# 1. Import neccessery libraries

In [1]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
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

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

# Problem

**Implement a KNN model to classify the animals in to categorie**

# 2. Import data

In [2]:

```
zoo_data = pd.read_csv('Zoo.csv')
zoo_data
```

Out[2]:

| | animal name | hair | feathers | eggs | milk | airborne | aquatic | predator | toothed | backbone | breat |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **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

# 3. Data understanding

In [3]:

```
zoo_data.shape
```

Out[3]: (101, 18)

In [4]:
```
zoo_data.isna().sum()
```

Out[4]:
```
animal name    0
hair           0
feathers       0
eggs           0
milk           0
airborne       0
aquatic        0
predator       0
toothed        0
backbone       0
breathes       0
venomous       0
fins           0
legs           0
tail           0
domestic       0
catsize        0
type           0
dtype: int64
```

In [5]:
```
zoo_data.dtypes
```

Out[5]:
```
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 [6]:
```
zoo_data.describe()
```

Out[6]:

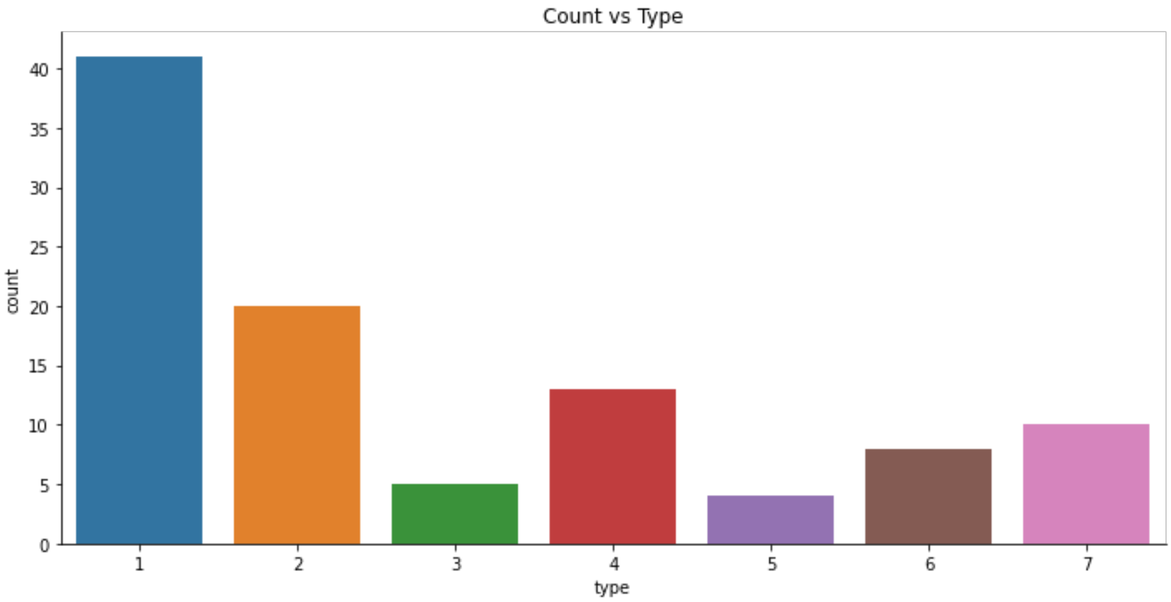
| | hair | feathers | eggs | milk | airborne | aquatic | predator | toc |
|---|---|---|---|---|---|---|---|---|
| **count** | 101.000000 | 101.000000 | 101.000000 | 101.000000 | 101.000000 | 101.000000 | 101.000000 | 101.0( |
| **mean** | 0.425743 | 0.198020 | 0.584158 | 0.405941 | 0.237624 | 0.356436 | 0.554455 | 0.6( |
| **std** | 0.496921 | 0.400495 | 0.495325 | 0.493522 | 0.427750 | 0.481335 | 0.499505 | 0.4! |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0( |
| **25%** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.0( |
| **50%** | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 1.000000 | 1.0( |

| | hair | feathers | eggs | milk | airborne | aquatic | predator | toc |
|---|---|---|---|---|---|---|---|---|
| **75%** | 1.000000 | 0.000000 | 1.000000 | 1.000000 | 0.000000 | 1.000000 | 1.000000 | 1.0( |

In [9]:

```
plt.figure(figsize=(12,8))
sns.factorplot('type',data=zoo_data,kind='count',size=5,aspect=2)
plt.title('Count vs Type' )
plt.show()
```

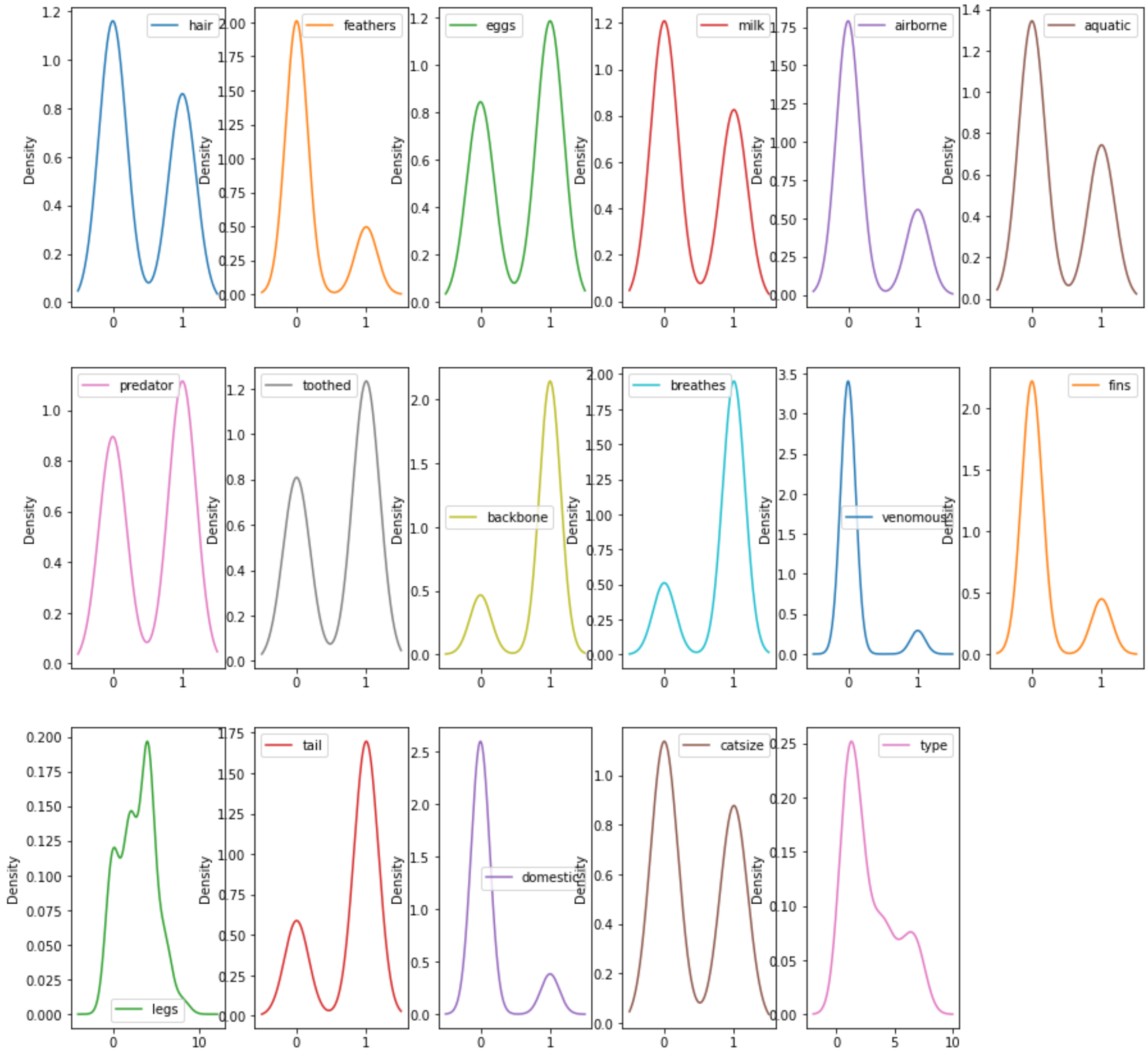
```
<Figure size 864x576 with 0 Axes>
```




# As shown in the graphs above, highest number of animals available in Zoo are Type 1 followed by 2, 4 and 7 respectively

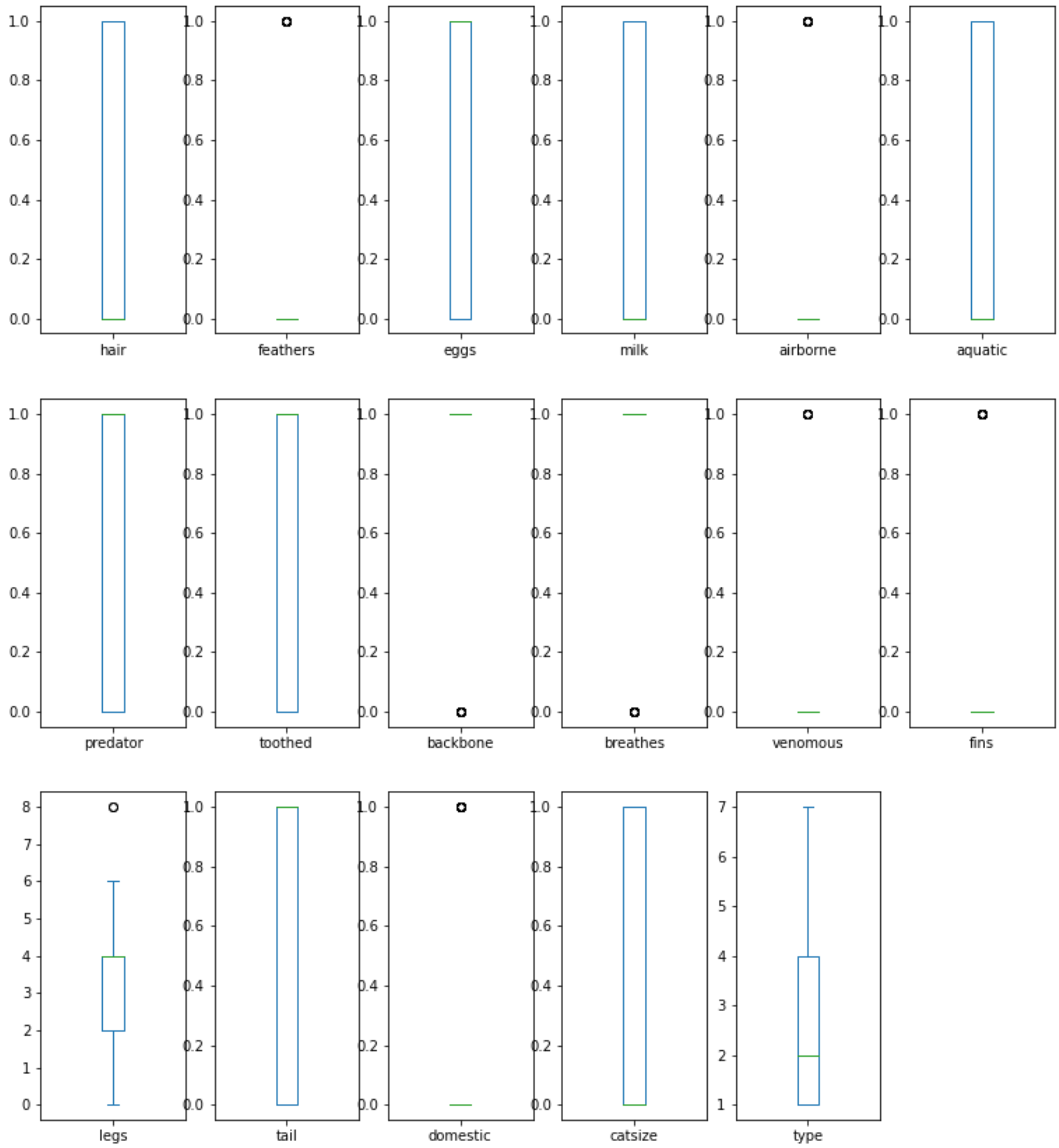
In [10]:

```
zoo_data.plot(kind='density', subplots=True, layout=(5,6),figsize=(15,25),s
plt.show()
```




In [11]:

```
zoo_data.plot(kind='box', subplots=True, layout=(5,6),figsize=(13,25),share
plt.show()
```




As shown in the graphs above, majority of the variables are evenly distributed amongst the animals. However some of the variables like airbone, backbone, breathes, venomous, fins, tail and domestic is not evenly distributed (i.e majority of animals either have these variable or dont)

## 4. Finding correlation between the variables in the data

In [12]:

```
cor = zoo_data.corr(method='pearson')
```

In [13]:

```
cor.style.background_gradient(cmap='coolwarm')
```

Out[13]:

| | 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 |

| | hair | feathers | eggs | milk | airborne | aquatic | predator | toothed |
|---|---|---|---|---|---|---|---|---|
| **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 |
| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |

there is a high correlation exists between some of the variables. We can use PCA to reduce the hight correlated variables

# 5. KNN

## 5.1 Finding optimal number of K

In [15]:

```
X = zoo_data.iloc[:,1:17]
y = zoo_data.iloc[:,17]
```

In [16]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [18]:

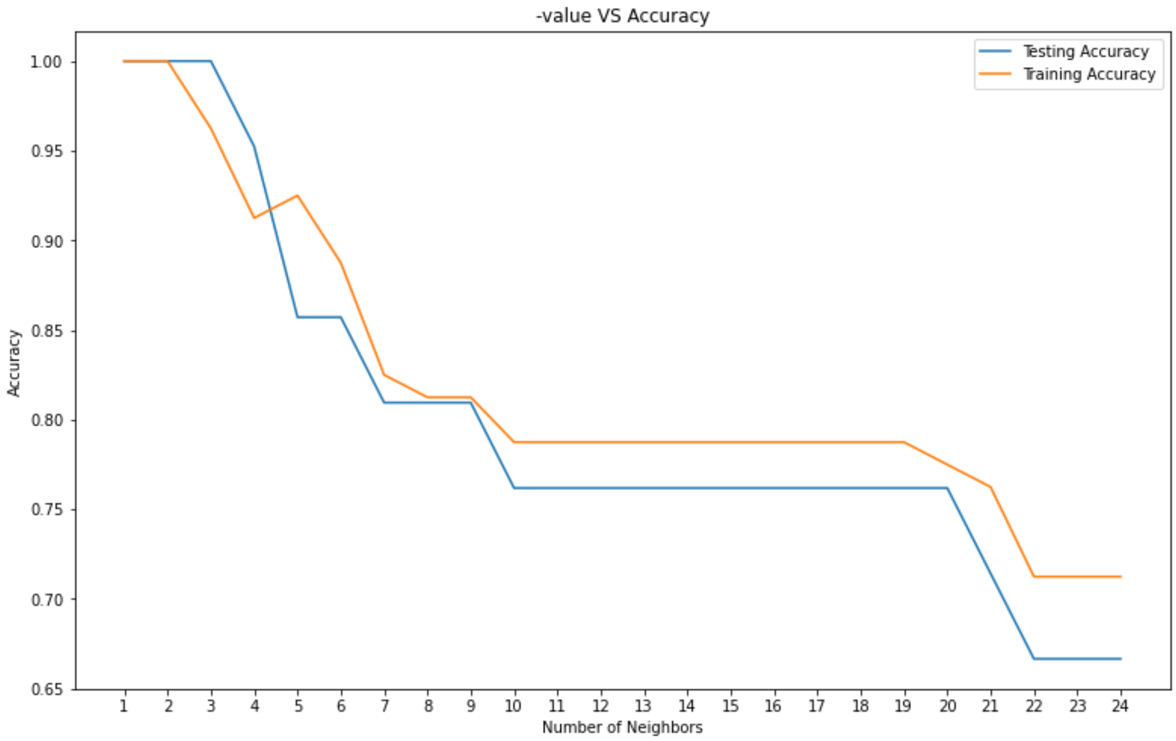
```
k_values = np.arange(1,25)
train_accuracy = []
test_accuracy = []
```

In [19]:

```
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 [21]:

```
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%.

## 5.2 Applying the algorithm

In [23]:

```
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)
y_pred_knn = knn.predict(X_test)
```

In [24]:

```
scores = []
cv_scores = []
```

In [26]:

```
score = accuracy_score(y_pred_knn,y_test)
scores.append(score)
```

In [27]:

```
score_knn=cross_val_score(knn, X,y, cv=10)
```

In [28]:

```
score_knn.mean()
```

Out[28]:

```
0.8809090909090909
```

In [29]:
```
score_knn.std()*2
```
Out[29]:
```
0.12072782037115655
```
In [30]:
```
cv_score = score_knn.mean()
```
In [31]:
```
cv_scores.append(cv_score)
```
In [32]:
```
cv_scores
```
Out[32]:
```
[0.8809090909090909]
```
# 5 - Conclusion

# Support Vector Machine Accuracy: 0.88 (+/- 0.11)

In [ ]:
```

```