In [39]:
```
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import KFold,cross_val_score
import seaborn as sns

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

In [40]:
```
data = pd.read_csv('Zoo.csv')
data
```

Out[40]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes
0	aardvark	1	0	0	1	0	0	1	1	1	1
1	antelope	1	0	0	1	0	0	0	1	1	1
2	bass	0	0	1	0	0	1	1	1	1	0
3	bear	1	0	0	1	0	0	1	1	1	1
4	boar	1	0	0	1	0	0	1	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...
96	wallaby	1	0	0	1	0	0	0	1	1	1
97	wasp	1	0	1	0	1	0	0	0	0	1
98	wolf	1	0	0	1	0	0	1	1	1	1
99	worm	0	0	1	0	0	0	0	0	0	1
100	wren	0	1	1	0	1	0	0	0	1	1

101 rows × 18 columns

In [41]:
```
data.describe()
```

Out[41]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	toc
count	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	101.00
mean	0.425743	0.198020	0.584158	0.405941	0.237624	0.356436	0.554455	0.60
std	0.496921	0.400495	0.495325	0.493522	0.427750	0.481335	0.499505	0.49
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00
50%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	1.00
75%	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.00
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00

In [42]:
```
data.shape
```

Out[42]: (101, 18)

In [43]:
```
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 [44]:
```
data.dtypes
```

Out[44]:
```
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
```

In [45]:
```
correlation = data.corr()
correlation
```

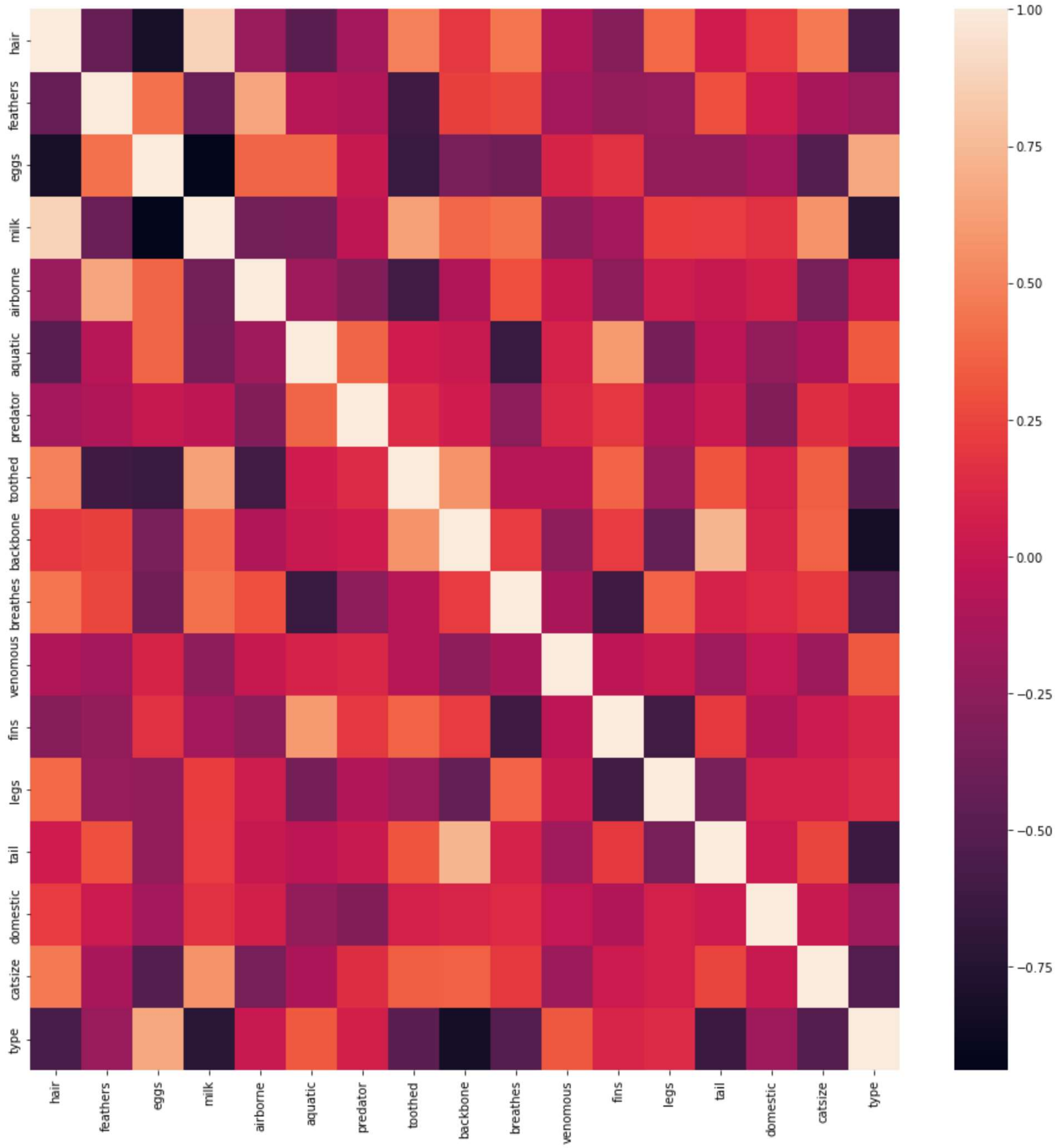
Out[45]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed
hair	1.000000	-0.427851	-0.817382	0.878503	-0.198431	-0.473554	-0.154769	0.492531
feathers	-0.427851	1.000000	0.419248	-0.410761	0.656553	-0.058552	-0.104430	-0.613631
eggs	-0.817382	0.419248	1.000000	-0.938848	0.376646	0.376244	0.011605	-0.642150
milk	0.878503	-0.410761	-0.938848	1.000000	-0.366765	-0.362613	-0.029721	0.628168
airborne	-0.198431	0.656553	0.376646	-0.366765	1.000000	-0.172638	-0.295181	-0.594311
aquatic	-0.473554	-0.058552	0.376244	-0.362613	-0.172638	1.000000	0.375978	0.053150
predator	-0.154769	-0.104430	0.011605	-0.029721	-0.295181	0.375978	1.000000	0.129452
toothed	0.492531	-0.613631	-0.642150	0.628168	-0.594311	0.053150	0.129452	1.000000
backbone	0.191681	0.231403	-0.340420	0.384958	-0.104718	0.022463	0.051022	0.575085
breathes	0.441149	0.254588	-0.382777	0.423527	0.286039	-0.637506	-0.262931	-0.065690
venomous	-0.104245	-0.145739	0.098689	-0.242449	0.008528	0.087915	0.115391	-0.062344
fins	-0.280313	-0.223541	0.164796	-0.156328	-0.251157	0.604492	0.190302	0.364292
legs	0.394009	-0.206686	-0.224918	0.214196	0.043712	-0.360638	-0.099723	-0.193476
tail	0.048973	0.292569	-0.221090	0.210026	0.009482	-0.034642	0.018947	0.310368
domestic	0.207208	0.031586	-0.155610	0.163928	0.063274	-0.224308	-0.309794	0.069430
catsize	0.455020	-0.135934	-0.514650	0.574906	-0.349768	-0.111866	0.144790	0.344010
type	-0.562384	-0.197520	0.661825	-0.723683	0.022677	0.326639	0.061179	-0.471527

In [46]:

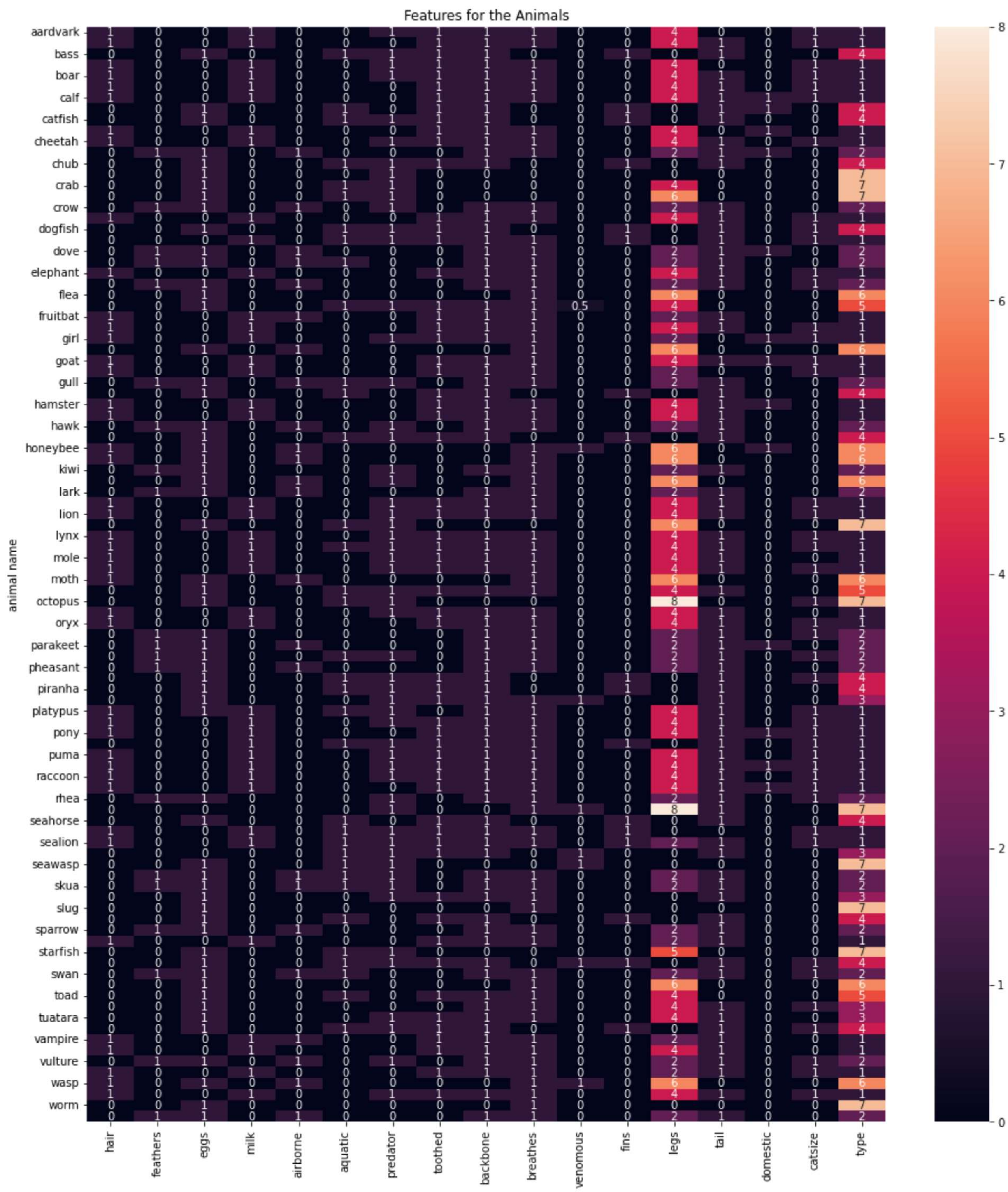
```
plt.figure(figsize=(16,16))
sns.heatmap(correlation)
```

Out[46]: <AxesSubplot:>



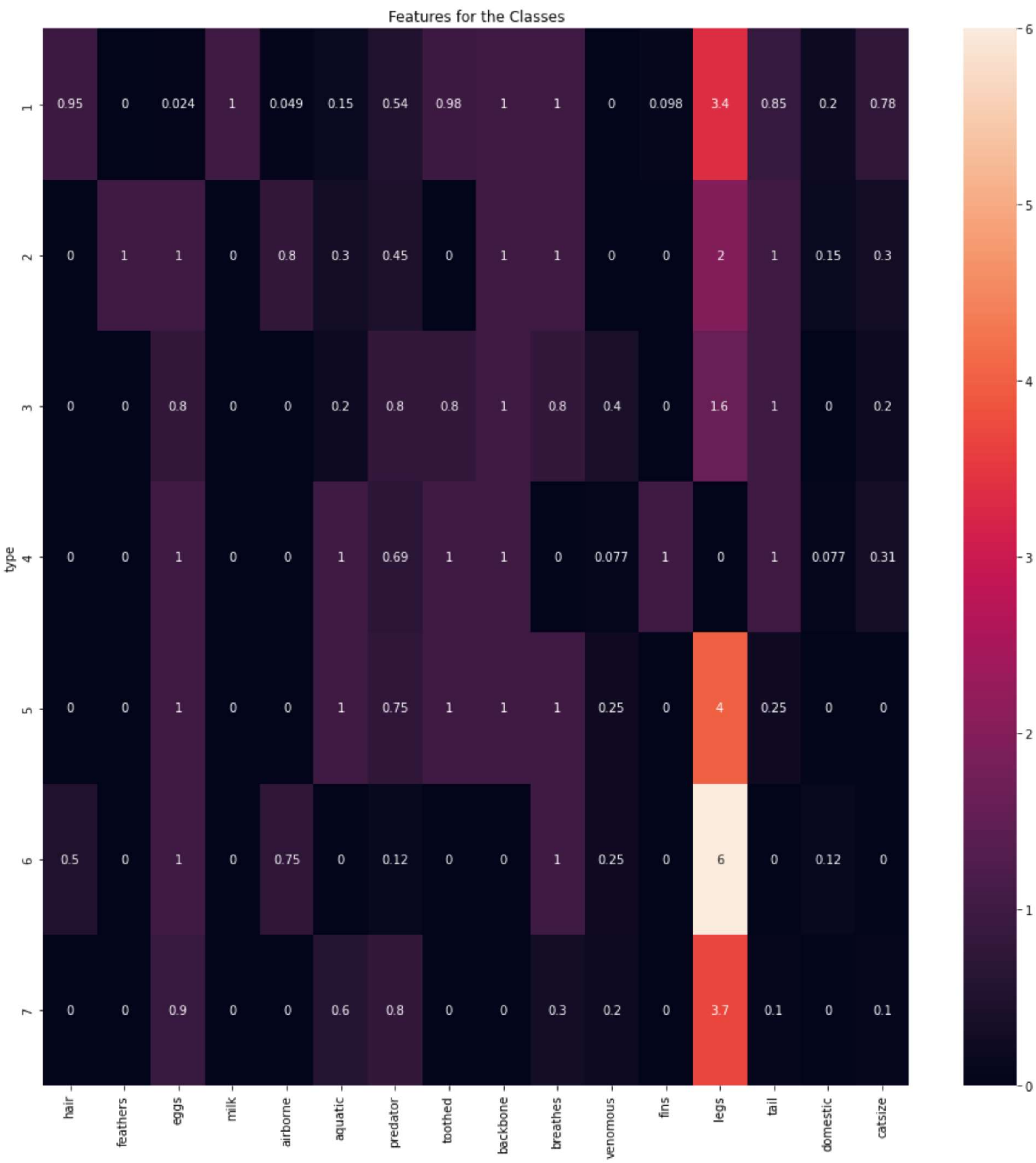
In [47]:
```
data_temp = data.groupby(by = 'animal name').mean()
plt.figure(figsize=(16,18))
sns.heatmap(data_temp,annot=True)
plt.title('Features for the Animals')
```

Out[47]: Text(0.5, 1.0, 'Features for the Animals')



In [48]:
```
data_temp = data.groupby(by='type').mean()
plt.figure(figsize=(16,16))
sns.heatmap(data_temp,annot = True)
plt.title('Features for the Classes')
```

Out[48]: `Text(0.5, 1.0, 'Features for the Classes')`



In [49]:
```
data_1 = data.drop('animal name',axis=1)
data_1
```

Out[49]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomc
0	1	0	0	1	0	0	1	1	1	1	
1	1	0	0	1	0	0	0	1	1	1	
2	0	0	1	0	0	1	1	1	1	0	
3	1	0	0	1	0	0	1	1	1	1	
4	1	0	0	1	0	0	1	1	1	1	
...	...	...	...	...	...	...	...	...	...	...	
96	1	0	0	1	0	0	0	1	1	1	
97	1	0	1	0	1	0	0	0	0	1	
98	1	0	0	1	0	0	1	1	1	1	
99	0	0	1	0	0	0	0	0	0	1	
100	0	1	1	0	1	0	0	0	1	1	

101 rows × 17 columns

In [50]:
```
x = data_1.drop('type',axis=1)
y = data_1[['type']]
```

In [51]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,random_state=0)
```

In [52]:
```
x_train.shape,y_train.shape
```

Out[52]: ((75, 16), (75, 1))

In [53]:
```
x_test.shape,y_test.shape
```

Out[53]: ((26, 16), (26, 1))

In [54]:
```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_train)
print('Accuracy score :',accuracy_score(y_train,y_pred))
```

```
Accuracy score : 0.9466666666666667
```

In [55]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_train)
print('Accuracy score :',accuracy_score(y_train,y_pred))
```

```
Accuracy score : 0.9466666666666667
```

In [56]:
```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_train)
print('Accuracy score :',accuracy_score(y_train,y_pred))
```

```
Accuracy score : 0.8533333333333334
```

In [57]:
```
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_train)
print('Accuracy score :',accuracy_score(y_train,y_pred))
```

```
Accuracy score : 0.8133333333333334
```

In [58]:
```
std_scalar = StandardScaler()
std_scalar = std_scalar.fit_transform(x)
x_scaled = pd.DataFrame(data = std_scalar,columns=x.columns)
x_scaled
```

Out[58]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backboı
0	1.161395	-0.496904	-1.185227	1.209717	-0.558291	-0.744208	0.896421	0.809776	0.465(
1	1.161395	-0.496904	-1.185227	1.209717	-0.558291	-0.744208	-1.115547	0.809776	0.465(
2	-0.861034	-0.496904	0.843721	-0.826640	-0.558291	1.343710	0.896421	0.809776	0.465(
3	1.161395	-0.496904	-1.185227	1.209717	-0.558291	-0.744208	0.896421	0.809776	0.465(
4	1.161395	-0.496904	-1.185227	1.209717	-0.558291	-0.744208	0.896421	0.809776	0.465(
...	...	...	...	...	...	...	...	...	
96	1.161395	-0.496904	-1.185227	1.209717	-0.558291	-0.744208	-1.115547	0.809776	0.465(
97	1.161395	-0.496904	0.843721	-0.826640	1.791182	-0.744208	-1.115547	-1.234909	-2.147:
98	1.161395	-0.496904	-1.185227	1.209717	-0.558291	-0.744208	0.896421	0.809776	0.465(
99	-0.861034	-0.496904	0.843721	-0.826640	-0.558291	-0.744208	-1.115547	-1.234909	-2.147:
100	-0.861034	2.012461	0.843721	-0.826640	1.791182	-0.744208	-1.115547	-1.234909	0.465(

101 rows × 16 columns

In [59]:
```
x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,random_state=0)
```

In [60]:
```
x_train.shape,y_train.shape
```

Out[60]:
```
((75, 16), (75, 1))
```

In [61]:
```
x_test.shape,y_test.shape
```

Out[61]:
```
((26, 16), (26, 1))
```

In [62]:
```
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_train)
print('Accuracy score :',accuracy_score(y_train,y_pred))
```

```
Accuracy score : 0.9733333333333334
```

In [63]:
```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_train)
print('Accuracy score :',accuracy_score(y_train,y_pred))
```

```
Accuracy score : 0.96
```

In [64]:
```
knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_train)
print('Accuracy score :',accuracy_score(y_train,y_pred))
```

```
Accuracy score : 0.8933333333333333
```

In [65]:
```
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(x_train,y_train)
y_pred = knn.predict(x_train)
print('Accuracy score :',accuracy_score(y_train,y_pred))
```

```
Accuracy score : 0.84
```

In [80]:

```
kfold = KFold(n_splits=5,shuffle=True,random_state=14)
cv_scores = []


for i in range(1,50,2):
    knn_model = KNeighborsClassifier(n_neighbors=i)
    cross_val_scores = cross_val_score(estimator = knn_model,X = x_scaled,y=y,cv=
    print(i,'th Iteration:\n',cross_val_scores.mean().round(4))
    cv_scores.append(cross_val_scores.mean().round(4))
```

```
1 th Iteration:
 0.96
3 th Iteration:
 0.9505
5 th Iteration:
 0.8905
7 th Iteration:
 0.861
9 th Iteration:
 0.8114
11 th Iteration:
 0.8019
13 th Iteration:
 0.8019
15 th Iteration:
 0.7919
17 th Iteration:
 0.7924
19 th Iteration:
 0.7924
21 th Iteration:
 0.7824
23 th Iteration:
 0.7824
25 th Iteration:
 0.7624
27 th Iteration:
 0.7433
29 th Iteration:
 0.7143
31 th Iteration:
 0.6652
33 th Iteration:
 0.6552
35 th Iteration:
 0.6352
37 th Iteration:
 0.6252
39 th Iteration:
 0.6152
41 th Iteration:
 0.5852
43 th Iteration:
 0.5552
45 th Iteration:
```

```
 0.5052
47 th Iteration:
 0.4552
49 th Iteration:
 0.4552
```

In [81]:
```
cv_scores
```

Out[81]:
```
[0.96,
 0.9505,
 0.8905,
 0.861,
 0.8114,
 0.8019,
 0.8019,
 0.7919,
 0.7924,
 0.7924,
 0.7824,
 0.7824,
 0.7624,
 0.7433,
 0.7143,
 0.6652,
 0.6552,
 0.6352,
 0.6252,
 0.6152,
 0.5852,
 0.5552,
 0.5052,
 0.4552,
 0.4552]
```

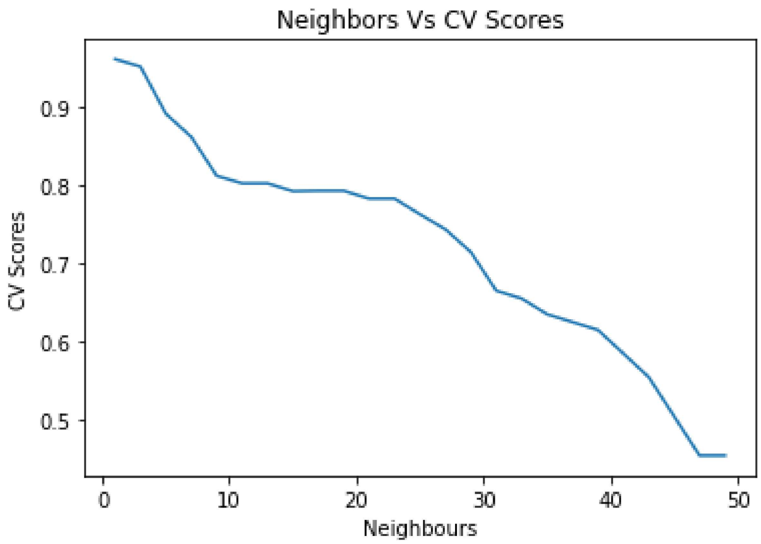
In [82]:
```
max(cv_scores)
```

Out[82]:
```
0.96
```

In [83]:
```
cv_scores.index(max(cv_scores))
```

Out[83]:
```
0
```

In [84]:
```
plt.plot(range(1,50,2),cv_scores)
plt.xlabel('Neighbours')
plt.ylabel('CV Scores')
plt.title('Neighbors Vs CV Scores')
plt.show()
```



In [85]:
```
knn.score(x_train,y_train)
```

Out[85]: 0.84

In [86]:
```
knn.score(x_test,y_test)
```

Out[86]: 0.9230769230769231

In [ ]: