In [1]:

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

In [3]:

```
zoo = pd.read_csv('C:/Users/acer/Sandesh Pal/Data Science Assgn/KNN/Zoo.csv')
```

In [4]:

```
zoo.head()
```

Out[4]:

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

In [5]:

```
# checking for null values
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.describe()
```

Out[6]:

	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
mean	0.425743	0.198020	0.584158	0.405941	0.237624	0.356436	0.554455	0.603960	0.821782	0.792079	0.079208
std	0.496921	0.400495	0.495325	0.493522	0.427750	0.481335	0.499505	0.491512	0.384605	0.407844	0.271410
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 [7]:

```
X = zoo.drop(['type','animal name'], axis =1)
X.head()
```

Out[7]:

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

In [8]:

```
Y = zoo['type']
Y
```

Out[8]:

```
0      1
1      1
2      4
3      1
4      1
      ..
96     1
97     6
98     1
99     7
100    2
Name: type, Length: 101, dtype: int64
```

In [9]:

```
# Let's use Grid search CV to find out best value for K
```

In [10]:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

In [11]:

```
n_neighbors = np.array(range(1,40))
param_grid = dict(n_neighbors=n_neighbors)
```

In [12]:

```
model = KNeighborsClassifier()
grid = GridSearchCV(estimator=model, param_grid=param_grid)
grid.fit(X, Y)
```

Out[12]:

```
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 [13]:

```
print(grid.best_score_)
print(grid.best_params_)
```

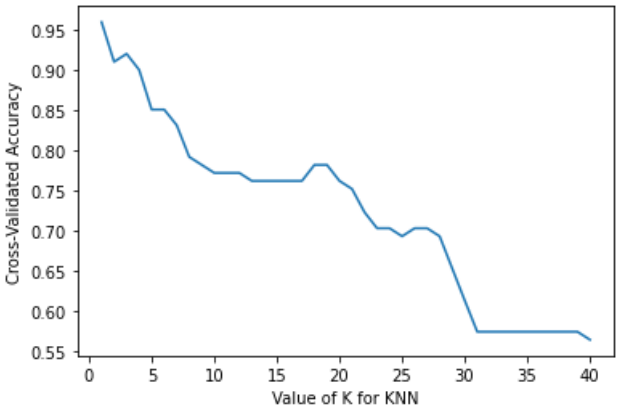
```
0.96
{'n_neighbors': 1}
```

In [14]:

```
# Visualizing the CV results
```

In [15]:

```
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
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()
```



In [16]:

```
#KNN Classification
num_folds = 30
kfold = KFold(n_splits=20)
model = KNeighborsClassifier(n_neighbors=1)
results = cross_val_score(model, X, Y, cv=kfold)
print(results.mean())
```

```
0.9800000000000001
```

In [ ]: