In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
```

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

In [3]:
```
zoo
```

Out[3]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	ca
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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
96	wallaby	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	
97	wasp	1	0	1	0	1	0	0	0	0	1	1	0	6	0	0	
98	wolf	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	
99	worm	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	
100	wren	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	

101 rows × 18 columns

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

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

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

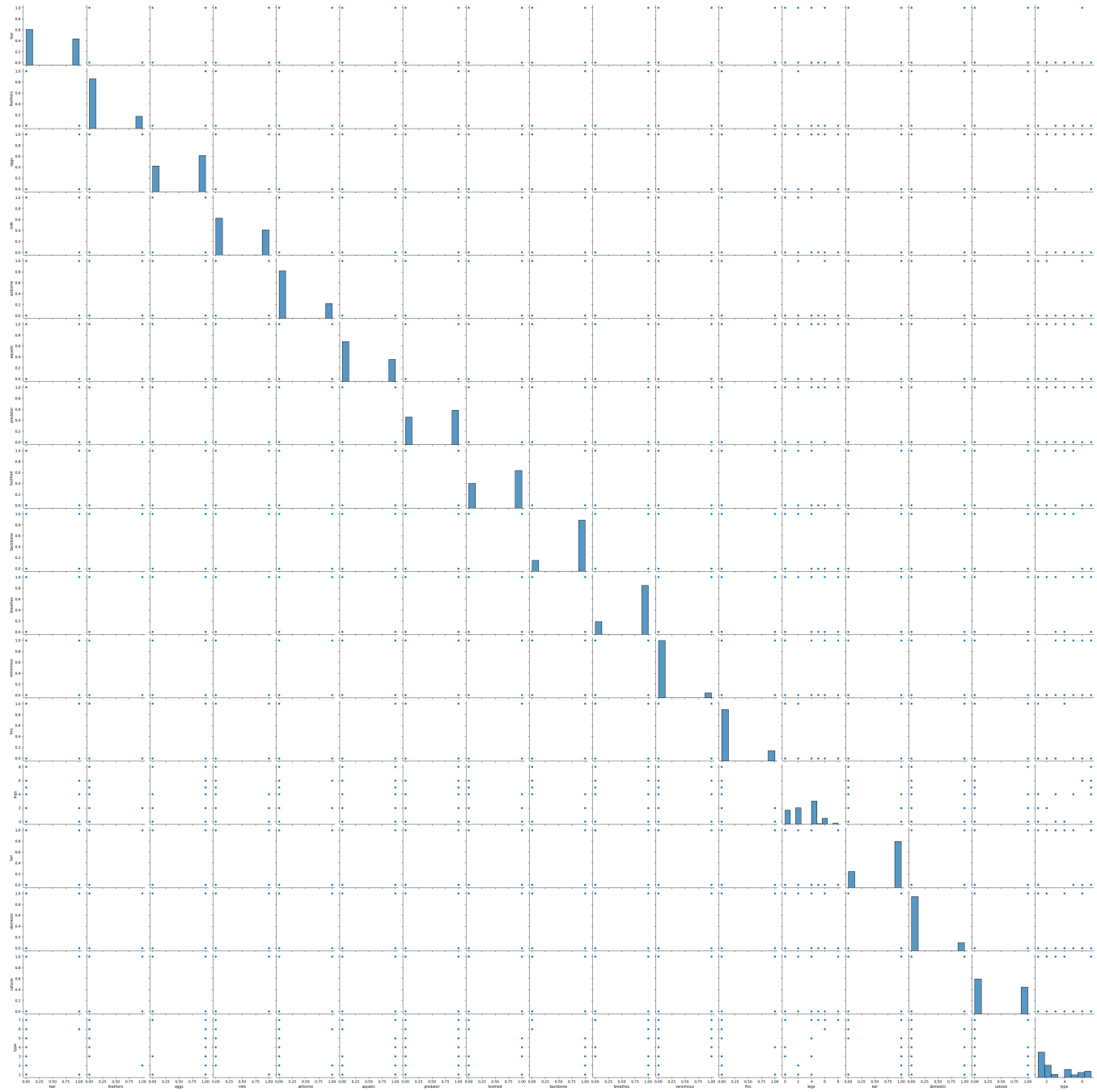
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101 entries, 0 to 100
Data columns (total 18 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   animal name  101 non-null    object
 1   hair         101 non-null    int64 
 2   feathers     101 non-null    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]:
```
zoo.duplicated()
```

Out[6]:
```
0      False
1      False
2      False
3      False
4      False
       ...
96     False
97     False
98     False
99     False
100    False
Length: 101, dtype: bool
```

In [7]:
```
sns.pairplot(zoo)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x2601c8cdeb0>



In [8]:
```
sns.heatmap(zoo.isnull(),cmap='Blues')
```

Out[8]: <AxesSubplot:>

In [9]:
```
zoo=zoo.drop("animal name",axis=1)
```

In [10]:
```
array = zoo.values
X = array[:, 0:16]
Y = array[:, 16]
```

In [11]:
```
X
```

Out[11]:
```
array([[1, 0, 0, ..., 0, 0, 1],
       [1, 0, 0, ..., 1, 0, 1],
       [0, 0, 1, ..., 1, 0, 0],
       ...,
       [1, 0, 0, ..., 1, 0, 1],
       [0, 0, 1, ..., 0, 0, 0],
       [0, 1, 1, ..., 1, 0, 0]], dtype=int64)
```

In [12]:
```
Y
```

Out[12]:
```
array([1, 1, 4, 1, 1, 1, 1, 4, 4, 1, 1, 2, 4, 7, 7, 7, 2, 1, 4, 1, 2, 2,
       1, 2, 6, 5, 5, 1, 1, 1, 6, 1, 1, 2, 4, 1, 1, 2, 4, 6, 6, 2, 6, 2,
       1, 1, 7, 1, 1, 1, 1, 6, 5, 7, 1, 1, 2, 2, 2, 2, 4, 4, 3, 1, 1, 1,
       1, 1, 1, 1, 1, 2, 7, 4, 1, 1, 3, 7, 2, 2, 3, 7, 4, 2, 1, 7, 4, 2,
       6, 5, 3, 3, 4, 1, 1, 2, 1, 6, 1, 7, 2], dtype=int64)
```

In [13]:
```
num_folds = 20
kfold = KFold(n_splits=20)
```

In [16]:
```
model = KNeighborsClassifier(n_neighbors=20)
results = cross_val_score(model, X, Y, cv=kfold)
import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [17]:
```
print(results.mean())
```

```
0.78
```

In [18]:
```
from sklearn.model_selection import GridSearchCV
import pandas as pd
import numpy
```

In [20]:
```
n_neighbors = numpy.array(range(1,40))
param_grid = dict(n_neighbors=n_neighbors)
```

In [21]:
```
model = KNeighborsClassifier()
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid.fit(X, Y)
```

```
C:\Users\ROHIT\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:676: UserWarning: The least populated class in y has only 4 members, which is less than n_splits=5.
  warnings.warn(
```

Out[21]:
```
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 [22]:
```
print(grid.best_score_)
print(grid.best_params_)
```

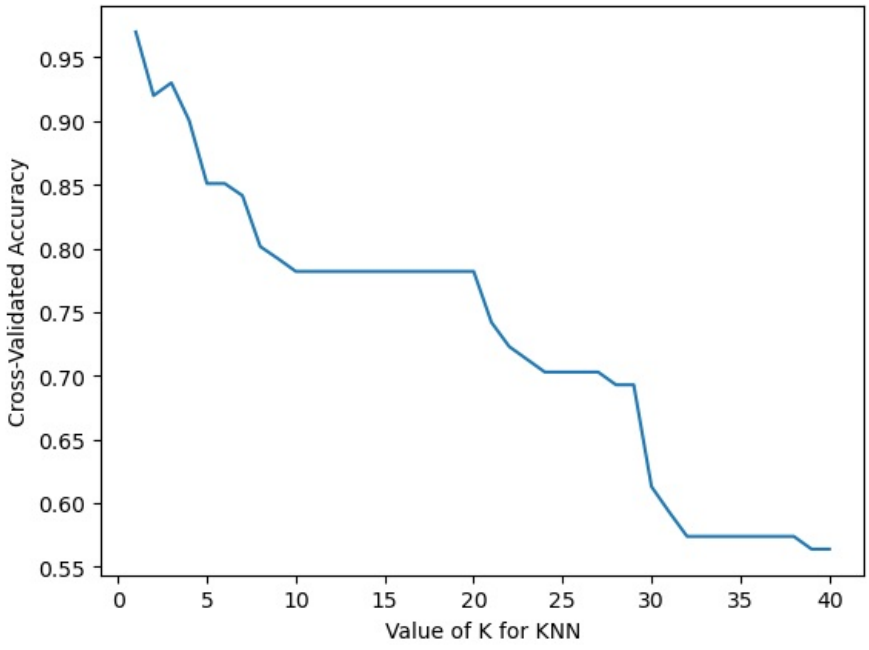
```
0.97
{'n_neighbors': 1}
```

In [23]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [24]:
```
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
# choose k between 1 to 41
k_range = range(1, 41)
k_scores = []
```

In [25]:
```
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, Y, cv=5)
    k_scores.append(scores.mean())
```

In [26]:
```
plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```



In [ ]: