In [19]:

```
# KNN Classification
from pandas import read_csv
import numpy as np
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn.neighbors import KNeighborsClassifier
```

In [23]:

```
zoo = read_csv('C:/Users/Hp/Downloads/Zoo.csv')
```

In [24]:

```
zoo.head(10)
```

Out[24]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breath
0	aardvark	1	0	0	1	0	0	1	1	1	
1	antelope	1	0	0	1	0	0	0	1	1	
2	bass	0	0	1	0	0	1	1	1	1	
3	bear	1	0	0	1	0	0	1	1	1	
4	boar	1	0	0	1	0	0	1	1	1	
5	buffalo	1	0	0	1	0	0	0	1	1	
6	calf	1	0	0	1	0	0	0	1	1	
7	carp	0	0	1	0	0	1	0	1	1	
8	catfish	0	0	1	0	0	1	1	1	1	
9	cavy	1	0	0	1	0	0	0	1	1	

In [25]:

```
zoo.shape
```

Out[25]:

```
(101, 18)
```

In [26]:

```
zoo.info
```

Out[26]:

```
<bound method DataFrame.info of     animal name  hair  feathers  eggs  milk
airborne  aquatic  predator  \
0      aardvark     1         0     0     1         0         0         1
1      antelope     1         0     0     1         0         0         0
2          bass     0         0     1     0         0         1         1
3          bear     1         0     0     1         0         0         1
4          boar     1         0     0     1         0         0         1
..          ...   ...       ...   ...   ...       ...       ...       ...
96      wallaby     1         0     0     1         0         0         0
97         wasp     1         0     1     0         1         0         0
98         wolf     1         0     0     1         0         0         1
99         worm     0         0     1     0         0         0         0
100        wren     0         1     1     0         1         0         0

     toothed  backbone  breathes  venomous  fins  legs  tail  domestic  \
0          1         1         1         0     0     4     0         0
1          1         1         1         0     0     4     1         0
2          1         1         0         0     1     0     1         0
3          1         1         1         0     0     4     0         0
4          1         1         1         0     0     4     1         0
..       ...       ...       ...       ...   ...   ...   ...       ...
96         1         1         1         0     0     2     1         0
97         0         0         1         1     0     6     0         0
98         1         1         1         0     0     4     1         0
99         0         0         1         0     0     0     0         0
100        0         1         1         0     0     2     1         0

     catsize  type
0          1     1
1          1     1
2          0     4
3          1     1
4          1     1
..       ...   ...
96         1     1
97         0     6
98         1     1
99         0     7
100        0     2

[101 rows x 18 columns]>
```

In [27]:

```
zoo.describe
```

Out[27]:

```
<bound method NDFrame.describe of     animal name  hair  feathers  eggs  mil
k  airborne  aquatic  predator  \
0      aardvark     1         0     0     1         0        0         1
1      antelope     1         0     0     1         0        0         0
2          bass     0         0     1     0         0        1         1
3          bear     1         0     0     1         0        0         1
4          boar     1         0     0     1         0        0         1
..          ...   ...       ...   ...   ...       ...      ...       ...
96      wallaby     1         0     0     1         0        0         0
97         wasp     1         0     1     0         1        0         0
98         wolf     1         0     0     1         0        0         1
99         worm     0         0     1     0         0        0         0
100        wren     0         1     1     0         1        0         0

     toothed  backbone  breathes  venomous  fins  legs  tail  domestic  \
0          1         1         1         0     0     4     0         0
1          1         1         1         0     0     4     1         0
2          1         1         0         0     1     0     1         0
3          1         1         1         0     0     4     0         0
4          1         1         1         0     0     4     1         0
..       ...       ...       ...       ...   ...   ...   ...       ...
96         1         1         1         0     0     2     1         0
97         0         0         1         1     0     6     0         0
98         1         1         1         0     0     4     1         0
99         0         0         1         0     0     0     0         0
100        0         1         1         0     0     2     1         0

     catsize  type
0          1     1
1          1     1
2          0     4
3          1     1
4          1     1
..       ...   ...
96         1     1
97         0     6
98         1     1
99         0     7
100        0     2

[101 rows x 18 columns]>
```

In [28]:

```
## Preprocessing
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
Zoo["animal name"] = label_encoder.fit_transform(Zoo["animal name"])
```

In [29]:

```
Zoo
```

Out[29]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breath
0	0	1	0	0	1	0	0	1	1	1	
1	1	1	0	0	1	0	0	0	1	1	
2	2	0	0	1	0	0	1	1	1	1	
3	3	1	0	0	1	0	0	1	1	1	
4	4	1	0	0	1	0	0	1	1	1	
...	...	...	...	...	...	...	...	...	...	...	
96	95	1	0	0	1	0	0	0	1	1	
97	96	1	0	1	0	1	0	0	0	0	
98	97	1	0	0	1	0	0	1	1	1	
99	98	0	0	1	0	0	0	0	0	0	
100	99	0	1	1	0	1	0	0	0	1	

101 rows × 18 columns

In [30]:

```
array = Zoo.values
X = array[:, 1:17]
X
```

Out[30]:

```
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 [31]:

```
Y = array[:, -1]
Y
```

Out[31]:

```
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 [32]:

```
kfold = KFold(n_splits=4)
```

In [33]:

```
model = KNeighborsClassifier(n_neighbors=13)
results = cross_val_score(model, X, Y, cv=kfold)
```

In [34]:

```
print(results.mean())
```

```
0.751923076923077
```

In [36]:

```
 # Grid Search for Algorithm Tuning
import numpy
from pandas import read_csv
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import GridSearchCV
```

In [37]:

```
n_neighbors1 = numpy.array(range(1,40))
param_grid = dict(n_neighbors=n_neighbors1)
```

In [38]:

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

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:66
6: UserWarning: The least populated class in y has only 4 members, which is
less than n_splits=5.
  warnings.warn(("The least populated class in y has only %d"
```

Out[38]:

```
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 [39]:

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

```
0.97
```

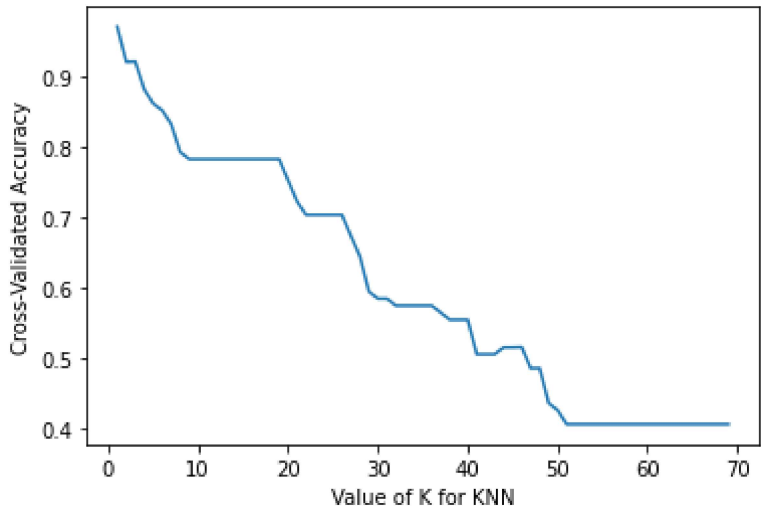
In [40]:

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

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

In [41]:

```
 ## Visualizing the CV results
import matplotlib.pyplot as plt
get_ipython().run_line_magic('matplotlib', 'inline')
# choose k between 1 to 70
k_range = range(1, 70)
k_scores = []
# use iteration to caclulator different k in models, then return the average accuracy based
for k in k_range:
    knn = KNeighborsClassifier(n_neighbors=k)
    scores = cross_val_score(knn, X, Y, cv=4)
    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 [ ]: