KNN Assignment

Data Set - Zoo

1. Import Necessary libraries

In [1]:
```
import pandas as pd
import numpy as np

from matplotlib import pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

2. Import Data

In [2]:
```
animals_data = pd.read_csv('Zoo.csv')
animals_data
```

Out[2]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	type
0	aardvark	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
1	antelope	1	0	0	1	0	0	0	1	1	1	0	0	4	1	0	1	1
2	bass	0	0	1	0	0	1	1	1	1	0	0	1	0	1	0	0	4
3	bear	1	0	0	1	0	0	1	1	1	1	0	0	4	0	0	1	1
4	boar	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
96	wallaby	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	1	1
97	wasp	1	0	1	0	1	0	0	0	0	1	1	0	6	0	0	0	6
98	wolf	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
99	worm	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	7
100	wren	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2

101 rows × 18 columns

3. Data Understanding

3.1 Initial Analysis :

In [3]:
```
animals_data.head()
```

Out[3]:

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

In [4]:
```
animals_data.shape
```

Out[4]:
```
(101, 18)
```

In [5]:
```
animals_data.info()
```

```
<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
 13  legs         101 non-null    int64
 14  tail         101 non-null    int64
 15  domestic     101 non-null    int64
 16  catsize      101 non-null    int64
 17  type         101 non-null    int64
dtypes: int64(17), object(1)
memory usage: 14.3+ KB
```

In [6]:
```
animals_data.isna().sum()
```

Out[6]:
```
animal name    0
hair           0
feathers       0
eggs           0
milk           0
airborne       0
aquatic        0
predator       0
toothed        0
backbone       0
breathes       0
venomous       0
fins           0
legs           0
tail           0
domestic       0
catsize        0
type           0
dtype: int64
```

In [7]:
```
animals_data.describe()
```

Out[7]:

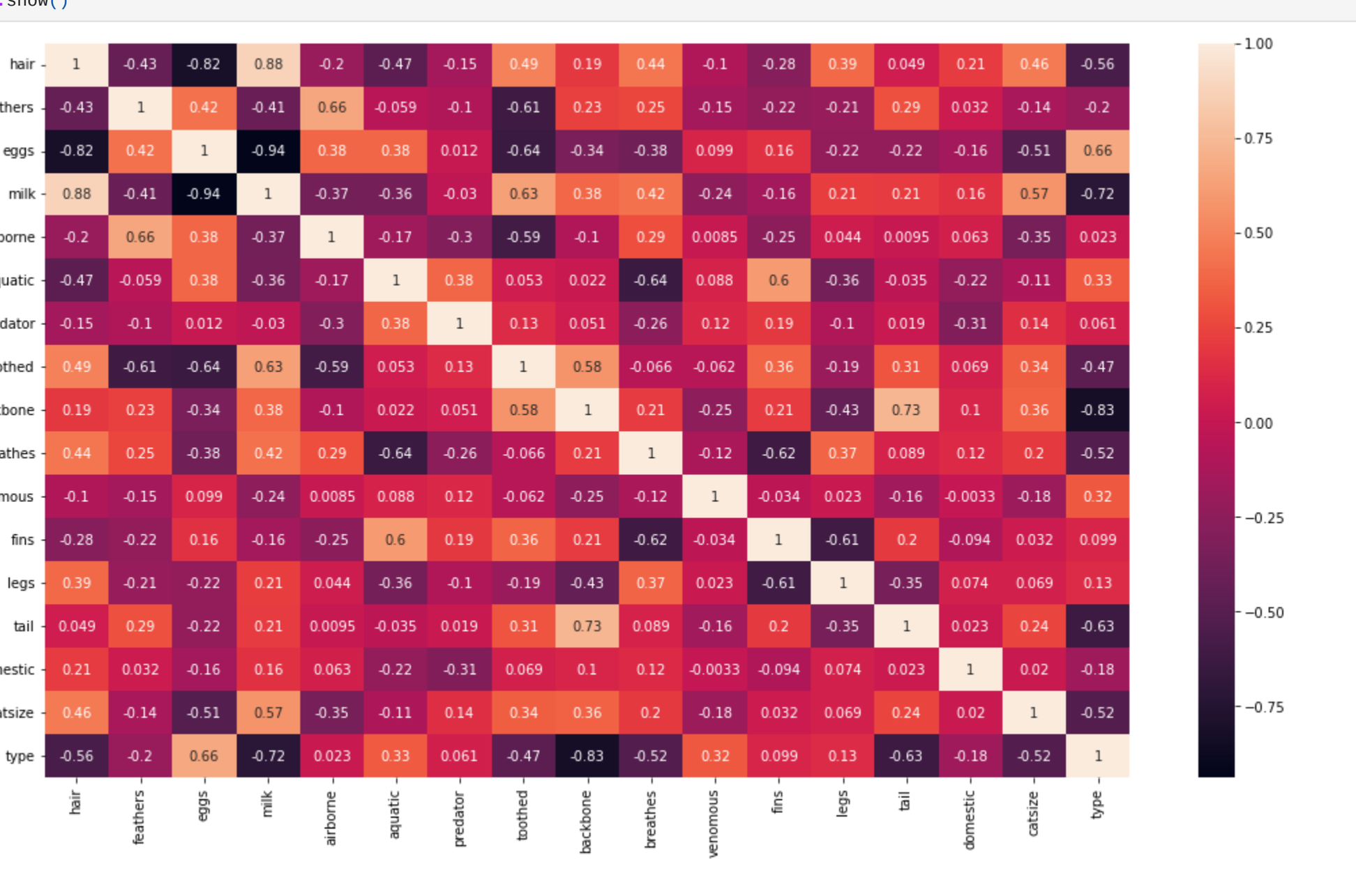
	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	type
count	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000
mean	0.425743	0.198020	0.584158	0.405941	0.237624	0.356436	0.554455	0.603960	0.821782	0.792079	0.079208	0.168317	2.841584	0.742574	0.128713	0.435644	2.831683
std	0.496921	0.400495	0.495325	0.493522	0.427750	0.481335	0.499505	0.491512	0.384605	0.407844	0.271410	0.376013	2.033385	0.439397	0.336552	0.498314	2.102709
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	2.000000	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	4.000000	1.000000	0.000000	0.000000	2.000000
75%	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	4.000000	1.000000	0.000000	1.000000	4.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	8.000000	1.000000	1.000000	1.000000	7.000000

In [8]:
```
animals_data.dtypes
```

Out[8]:
```
animal name    object
hair            int64
feathers        int64
eggs            int64
milk            int64
airborne        int64
aquatic         int64
predator        int64
toothed         int64
backbone        int64
breathes        int64
venomous        int64
fins            int64
legs            int64
tail            int64
domestic        int64
catsize         int64
type            int64
dtype: object
```

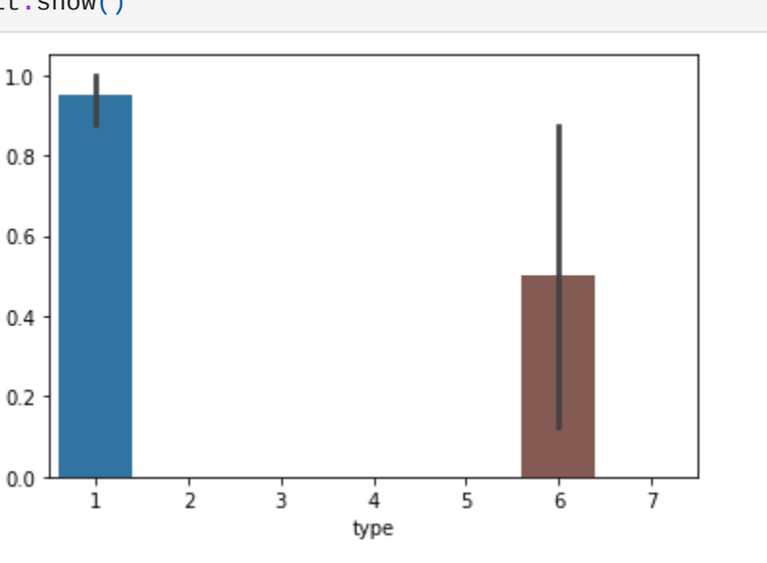
3.2 Correlation Matrix :

In [9]:
```
plt.figure(figsize = (18,10))
sns.heatmap(animals_data.corr(),annot = True)
plt.show()
```

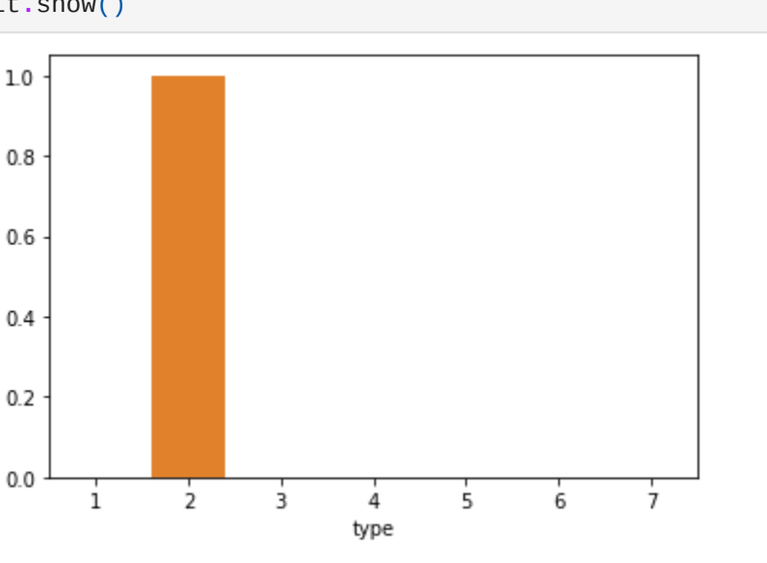


4. Perform Assumption Check

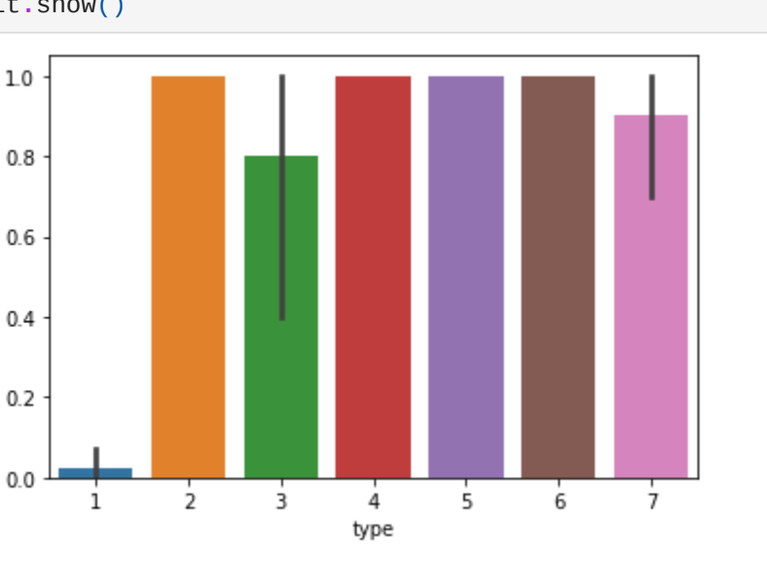
In [10]:
```
sns.barplot(x = 'type' , y = 'hair',data = animals_data)
plt.show()
```



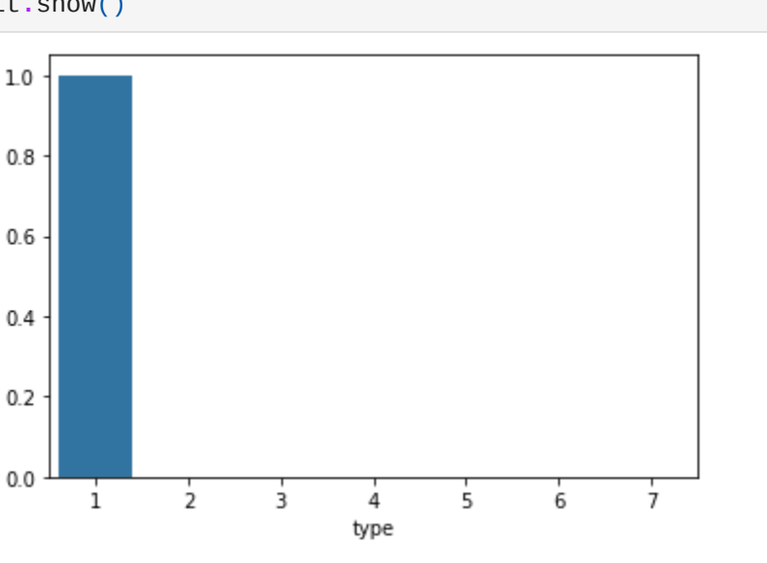
In [11]:
```
sns.barplot(x = 'type' , y = 'feathers',data = animals_data)
plt.show()
```



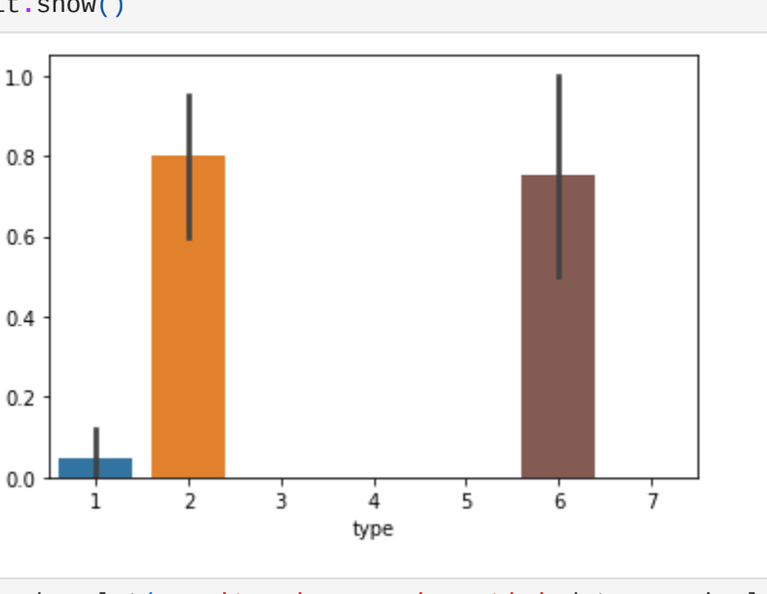
In [12]:
```
sns.barplot(x = 'type' , y = 'eggs',data = animals_data)
plt.show()
```



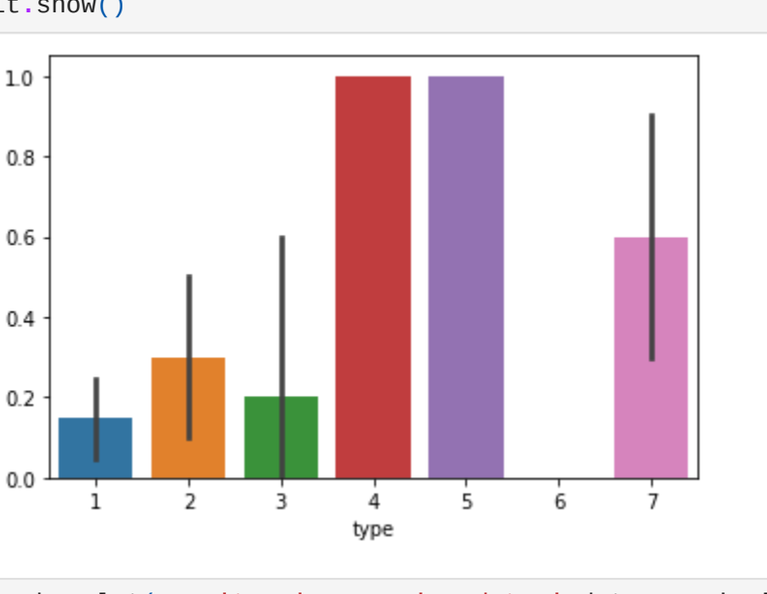
In [13]:
```
sns.barplot(x = 'type' , y = 'milk',data = animals_data)
plt.show()
```



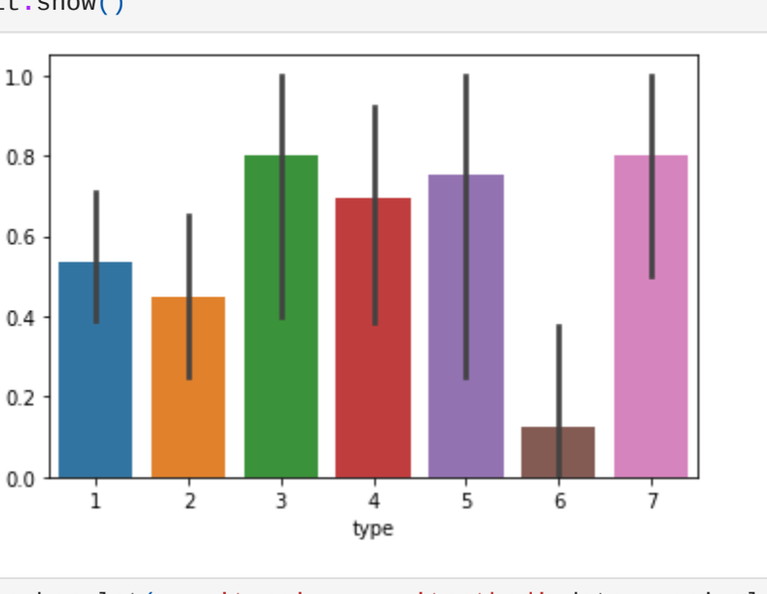
In [14]:
```
sns.barplot(x = 'type' , y = 'airborne',data = animals_data)
plt.show()
```



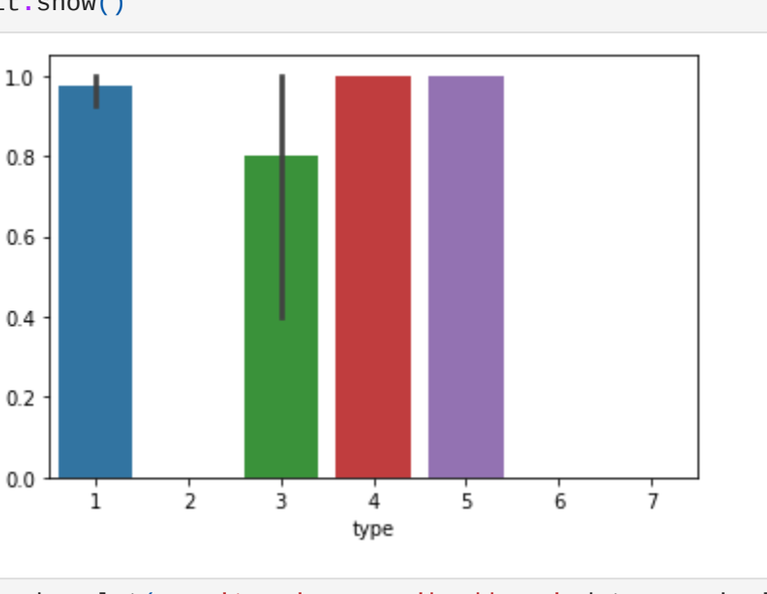
In [15]:
```
sns.barplot(x = 'type' , y = 'aquatic',data = animals_data)
plt.show()
```



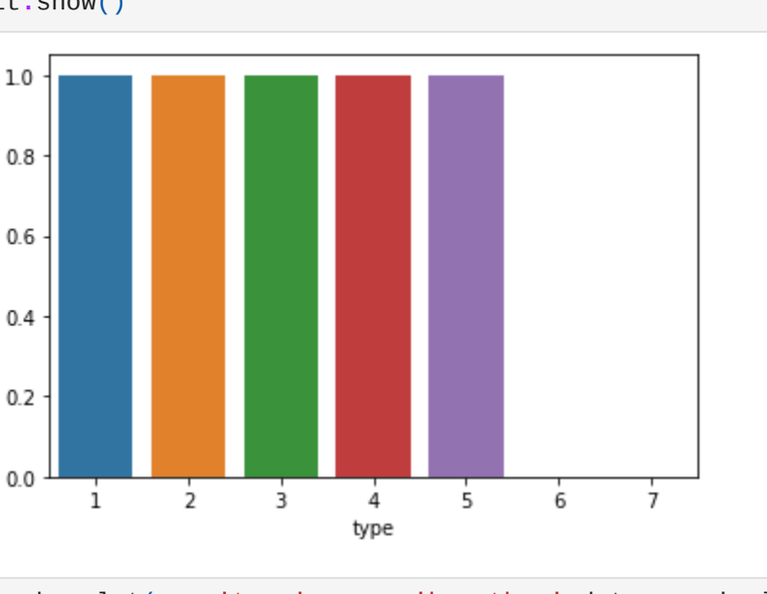
In [16]:
```
sns.barplot(x = 'type' , y = 'predator',data = animals_data)
plt.show()
```



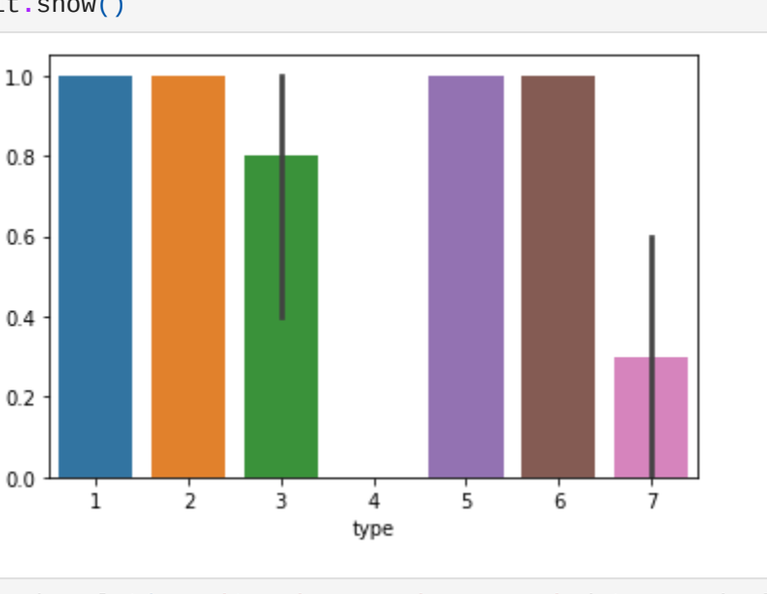
In [17]:
```
sns.barplot(x = 'type' , y = 'toothed',data = animals_data)
plt.show()
```



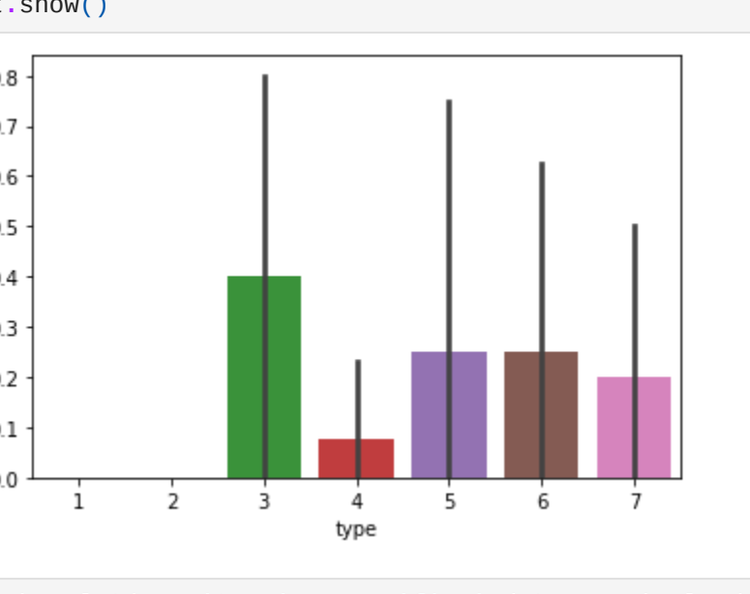
In [18]:
```
sns.barplot(x = 'type' , y = 'backbone',data = animals_data)
plt.show()
```



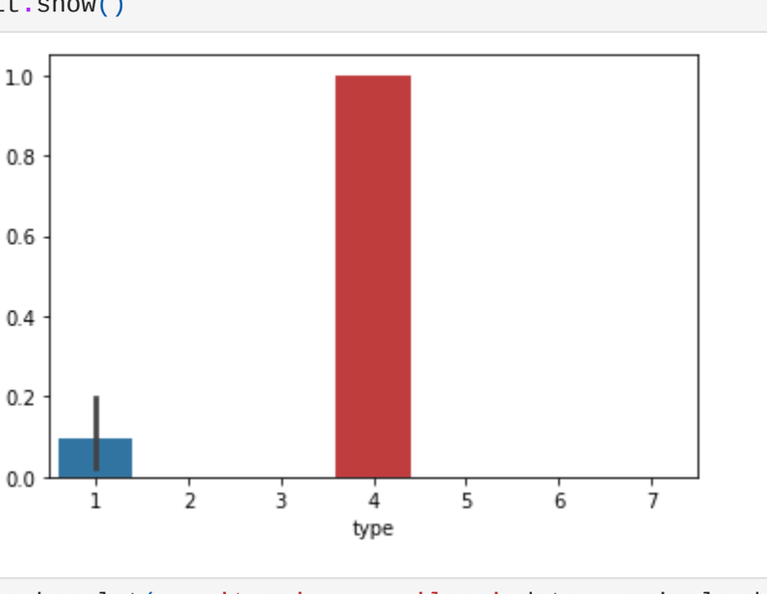
In [19]:
```
sns.barplot(x = 'type' , y = 'breathes',data = animals_data)
plt.show()
```



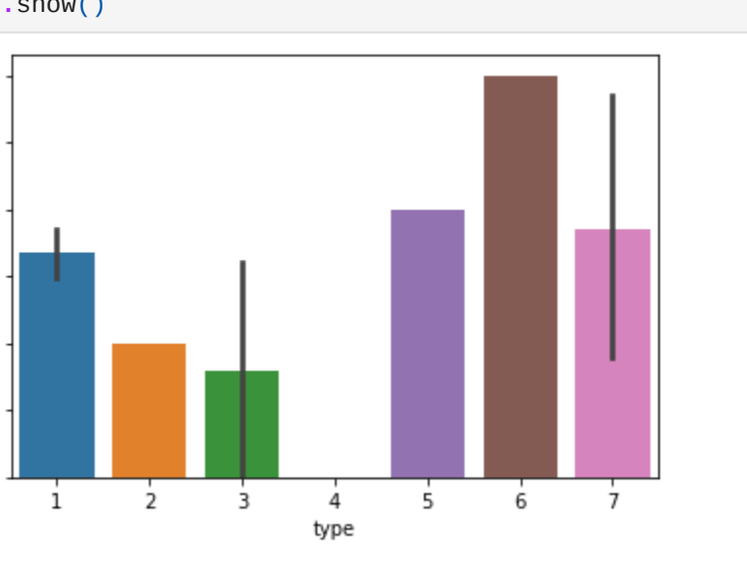
In [20]:
```
sns.barplot(x = 'type' , y = 'venomous',data = animals_data)
plt.show()
```



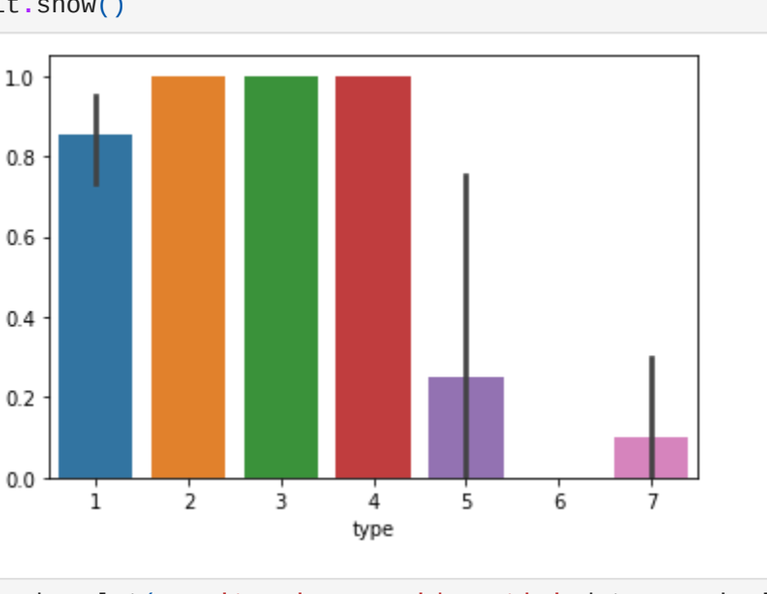
In [21]:
```
sns.barplot(x = 'type' , y = 'fins',data = animals_data)
plt.show()
```



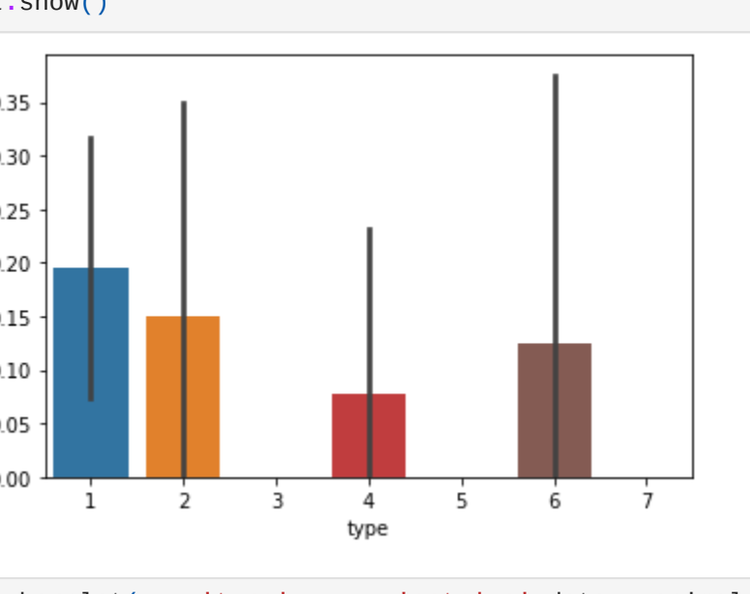
In [22]:
```
sns.barplot(x = 'type' , y = 'legs',data = animals_data)
plt.show()
```



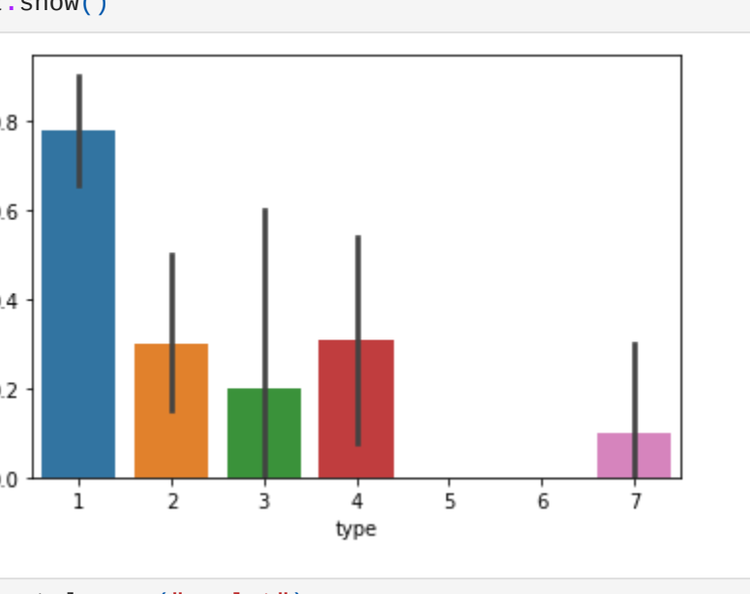
In [23]:
```
sns.barplot(x = 'type' , y = 'tail',data = animals_data)
plt.show()
```



In [24]:
```
sns.barplot(x = 'type' , y = 'domestic',data = animals_data)
plt.show()
```



In [25]:
```
sns.barplot(x = 'type' , y = 'catsize',data = animals_data)
plt.show()
```



In [26]:
```
plt.style.use("ggplot")
fig, ax = plt.subplots(4,4, figsize = (20,25))

sns.countplot(animals_data['hair'], palette = 'viridis', ax = ax[0,0])
sns.countplot(animals_data['feathers'], palette = 'viridis', ax = ax[0,1])
sns.countplot(animals_data['eggs'], palette = 'viridis', ax = ax[0,2])
sns.countplot(animals_data['milk'], palette = 'viridis', ax = ax[0,3])

sns.countplot(animals_data['airborne'], palette = 'viridis', ax = ax[1,0])
sns.countplot(animals_data['aquatic'], palette = 'viridis', ax = ax[1,1])
sns.countplot(animals_data['predator'], palette = 'viridis', ax = ax[1,2])
sns.countplot(animals_data['toothed'], palette = 'viridis', ax = ax[1,3])

sns.countplot(animals_data['backbone'], palette = 'viridis', ax = ax[2,0])
sns.countplot(animals_data["breathes"], palette = 'viridis', ax = ax[2,1])
sns.countplot(animals_data["venomous"], palette = 'viridis', ax = ax[2,2])
sns.countplot(animals_data["fins"], palette = 'viridis', ax = ax[2,3])

sns.countplot(animals_data['legs'], palette = 'viridis', ax = ax[3,0])
sns.countplot(animals_data['tail'], palette = 'viridis', ax = ax[3,1])
sns.countplot(animals_data['domestic'], palette = 'viridis', ax = ax[3,2]);
sns.countplot(animals_data['catsize'], palette = 'viridis', ax = ax[3,3]);
```

5. Train Test Split :

In [27]:
```
from sklearn.model_selection import train_test_split
```

In [28]:
```
animals_data_df = animals_data.drop('animal name', axis = 1)
animals_data_df.head()
```

Out[28]:

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

In [29]:
```
X = animals_data_df.drop('type', axis = 1)
y = animals_data_df['type']
```

In [30]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state = 0)
```

In [31]:
```
X_train.shape,X_test.shape
```

Out[31]:
```
((75, 16), (26, 16))
```

In [32]:
```
y_test.shape,y_train.shape
```

Out[32]:
```
((26,), (75,))
```

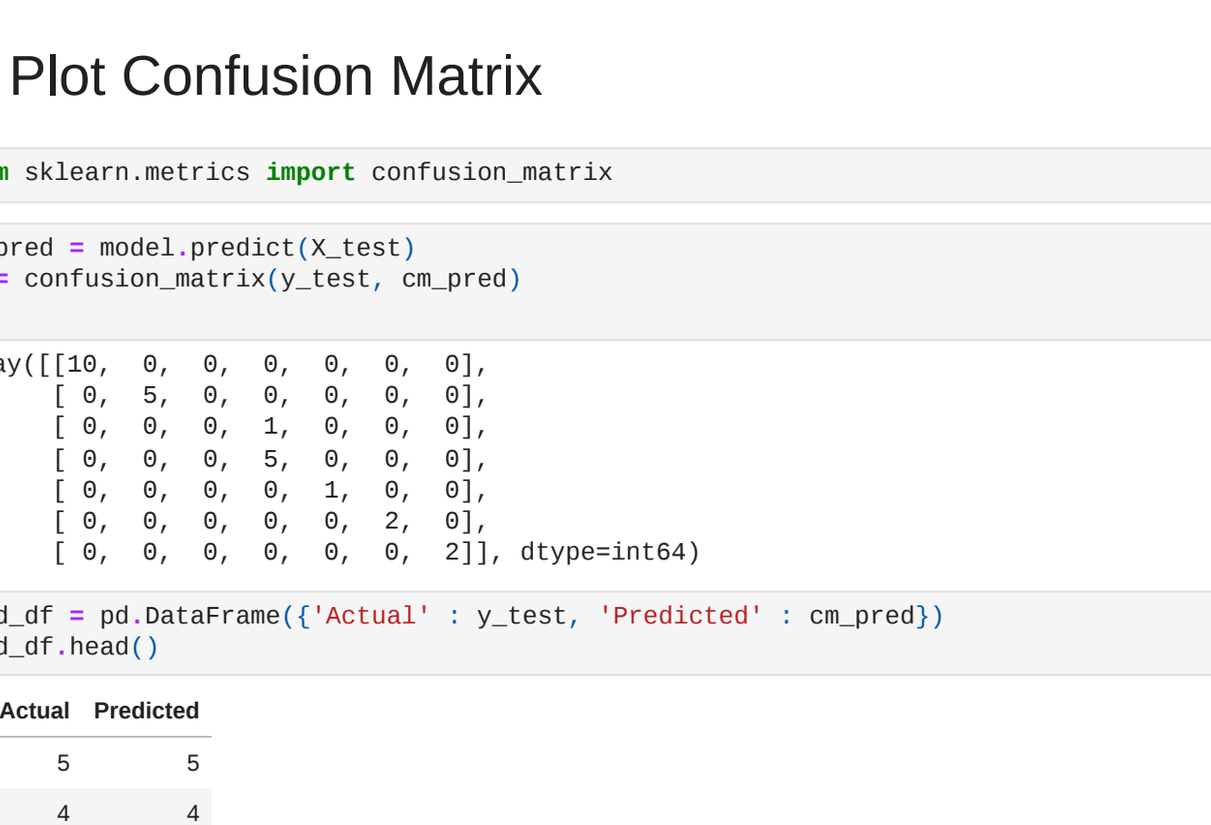
6. KNN (K Neighrest Neighbour Classifier)

In [33]:
```
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
```

In [34]:
```
# choose k between 1 to 41
k_range = range(1, 31)
k_scores = []

for k in k_range:
    knn = KNeighborsClassifier(n_neighbors = k)
    train_scores = cross_val_score(knn, X_train, y_train, cv = 5)
    k_scores.append(train_scores.mean())

plt.figure(figsize = (10,6))
plt.plot(k_range, k_scores,marker = "o")
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```



Model :

In [35]:
```
model = KNeighborsClassifier(n_neighbors = 5)
model
```

Out[35]:
```
KNeighborsClassifier()
```

In [36]:
```
model_pred = model.fit(X_train,y_train).predict(X_train)
model_pred
```

Out[36]:
```
array([1, 2, 4, 4, 4, 4, 5, 1, 6, 1, 4, 2, 1, 1, 1, 4, 6, 2, 1, 7, 1, 6,
       4, 1, 4, 2, 1, 4, 1, 1, 2, 1, 2, 5, 1, 1, 1, 2, 2, 7, 1, 1, 7, 1,
       1, 1, 1, 1, 7, 1, 2, 2, 7, 7, 5, 2, 7, 7, 6, 1, 2, 4, 5, 6, 1, 2,
       1, 2, 2, 1, 6, 1, 1, 1, 1], dtype=int64)
```

In [37]:
```
model_accuracy = model.score(X_test, y_test)
model_accuracy
print('Model accuracy is:',model_accuracy)
```

```
Model accuracy is: 0.9615384615384616
```

7. Plot Confusion Matrix

In [38]:
```
from sklearn.metrics import confusion_matrix
```

In [39]:
```
cm_pred = model.predict(X_test)
cm = confusion_matrix(y_test, cm_pred)
cm
```

Out[39]:
```
array([[10,  0,  0,  0,  0,  0,  0],
       [ 0,  5,  0,  0,  0,  0,  0],
       [ 0,  0,  0,  1,  0,  0,  0],
       [ 0,  0,  0,  5,  0,  0,  0],
       [ 0,  0,  0,  0,  1,  0,  0],
       [ 0,  0,  0,  0,  0,  2,  0],
       [ 0,  0,  0,  0,  0,  0,  2]], dtype=int64)
```

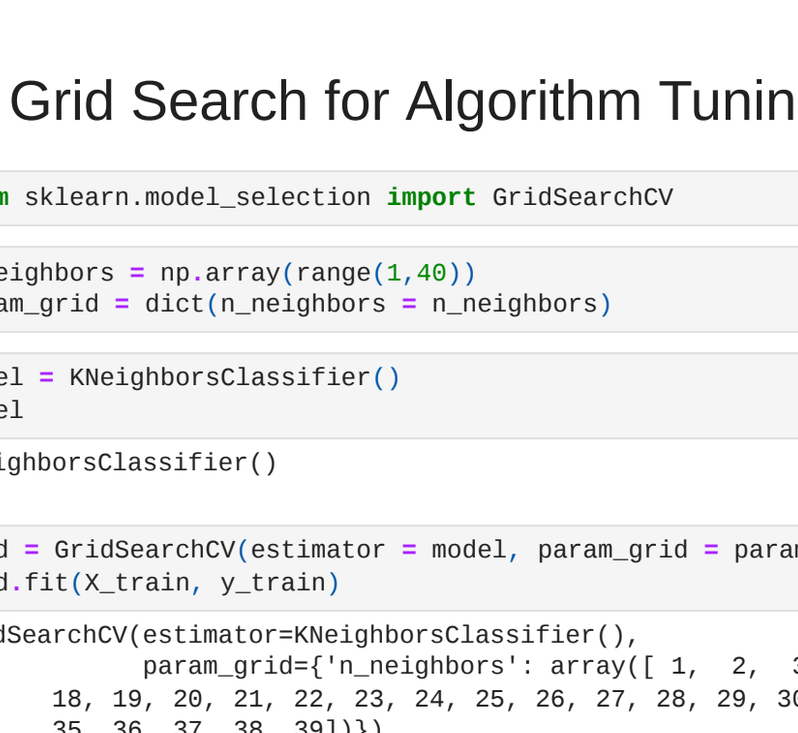
In [40]:
```
pred_df = pd.DataFrame({'Actual' : y_test, 'Predicted' : cm_pred})
pred_df.head()
```

Out[40]:

	Actual	Predicted
26	5	5
86	4	4
2	4	4
55	1	1
75	1	1

In [41]:
```
plt.figure(figsize = (7,5))

sns.heatmap(cm, annot = True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()
```



8. Grid Search for Algorithm Tuning

In [42]:
```
from sklearn.model_selection import GridSearchCV
```

In [43]:
```
n_neighbors = np.array(range(1,40))
param_grid = dict(n_neighbors = n_neighbors)
```

In [44]:
```
model = KNeighborsClassifier()
model
```

Out[44]:
```
KNeighborsClassifier()
```

In [45]:
```
grid = GridSearchCV(estimator = model, param_grid = param_grid)
grid.fit(X_train, y_train)
```

Out[45]:
```
GridSearchCV(estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39])})
```

In [46]:
```
print('The Best Score Accuracy :',grid.best_score_)
print('The Best Parameter  :',grid.best_params_)
```

```
The Best Score Accuracy : 0.9333333333333333
The Best Parameter  : {'n_neighbors': 1}
```

Conclusion :

```
a) For KNN Model Accuracy of Animals  : 0.9615

b) The Grid Best Score Accuracy       : 0.9333
```