In [1]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
import pandas as pd
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
```

In [3]:
```
df = pd.read_csv("Zoo.csv")
df
```

Out[3]:

	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	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
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]:
```
df.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 [5]:
```
df.describe().round(2).style.background_gradient(cmap = 'Oranges')
```

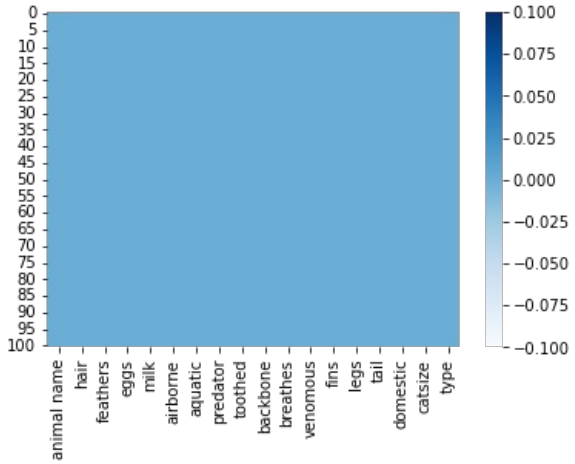
Out[5]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	
count	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	10
mean	0.430000	0.200000	0.580000	0.410000	0.240000	0.360000	0.550000	0.600000	0.820000	0.790000	0.080000	
std	0.500000	0.400000	0.500000	0.490000	0.430000	0.480000	0.500000	0.490000	0.380000	0.410000	0.270000	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	
50%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000000	

75%	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

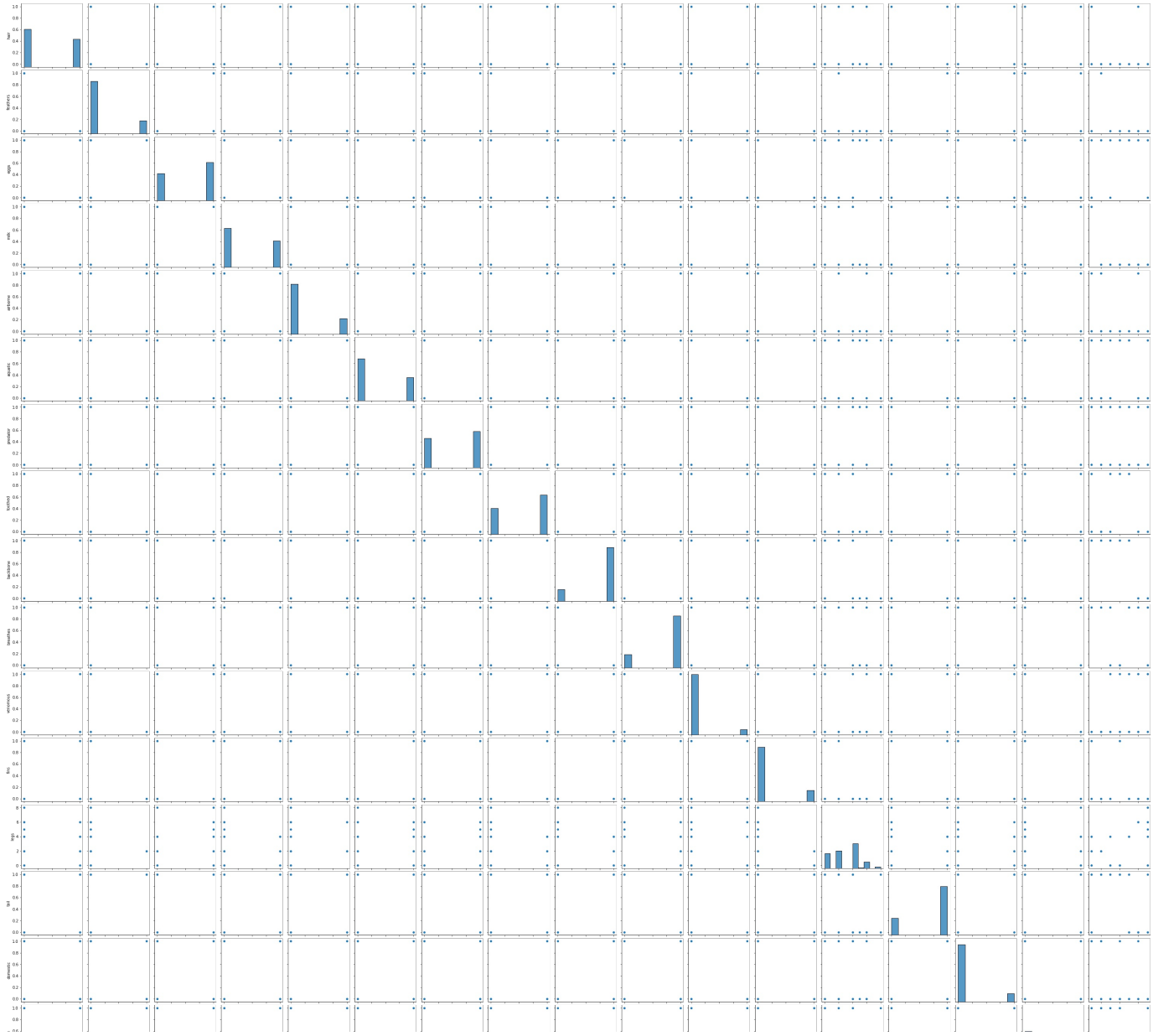
In [6]:
```
import seaborn as sns
sns.heatmap(df.isnull(),cmap='Blues')
```

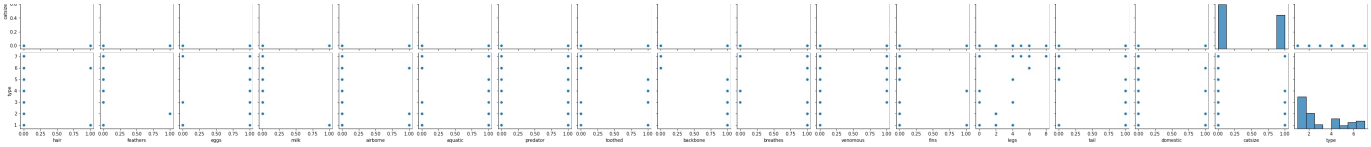
Out[6]:
```
<AxesSubplot:>
```



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

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x1e91c14af70>
```





In [8]:
```
df.duplicated()
```

Out[8]:
```
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 [9]:
```
data = df.drop("animal name",axis=1)
```

In [10]:
```
data
```

Out[10]:

	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
96	1	0	0	1	0	0	0	1	1	1	0	0	2	1	0	1	1
97	1	0	1	0	1	0	0	0	0	1	1	0	6	0	0	0	6
98	1	0	0	1	0	0	1	1	1	1	0	0	4	1	0	1	1
99	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	7
100	0	1	1	0	1	0	0	0	1	1	0	0	2	1	0	0	2

101 rows × 17 columns

In [11]:
```
array = data.values
x= array[:,0:16]
y= array[:,16]
```

In [12]:
```
x
```

Out[12]:
```
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 [13]:
```
y
```

Out[13]:
```
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 [14]:
```
x = (x-x.min(axis=0))/(x.max(axis=0))-x.min(axis=0)
```

In [15]:
```
x
```

Out[15]:
```
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.]])
```

In [16]:
```
num_folds = 30
KFold = KFold(n_splits=30)
```

In [17]:
```
model=KNeighborsClassifier(n_neighbors=20)
results=cross_val_score(model,x,y,cv=KFold)
```

In [18]:
```
print(results.mean())
```

```
0.8111111111111113
```

grid search for algorithm tunning

In [19]:
```
import numpy
from pandas import read_csv
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

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

In [21]:
```
n_neighbors
```

Out[21]:
```
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]:
```
model = KNeighborsClassifier()
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid.fit(x, y)
```

Out[22]:
```
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])})
```

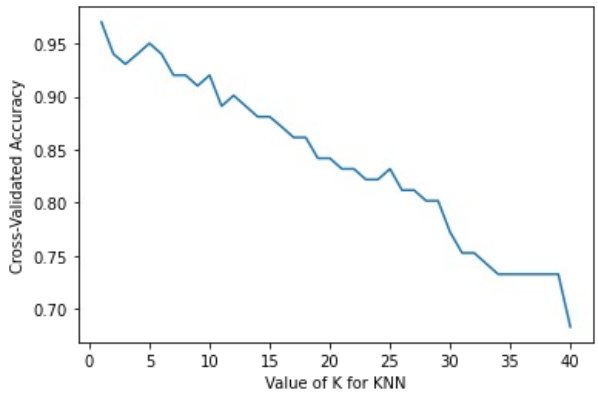
In [23]:
```
print(grid.best_score_)
print(grid.best_params_)
```

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

Visuvilazing the cv results

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

```
k_scores = []
# use iteration to caclulator different k in models, then return the average accuracy based on the cross validati
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, x, y, cv=5)
    k_scores.append(scores.mean())
# plot to see clearly
plt.plot(k_range, k_scores)
plt.xlabel('Value of K for KNN')
plt.ylabel('Cross-Validated Accuracy')
plt.show()
```



hence from the above graph we casn see that best value for k was 1 which gave us classification accuracy as 0.97

In [ ]: