10]
unique value pada feature Class_Type adalah: ['Mammal' 'Bird' 'Reptile' 'Fish' 'Amphibian' 'Bug' 'Invertebrate']
```

In [5]:
```
# Explorattory Data Analysis

# join zoo dataframe dengan classes dataframe untuk menunjukkan nama origin class_type pada setiap row
merge_df=pd.merge(zoo,classes,how='left',left_on='class_type',right_on='Class_Number')
# delete column class_type pada dataframe zoo dan keep class_type pada dataframe classes
del merge_df['class_type']
merge_df.rename(columns = {'Class_Number':'class_number'}, inplace = True)
merge_df.rename(columns = {'Class_Type':'class_type'}, inplace = True)
# ganti column legs dengan has_legs agar datanya binary seperti data yang lain
merge_df['has_legs'] = np.where(merge_df['legs']>0,1,0)
df = merge_df[['animal_name','hair','feathers','eggs','milk','airborne','aquatic','predator','toothed','backbone','breatl
               'venomous','fins','has_legs','tail','domestic','catsize','class_number','class_type',
               'Number_Of_Animal_Species_In_Class','Animal_Names']]
display(df.head())

# summary origin animal_type dari dataset
print("Dataset Zoo ini memiliki",len(zoo),"rows.")
sns.catplot('class_type', data=df,kind="count", aspect=2)

# correlation plot dari 16 feature
corr = zoo.iloc[:,1:-1].corr()
colormap = sns.diverging_palette(20, 120, as_cmap = True)
plt.figure(figsize=(14,13))
sns.heatmap(corr, cbar = True,  square = True, annot=True, fmt= '.2f',annot_kws={'size': 14},
            cmap = colormap, linewidths=0.1, linecolor='white')
plt.title('Correlation of ZOO Features', y=1.05, size=15)
```

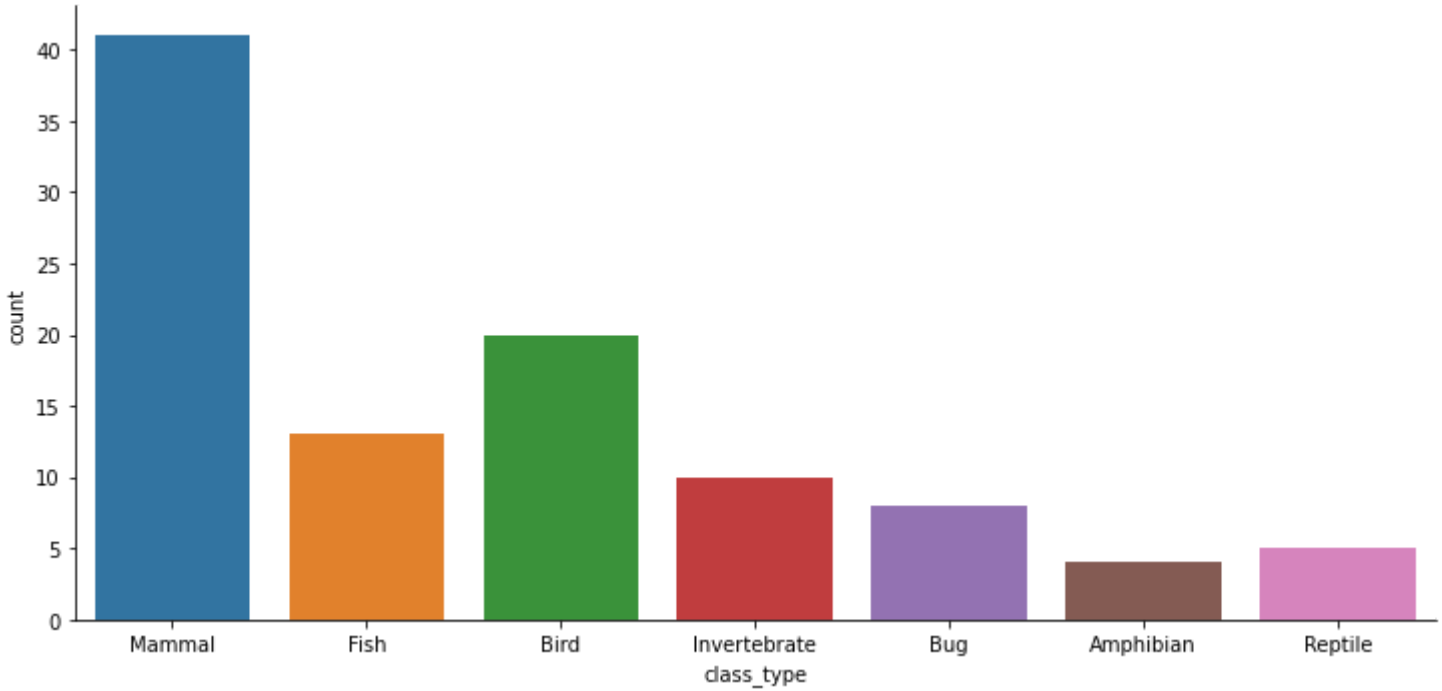
	animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	...	venomous	fins	has_legs	tail	domestic	catsiz
0	aardvark	1	0	0	1	0	0	1	1	1	...	0	0	1	0	0	
1	antelope	1	0	0	1	0	0	0	1	1	...	0	0	1	1	0	

	animal_name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	...	venomous	fins	has_legs	tail	domestic	catsiz
2	bass	0	0	1	0	0	1	1	1	1	...	0	1	0	1	0	
3	bear	1	0	0	1	0	0	1	1	1	...	0	0	1	0	0	
4	boar	1	0	0	1	0	0	1	1	1	...	0	0	1	1	0	

5 rows × 21 columns

```
Dataset Zoo ini memiliki 101 rows.
```

Out[5]: `Text(0.5, 1.05, 'Correlation of ZOO Features')`




Correlation of ZOO Features
hair 1.00 -0.43 -0.82 0.88 -0.20 -0.47 -0.15 0.49 0.19 0.44 -0.10 -0.28 0.39 0.05 0.21 0.46
feathers -0.43 1.00 0.42 -0.41 0.66 -0.06 -0.10 -0.61 0.23 0.25 -0.15 -0.22 -0.21 0.29 0.03 -0.14
eggs -0.82 0.42 1.00 -0.94 0.38 0.38 0.01 -0.64 -0.34 -0.38 0.10 0.16 -0.22 -0.22 -0.16 -0.51
milk 0.88 -0.41 -0.94 1.00 -0.37 -0.36 -0.03 0.63 0.38 0.42 -0.24 -0.16 0.21 0.21 0.16 0.57
airborne -0.20 0.66 0.38 -0.37 1.00 -0.17 -0.30 -0.59 -0.10 0.29 0.01 -0.25 0.04 0.01 0.06 -0.35
aquatic -0.47 -0.06 0.38 -0.36 -0.17 1.00 0.38 0.05 0.02 -0.64 0.09 0.60 -0.36 -0.03 -0.22 -0.11
predator -0.15 -0.10 0.01 -0.03 -0.30 0.38 1.00 0.13 0.05 -0.26 0.12 0.19 -0.10 0.02 -0.31 0.14
toothed 0.49 -0.61 -0.64 0.63 -0.59 0.05 0.13 1.00 0.58 -0.07 -0.06 0.36 -0.19 0.31 0.07 0.34
backbone 0.19 0.23 -0.34 0.38 -0.10 0.02 0.05 0.58 1.00 0.21 -0.25 0.21 -0.43 0.73 0.10 0.36
breathes 0.44 0.25 -0.38 0.42 0.29 -0.64 -0.26 -0.07 0.21 1.00 -0.12 -0.62 0.37 0.09 0.12 0.20
venomous -0.10 -0.15 0.10 -0.24 0.01 0.09 0.12 -0.06 -0.25 -0.12 1.00 -0.03 0.02 -0.16 -0.00 -0.18
fins -0.28 -0.22 0.16 -0.16 -0.25 0.60 0.19 0.36 0.21 -0.62 -0.03 1.00 -0.61 0.20 -0.09 0.03
legs 0.39 -0.21 -0.22 0.21 0.04 -0.36 -0.10 -0.19 -0.43 0.37 0.02 -0.61 1.00 -0.35 0.07 0.07
tail 0.05 0.29 -0.22 0.21 0.01 -0.03 0.02 0.31 0.73 0.09 -0.16 0.20 -0.35 1.00 0.02 0.24
domestic 0.21 0.03 -0.16 0.16 0.06 -0.22 -0.31 0.07 0.10 0.12 -0.00 -0.09 0.07 0.02 1.00 0.02
catsize 0.46 -0.14 -0.51 0.57 -0.35 -0.11 0.14 0.34 0.36 0.20 -0.18 0.03 0.07 0.24 0.02 1.00
hair
eggs
milk
fins
legs
tail
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

In [6]:
```
# hasil correlation yang positif atau negatif (memiliki nilai lebih dari 0.7)
corr = corr[corr != 1][abs(corr)> 0.7].dropna(how='all', axis=1).dropna(how='all', axis=0)
display(corr)
```

	hair	eggs	milk	backbone	tail
hair	NaN	-0.817382	0.878503	NaN	NaN
eggs	-0.817382	NaN	-0.938848	NaN	NaN
milk	0.878503	-0.938848	NaN	NaN	NaN
backbone	NaN	NaN	NaN	NaN	0.731762
tail	NaN	NaN	NaN	0.731762	NaN

In [7]:
```
# hasil check feature setiap class_type:
# 1. feature feathers menentukan class_type BIRD
# 2. feature milk menentukan class_type MAMMAL
# 3. feature has_legs menentukan class_type FISH

df.groupby('class_type').mean()
```

Out[7]:

class_type	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	has_legs	tail
Amphibian	0.00000	0.0	1.00000	0.0	0.00000	1.000000	0.750000	1.00000	1.0	1.0	0.250000	0.000000	1.000000	0.250000
Bird	0.00000	1.0	1.00000	0.0	0.80000	0.300000	0.450000	0.00000	1.0	1.0	0.000000	0.000000	1.000000	1.000000
Bug	0.50000	0.0	1.00000	0.0	0.75000	0.000000	0.125000	0.00000	0.0	1.0	0.250000	0.000000	1.000000	0.000000
Fish	0.00000	0.0	1.00000	0.0	0.00000	1.000000	0.692308	1.00000	1.0	0.0	0.076923	1.000000	0.000000	1.000000
Invertebrate	0.00000	0.0	0.90000	0.0	0.00000	0.600000	0.800000	0.00000	0.0	0.3	0.200000	0.000000	0.600000	0.100000
Mammal	0.95122	0.0	0.02439	1.0	0.04878	0.146341	0.536585	0.97561	1.0	1.0	0.000000	0.097561	0.926829	0.853659
Reptile	0.00000	0.0	0.80000	0.0	0.00000	0.200000	0.800000	0.80000	1.0	0.8	0.400000	0.000000	0.400000	1.000000

In [8]:
```
# tentukan data x (feature) dan data y (label data) untuk pemodelan menggunakan metode machine learning

features = list(df.columns.values)
features.remove('animal_name')
features.remove('class_type')
features.remove('class_number')
features.remove('Number_Of_Animal_Species_In_Class')
features.remove('Animal_Names')

x = df[features]
y = df['class_number']

display(x.shape)
display(x.shape)
display(x.head())
display(y.head())
```

```
(101, 16)
```

```
(101, 16)
```

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	has_legs	tail	domestic	catsize
0	1	0	0	1	0	0	1	1	1	1	0	0	1	0	0	1
1	1	0	0	1	0	0	0	1	1	1	0	0	1	1	0	1
2	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0
3	1	0	0	1	0	0	1	1	1	1	0	0	1	0	0	1
4	1	0	0	1	0	0	1	1	1	1	0	0	1	1	0	1

```
0    1
1    1
2    4
3    1
4    1
Name: class_number, dtype: int64
```

In [9]:
```
from sklearn.model_selection import train_test_split

# melakukan evaluasi data dengan menggunakan tipe TRAIN/TEST SPLIT
# split : 70% Train, 30% Test
train_x, test_x, train_y, test_y = train_test_split(x, y, test_size=0.3, random_state=42, stratify=y)

print("Training Data memiliki",train_x.shape)
print("Testing Data memiliki",test_x.shape)
print("Training Target memiliki",train_y.shape)
print("Testing Target memiliki",test_y.shape)
display (train_x)
display (test_x)
display (train_y)
display (test_y)
```

```
Training Data memiliki (70, 16)
Testing Data memiliki (31, 16)
Training Target memiliki (70,)
Testing Target memiliki (31,)
```

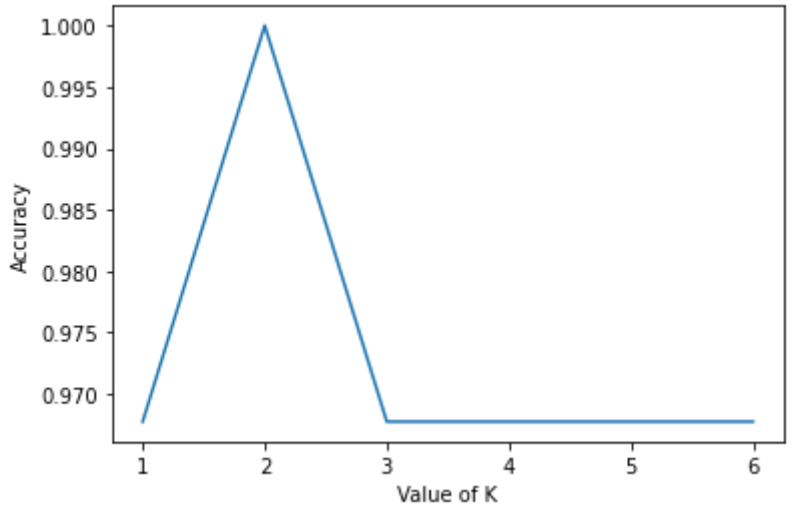
In [10]:

```
from sklearn import preprocessing
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.neighbors import KNeighborsClassifier

range_k = range(1,7)
scores = {}
scores_list = []
for k in range_k:
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(train_x, train_y)
    pred_y = knn.predict(test_x)
    scores = metrics.accuracy_score(test_y,pred_y)
    scores_list.append(metrics.accuracy_score(test_y,pred_y))
result = metrics.classification_report(test_y,pred_y)

print("Classification Report:",)
print (result)

print('Accuracy vs Value of K')
%matplotlib inline
plt.plot(range_k,scores_list)
plt.xlabel("Value of K")
plt.ylabel("Accuracy")
```

```
Classification Report:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        13
           2       1.00      1.00      1.00         6
           3       1.00      0.50      0.67         2
           4       0.80      1.00      0.89         4
           5       1.00      1.00      1.00         1
           6       1.00      1.00      1.00         2
           7       1.00      1.00      1.00         3

    accuracy                           0.97        31
   macro avg       0.97      0.93      0.94        31
weighted avg       0.97      0.97      0.96        31
```

```
Accuracy vs Value of K
```

Out[10]: Text(0, 0.5, 'Accuracy')

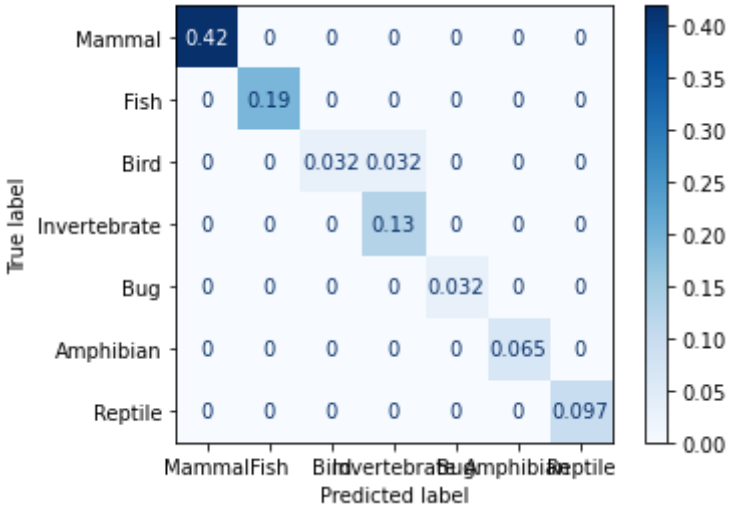


In [11]:
```
# hasil iterasi pergantian nilai K pada model, menunjukkan K = 2 memiliki nilai accuracy tertinggi
# lakukan pemodelan pada K = 2

from sklearn import preprocessing
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import accuracy_score

class_type = df['class_type'].unique()

x2 = df[features]
y2 = df['class_number']

train_x2, test_x2, train_y2, test_y2 = train_test_split(x2, y2, test_size=0.3, random_state=42, stratify=y)

knn2 = KNeighborsClassifier(n_neighbors=1)
knn2.fit(train_x2, train_y2)
pred_y2 = knn2.predict(test_x2)
scores2 = metrics.accuracy_score(test_y2,pred_y2)
result2 = metrics.classification_report(test_y2,pred_y2, target_names=(df['class_type'].unique()))

cnf_mat = metrics.confusion_matrix(test_y2,pred_y2)
plot_conf_mat = plot_confusion_matrix(knn2, test_x2, test_y2,
                                      display_labels=class_type,
                                      cmap=plt.cm.Blues, normalize='all')

print('Accuracy:',metrics.accuracy_score(test_y2,pred_y2))
print("Confusion Matrix:")
print(cnf_mat)
print("Classification Report:",)
print (result2)
```

```
Accuracy: 0.967741935483871
Confusion Matrix:
[[13  0  0  0  0  0  0]
 [ 0  6  0  0  0  0  0]
 [ 0  0  1  1  0  0  0]
 [ 0  0  0  4  0  0  0]
 [ 0  0  0  0  1  0  0]
 [ 0  0  0  0  0  2  0]
 [ 0  0  0  0  0  0  3]]
```

Classification Report:

	precision	recall	f1-score	support
Mammal	1.00	1.00	1.00	13
Fish	1.00	1.00	1.00	6
Bird	1.00	0.50	0.67	2
Invertebrate	0.80	1.00	0.89	4
Bug	1.00	1.00	1.00	1
Amphibian	1.00	1.00	1.00	2
Reptile	1.00	1.00	1.00	3
accuracy			0.97	31
macro avg	0.97	0.93	0.94	31
weighted avg	0.97	0.97	0.96	31



In [12]:
```
# komparasi nilai target actual (test_y2) dengan nilai target prediksi (pred_y2)

# predicted_class = pred_y2
print('nilai predicted_class')
display(pred_y2)

# actual_class = numpy array test_y2
actual_y2 = np.array(test_y2)
print('nilai actual_class:')
display(actual_y2)
```

```
nilai predicted_class

array([2, 2, 2, 2, 1, 6, 1, 1, 2, 3, 1, 2, 4, 1, 7, 4, 1, 1, 1, 5, 4, 1,
       1, 4, 1, 4, 7, 1, 7, 6, 1], dtype=int64)

nilai actual_class:

array([2, 2, 2, 2, 1, 6, 1, 1, 2, 3, 1, 2, 4, 1, 7, 4, 1, 1, 1, 5, 4, 1,
       1, 4, 1, 3, 7, 1, 7, 6, 1], dtype=int64)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: