Import Necessary Libraries

In [34]:
```python
import pandas as pd
import seaborn as sns
import numpy as np
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import cross_val_score
```

Data collection

In [3]:
```python
zoodata=pd.read_csv('Zoo.csv')
zoodata
```

Out[3]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone
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
...	...	...	...	...	...	...	...	...	...	...
96	wallaby	1	0	0	1	0	0	0	1	1
97	wasp	1	0	1	0	1	0	0	0	0
98	wolf	1	0	0	1	0	0	1	1	1
99	worm	0	0	1	0	0	0	0	0	0
100	wren	0	1	1	0	1	0	0	0	1

101 rows × 18 columns

In [4]:
```python
zoodata.head()
```

Out[4]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone
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

In [5]:
```python
zoodata.describe()
```

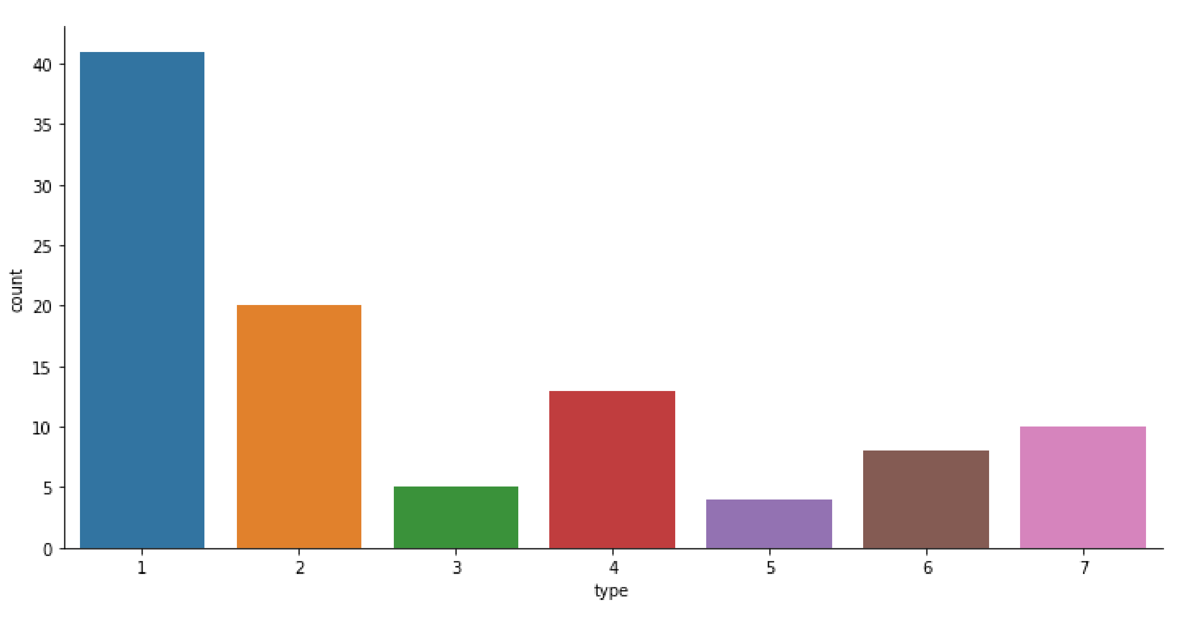
Out[5]:

	hair	feathers	eggs	milk	airborne	aquatic	predator
count	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
std	0.496921	0.400495	0.495325	0.493522	0.427750	0.481335	0.499505
min	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
50%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000
75%	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

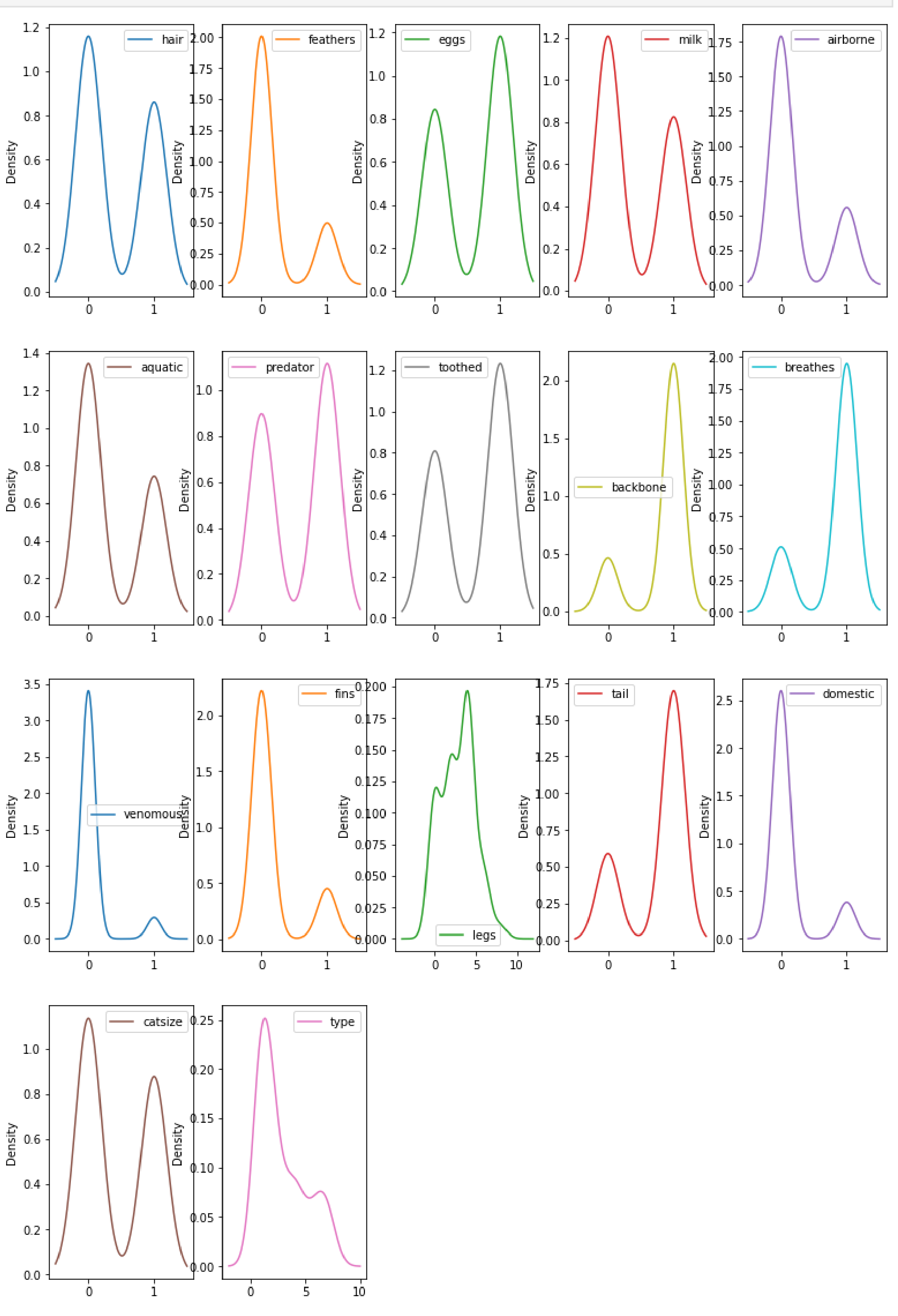
In [10]:
```python
import warnings
warnings.filterwarnings('ignore')
```

In [11]:
```python
sns.factorplot('type', data=zoodata, kind="count",size = 5,aspect = 2)
```

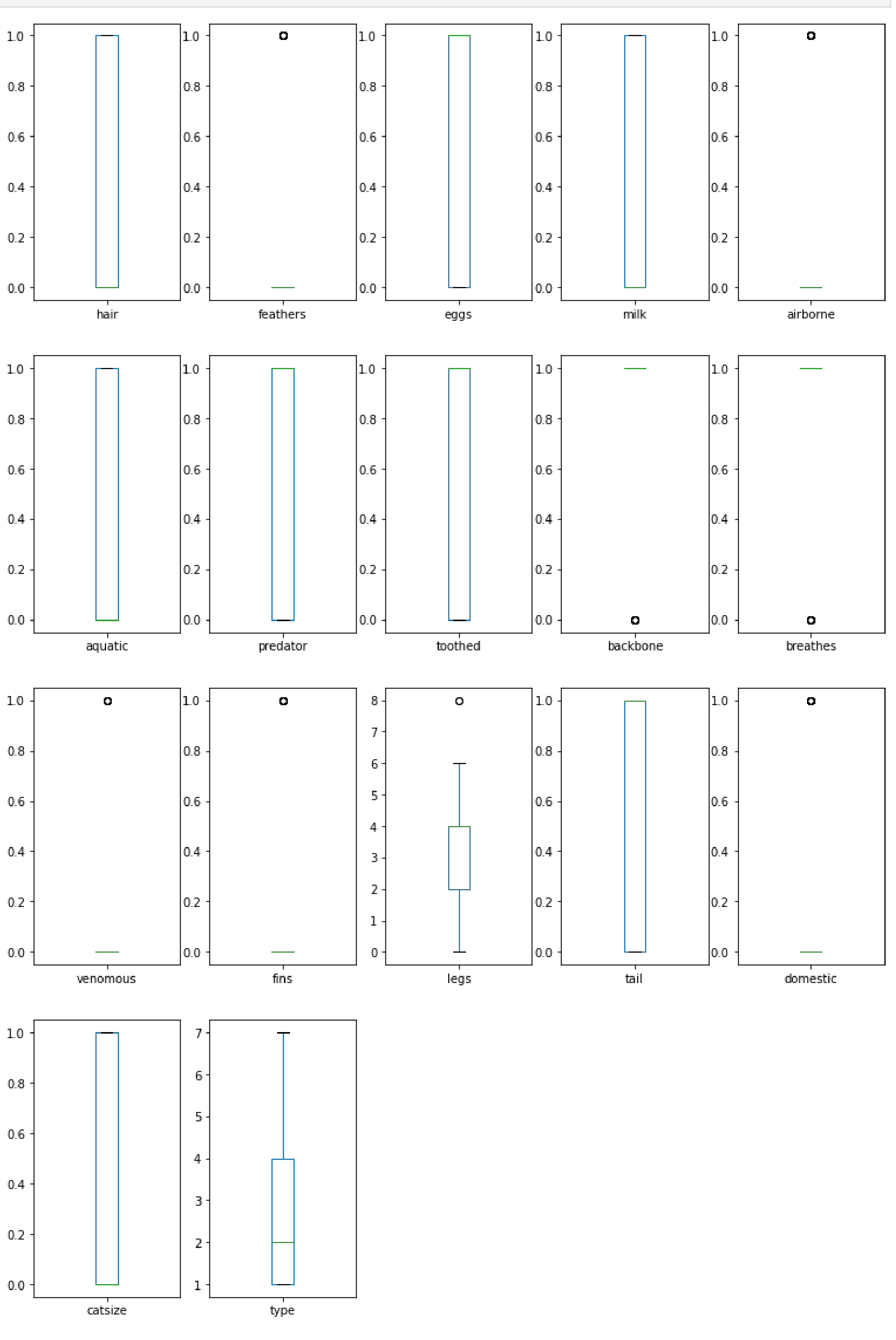
Out[11]: <seaborn.axisgrid.FacetGrid at 0x19ede65f430>



In [14]:
```python
zoodata.plot(kind='density', subplots=True, layout=(4,5), figsize=(13,20),
plt.show()
```



In [15]:
```python
zoodata.plot(kind='box', subplots=True, layout=(4,5), figsize=(13,20), sha
plt.show()
```



Finding correlation between the variables in data

In [16]:
```python
cor = zoodata.corr(method='pearson')
```

In [17]:
```python
cor.style.background_gradient(cmap='coolwarm')
```

Out[17]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	tooth
hair	1.000000	-0.427851	-0.817382	0.878503	-0.198431	-0.473554	-0.154769	0.4925
feathers	-0.427851	1.000000	0.419248	-0.410761	0.656553	-0.058552	-0.104430	-0.6136
eggs	-0.817382	0.419248	1.000000	-0.938848	0.376646	0.376244	0.011605	-0.6421
milk	0.878503	-0.410761	-0.938848	1.000000	-0.366765	-0.362613	-0.029721	0.6281
airborne	-0.198431	0.656553	0.376646	-0.366765	1.000000	-0.172638	-0.295181	-0.5943
aquatic	-0.473554	-0.058552	0.376244	-0.362613	-0.172638	1.000000	0.375978	0.0531
predator	-0.154769	-0.104430	0.011605	-0.029721	-0.295181	0.375978	1.000000	0.1294
toothed	0.492531	-0.613631	-0.642150	0.628168	-0.594311	0.053150	0.129452	1.0000
backbone	0.191681	0.231403	-0.340420	0.384958	-0.104718	0.022463	0.051022	0.5750
breathes	0.441149	0.254588	-0.382777	0.423527	0.286039	-0.637506	-0.262931	-0.0656
venomous	-0.104245	-0.145739	0.098689	-0.242449	0.008528	0.087915	0.115391	-0.0623
fins	-0.280313	-0.223541	0.164796	-0.156328	-0.251157	0.604492	0.190302	0.3642
legs	0.394009	-0.206686	-0.224918	0.214196	0.043712	-0.360638	-0.099723	-0.1934
tail	0.048973	0.292569	-0.221090	0.210026	0.009482	-0.034642	0.018947	0.3103
domestic	0.207208	0.031586	-0.155610	0.163928	0.063274	-0.224308	-0.309794	0.0694
catsize	0.455020	-0.135934	-0.514650	0.574906	-0.349768	-0.111866	0.144790	0.3440
type	-0.562384	-0.197520	0.661825	-0.723683	0.022677	0.326639	0.061179	-0.4715

KNN

Finding optimal number of K

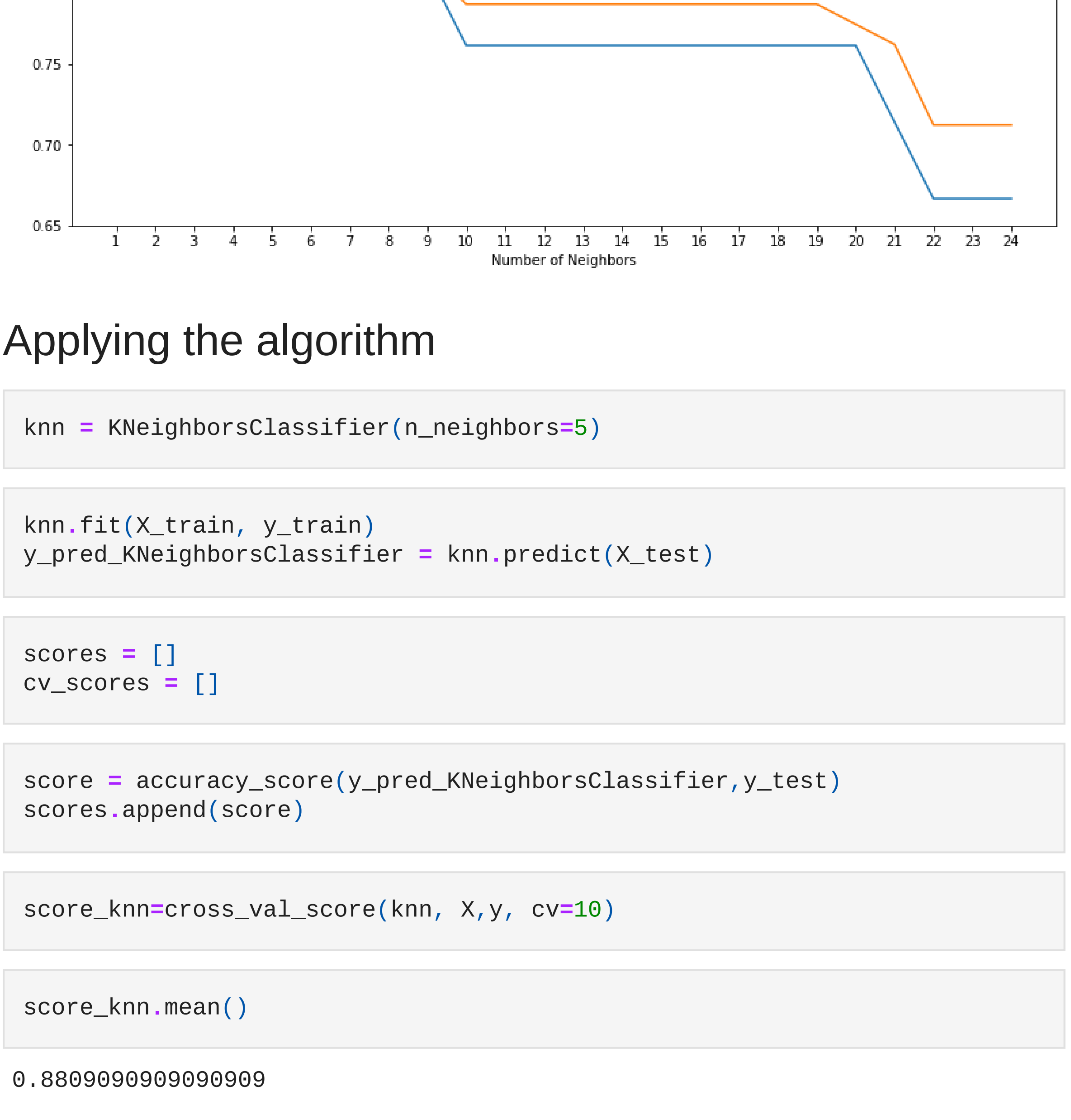
In [18]:
```python
X = zoodata.iloc[:,1:17]
y = zoodata.iloc[:,17]
```

In [21]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
```

In [23]:
```python
k_values = np.arange(1,25)
train_accuracy = []
test_accuracy = []
```

In [26]:
```python
for i, k in enumerate(k_values):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train,y_train)
    train_accuracy.append(knn.score(X_train, y_train))
    test_accuracy.append(knn.score(X_test, y_test))
```

In [28]:
```python
plt.figure(figsize=[13,8])
plt.plot(k_values, test_accuracy, label = 'Testing Accuracy')
plt.plot(k_values, train_accuracy, label = 'Training Accuracy')
plt.legend()
plt.title('-value VS Accuracy')
plt.xlabel('Number of Neighbors')
plt.ylabel('Accuracy')
plt.xticks(k_values)
plt.show()
```



Applying the algorithm

In [29]:
```python
knn = KNeighborsClassifier(n_neighbors=5)
```

In [30]:
```python
knn.fit(X_train, y_train)
y_pred_KNeighborsClassifier = knn.predict(X_test)
```

In [31]:
```python
scores = []
cv_scores = []
```

In [32]:
```python
score = accuracy_score(y_pred_KNeighborsClassifier,y_test)
scores.append(score)
```

In [35]:
```python
score_knn=cross_val_score(knn, X,y, cv=10)
```

In [36]:
```python
score_knn.mean()
```

Out[36]: 0.8809090909090909

In [37]:
```python
score_knn.std()*2
```

Out[37]: 0.12072782037115655

In [38]:
```python
cv_score = score_knn.mean()
```

In [39]:
```python
cv_scores.append(cv_score)
```

In [40]:
```python
cv_scores
```

Out[40]: [0.8809090909090909]

In [ ]: