dataminingprog3

March 1, 2022

Name:

Mohith Krishna Behata

Email:

mbm2b@mst.edu

Course:

CS 5402

Assignment:

Programming assignment 3

Date:

2022-02-24

GitHublink:

https://git-classes.mst.edu/mbm2b/dataminingprog3

[1]:
```
# Imported for data management (dataframes)
import pandas as pd
import numpy as np
import sklearn as sk
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
```

0.1 Concept Description:

This assignment is about understanding and running a 1-R classifier, also known as a One rule classifier. It generates rules based on each attribute. Each rule is then used to predict the final value. Using this analysis, we can see which attributes are enabling us to provide a more accurate prediction of the final data value

0.2 Data Collection:

The data set was provided by our Instructor Mr.Peery Koob The data given in the dataset has values that can help us build a classifier that can classify the animals based on various attributes.

0.3 Example Description:

animal name This animal name attribute has names of various animals.This is a Nominal Data type. hair The hair attribute has true or false as values which indicate whether the animal has hair or not.This is a Nominal Data type. feathers The feathers attribute has true or false as values which indicate whether the animal has feathers or not.This is a Nominal data type. eggs The egg attribute has true or false as values which indicate whether the animal can lay eggs or not.This is a Nominal data type. milk The milk attribute has true or false as values which indicate whether the animal can give milk or not.This is a Nominal Data type. airborne The airborne attribute has true or false as values which indicate whether the animal moves from one point to other point by air or not.This is a Nominal Data type. aquatic The aquatic attribute has true or false as values which indicate whether the animal is aquatic or not.This is a Nominal Data type. preadator The predator attribute has true or false as values which indicate whether the animal is predator or not.This is a Nominal Data type. toothed The toothed attribute has true or false as values which indicate whether the animal has teeth or not.This is a Nominal Data type. backbone The backbone attribute has true or false as values which indicate whether the animal has backbone or not.This is a Nominal Data type. breathes The breathes attribute has true or false as values which indicate whether the animal breathes or not.This is a Nominal Data type. venomous The venomous attribute has true or false as values which indicate whether the animal is venomous or not.This is a Nominal Data type. fins The fins attribute has true or false as values which indicate whether the animal has fins or not.This is a Nominal Data type. legs The legs attribute has values which indicate how many legs does the animal have.This is a Nominal Data type but numeric. tail The tail attribute has true or false as values which indicate whether the animal has tail or not.This is a Nominal Data type. domestic The domestic attribute has true or false as values which indicate whether the animal is domestic or not.This is a Nominal Data type. catsize The catsize attribute has true or false as values .This is a Nominal Data type. gestation The hair gestation has various values which are numeric these values which indicate the number of days animal takes between conception and birth.This is a Nominal Data type.[0-720]days type The type attribute has values that describe various types of animals (amphibian, arthropod, bird, fish, insect, mammal, reptile).This is a Nominal Data type.

Attributes values level of measurement - animal_name: Unique for each instance.Nominal datatype. - hair:Nominal datatype. - feathers: Nominal datatype. - eggs: Nominal datatype. - milk: Nominal datatype. - airborne: Nominal datatype. - aquatic: Nominal datatype. - predator: Nominal datatype. - toothed: Nominal datatype. - backbone: Nominal datatype. - breathes: Nominal datatype. - venomous: Nominal datatype. - fins: Nominal datatype. - legs: Nominal Data type but Numeric (set of values: {0,2,4,5,6,8}). - tail: Nominal datatype. - domestic: Nominal datatype. - catsize: Nominal datatype. - gestation: Nominal datatype. - type: Nominal datatype.

0.4 Data Import and Wrangling:

[2]:
```
# load dataset
animal = pd.read_excel("animal-taxonomy.xlsx")
```

[3]:
```
#Understand the data head()returns the First 5 rows of the dataset
animal.head()
```

[3]:
```
  animal name   hair  feathers   eggs   milk  airborne  aquatic  predator  \
0    aardvark   True     False  False   True     False    False      True
1       anole  False     False   True  False     False    False     False
2    antelope   True     False  False   True     False    False     False
3     axolotl  False     False   True  False     False     True     False
4        bass  False     False   True  False     False     True      True

   toothed  backbone  breathes  venomous   fins  legs   tail  domestic  \
0     True      True      True     False  False     4  False     False
1     True      True      True     False  False     4   True     False
2     True      True      True     False  False     4   True     False
3     True      True     False     False  False     4   True     False
4     True      True     False     False   True     0   True     False

   catsize  gestation       type
0     True      213.0     mammal
1    False       42.0    reptile
2     True      274.0     mammal
3    False       17.0  amphibian
4    False        5.0       fish
```

[4]:
```
#for understanding the shape of dataset
print("Shape:", animal.shape)
```

```
Shape: (132, 19)
```

By this we understood there are 19 columns in this dataset and 132 rows in the dataset.

0.5 Exploratory Data Analysis:

[5]:
```
animal['type'].unique()
```

[5]:
```
array(['mammal', 'reptile', 'amphibian', 'fish', 'insect', 'bird',
       'arthropod', 'mamal', 'fis'], dtype=object)
```

from above we can see that there are erros in the spelling of type of animals we need to rectify these for example fis->fish for example mamal->mammal

[6]:
```
animal['type']=animal['type'].replace(['fis'],'fish')
animal['type']=animal['type'].replace(['mamal'],'mammal')
```

Now we can see that the mistakes in the dataset had beeen rectified

[7]:
```
animal['type'].unique()
```

[7]:
```
array(['mammal', 'reptile', 'amphibian', 'fish', 'insect', 'bird',
       'arthropod'], dtype=object)
```

In 1-R, binary classification is performed. The categories bird, fish, anthropod, insect, reptile, and amphibian are replaced by 'non-mammal'

[8]:
```
animal['type'] = animal['type'].
↪replace(['bird','fish','arthropod','insect','reptile','amphibian'],'non-mammal')
```

[9]:
```
animal['type'].unique()
```

[9]:
```
array(['mammal', 'non-mammal'], dtype=object)
```

Creating two classes mammalclass and nonmammalclass

We will create Two classes that has all Mammal and Nonmammal type of animals separtely. 1. Mammal is Mammal Class 2. NonMammal is Non-Mammal Class

[10]:
```
grouped = animal.groupby(animal.type)
mammal = grouped.get_group("mammal")
nonmammal=grouped.get_group("non-mammal")
```

[11]:
```
class mammalclass:

    def __init__(self, mammal):
        self.mammal = mammal
    def show(self):
        return print(mammal)
    def count(self):
        return mammal['type'].value_counts()
Mammal = mammalclass(mammal)
```

[12]:
```
class nonmammalclass:

    def __init__(self,nonmammal):
        self.nonmammal = nonmammal
    def show(self):
        return print(nonmammal)
    def count(self):
        return nonmammal['type'].value_counts()
NonMammal = nonmammalclass(nonmammal)
```

[13]:
```
mammalclass.count(0)
```

[13]:
```
mammal    51
Name: type, dtype: int64
```

[14]:
```
nonmammalclass.count(0)
```

[14]:
```
non-mammal    81
Name: type, dtype: int64
```

By the example we can see with help of count method in mammal and non mammal class we are able to see the number of animals available for each data type.

0.5.1 Values for each attribute

Determining the possible values or range of the values for each attribute

For all the numeric data the minimum and maximum values would be provided as those values fall within the range.

For all the nominal data the unique values available in the dataframe animal would be provided.

[15]:
```
num=['legs','gestation']
obj=['animal␣
 ↪name','hair','feathers','eggs','milk','airborne','aquatic','predator','toothed','backbone',
 ↪]

for a in num:
    print(a, 'has values that fall between',animal[a].min(),'and', animal[a].
 ↪max())

for a in obj:
    print('The possible values for ',a,' are: ',animal[a].unique())
```

```
legs has values that fall between 0 and 12
gestation has values that fall between 0.0 and 720.0
The possible values for  animal name  are:  ['aardvark' 'anole' 'antelope'
'axolotl' 'bass' 'bear' 'blue dragon'
 'boar' 'buffalo' 'caecilians' 'capybara' 'carp' 'catfish' 'cavy' 'cayman'
 'cheetah' 'chicken' 'chinese giant salamander' 'chub' 'chupacabra' 'clam'
 'cow' 'crab' 'crayfish' 'crow' 'cuttlefish' 'dartfrog' 'deer' 'dogfish'
 'dolphin' 'dove' 'duck' 'elephant' 'flamingo' 'flea' 'fox' 'frog'
 'fruitbat' 'giant panda' 'giraffe' 'gnat' 'goat' 'gorilla'
 'great white shark' 'gull' 'haddock' 'hagfish' 'hamster' 'hare' 'hawk'
 'hellbender' 'herring' 'honeybee' 'horse' 'housecat' 'housefly' 'human'
 'kiwi' 'komodo dragon' 'ladybird' 'lark' 'leopard' 'lion' 'lobster'
 'lynx' 'malayan sun bear' 'manta ray' 'mantis' 'mink' 'mole' 'momo'
 'mongoose' 'moth' 'nessie' 'newt' 'octopus' 'opossum' 'orangutan' 'oryx'
 'ostrich' 'pangolin' 'parakeet' 'penguin' 'pheasant' 'pike' 'piranha'
 'pitviper' 'platypus' 'polecat' 'porpoise' "portuguese man o' war" 'puma'
 'raccoon' 'red panda' 'reindeer' 'rhea' 'scorpion' 'sea anemone'
 'sea cucumber' 'seahorse' 'seal' 'sealion' 'seasnake' 'seawasp'
 'shai-hulud' 'skimmer' 'skink' 'skua' 'slowworm' 'slug' 'sole' 'sparrow'
 'squid' 'squirrel' 'starfish' 'stingray' 'swan' 'termite' 'thylacine'
 'toad' 'tortoise' 'tuatara' 'tuna' 'vampire' 'vole' 'vulture' 'wallaby'
 'wasp' 'whale shark' 'wolf' 'worm' 'wren']
The possible values for  hair  are:  [ True False]
The possible values for  feathers  are:  [False  True]
The possible values for  eggs  are:  [False  True]
```

```
The possible values for  milk  are:  [ True False]
The possible values for  airborne  are:  [False  True]
The possible values for  aquatic  are:  [False  True]
The possible values for  predator  are:  [ True False]
The possible values for  toothed  are:  [ True False]
The possible values for  backbone  are:  [ True False]
The possible values for  breathes  are:  [ True False]
The possible values for  venomous  are:  [False  True]
The possible values for  fins  are:  [False  True]
The possible values for  domestic  are:  [False  True]
The possible values for  catsize  are:  [ True False]
The possible values for  type  are:  ['mammal' 'non-mammal']
```

[16]:
```
#Now we will check for unique values for each attribute present in the forest␣
↪fire dataset
animal.nunique()
```

[16]:
```
animal name    132
hair             2
feathers         2
eggs             2
milk             2
airborne         2
aquatic          2
predator         2
toothed          2
backbone         2
breathes         2
venomous         2
fins             2
legs             8
tail             2
domestic         2
catsize          2
gestation       80
type             2
dtype: int64
```

0.5.2 Checking if there are any missing values in dataset and handling them

[17]:
```
column = animal.columns
for a in column:
    if animal[a].isnull().any().sum():
        print(a,' has: ',animal[a].isnull().sum(),' missing values')
    else:
        print(a,'has no missing values')
```

```
animal name has no missing values
hair has no missing values
feathers has no missing values
eggs has no missing values
milk has no missing values
airborne has no missing values
aquatic has no missing values
predator has no missing values
toothed has no missing values
backbone has no missing values
breathes has no missing values
venomous has no missing values
fins has no missing values
legs has no missing values
tail has no missing values
domestic has no missing values
catsize has no missing values
gestation  has:  6  missing values
type has no missing values
```

"gestation" attribute is the only attribute with missing value.

These missing values can cause serious trouble if they are not handled

So we would fill the missing values of gestation with the mean of gestation

[18]:
```
animal = animal.fillna(np.mean(animal.gestation))
```

[19]:
```
column = animal.columns
for a in column:
    if animal[a].isnull().any().sum():
        print(a,' has: ',animal[a].isnull().sum(),' missing values')
    else:
        print(a,'has no missing values')
```

```
animal name has no missing values
hair has no missing values
feathers has no missing values
eggs has no missing values
milk has no missing values
airborne has no missing values
aquatic has no missing values
predator has no missing values
toothed has no missing values
backbone has no missing values
breathes has no missing values
venomous has no missing values
fins has no missing values
legs has no missing values
```

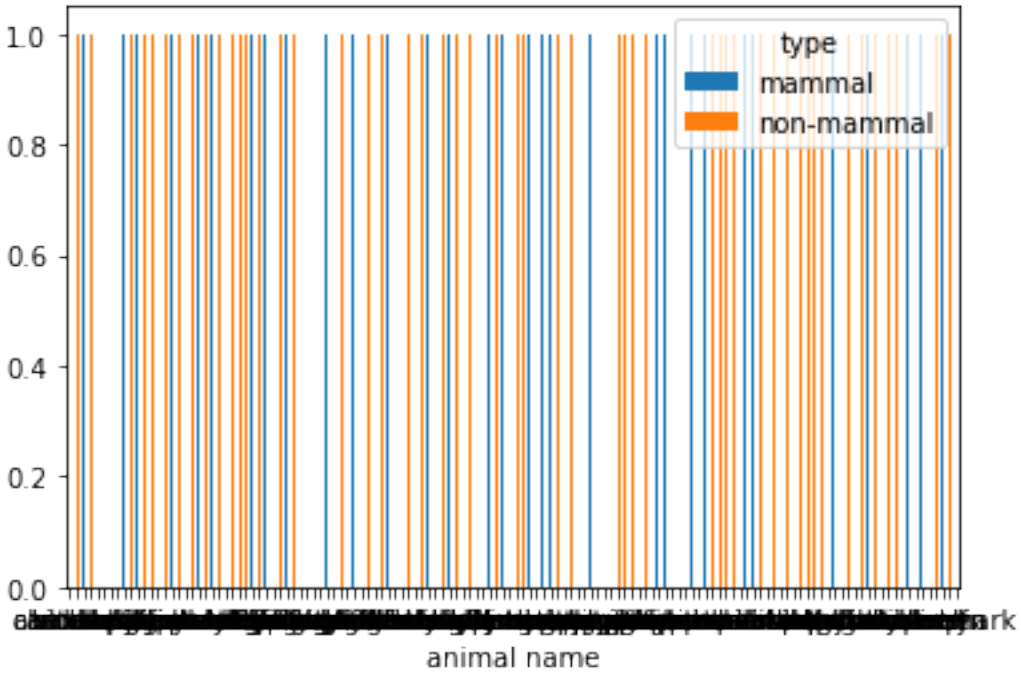
```
tail has no missing values
domestic has no missing values
catsize has no missing values
gestation has no missing values
type has no missing values
```

Now we can see that there are no missing values in the dataset

0.5.3 Scatter plot and Bar charts

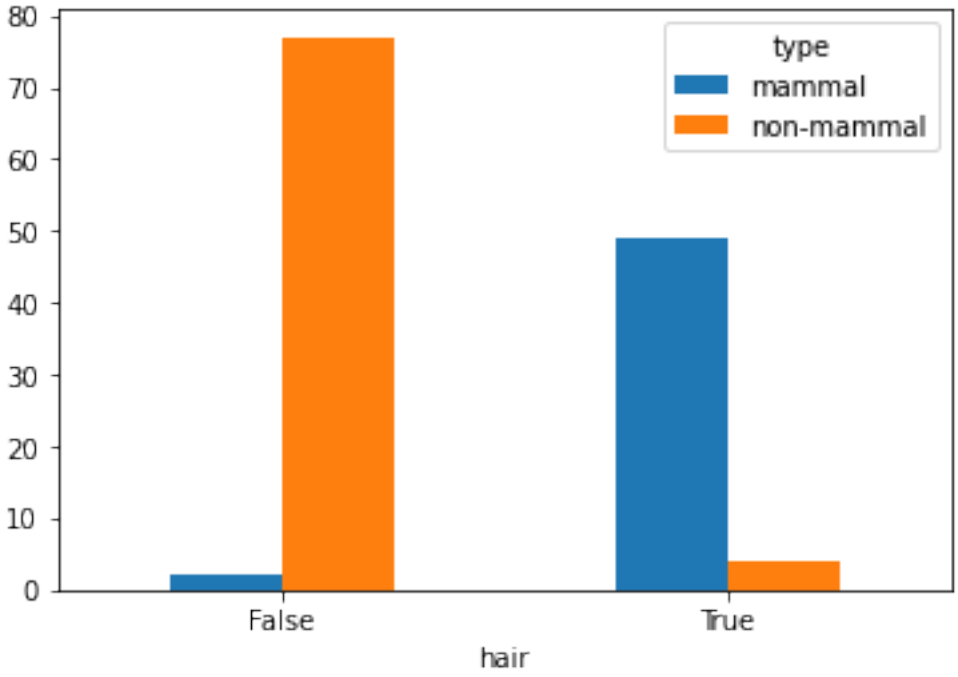
- Visualizing the data for each attribute
- the Below graphs show the attribute value occurences in the dataset.

[20]:
```
crosstb =pd.crosstab(animal['animal name'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```

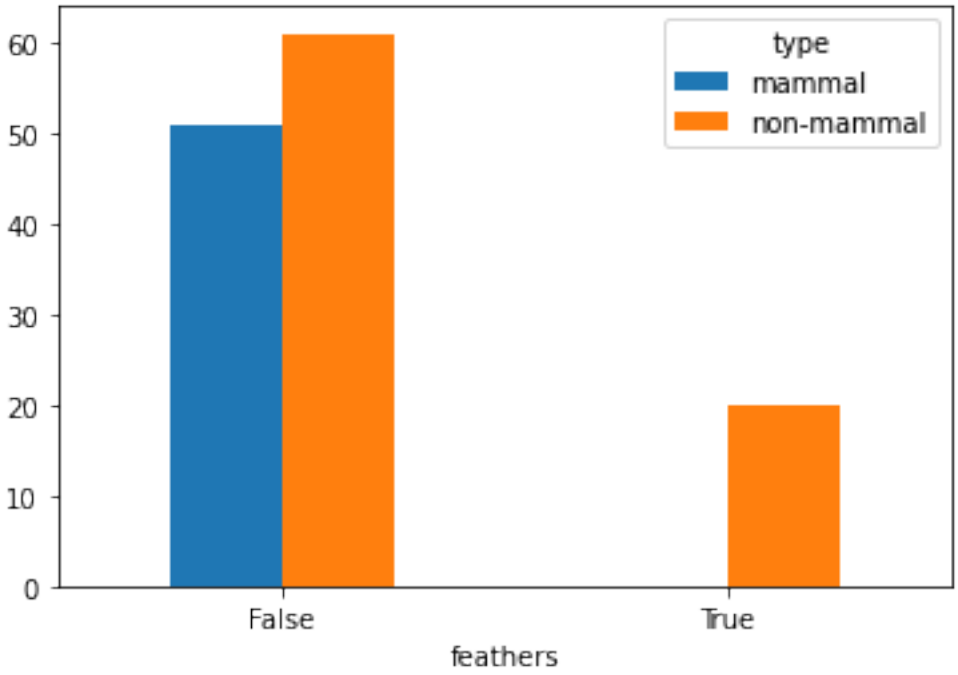


As there are so many unique values for attribute animal name the bar plot is unable to show all the values clearly.

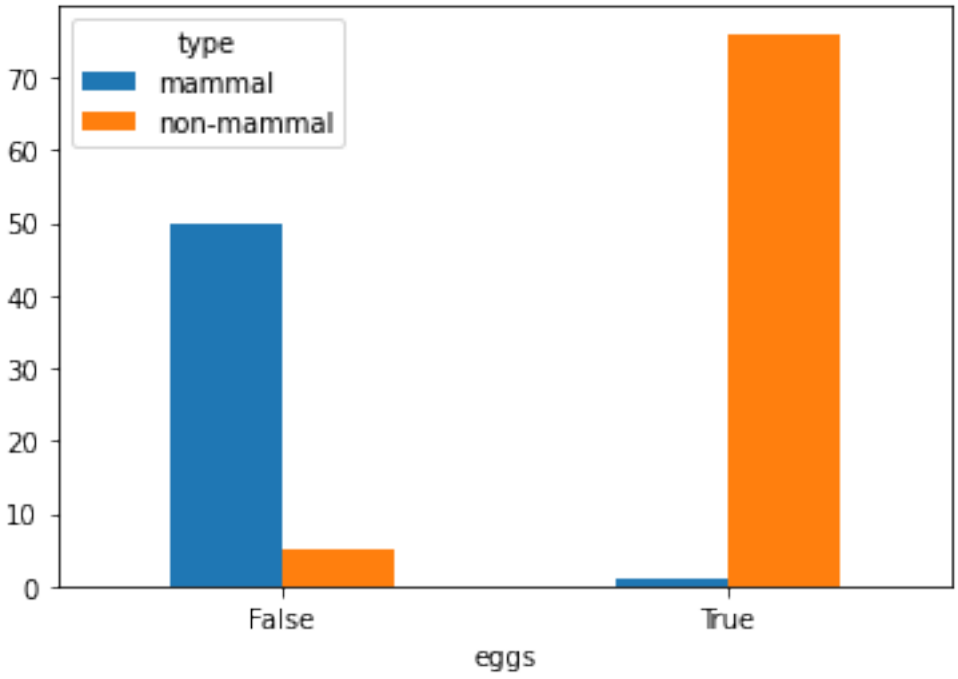
[21]:
```
crosstb =pd.crosstab(animal['hair'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



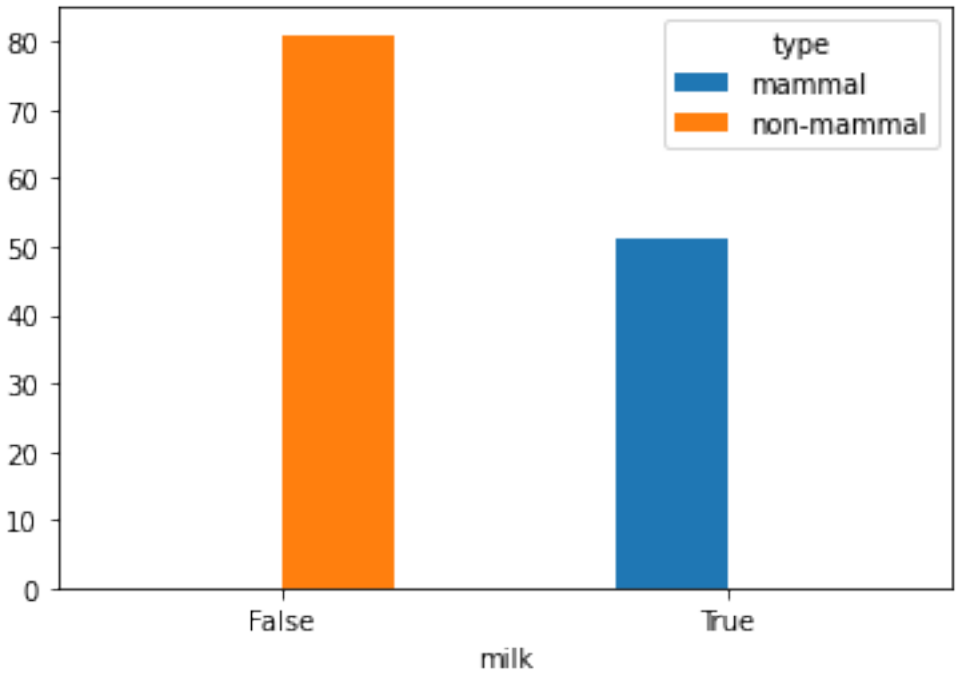
[22]:
```
crosstb =pd.crosstab(animal['feathers'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



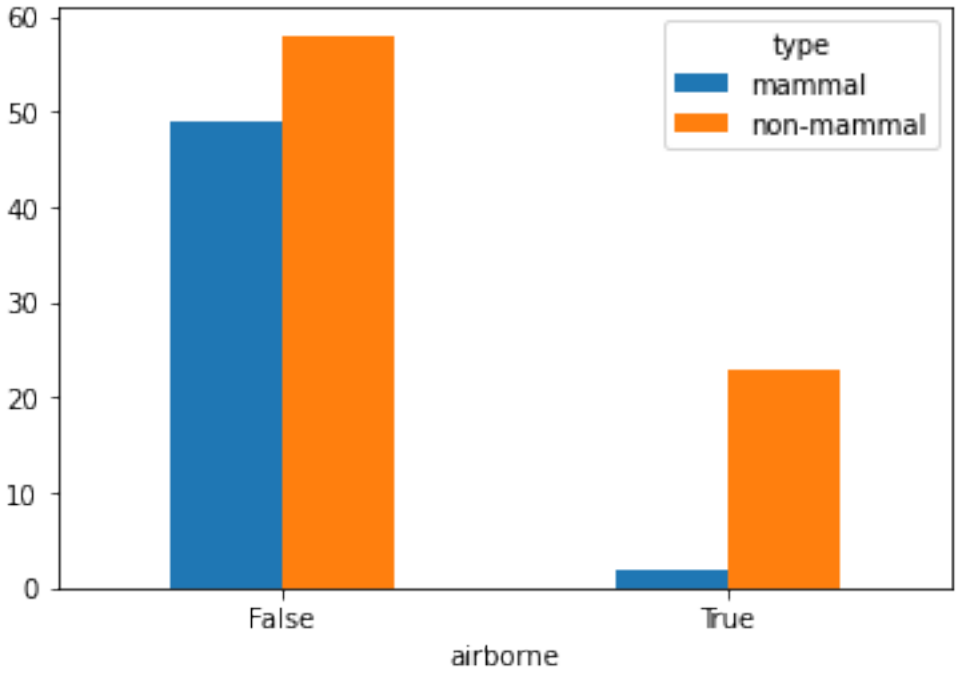
[23]:
```
crosstb =pd.crosstab(animal['eggs'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



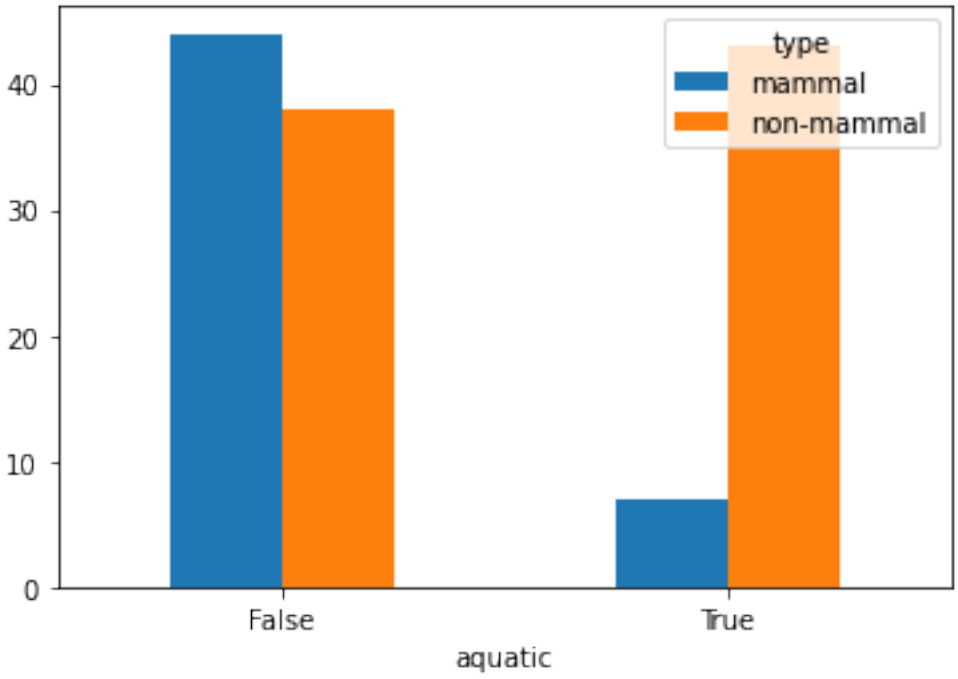
[24]:
```
crosstb =pd.crosstab(animal['milk'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



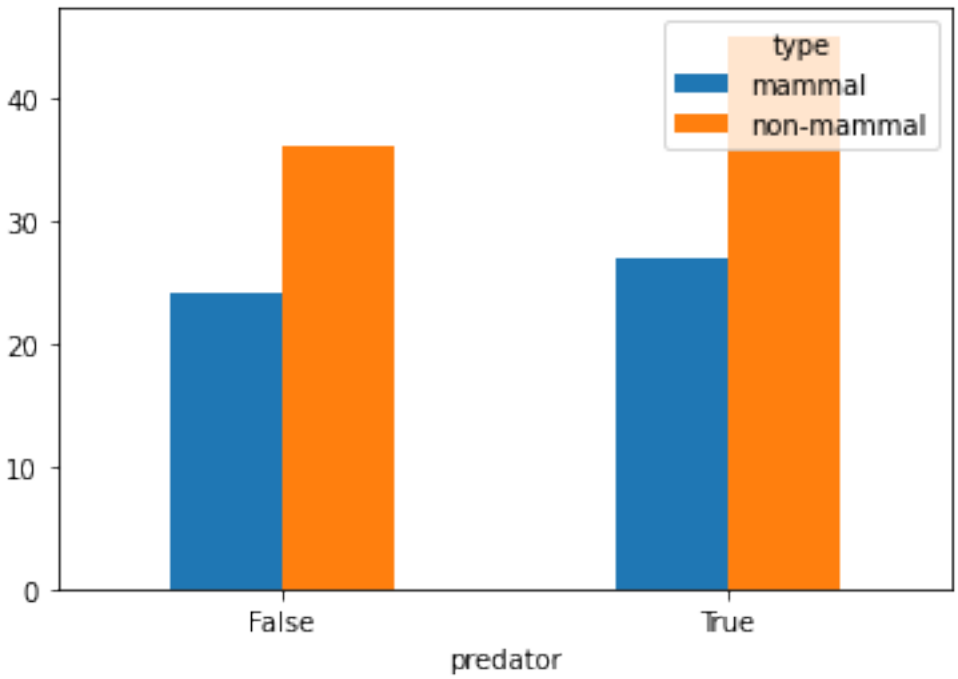
[25]:
```
crosstb =pd.crosstab(animal['airborne'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



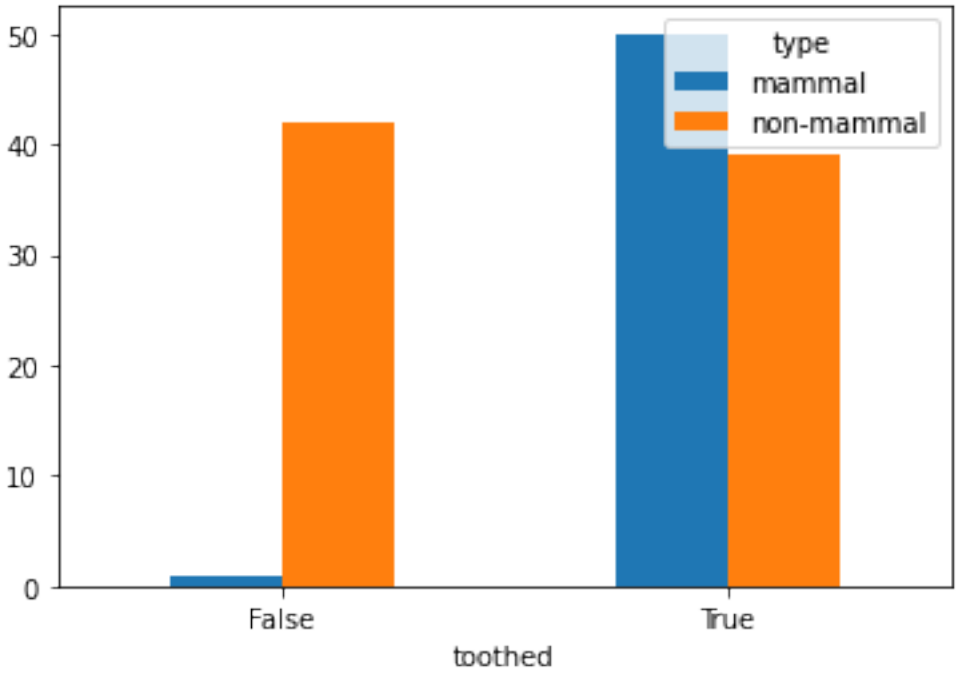
[26]:
```
crosstb =pd.crosstab(animal['aquatic'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



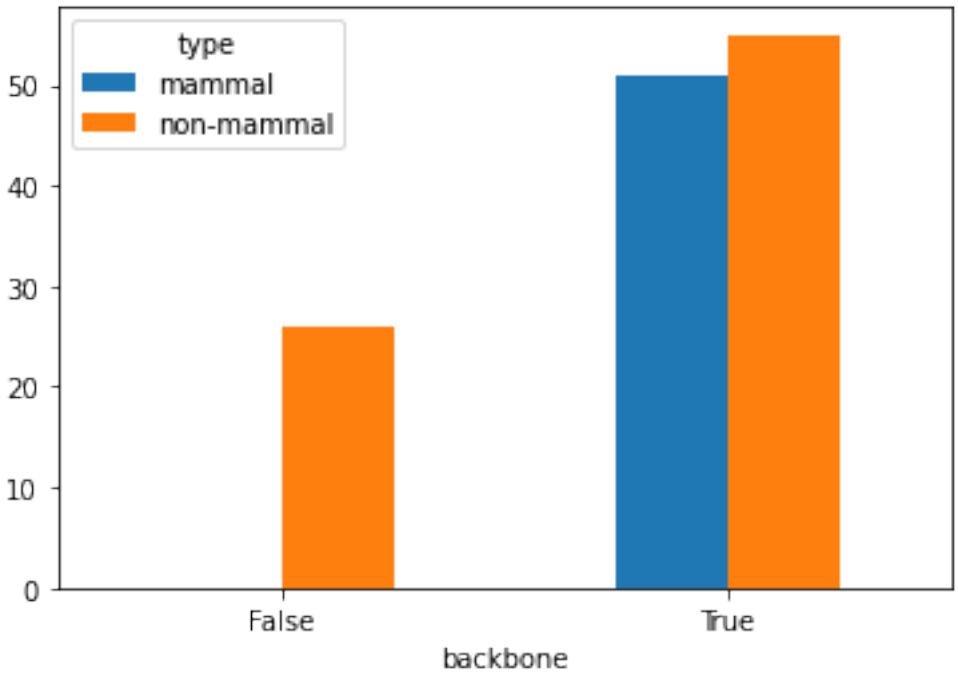
[27]:
```
crosstb =pd.crosstab(animal['predator'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



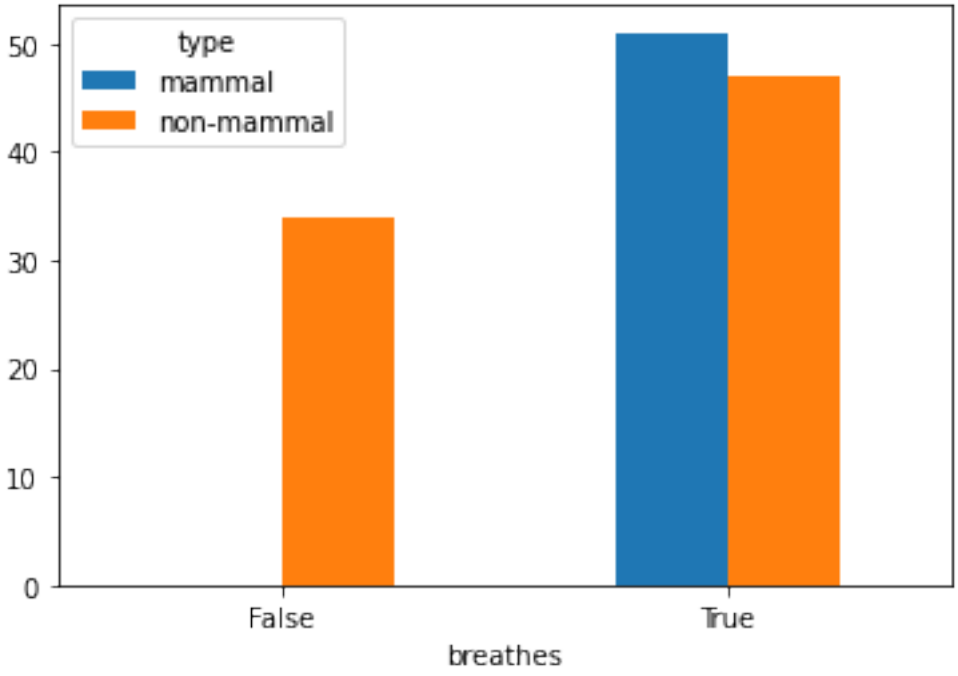
[28]:
```
crosstb =pd.crosstab(animal['toothed'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



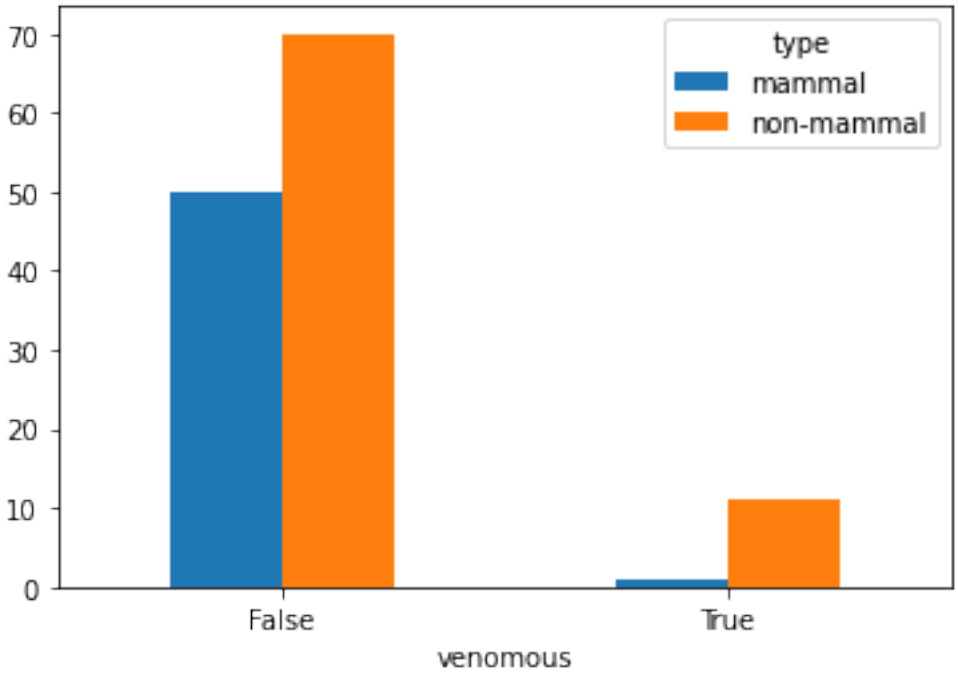
[29]:
```
crosstb =pd.crosstab(animal['backbone'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



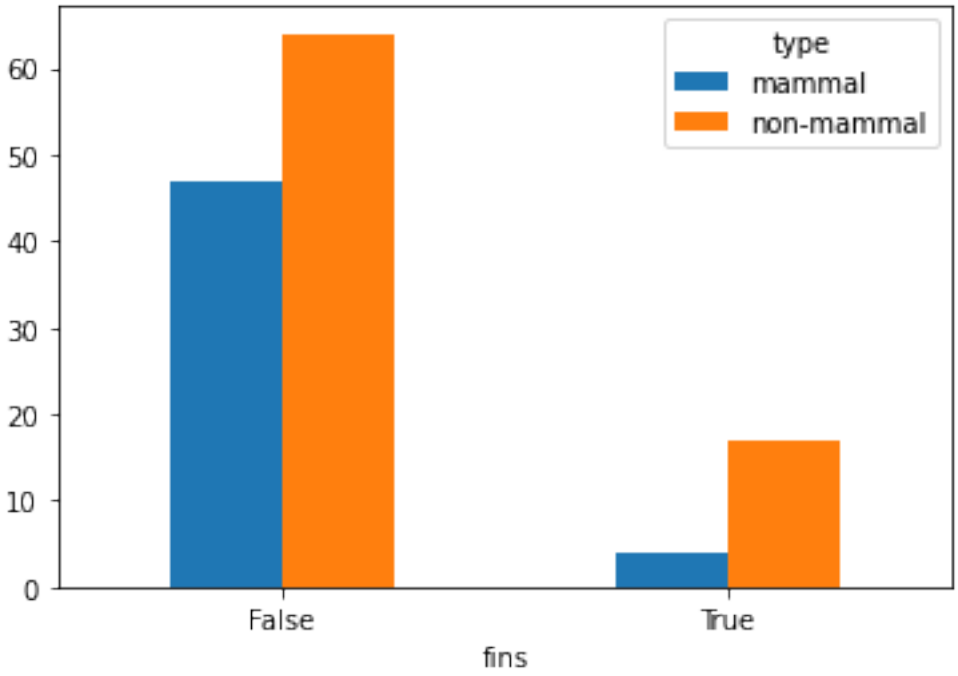
[30]:
```
crosstb =pd.crosstab(animal['breathes'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



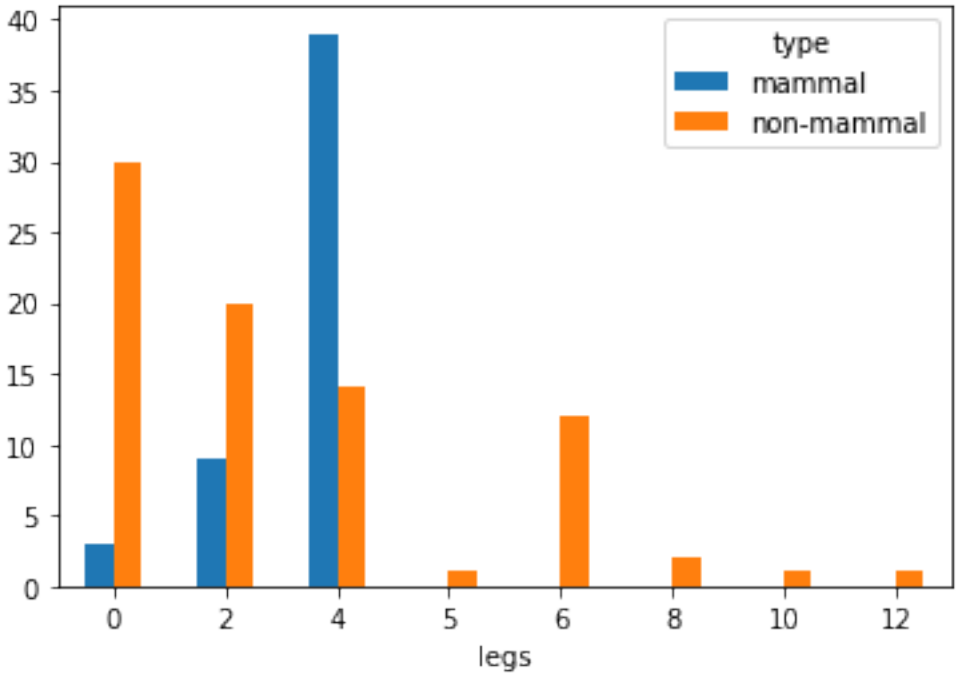
[31]:
```
crosstb =pd.crosstab(animal['venomous'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



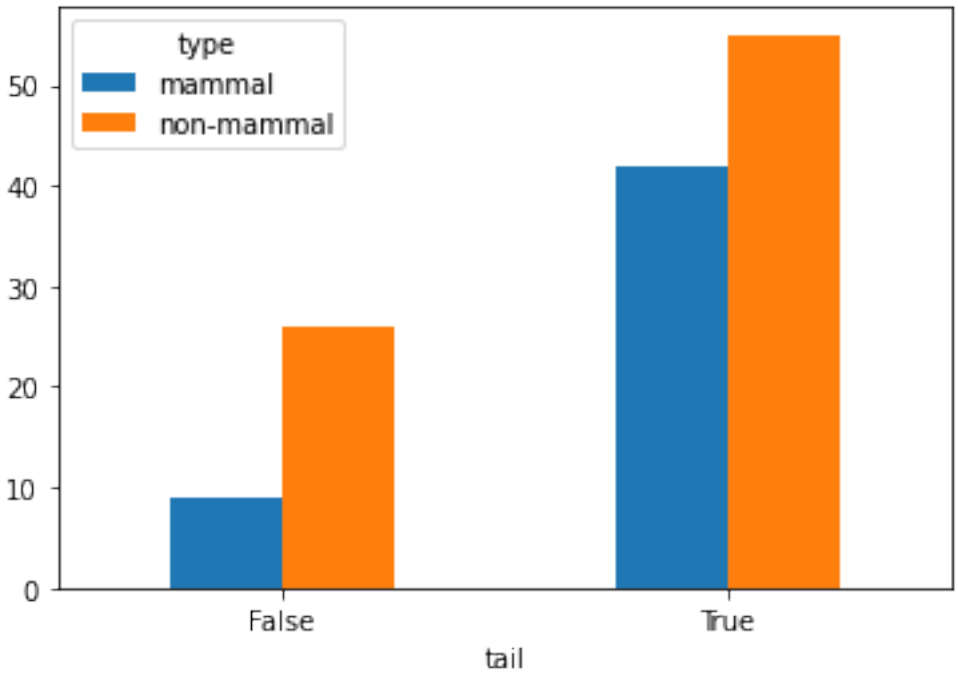
[32]:
```
crosstb =pd.crosstab(animal['fins'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



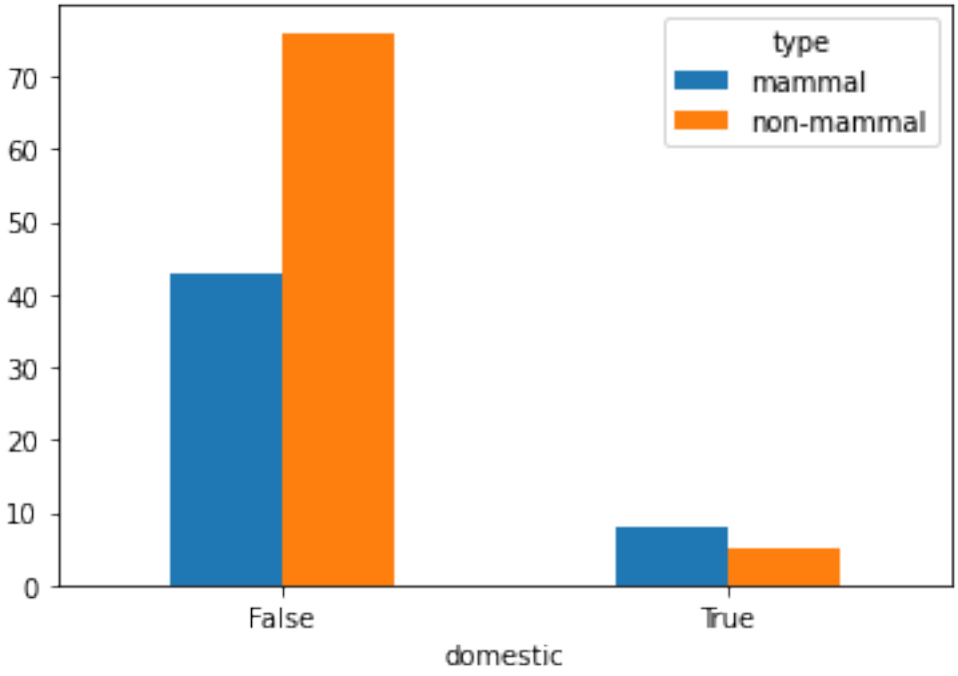
[33]:
```
crosstb =pd.crosstab(animal['legs'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



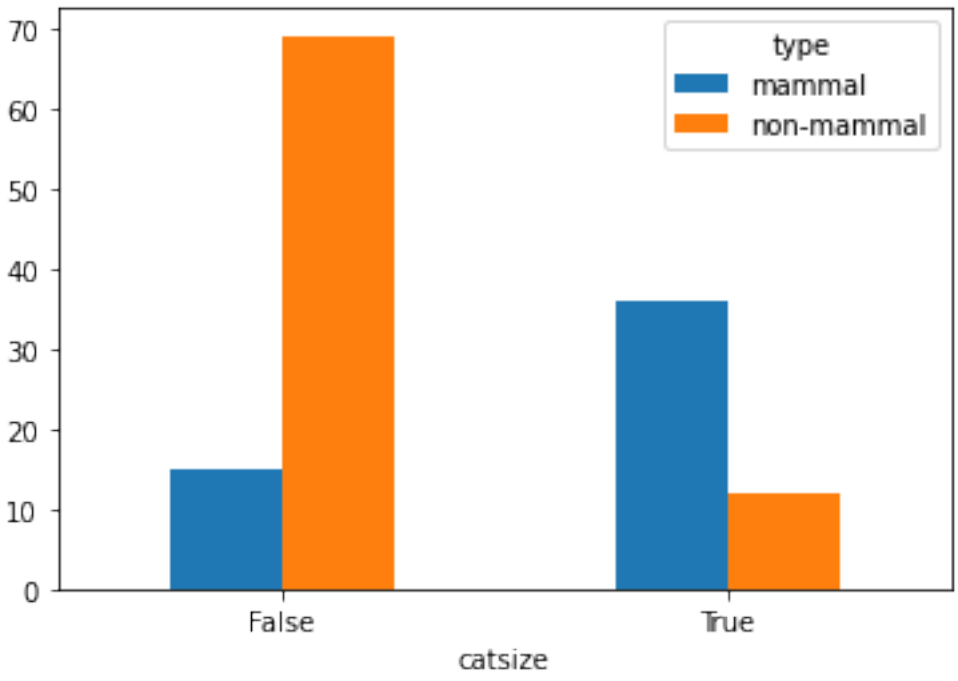
[34]:
```
crosstb =pd.crosstab(animal['tail'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



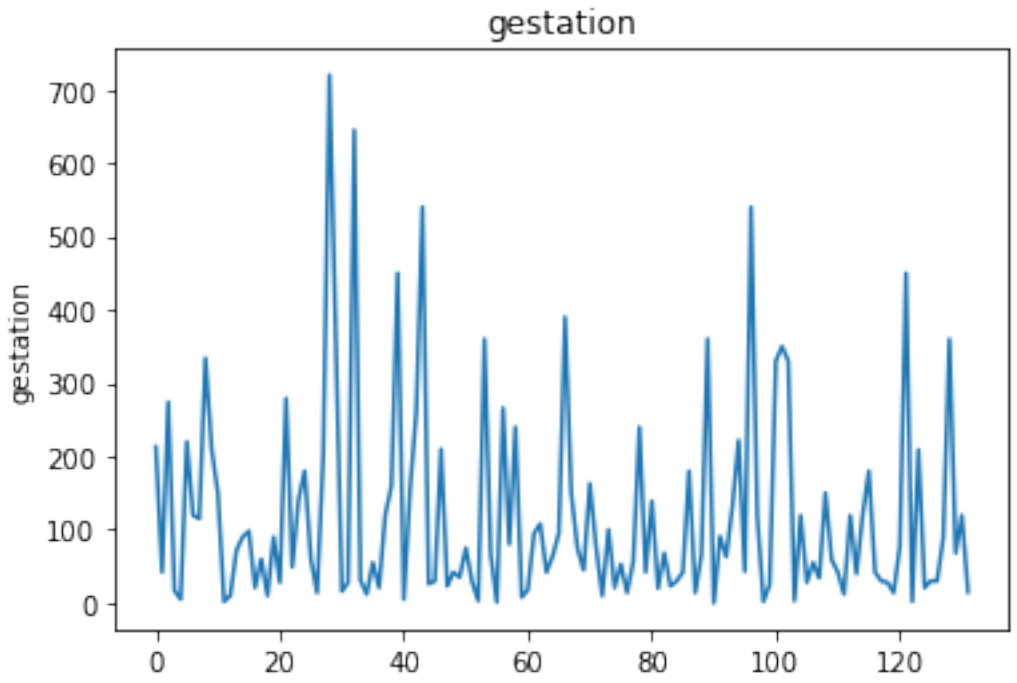
[35]:
```
crosstb =pd.crosstab(animal['domestic'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



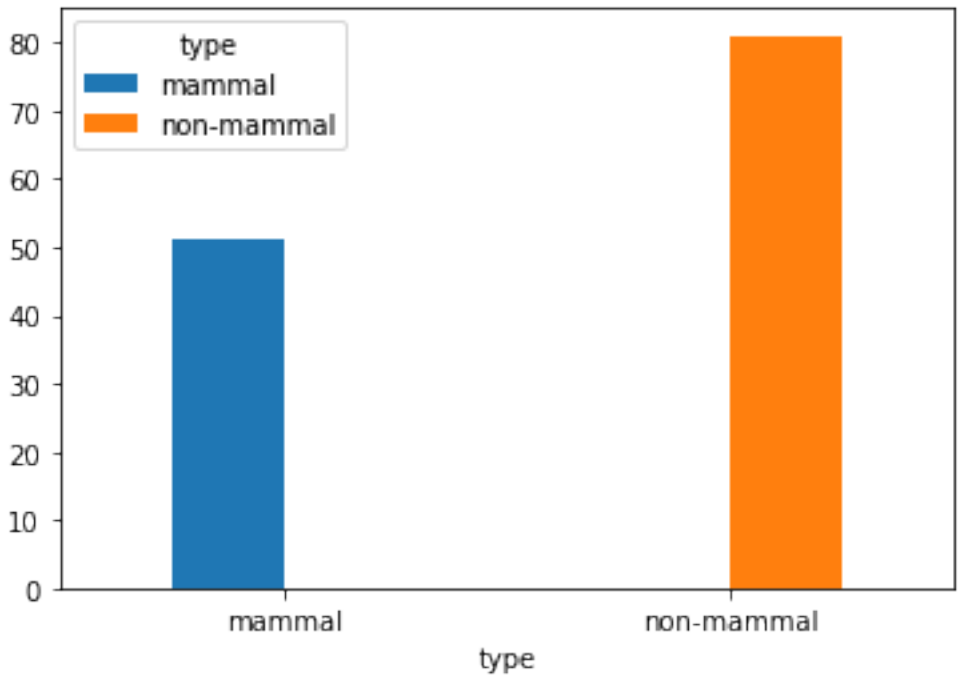
[36]:
```
crosstb =pd.crosstab(animal['catsize'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



[37]:
```
xaxis=np.arange(0,len(animal))
yaxis=np.array(animal['gestation'])
plt.plot(xaxis,yaxis)
plt.title('gestation')
plt.ylabel('gestation')
plt.show()
```



[38]:
```
crosstb =pd.crosstab(animal['type'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```

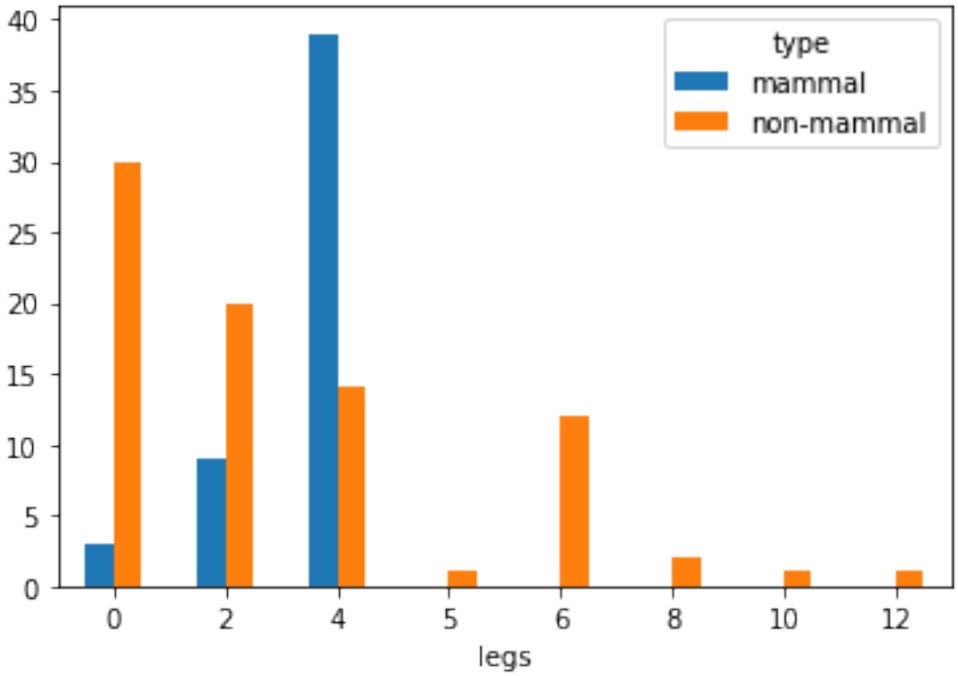


The Pie graph shows the total percentage of values of each type of animal available in the dataset.

[39]:
```
labels = ["NonMammal","Mammal"]
values = animal['type'].value_counts().tolist()
px.pie(animal, values=values, names=labels, title="Animal Class Type␣
↪Distribution Pie Chart")
```

Legs as factors

[40]:
```
crosstb =pd.crosstab(animal['legs'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```

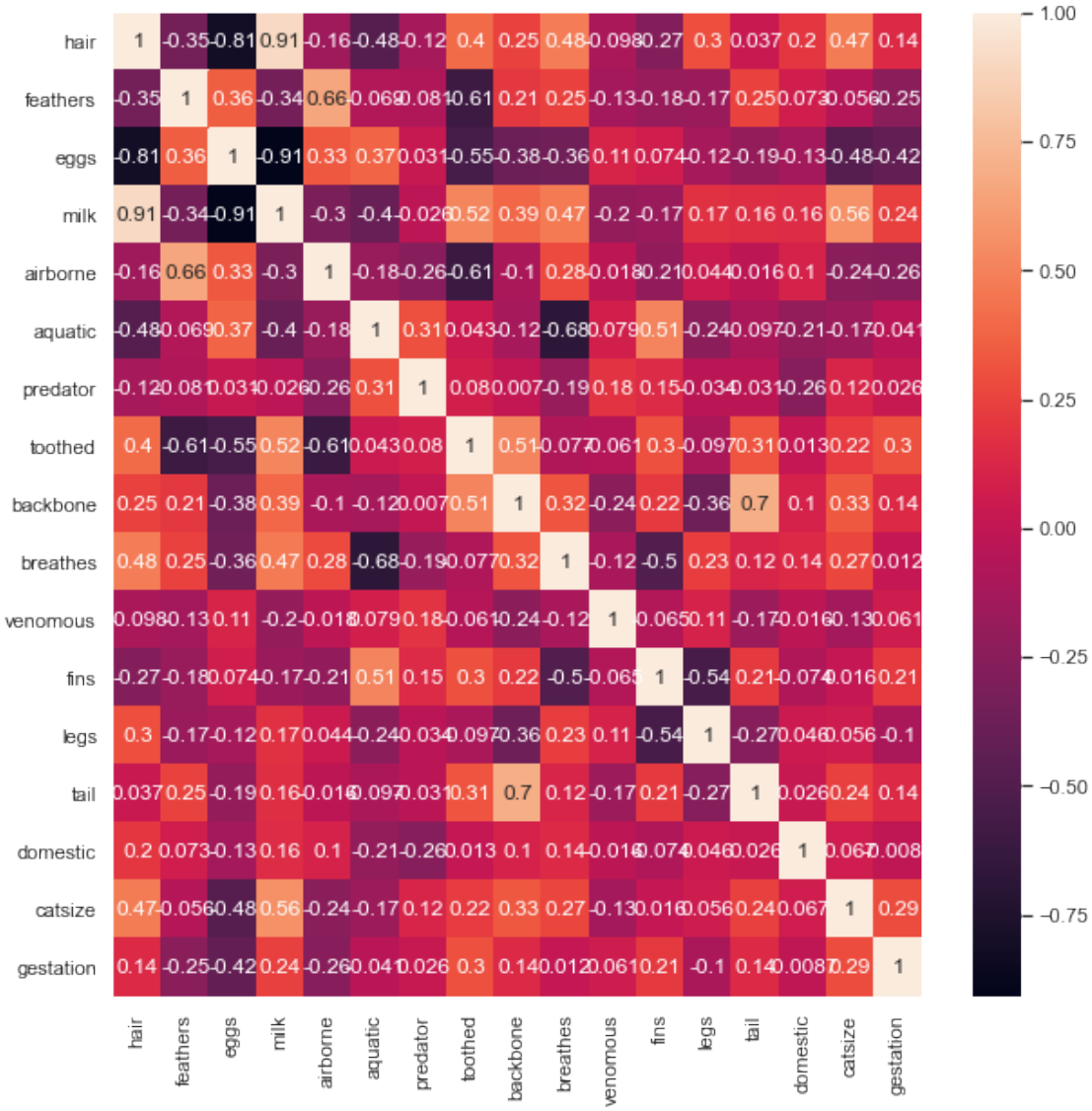


- Most of the animals have 4 legs and this can be taken into factor for classifying them as mammals or not.
- 0,2 legs have more values as non mammals so though there are certain species that are mammals with 0 and 2 legs this cannot be taken into consideration
- Least number of animals have 5 legs
- The animals with 5,6,8,10,12 are considered to be non mammals.
- we can write a RULE : if legs == 4 -> mammal if legs == 0/2/5/6/8/10/12 -> non-mammal

Identifying relationships To identify the other possible relations we will use Pearson coefficient.To identify relations among different features within a dataset.

[41]:
```
# Correlation Heatmap of the features in the dataset
plt.figure(figsize=(10,10))
sns.set(font_scale = 1)
```

```
sns.heatmap(animal.corr(),annot = True);
```



Discretizing gestation

[42]:
```
df=pd.DataFrame(animal['gestation'])
df['binned']=pd.cut(x=df['gestation'], bins=[0,100,200,300,400,500,600,700,800])
df['gestation_level']=pd.cut(x = df['gestation'], bins =␣
↪[0,100,200,300,400,500,600,700,800],labels = [0,1,2,3,4,5,6,7])
df
```

[42]:

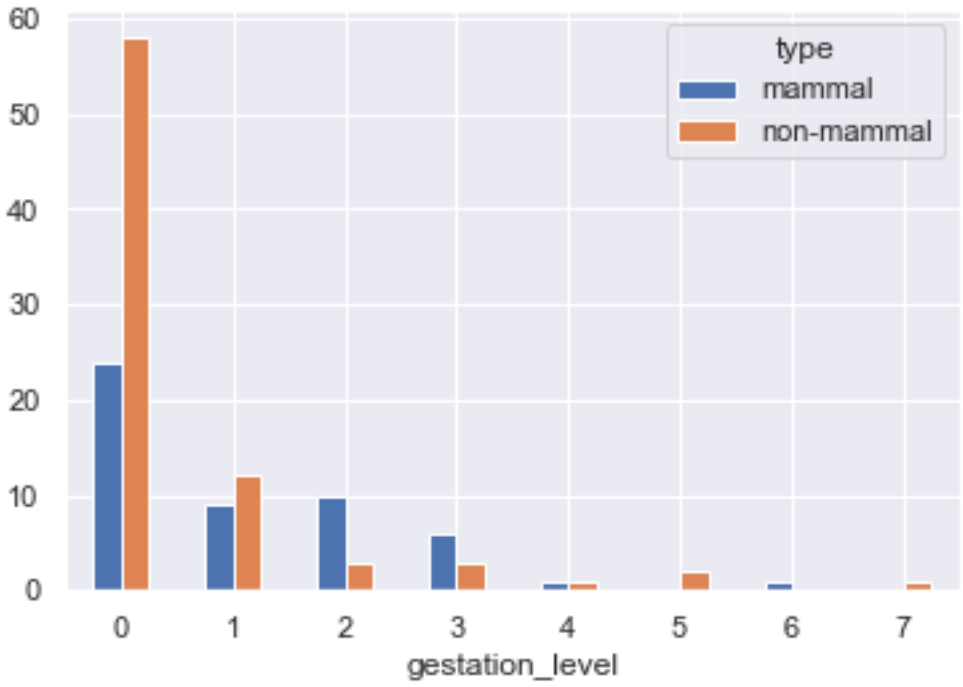
	gestation	binned	gestation_level
0	213.000000	(200, 300]	2
1	42.000000	(0, 100]	0

```
2    274.000000  (200, 300]               2
3     17.000000    (0, 100]               0
4      5.000000    (0, 100]               0
..          …          …              …
127   88.000000    (0, 100]               0
128  360.000000  (300, 400]               3
129   68.000000    (0, 100]               0
130  119.460317  (100, 200]               1
131   15.000000    (0, 100]               0

[132 rows x 3 columns]
```

[43]:
```
animal['gestation_level']=df['gestation_level']
```

visualizing gestation_level

[44]:
```
crosstb =pd.crosstab(animal['gestation_level'],animal['type'])
barplot = crosstb.plot.bar(rot=0)
```



In the above table,as we can see 'gestation' is a continous attribute. Since the 1-R classifier only deals with nominal values which are not continous , we can safely omit this feature from our classifier, and therefore, it's having missing values should not pose any issues. Therefore, the feature will not be included in my dataframe

[45]:
```
animal = animal.drop(columns=['gestation'])
animal.head()
```

[45]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes	venomous	fins	legs	tail	domestic	catsize	type	gestation_level
0	aardvark	True	False	False	True	False	False	True	True	True	True	False	False	4	False	False	True	mammal	2
1	anole	False	False	True	False	False	False	False	True	True	True	False	False	4	True	False	False	non-mammal	0
2	antelope	True	False	False	True	False	False	False	True	True	True	False	False	4	True	False	True	mammal	2
3	axolotl	False	False	True	False	False	True	False	True	True	False	False	False	4	True	False	False	non-mammal	0
4	bass	False	False	True	False	False	True	True	True	True	False	False	True	0	True	False	False	non-mammal	0

0.6 Mining Analytics

0.6.1 Training and Testing split

Spliting the data into Train and Test data

[46]:
```
from sklearn.model_selection import train_test_split
X=animal[['animal␣
 ↪name','hair','eggs','milk','feathers','airborne','aquatic','predator','toothed','backbone',
y=animal[['type']]
X_train, X_test, Y_train, Y_test = train_test_split(X, y, test_size=0.4,␣
 ↪random_state=1,shuffle=True)
```

0.6.2 Frequency Tables

Generating label frequencies for all the attributes

[47]:
```
freq_tab=animal['animal name'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'animal name','values':mykeys,'Frequecy':
 ↪myval})
freq_tab2=pd.crosstab(animal['animal name'],animal['type'])
display(freq_tab)
```

```
display(freq_tab2)
```

```
       Attribute          values  Frequecy
0    animal name            goat         1
1    animal name             fox         1
2    animal name        housefly         1
3    animal name         opossum         1
4    animal name            gull         1
..           …             …       …
127  animal name         penguin         1
128  animal name           human         1
129  animal name           anole         1
130  animal name        dartfrog         1
131  animal name  komodo dragon         1

[132 rows x 3 columns]
```

```
type         mammal  non-mammal
animal name
aardvark          1           0
anole             0           1
antelope          1           0
axolotl           0           1
bass              0           1
…               …           …
wasp              0           1
whale shark       0           1
wolf              1           0
worm              0           1
wren              0           1

[132 rows x 2 columns]
```

[48]:
```
freq_tab=animal['hair'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab= pd.DataFrame({'Attribute':'hair','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['hair'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0      hair  False        79
1      hair   True        53
```

```
type   mammal  non-mammal
hair
False       2          77
True       49           4
```

[49]:
```
freq_tab=animal['feathers'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'feathers','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['feathers'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0  feathers  False       112
1  feathers   True        20
```

```
type      mammal  non-mammal
feathers
False         51          61
True           0          20
```

[50]:
```
freq_tab=animal['eggs'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'eggs','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['eggs'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0      eggs   True        77
1      eggs  False        55
```

```
type   mammal  non-mammal
eggs
False      50           5
True        1          76
```

[51]:
```
freq_tab=animal['aquatic'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'aquatic','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['aquatic'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0   aquatic  False        82
1   aquatic   True        50
```

```
type     mammal  non-mammal
aquatic
False        44          38
```

```
True          7          43
```

[52]:
```
freq_tab=animal['predator'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'predator','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['predator'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0  predator   True        72
1  predator  False        60
```

```
type      mammal  non-mammal
predator
False         24          36
True          27          45
```

[53]:
```
freq_tab=animal['toothed'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'toothed','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['toothed'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0   toothed   True        89
1   toothed  False        43
```

```
type     mammal  non-mammal
toothed
False         1          42
True         50          39
```

[54]:
```
freq_tab=animal['backbone'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'backbone','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['backbone'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0  backbone   True       106
1  backbone  False        26
```

```
type      mammal  non-mammal
backbone
```

```
False          0          26
True          51          55
```

[55]:
```
freq_tab=animal['breathes'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'breathes','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['breathes'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0  breathes   True        98
1  breathes  False        34
```

```
type      mammal  non-mammal
breathes
False          0          34
True          51          47
```

[56]:
```
freq_tab=animal['venomous'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'venomous','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['venomous'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0  venomous  False       120
1  venomous   True        12
```

```
type      mammal  non-mammal
venomous
False         50          70
True           1          11
```

[57]:
```
freq_tab=animal['fins'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'fins','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['fins'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0      fins  False       111
1      fins   True        21
```

```
type   mammal  non-mammal
fins
False      47          64
True        4          17
```

[58]:
```
freq_tab=animal['legs'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'legs','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['legs'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute  values  Frequecy
0      legs       4        53
1      legs       0        33
2      legs       2        29
3      legs       6        12
4      legs       8         2
5      legs       5         1
6      legs      10         1
7      legs      12         1
```

```
type  mammal  non-mammal
legs
0          3          30
2          9          20
4         39          14
5          0           1
6          0          12
8          0           2
10         0           1
12         0           1
```

[59]:
```
freq_tab=animal['tail'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'tail','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['tail'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0      tail   True        97
1      tail  False        35
```

```
type   mammal  non-mammal
tail
False      9          26
```

```
True       42          55
```

[60]:
```
freq_tab=animal['domestic'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'domestic','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['domestic'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0  domestic  False       119
1  domestic   True        13
```

```
type      mammal  non-mammal
domestic
False         43          76
True           8           5
```

[61]:
```
freq_tab=animal['catsize'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'catsize','values':mykeys,'Frequecy':myval})
freq_tab2=pd.crosstab(animal['catsize'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
  Attribute values  Frequecy
0   catsize  False        84
1   catsize   True        48
```

```
type     mammal  non-mammal
catsize
False        15          69
True         36          12
```

[62]:
```
freq_tab=animal['gestation_level'].value_counts()
mykeys=freq_tab.keys()
myval=freq_tab.values
freq_tab=pd.DataFrame({'Attribute':'gestation_level','values':mykeys,'Frequecy':
 ↪myval})
freq_tab2=pd.crosstab(animal['gestation_level'],animal['type'])
display(freq_tab)
display(freq_tab2)
```

```
         Attribute values  Frequecy
0  gestation_level      0        82
1  gestation_level      1        21
2  gestation_level      2        13
```

```
3  gestation_level      3         9
4  gestation_level      4         2
5  gestation_level      5         2
6  gestation_level      6         1
7  gestation_level      7         1
```

type	mammal	non-mammal
gestation_level		
0	24	58
1	9	12
2	10	3
3	6	3
4	1	1
5	0	2
6	1	0
7	0	1

The Frequency table is generated from the above data Attribute

labels

Mammal

Non-mammal

hair

True

49

4

False

2

77

eggs

True

1

76

False

50

5

milk

True

51

0

False

0

81

feathers

True

0

20

False

51

61

Airborne

True

2

23

False

49

58

aquatic

True

7

43

False

44

38

predator

True

27

45

False

24

36

Toothed

True

50

39

False

1

42

backbone

True

51

55

False

0

26

breathes

True

51

47

False

0

34

venomous

True

1

11

False

50

70

fins

True

4

17

False

47

64

tail

True

42

55

False

9

26

domestic

True

8

5

False

43

46

catsize

True

36

12

False

15

69

legs

0

3

30

2

9

20

4

39

14

5

0

1

6

0

12

8

0

2

10

0

1

12

0

1

gestation_level

0

24

58

1

9

12

2

10

3

3

6

3

4

1

1

5

0

2

6

1

0

7

0

1

0.6.3 Rules

The rules are made from the Frequency tables data. #### Hair attribute RULE : if hair == True -> mammal if hair == False -> non-mammal

Eggs attribute RULE: if eggs == False -> mammal if eggs == True -> non-mammal

Milk Attribute RULE: if milk == True -> mammal if milk == False -> non-mammal

Aquatic attribute RULE : if aquatic == False -> mammal if aquatic == True -> non-mammal

Toothed Attribute RULE : if toothed == True -> mammal if aquatic == False -> non-mammal

Breathes attributes RULE : if breathes == True -> mammal if breathes == False -> non-mammal

Domestic attribute RULE: if domestic == True -> Mammal if domestic == False -> non-mammal

Catsize attribute RULE : if catsize == True -> mammal if catsize == False -> non-mammal

Legs attribute RULE : if legs == 4 -> mammal if legs == 0/2/5/6/8/10/12 -> non-mammal

Feathers attribute RULE: if feathers == true or false -> non-mammals

Airborne attribute RULE: if airborne == true or false -> non-mammals

Predator attribute RULE: if predator == true or false -> non-mammals

venoumous attribute RULE: if venomous == true or false -> non-mammals

Backbone attribute RULE: if backbone == true or false -> non-mammals

Tail attribute RULE: if tail == true or false -> non-mammals

Fins attribute RULE: if fins == true or false -> non-mammals

gestation_level attribute RULE : if gestation_level == 2 -> mammal if legs == 0/1/3/4/5/6/7 -> non-mammal

0.7 Evaluation

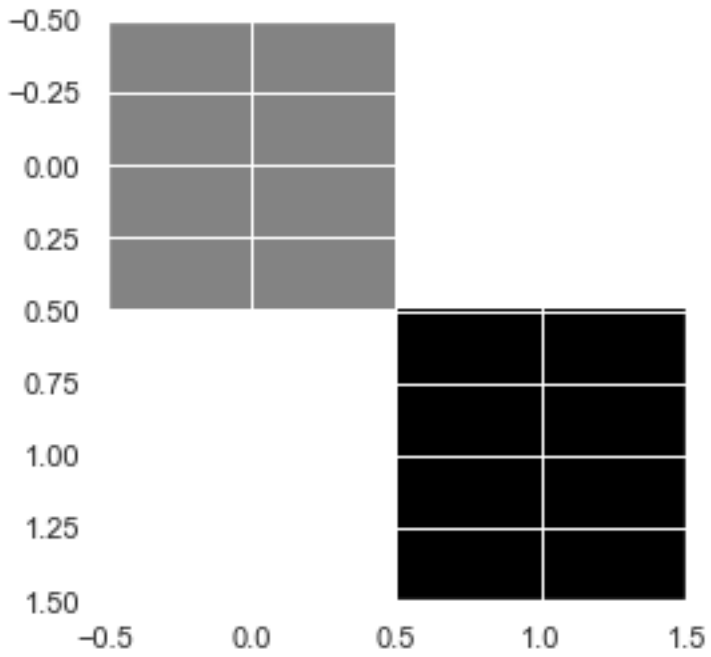
[63]:
```
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix

Yarray = np.array(Y_test)
```

Hair attribute RULE : if hair == True -> mammal if hair == False -> non-mammal

[64]:
```
hairid = X_test.columns.get_loc('hair')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,hairid]== True:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("Hair F1 score:", f1_score(Yarray, pred, average='macro'))
print("Hair Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Hair F1 score: 0.9579365079365079
Hair Accuracy score:  0.9622641509433962
[[17  1]
 [ 1 34]]
```

[64]: <matplotlib.image.AxesImage at 0x23584fd94f0>

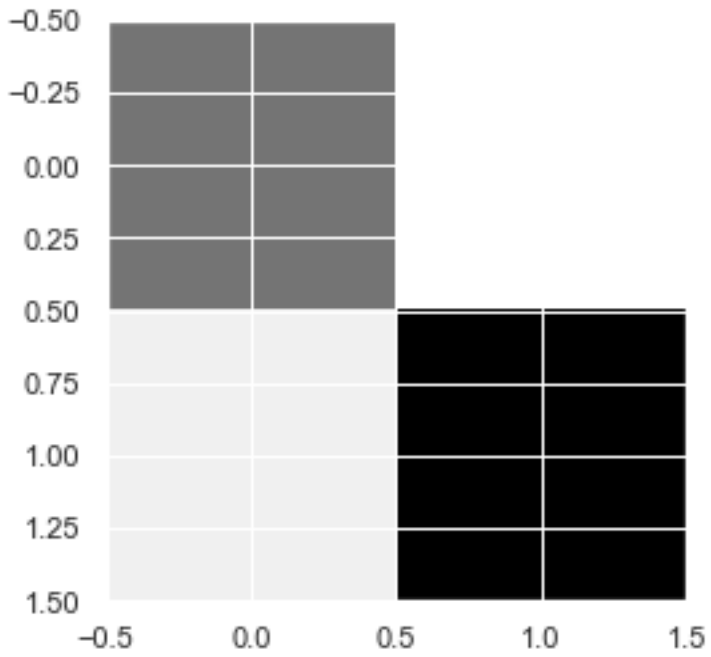


Eggs attribute RULE: if eggs == False -> mammal if eggs == True -> non-mammal

[65]:
```
eggsid = X_test.columns.get_loc('eggs')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,eggsid]== False:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("Eggs F1 score:", f1_score(Yarray, pred, average='macro'))
print("Eggs Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Eggs F1 score: 0.9589783281733746
Eggs Accuracy score:  0.9622641509433962
[[18  0]
 [ 2 33]]
```

[65]: `<matplotlib.image.AxesImage at 0x235873fb760>`

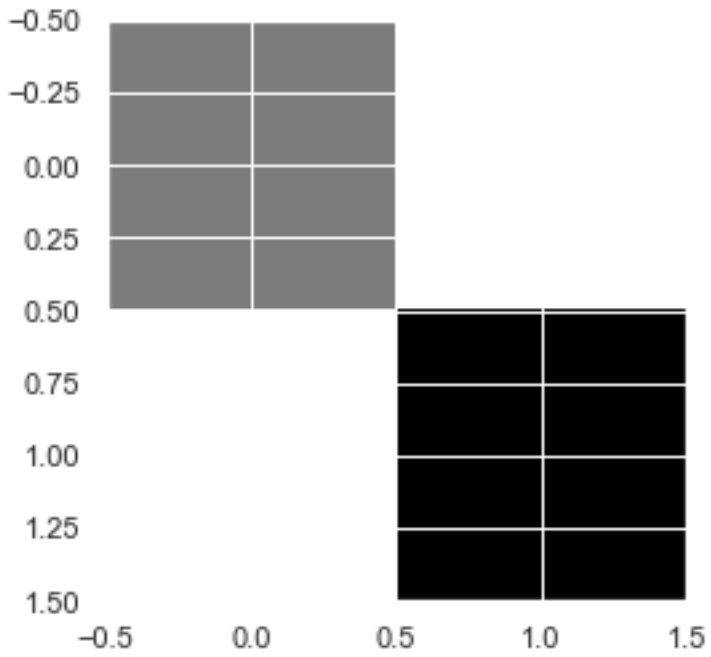


Milk Attribute RULE: if milk == True -> mammal if milk == False -> non-mammal

[66]:
```
milkid = X_test.columns.get_loc('milk')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,milkid]== True:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("Milk F1 score:", f1_score(Yarray, pred, average='macro'))
print("Milk Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Milk F1 score: 1.0
Milk Accuracy score:  1.0
[[18  0]
 [ 0 35]]
```

[66]: <matplotlib.image.AxesImage at 0x23586fe58e0>

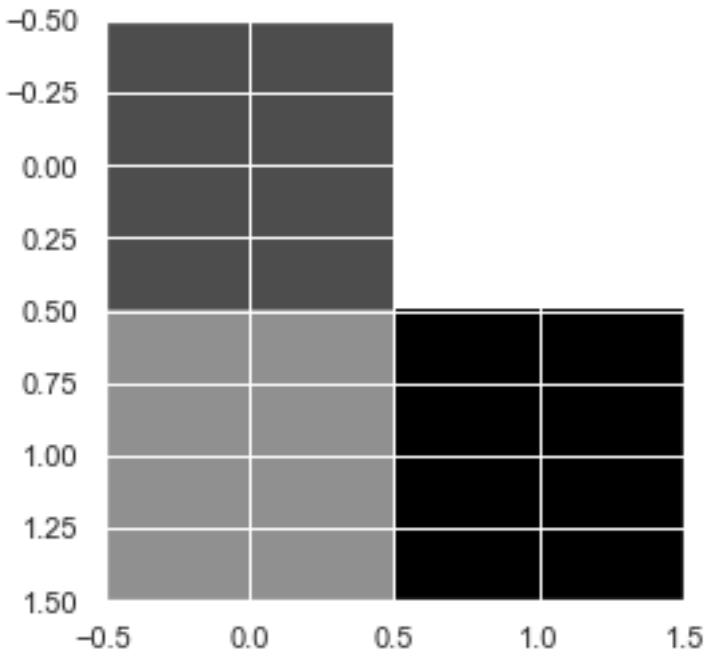


Aquatic attribute RULE : if aquatic == False -> mammal if aquatic == True -> non-mammal

[67]:
```
aquaid = X_test.columns.get_loc('aquatic')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,aquaid]== False:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("Aquatic F1 score:", f1_score(Yarray, pred, average='macro'))
print("Aquatic Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Aquatic F1 score: 0.7695652173913043
Aquatic Accuracy score:  0.7735849056603774
[[17  1]
 [11 24]]
```

[67]: `<matplotlib.image.AxesImage at 0x235873b7820>`

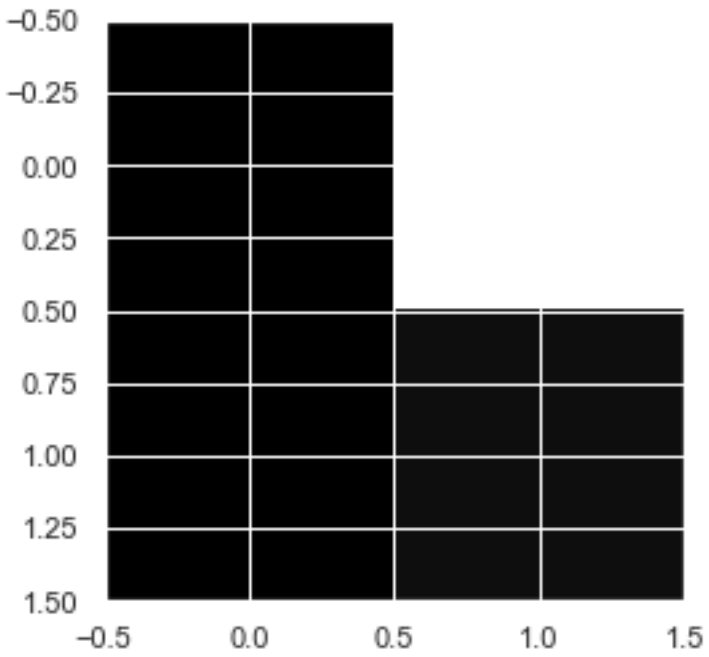


Toothed Attribute RULE : if toothed == True -> mammal if aquatic == False -> non-mammal

[68]:
```
toothid = X_test.columns.get_loc('toothed')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,toothid]== True:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("Toothed F1 score:", f1_score(Yarray, pred, average='macro'))
print("Toothed Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Toothed F1 score: 0.6602564102564102
Toothed Accuracy score:  0.660377358490566
[[18  0]
 [18 17]]
```

[68]: <matplotlib.image.AxesImage at 0x23586f4ea60>

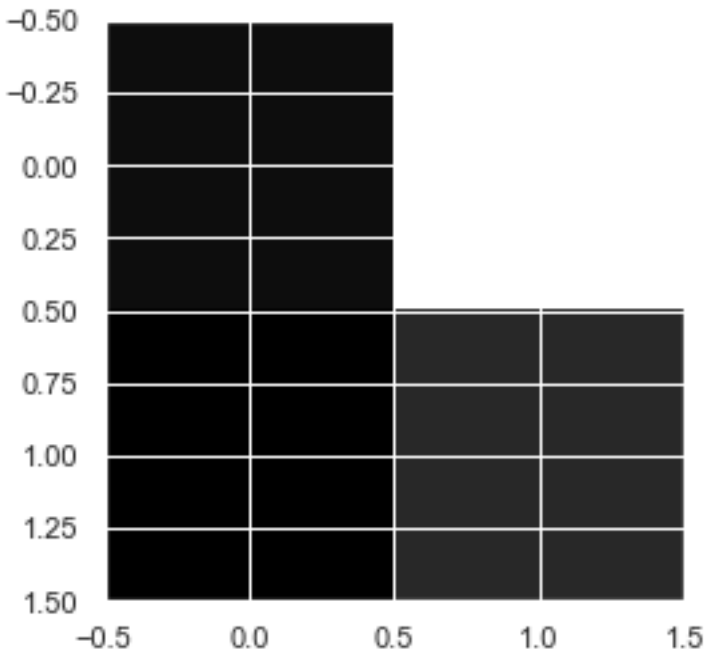


Breathes attributes RULE : if breathes == True -> mammal if breathes == False -> non-mammal

[69]:
```
breathid = X_test.columns.get_loc('breathes')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,breathid]== True:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("Breathes F1 score:", f1_score(Yarray, pred, average='macro'))
print("Breathes Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Breathes F1 score: 0.6409982174688058
Breathes Accuracy score:  0.6415094339622641
[[18  0]
 [19 16]]
```

[69]:
```
<matplotlib.image.AxesImage at 0x235870e6cd0>
```

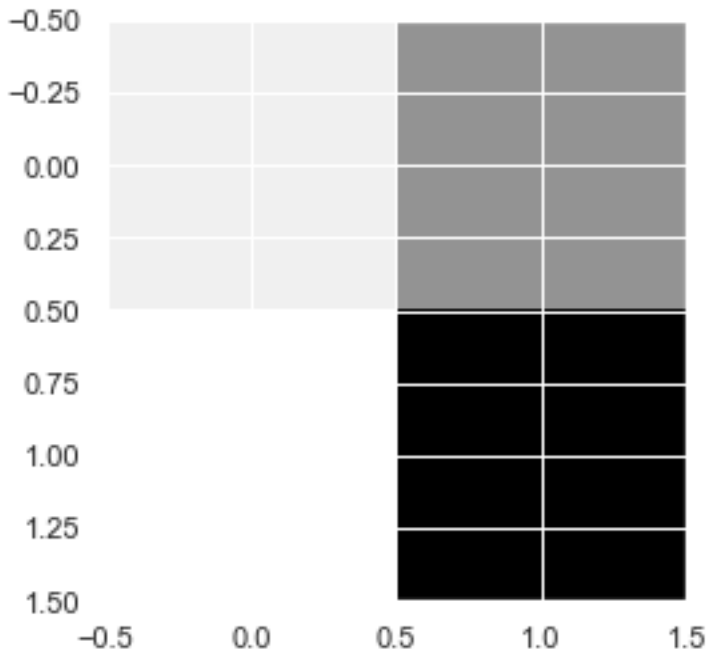


Domestic attribute RULE: if domestic == True -> Mammal if domestic == False -> non-mammal

[70]:
```
domesticid = X_test.columns.get_loc('domestic')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,domesticid]== True:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("Domestic F1 score:", f1_score(Yarray, pred, average='macro'))
print("Domestic Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Domestic F1 score: 0.5411255411255411
Domestic Accuracy score:  0.6981132075471698
[[ 3 15]
 [ 1 34]]
```

[70]: `<matplotlib.image.AxesImage at 0x23587138f70>`

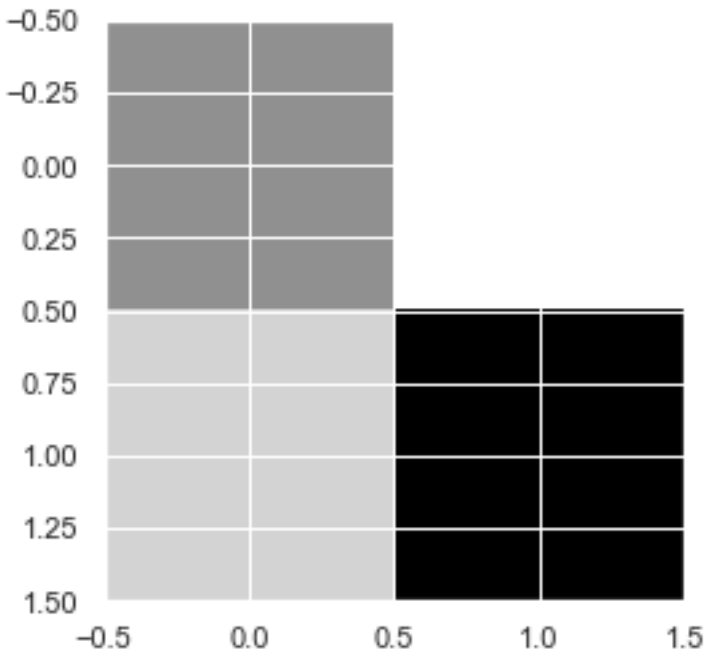


Catsize attribute RULE : if catsize == True -> mammal if catsize == False -> non-mammal

[71]:
```
catsizeid = X_test.columns.get_loc('catsize')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,catsizeid]== True:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("Catsize F1 score:", f1_score(Yarray, pred, average='macro'))
print("Catsize Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Catsize F1 score: 0.7590909090909091
Catsize Accuracy score:  0.7735849056603774
[[14  4]
 [ 8 27]]
```

[71]: `<matplotlib.image.AxesImage at 0x23587184cd0>`

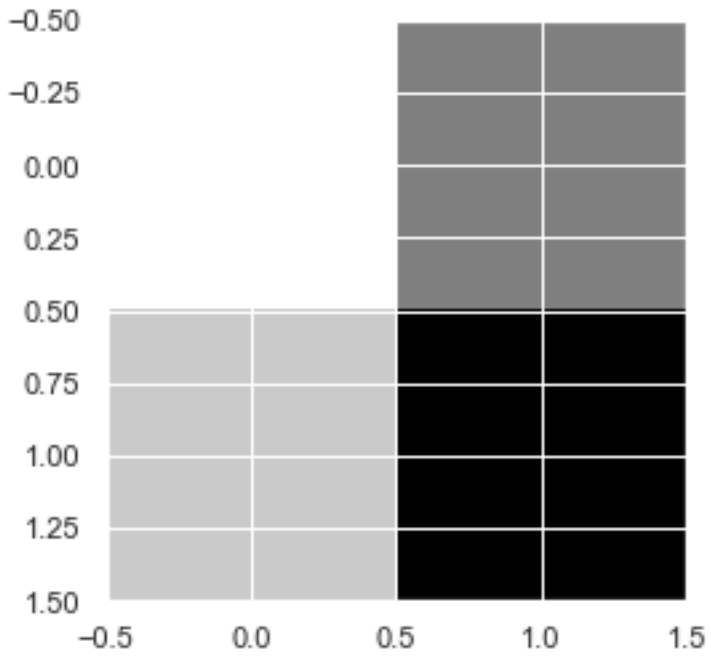


Legs attribute RULE : if legs == 4 -> mammal if legs == 0/2/5/6/8/10/12 -> non-mammal

[72]:
```
legsid = X_test.columns.get_loc('legs')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,legsid]== 2:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("Legs F1 score:", f1_score(Yarray, pred, average='macro'))
print("Legs Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Legs F1 score: 0.4540976265114196
Legs Accuracy score:  0.5660377358490566
[[ 3 15]
 [ 8 27]]
```

[72]: <matplotlib.image.AxesImage at 0x235871e32e0>



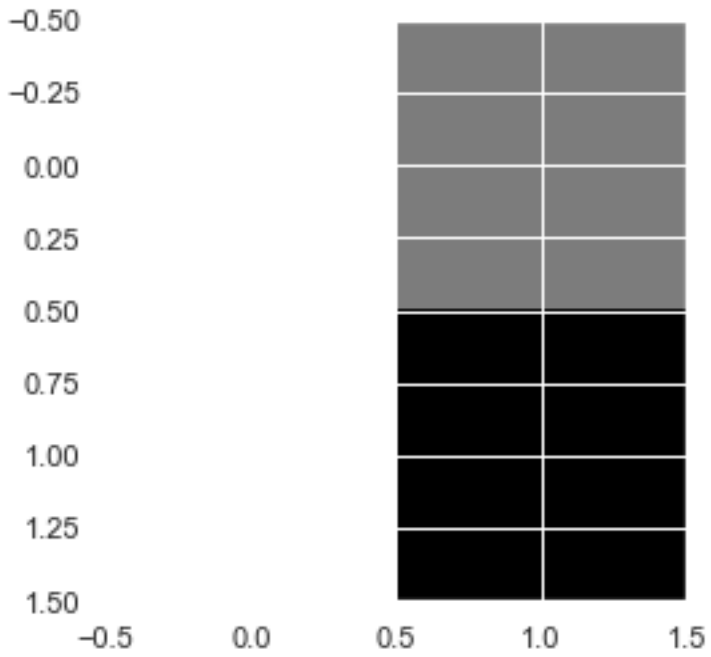
For the attributes feathers, airborne, predator, backbone, venomous, fins and tail, both True and False values showing majority in Non-mammals. Therefore, for these categories True == non-mammal and also False == non-mammal

Feathers attribute RULE: if feathers == true or false -> non-mammals

[73]:
```
featherid = X_test.columns.get_loc('feathers')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,featherid]== True or X_test.iloc[r,featherid]== False:
        pred.append('non-mammal')
print("Feathers F1 score:", f1_score(Yarray, pred, average='macro'))
print("Feathers Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Feathers F1 score: 0.3977272727272727
Feathers Accuracy score:  0.660377358490566
[[ 0 18]
 [ 0 35]]
```

[73]: `<matplotlib.image.AxesImage at 0x23587237100>`

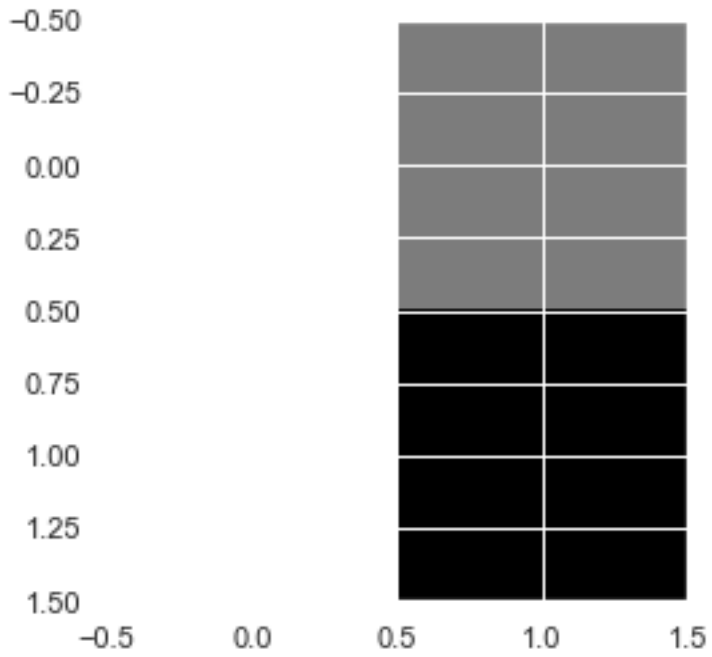


Airborne attribute RULE: if airborne == true or false -> non-mammals

[74]:
```
airid = X_test.columns.get_loc('airborne')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,airid]== True or X_test.iloc[r,airid]== False:
        pred.append('non-mammal')
print("Airborne F1 score:", f1_score(Yarray, pred, average='macro'))
print("Airborne Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Airborne F1 score: 0.3977272727272727
Airborne Accuracy score:  0.660377358490566
[[ 0 18]
 [ 0 35]]
```

[74]: <matplotlib.image.AxesImage at 0x235872881c0>

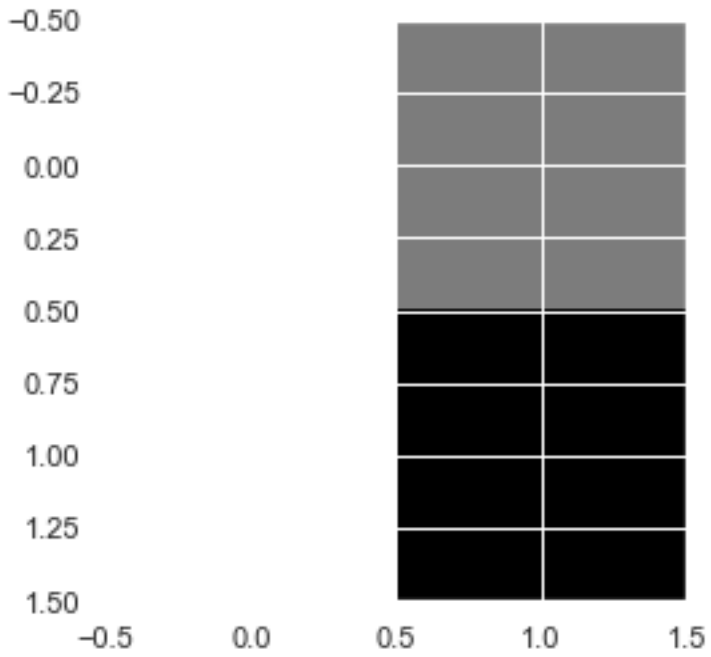


Predator attribute RULE: if predator == true or false -> non-mammal

[75]:
```
predid = X_test.columns.get_loc('predator')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,predid]== True or X_test.iloc[r,predid]== False:
        pred.append('non-mammal')
print("Predator F1 score:", f1_score(Yarray, pred, average='macro'))
print("Predator Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Predator F1 score: 0.3977272727272727
Predator Accuracy score:  0.660377358490566
[[ 0 18]
 [ 0 35]]
```

[75]: <matplotlib.image.AxesImage at 0x235872d8220>

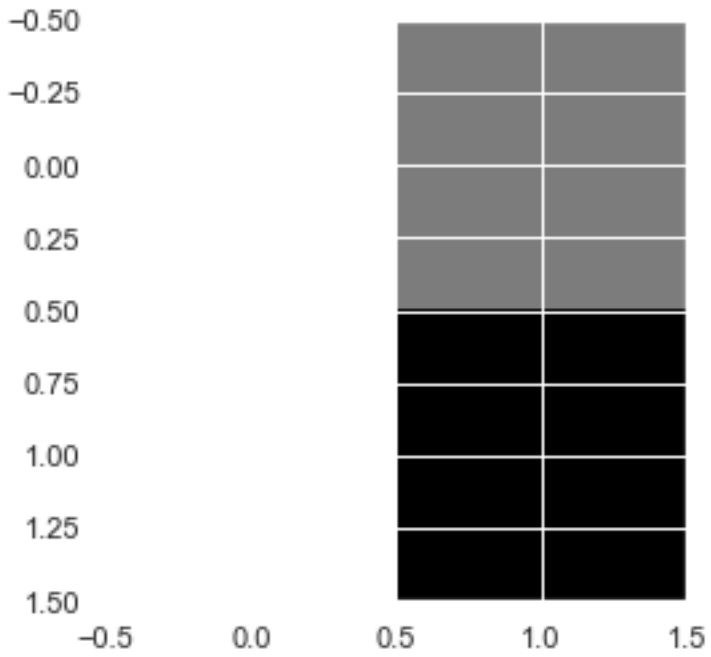


Backbone attribute RULE : if backbone == true or false -> non-mammal

[76]:
```
bbid = X_test.columns.get_loc('backbone')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,bbid]== True or X_test.iloc[r,bbid]== False:
        pred.append('non-mammal')
print("Backbone F1 score:", f1_score(Yarray, pred, average='macro'))
print("Backbone Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Backbone F1 score: 0.3977272727272727
Backbone Accuracy score:  0.660377358490566
[[ 0 18]
 [ 0 35]]
```

[76]:
```
<matplotlib.image.AxesImage at 0x2358749a310>
```

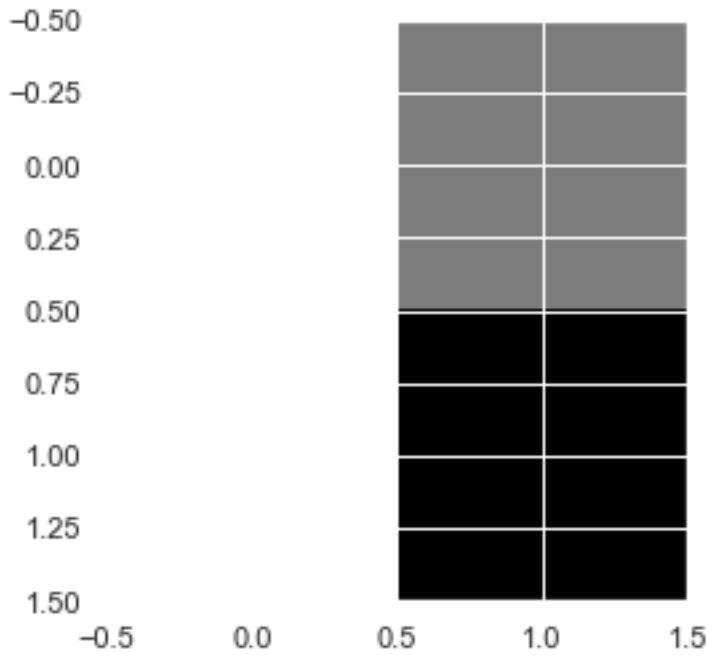


Venomous attribute RULE : if venomous == True or False -> non-mammal

[77]:
```
vid = X_test.columns.get_loc('venomous')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,vid]== True or X_test.iloc[r,vid]== False:
        pred.append('non-mammal')
print("Venomous F1 score:", f1_score(Yarray, pred, average='macro'))
print("Venomous Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Venomous F1 score: 0.3977272727272727
Venomous Accuracy score:  0.660377358490566
[[ 0 18]
 [ 0 35]]
```

[77]:
```
<matplotlib.image.AxesImage at 0x235874e97c0>
```

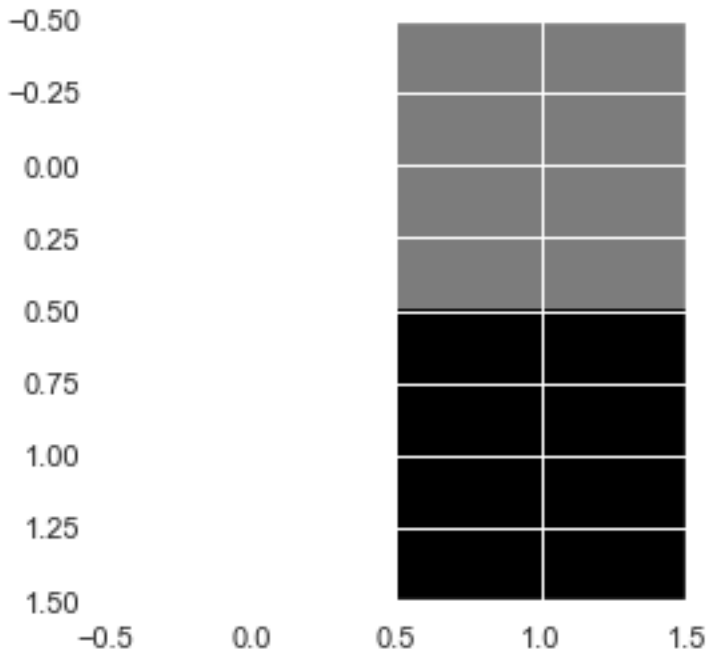


Tail attribute RULE : if tail == true or false -> non-mammal

[78]:
```
tid = X_test.columns.get_loc('tail')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,tid]== True or X_test.iloc[r,tid]== False:
        pred.append('non-mammal')
print("Tail F1 score:", f1_score(Yarray, pred, average='macro'))
print("Tail Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Tail F1 score: 0.3977272727272727
Tail Accuracy score:  0.660377358490566
[[ 0 18]
 [ 0 35]]
```

[78]:
```
<matplotlib.image.AxesImage at 0x23587536d00>
```

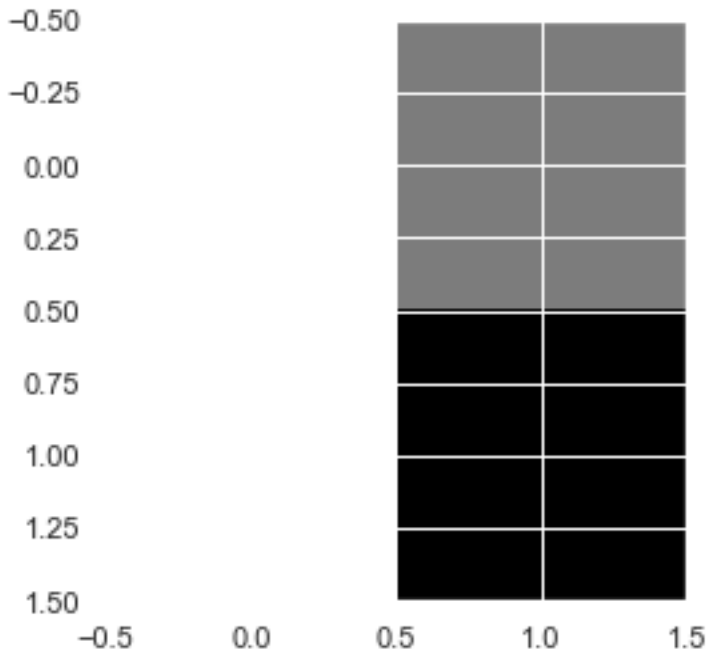


Fins attribute RULE : if fins == true or false -> non-mammal

[79]:
```
fid = X_test.columns.get_loc('fins')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,fid]== True or X_test.iloc[r,fid]== False:
        pred.append('non-mammal')
print("Fins F1 score:", f1_score(Yarray, pred, average='macro'))
print("Fins Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
Fins F1 score: 0.3977272727272727
Fins Accuracy score:  0.660377358490566
[[ 0 18]
 [ 0 35]]
```

[79]: <matplotlib.image.AxesImage at 0x23587590190>

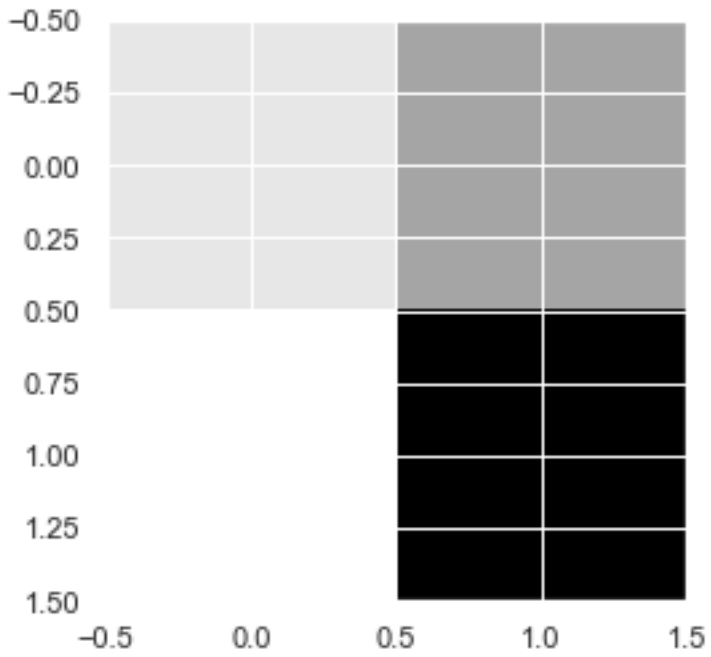


gestation_level RULE : if gestation_level == 2 -> mammal if legs == 0/1/3/4/5/6/7 -> non-mammal

[80]:
```
gid = X_test.columns.get_loc('gestation_level')
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,gid]== 2:
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("gestation_level F1 score:", f1_score(Yarray, pred, average='macro'))
print("gestation_level Legs Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)
plt.imshow(cmatrix, cmap='binary')
```

```
gestation_level F1 score: 0.6074074074074074
gestation_level Legs Accuracy score:  0.7169811320754716
[[ 5 13]
 [ 2 33]]
```

[80]:
```
<matplotlib.image.AxesImage at 0x23585053d60>
```



The f1-scores and accuracies among the attributes feathers, airborne, predator, backbone, venomous, fins and tail match. This is becuase, as per the rules, both True and False indicate 'non-mammals'. This means that the 1-R rule for each of these attributes wrongly classfies all the 'mammals', and correctly classifies all the 'non-mammals'. Since the number of mammals and non-mammals are fixed, the accuracies and f1-scores also dont change.

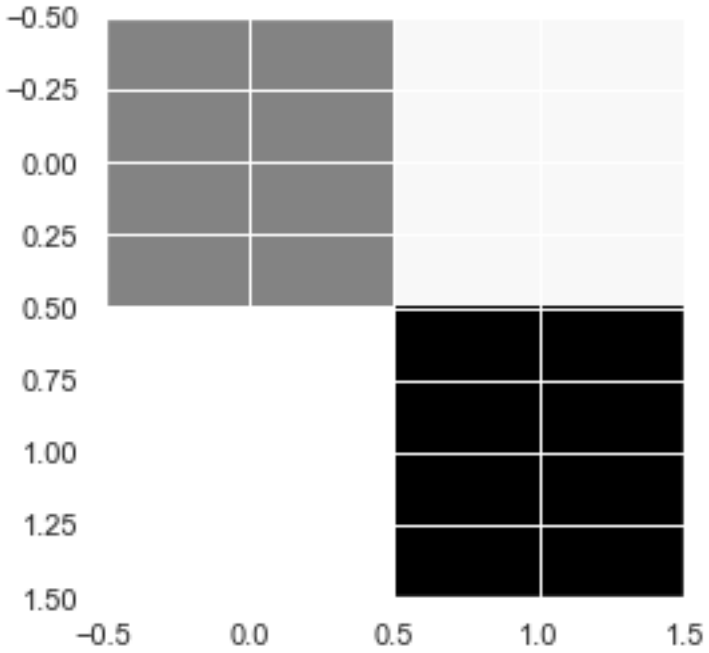
0.7.1 1-R classifier

[81]:
```
pred=[]
for r in range(0,len(X_test)):
    if X_test.iloc[r,hairid] == True and X_test.iloc[r,milkid]==True and X_test.
 ↪iloc[r,eggsid]==False and X_test.iloc[r,aquaid]==False :
        pred.append('mammal')
    else:
        pred.append('non-mammal')

print("1-R F1 score:", f1_score(Yarray, pred, average='macro'))
print("1-R Accuracy score: ",accuracy_score(Yarray,pred))
cmatrix = confusion_matrix(Yarray, pred)
print(cmatrix)

plt.imshow(cmatrix, cmap='binary')
```

```
1-R F1 score: 0.9786720321931589
1-R Accuracy score:  0.9811320754716981
[[17  1]
```

```
 [ 0 35]]
```

[81]: <matplotlib.image.AxesImage at 0x23584ffc880>



0.8 Results:

Rules related to Hair, Milk, Eggs, and Aquatic provide the highest accuracy and F1-score of all rules. Therefore, these features will result in the most accurate predictions when they are used in the classifier. So, here are the rules and their scores: Set of 1-R classifier rules:

Hair -> mammal f1 : 0.957, acc : 0.962

Milk -> mammal f1: 1.0 acc: 1.0

Eggs -> non-mammal f1: 0.958 acc: 0.962

Aquatic -> non-mammal f1-0.75 acc:0.76

I found that the accuracy of the classifier with combined rules was 98.1 percent when using the 1-R rule set.

0.9 References:

Anaconda Cloud. Wordcloud. Retrieved (2022, January 27) from https://anaconda.org/conda-forge/wordcloud

Python pool. (2021). Retrieved (2022, February 17) from https://www.pythonpool.com/matplotlib-figsize

Justify the text in jupyter.Retrieved (2022, February 17) from https://stackoverflow.com/questions/35077507/how-to-right-align-and-justify-align-in-markdown

NumPy.Retrieved (2022, February 17) from https://numpy.org/doc/stable/index.html

Pandas Package. Retrieved (2022, February 17) from https://pandas.pydata.org/

Matplotlib.Retrieved (2022, February 17) from https://matplotlib.org/

Seaborn.Retrieved (2022, February 17) from https://seaborn.pydata.org/

Sk learn. Retrieved (2022, February 17) from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

iloc.Retrieved (2022, February 17) from https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.htn

Using pandas crosstab to create a bar plot.Retrieved(2022, February 28) from https://www.geeksforgeeks.org/using-pandas-crosstab-to-create-a-bar-plot/#

How to use confusion matrix in python example.Retrieved (2022, February 28) from https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/

Confusion Matrix.Retrieved (2022, February 28)from http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

One R. Retrieved (2022, February 28) from https://www.saedsayad.com/oner.htm