In [9]:
```python
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn import preprocessing
import seaborn as sns
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.model_selection import cross_val_score
sns.set()
```

In [11]:
```python
df = pd.read_csv('Zoo.csv')
```

In [14]:
```python
df1 = df.copy()
```

In [15]:
```python
df1.head()
```

Out[15]:

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

In [17]:
```python
df1.describe()
```

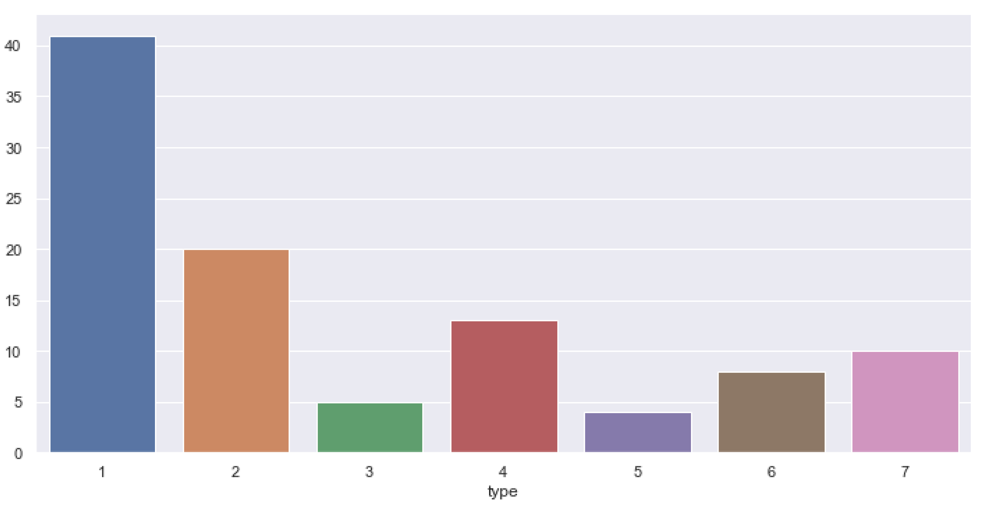
Out[17]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	type
count	101.000000	101.000000	101.000000	101.000000	101.000000	101.000000	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	0.168317	2.841584	0.742574	0.128713	0.435644	2.831683
std	0.496921	0.400495	0.495325	0.493522	0.427750	0.481335	0.499505	0.491512	0.384605	0.407844	0.271410	0.376013	2.033385	0.439397	0.336552	0.498314	2.102709
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	2.000000	0.000000	0.000000	0.000000	1.000000
50%	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	4.000000	1.000000	0.000000	0.000000	2.000000
75%	1.000000	0.000000	1.000000	1.000000	0.000000	1.000000	1.000000	1.000000	1.000000	1.000000	0.000000	0.000000	4.000000	1.000000	0.000000	1.000000	4.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	8.000000	1.000000	1.000000	1.000000	7.000000

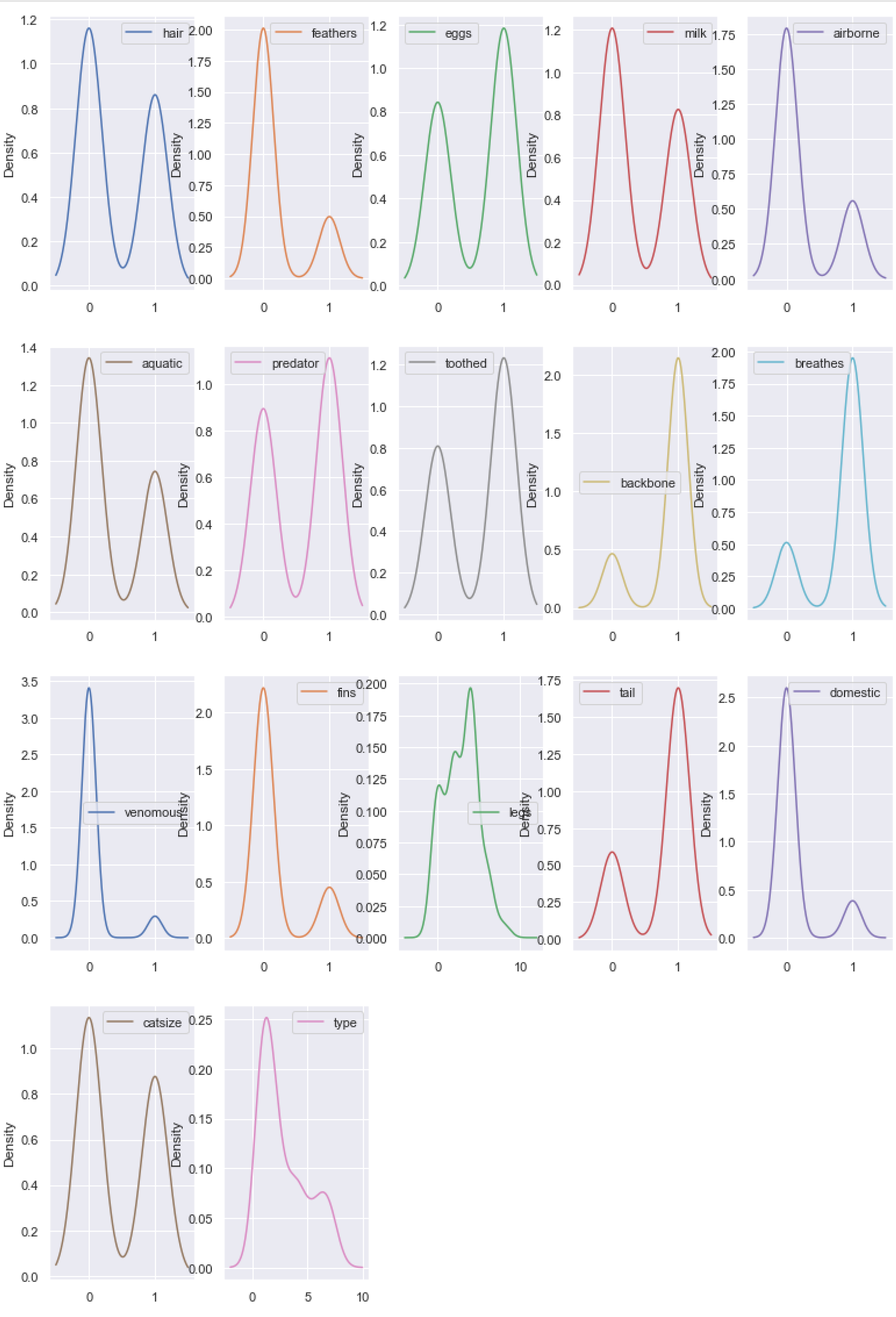
In [18]:
```python
sns.factorplot('type', data=df1, kind="count",size = 5,aspect = 2)
```

```
C:\Users\HP\anaconda3\lib\site-packages\seaborn\categorical.py:3704: UserWarning: The `factorplot` function has been renamed to `catplot`. The original name will be removed in a fu
ture release. Please update your code. Note that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in `catplot`.
  warnings.warn(msg)
C:\Users\HP\anaconda3\lib\site-packages\seaborn\categorical.py:3710: UserWarning: The `size` parameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
C:\Users\HP\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argu
ment will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

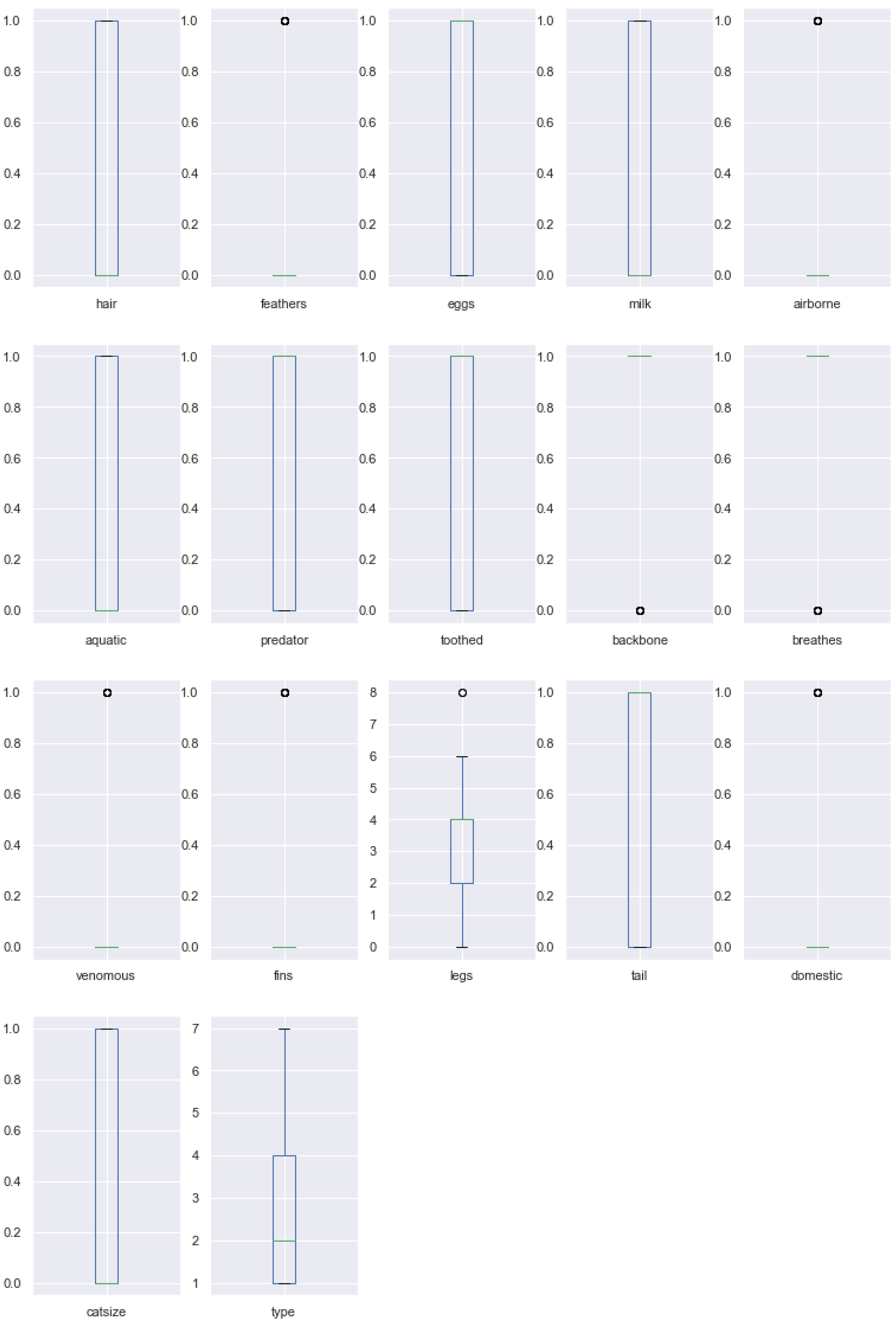
Out[18]: <seaborn.axisgrid.FacetGrid at 0x1741ec5ed00>



In [19]:
```python
df1.plot(kind='density', subplots=True, layout=(4,5), figsize=(13,20), sharex=False, sharey=False)
plt.show()
```



In [20]:
```python
df1.plot(kind='box', subplots=True, layout=(4,5), figsize=(13,20), sharex=False, sharey=False)
plt.show()
```



Finding correlation between the variables in the data

In [21]:
```python
cor = df1.corr(method='pearson')
```

In [22]:
```python
cor.style.background_gradient(cmap='coolwarm')
```

Out[22]:

	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	type
hair	1.000000	-0.427851	-0.817382	0.878503	-0.198431	-0.473554	-0.154769	0.492531	0.191681	0.441149	-0.104245	-0.280313	0.394009	0.048973	0.207208	0.455020	-0.562384
feathers	-0.427851	1.000000	0.419248	-0.410761	0.656553	-0.058552	-0.104430	-0.613631	0.231403	0.254588	-0.145739	-0.223541	-0.206686	0.292569	0.031586	-0.135934	-0.197520
eggs	-0.817382	0.419248	1.000000	-0.938848	0.376646	0.376244	0.011605	-0.642150	-0.340420	-0.382777	0.098689	0.164796	-0.224918	-0.221090	-0.155610	-0.514650	0.661825
milk	0.878503	-0.410761	-0.938848	1.000000	-0.366765	-0.362613	-0.029721	0.628168	0.384958	0.423527	-0.242449	-0.156328	0.214196	0.210026	0.163928	0.574906	-0.723683
airborne	-0.198431	0.656553	0.376646	-0.366765	1.000000	-0.172638	-0.295181	-0.594311	-0.104718	0.286039	0.008528	-0.251157	0.043712	0.009482	0.063274	-0.349768	0.022677
aquatic	-0.473554	-0.058552	0.376244	-0.362613	-0.172638	1.000000	0.375978	0.053150	0.022463	-0.637506	0.087915	0.604492	-0.360638	-0.034642	-0.224308	-0.111866	0.326639
predator	-0.154769	-0.104430	0.011605	-0.029721	-0.295181	0.375978	1.000000	0.129452	0.051022	-0.262931	0.115391	0.190302	-0.099723	0.018947	-0.309794	0.144790	0.061179
toothed	0.492531	-0.613631	-0.642150	0.628168	-0.594311	0.053150	0.129452	1.000000	0.575085	-0.065690	-0.062344	0.364292	-0.193476	0.310368	0.069430	0.344010	-0.471527
backbone	0.191681	0.231403	-0.340420	0.384958	-0.104718	0.022463	0.051022	0.575085	1.000000	0.207666	-0.246611	0.209499	-0.432856	0.731762	0.101733	0.356976	-0.828845
breathes	0.441149	0.254588	-0.382777	0.423527	0.286039	-0.637506	-0.262931	-0.065690	0.207666	1.000000	-0.120752	-0.617219	0.369868	0.088952	0.124068	0.204125	-0.519308
venomous	-0.104245	-0.145739	0.098689	-0.242449	0.008528	0.087915	0.115391	-0.062344	-0.246611	-0.120752	1.000000	-0.033956	0.022964	-0.162724	-0.003252	-0.183748	0.321476
fins	-0.280313	-0.223541	0.164796	-0.156328	-0.251157	0.604492	0.190302	0.364292	0.209499	-0.617219	-0.033956	1.000000	-0.605652	0.204349	-0.093887	0.031705	0.099430
legs	0.394009	-0.206686	-0.224918	0.214196	0.043712	-0.360638	-0.099723	-0.193476	-0.432856	0.369868	0.022964	-0.605652	1.000000	-0.348295	0.073931	0.068791	0.131693
tail	0.048973	0.292569	-0.221090	0.210026	0.009482	-0.034642	0.018947	0.310368	0.731762	0.088952	-0.162724	0.204349	-0.348295	1.000000	0.023434	0.243277	-0.631830
domestic	0.207208	0.031586	-0.155610	0.163928	0.063274	-0.224308	-0.309794	0.069430	0.101733	0.124068	-0.003252	-0.093887	0.073931	0.023434	1.000000	0.020073	-0.181043
catsize	0.455020	-0.135934	-0.514650	0.574906	-0.349768	-0.111866	0.144790	0.344010	0.356976	0.204125	-0.183748	0.031705	0.068791	0.243277	0.020073	1.000000	-0.521030
type	-0.562384	-0.197520	0.661825	-0.723683	0.022677	0.326639	0.061179	-0.471527	-0.828845	-0.519308	0.321476	0.099430	0.131693	-0.631830	-0.181043	-0.521030	1.000000

As seen in the above graph, there is a high correlation exists between some of the variables. We can use PCA to reduce the hight correlated variables

KNN

Finding optimal number of K

In [23]:
```python
X = df1.iloc[:,1:17]
y = df1.iloc[:,17]
```

In [25]:
```python
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1, stratify=y)
```

In [26]:
```python
k_values = np.arange(1,25)
train_accuracy = []
test_accuracy = []
```

In [27]:
```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()
```

As shown in the graph, with K=5 we can achive accurary of 90%.

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 [33]:
```python
score_knn=cross_val_score(knn, X,y, cv=10)
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\model_selection\_split.py:670: UserWarning: The least populated class in y has only 4 members, which is less than n_splits=10.
  warnings.warn(("The least populated class in y has only %d"
```

In [34]:
```python
score_knn.mean()
```

Out[34]: 0.890909090909091

In [35]:
```python
score_knn.std()*2
```

Out[35]: 0.14036552047722747

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

In [41]:
```python
cv_scores.append(cv_score)
```

In [42]:
```python
cv_scores
```

Out[42]: [0.890909090909091]

Conclusion

Support Vector Machine Accuracy: 0.90 (+/- 0.14)