Name:	Bikis Muhammed
Email:	bmpb8@mst.edu
Course:	CS 5402
Assignment:	Classification System
Date:	2021-07-08

In [434]:
```
# For working with dataframe
import pandas as pd
# Working with numbers
import numpy as ny
# Imported for spliting a data to traing and test dataset
from sklearn.model_selection import train_test_split
# Imported for countplot
import seaborn as sns
# Imported for drawing plots
import matplotlib.pyplot as plt
# For imputing missing attribute values
from sklearn.impute import SimpleImputer
# Imported to create confusion matrix (did not used)
from sklearn.metrics import confusion_matrix
# Imported to find the accuracy score  value of the generated model
from sklearn.metrics import accuracy_score
# Imported to find precision, recall, and f1-score
from sklearn.metrics import classification_report
# For getting minute and second for file name
import time
# used to display dataframes in a single lines
from IPython.display import display, HTML, Markdown

# Source: Stackoverflow
CSS = """
.output {
    flex-direction: row;
}
"""
HTML('<style>{}</style>'.format(CSS))
```

Out[434]:

Concept Description:

This project aims to classify different unknown animals as mammals and non-mammals based on their attributes. It is a binary classifier, and almost all attribute of the given data is nominal. Therefore, a 1R classifier will be used to accomplish this task. The gestation and leg attributes are numeric, and we may do some discretization to make them vital for our classification.

Data Collection:

The data has been provided by Perry B. Koob as an excel file (in .xlsx extension).

Example Description:

Level of measurements	Discription
Nominal	Just a label name E.g. True/False
Ordinal	Name (Nominal) + Order Named label + ordering E.g. : Minimum < medium < Maximum
Interval	Nominal + Ordinal + Fixed distance between each attribute values and no true zero E.g. Tempreture in degree celcious (65 < 70 < 75)
Ratio	Nominal + Ordinal + Interval + Meaningful zero value E.g Number of students.

Attribute	Level of measurement	Attribute Values	Note
animal name	Nominal	Example: 'aardvark', 'anole','antelope','axolotl', ...	It describes whether the animal has hair or not. It is a just a label, and it does not have a sense of order.
hair	Nominal	True or False	True/False attribute values are a good indicator of nominal attribute
feathers	Nominal	True or False	Animals that are covered with feathers and not hairs
eggs	Nominal	True or False	Animal that lay eggs for reproduction
milk	Nominal	True or False	Animal that breastfead their child
airborne	Nominal	True or False	Animals that can fly
aquatic	Nominal	True or False	Live in the water
predator	Nominal	True or False	Prey or eat other animals for servival
toothed	Nominal	True or False	have teeth
backbone	Nominal	True or False	Have backbnone
breathes	Nominal	True or False	Animals that breathes
venomous	Nominal	True or False	Animals that produce poison
fins	Nominal	True or False	Mostly in aquatic animal that is used for stablity and easy manuvering
legs	Ordinal	'4', '0', '6', '2', '10', '8', '12', '5	There is a sense of order here
tail	Nominal	True or False	Animals that have tails
domestic	Nominal	True or False	animals that live with human
catsize	Nominal	True or False	catsize false are for small animal, and catsize true is for considerebly big animals
gestation	Interval data	Exmple: '213','42','274','17','5','220','115','334','210','150', ...	It is a numeric attribute, and it does not have a true zero. It is the time between conception and birth in days
type	Nominal	Example: 'mammal', 'reptile', 'amphibian', 'fish', 'insect', 'bird', 'arthropod', 'mamal'	Animal catagories

Example

Sealipon attributes: hair, milk, aquatic, predator, toothed, backbone, beathes, fines, 2 legs, tial, catsize,350 gestation, and mammal

Data Import and Wrangling:

In [435]:
```
data = pd.read_excel("../src-data/animal-taxonomy.xlsx", engine = "openpyxl")
data.sample(10)
```

Out[435]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes
54	housecat	True	False	False	True	False	False	True	True	True	True
120	tuna	False	False	True	False	False	True	True	True	True	False
124	wallaby	True	False	False	True	False	False	False	True	True	True
98	seahorse	False	False	True	False	False	True	False	True	True	False
37	fruitbat	True	False	False	True	True	False	False	True	True	True
48	hare	True	False	False	True	False	False	False	True	True	True
112	starfish	False	False	True	False	False	True	True	False	False	False
6	blue dragon	False	False	True	False	False	True	True	True	False	False
91	raccoon	True	False	False	True	False	False	True	True	True	True
78	ostrich	False	True	True	False	False	False	False	False	True	True

Describe the data

In [436]:
```
data.describe().T
```

Out[436]:

	count	unique	top	freq
animal name	130	130	cayman	1
hair	130	2	False	77
feathers	130	2	False	110
eggs	130	2	True	75
milk	130	2	False	79
airborne	130	2	False	106
aquatic	130	2	False	80
predator	130	2	True	71
toothed	130	2	True	88
backbone	130	2	True	105
breathes	130	2	True	96
venomous	130	2	False	118

Note:

- Attributes with 2 unique (True/False) values are definitely nominal (categorical) attributes.
- Animal name is also nominal attributes since it is just a label name.
- Type of an animal also categorical since it does not have any ordering.
- Legs and gestations are numerical attribute and they have zero values.
  - Gestation can be considered as ratio attribute.
  - Number of legs is also ratio attribute
- Since I am using 1R classifier, I can consider the leg attribute as a categorical/nominal attribute.
- In addition, the gestation class have many unique values, and it is not practical to classify element with this many classes.
  - Therefore, I will divide the the data into two gestation periods such as short gestation period and log gestation period.
- Besides, there are 5 missing values in the gestation attribute, and I will impute those values using a mean distribution.

Unique Values in the data

In [437]:
```
dictionary = {}
for column in list(data):
    dictionary[column] = list(data[column].unique())

dictionary
```

Out[437]:
```
{'animal name': ['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',
  'mink',
  'mole',
  'mongoose',
  'momo',
  'moth',
```

```
 'moth',
 'newt',
 'nessie',
 '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',
 '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'],
'hair': ['True', 'False'],
'feathers': ['False', 'True'],
'eggs': ['False', 'True'],
'milk': ['True', 'False'],
'airborne': ['False', 'True'],
'aquatic': ['False', 'True'],
'predator': ['True', 'False'],
'toothed': ['True', 'False'],
'backbone': ['True', 'False'],
'breathes': ['True', 'False'],
'venomous': ['False', 'True'],
'fins': ['False', 'True'],
'legs': ['4', '0', '6', '2', '10', '8', '12', '5'],
'tail': ['False', 'True'],
'domestic': ['False', 'True'],
'catsize': ['True', 'False'],
'gestation': ['213',
 '42',
 '274',
 '17',
 '5',
 '220',
 nan
```

```
 '330',
 '350',
 '34',
 '59',
 '40',
 '209',
 '88',
 '15'],
'type': ['mammal',
 'reptile',
 'amphibian',
 'fish',
 'insect',
 'bird',
 'arthropod',
 'mamal']}
```

The unique value of the type attribute has a value error (inaccurate value). For the type attribute value, **mammal** and **mamal** are used to reference the same thing. The correct spelling for this attribute example is **mammal** .

In [438]:
```
for i, d in data.iterrows():
    if d['type']  == 'mamal':
        data.at[i,'type'] = 'mammal'
```

In [439]:
```
display(data.describe().T)
```

aquatic	130	2	False	80
predator	130	2	True	71
toothed	130	2	True	88
backbone	130	2	True	105
breathes	130	2	True	96
venomous	130	2	False	118
fins	130	2	False	109
legs	130	8	4	53
tail	130	2	True	96
domestic	130	2	False	117
catsize	130	2	False	82
gestation	125	80	42	7
type	130	7	mammal	51

Check for missing values

Even though the 1R classifier assumes missing values as another "valid" value, I will just impute them.

In [440]:
```
data.isnull().sum().sort_values(ascending=False)
```

Out[440]:
```
gestation      5
animal name    0
breathes       0
catsize        0
domestic       0
tail           0
legs           0
fins           0
venomous       0
backbone       0
hair           0
toothed        0
predator       0
aquatic        0
airborne       0
milk           0
eggs           0
feathers       0
type           0
dtype: int64
```

There are 5 missing values in **gestation** attribute, and that value need imputaion.

In [441]:
```
imputer = SimpleImputer(missing_values=ny.nan,strategy = "mean")
imputer.fit(data['gestation'].values.reshape(-1, 1))
data['gestation'] = imputer.transform(data['gestation'].values.reshape(-1, 1
```

In [442]:
```
data.isnull().sum().sort_values(ascending=False)
```

Out[442]:
```
animal name    0
breathes       0
gestation      0
catsize        0
domestic       0
tail           0
legs           0
fins           0
venomous       0
backbone       0
hair           0
toothed        0
predator       0
aquatic        0
airborne       0
milk           0
eggs           0
feathers       0
type           0
dtype: int64
```

In [443]:
```
data.describe().loc[['count']]
```

Out[443]:

	gestation
count	130.0

There was 125 gestation values in the given data set. Now it is 130.

The value of gestation need fixing

The algorithm assumes all values are discrete. If not, they need discretization.

The value of gestation is numerical (interval), and it needs to be changed to categorical. The data set will be split in half. Animals with gestation value greater than or equal to 165 will be considered as longer gestation times and the rest will be considered as shorter gestation period.

In [444]:
```
# # xtrain['gestation'] = [1 if float(x) > 170  else 0 for x in xtrain['geste
for i, d in data.iterrows():
    if pd.to_numeric(d['gestation'])  > float( data['gestation'].mean()): #1
        data.at[i,'GESTATION'] = 'long'
    else:
        data.at[i,'GESTATION'] = 'short'
```

In [445]:
```
data = data.drop('gestation', 1)
data = data.rename(columns={'GESTATION': 'gestation'})
data
```

Out[445]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone
0	aardvark	True	False	False	True	False	False	True	True	True
1	anole	False	False	True	False	False	False	False	True	True
2	antelope	True	False	False	True	False	False	False	True	True
3	axolotl	False	False	True	False	False	True	False	True	True
4	bass	False	False	True	False	False	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...
125	wasp	True	False	True	False	True	False	False	False	False
126	whale shark	False	False	False	False	False	False	False	True	True
127	wolf	True	False	False	True	False	False	True	True	True

Exploratory Data Analysis:

Classify

In [446]:
```
data.columns
```

Out[446]:
```
Index(['animal name', 'hair', 'feathers', 'eggs', 'milk', 'airborne',
       'aquatic', 'predator', 'toothed', 'backbone', 'breathes', 'venomou
s',
       'fins', 'legs', 'tail', 'domestic', 'catsize', 'type', 'gestation'],
      dtype='object')
```

In [447]:
```
for i, d in data.iterrows():
    if d['type'] == 'mammal':
        data.at[i,'mammal'] = 'mammal'
    else:
        data.at[i,'mammal'] = 'non-mammal'
```

In [448]:
```
data.columns
```

Out[448]:
```
Index(['animal name', 'hair', 'feathers', 'eggs', 'milk', 'airborne',
       'aquatic', 'predator', 'toothed', 'backbone', 'breathes', 'venomou
s',
       'fins', 'legs', 'tail', 'domestic', 'catsize', 'type', 'gestation',
       'mammal'],
      dtype='object')
```

In [449]:
```
data
```

Out[449]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone
0	aardvark	True	False	False	True	False	False	True	True	True
1	anole	False	False	True	False	False	False	False	True	True
2	antelope	True	False	False	True	False	False	False	True	True
3	axolotl	False	False	True	False	False	True	False	True	True
4	bass	False	False	True	False	False	True	True	True	True
...	...	...	...	...	...	...	...	...	...	...
125	wasp	True	False	True	False	True	False	False	False	False
126	whale shark	False	False	False	False	False	False	False	True	True

Splitting the data to training and testing data set

In [450]:
```
y = data[['mammal']]
x = data.drop(columns=['type','mammal'])
xtrain, xtest, ytrain, ytest = train_test_split(x, y, train_size = 0.80, ran
```

In [451]:
```
xtrain.sample(10)
```

Out[451]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes
113	stingray	False	False	True	False	False	True	True	True	True	False
13	cavy	True	False	False	True	False	False	False	True	True	True
96	sea anemone	False	False	True	False	False	True	True	False	False	False
116	thylacine	True	False	False	True	False	False	True	True	True	True
61	leopard	True	False	False	True	False	False	True	True	True	True
86	platypus	True	False	True	True	False	True	True	False	True	True
30	dove	False	True	True	False	True	False	False	False	True	True
50	hellbender	False	False	True	False	False	True	True	True	True	False
52	honeybee	True	False	True	False	True	False	False	False	False	True

Out[452]:

	animal name	hair	feathers	eggs	milk	airborne	aquatic	predator	toothed	backbone	breathes
42	gorilla	True	False	False	True	False	False	False	True	True	True
67	mink	True	False	False	True	False	True	True	True	True	True
108	sole	False	False	True	False	False	True	False	True	True	False
46	hagfish	False	False	True	False	False	True	False	True	False	False
83	pike	False	False	True	False	False	True	True	True	True	False
88	porpoise	False	False	False	True	False	True	True	True	True	True
54	housecat	True	False	False	True	False	False	True	True	True	True
4	bass	False	False	True	False	False	True	True	True	True	False
44	gull	False	True	True	False	True	True	True	False	True	True
33	flamingo	False	True	True	False	True	False	False	False	True	True

Count Plot of Each Attribute

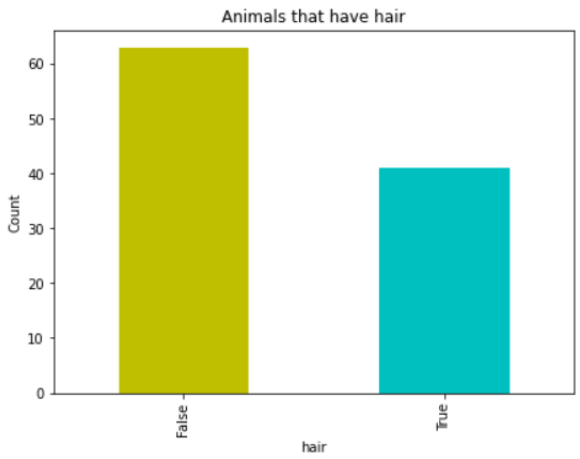
In [453]:
```
# def plot (attribute, dataValue):
#     bar = sns.countplot(x = attribute, data = dataValue)
#     bar.set_xticklabels(bar.get_xticklabels(), rotation=40, ha = 'right')
#     plt.tight_layout()
#     timeval = time.strftime("%M_%S")
#     plt.savefig("../generated-output/counterplot"+ str(timeval)+".png")
```

In [454]:
```
# colors = [k for k,v in pltcolor.cnames.items()]
```

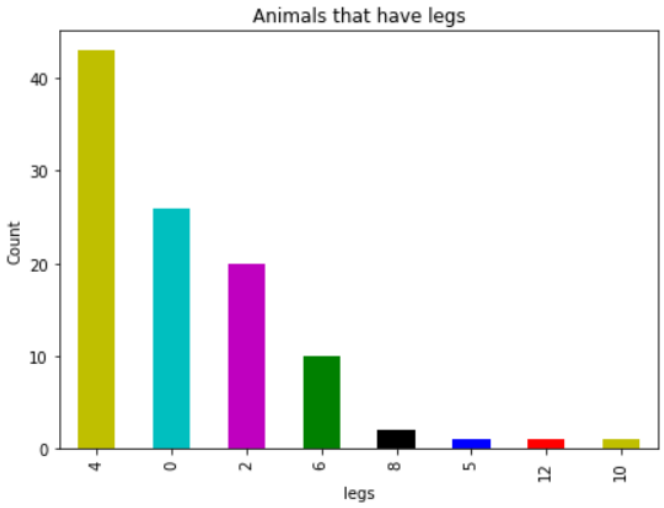
In [455]:
```
def plot(attribute, dataValue):
    titleName = ''
    if attribute == 'animal name':
        titleName  = 'Animal Names'
    elif attribute in ['hair', 'feathers','toothed','legs', 'tail','backbone
        titleName = 'Animals that have ' + attribute
    elif attribute in  [ 'airborne', 'aquatic', 'venomous', 'domestic', 'pre
        titleName = 'Animals that are ' + attribute
    elif attribute in ['breathes']:
        titleName = 'Animals that ' + attribute
    elif attribute in ['gestation']:
        titleName =  'Animal with gestation period'


    plt = dataValue[attribute].value_counts().plot(kind='bar', figsize=(7,5)
    plt.set_xlabel(attribute)
    plt.set_ylabel("Count")
    timeval = time.strftime("%M_%S")
    plt.figure.savefig("../generated-output/counterplot"+ str(timeval)+".png'
```

Hair

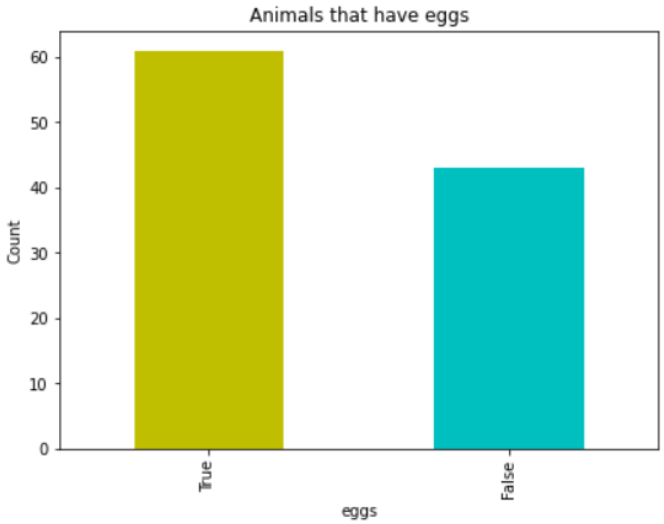
In [515]:
```
plot('hair', xtrain)
```



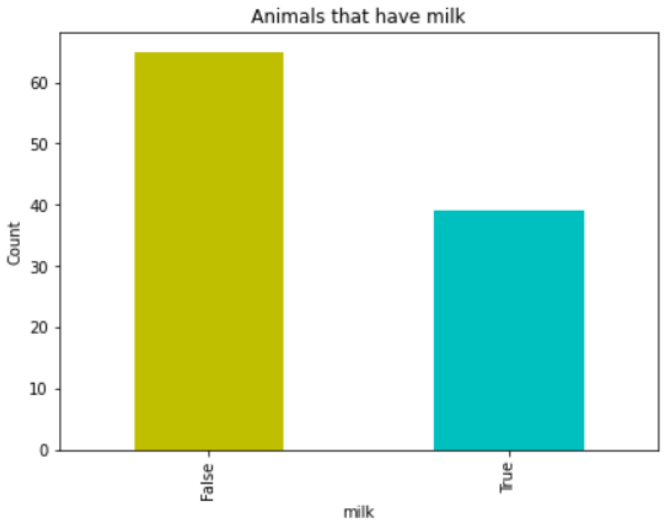
In [516]:
```
plot('legs', xtrain)
```



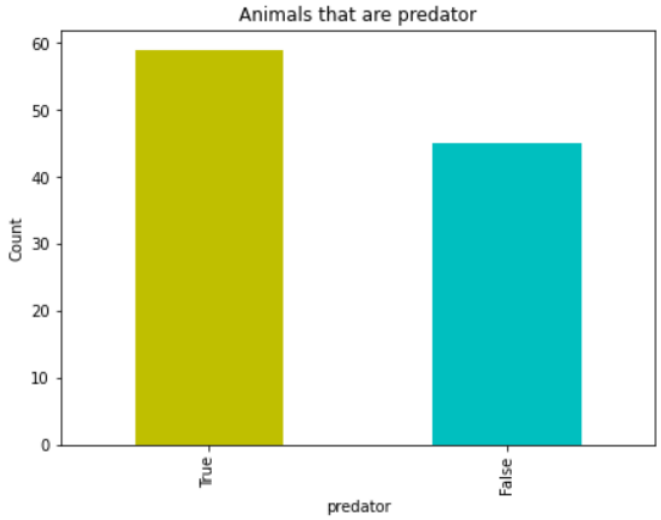
In [517]:
```
plot('eggs', xtrain)
```



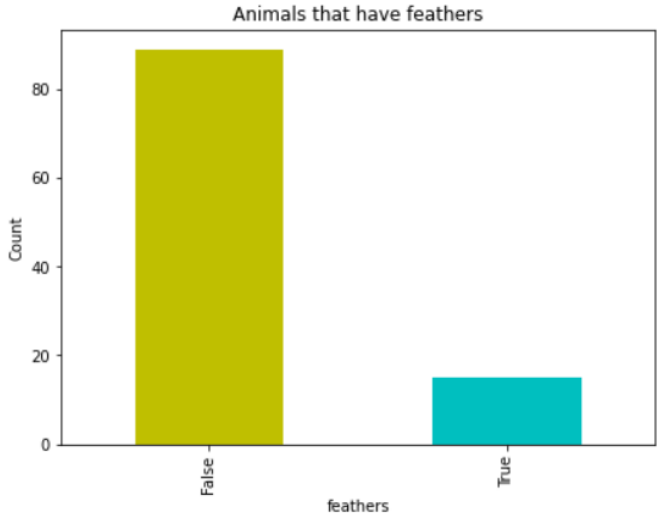
In [518]:
```
plot('milk', xtrain)
```



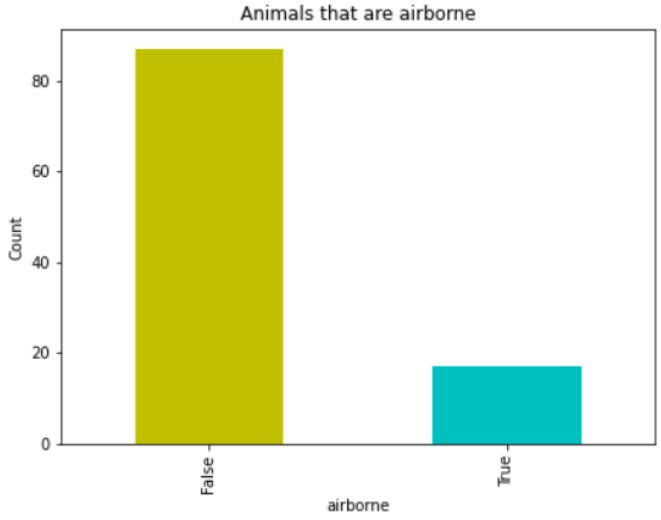
In [519]:
```
plot('predator', xtrain)
```



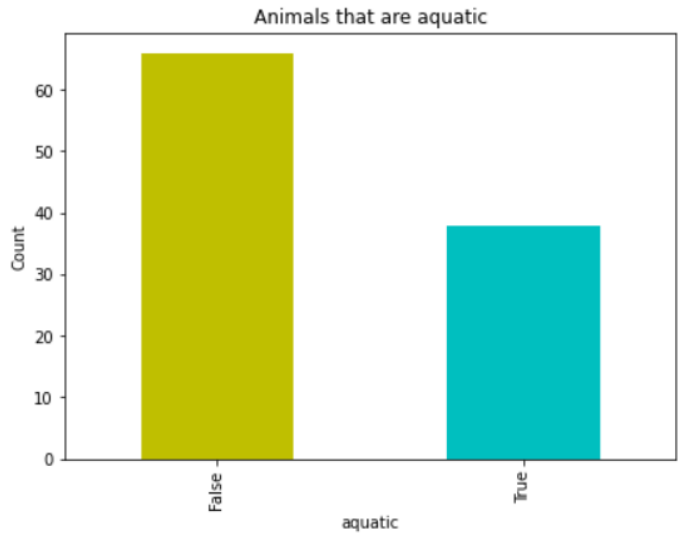
In [520]:
```
plot('feathers', xtrain)
```



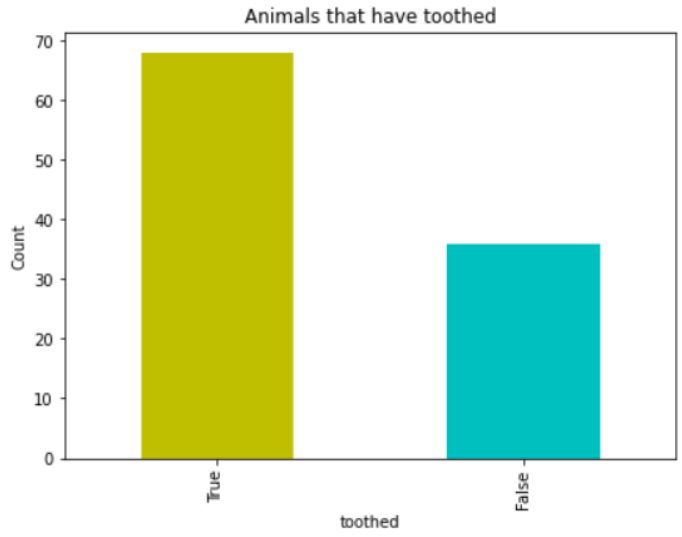
In [521]:
```
plot('airborne', xtrain)
```



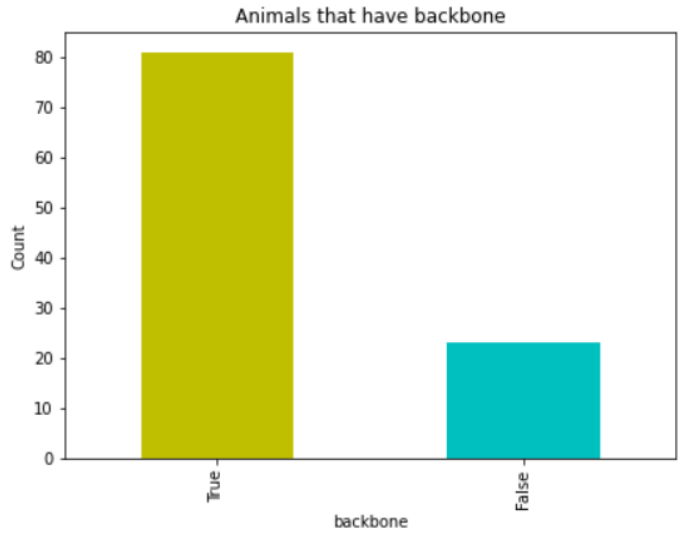
In [522]:
```
plot('aquatic', xtrain)
```



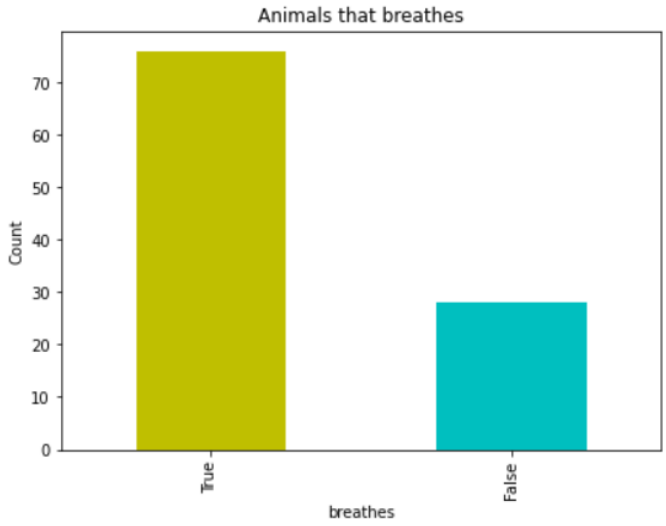
In [523]:
```
plot('toothed', xtrain)
```



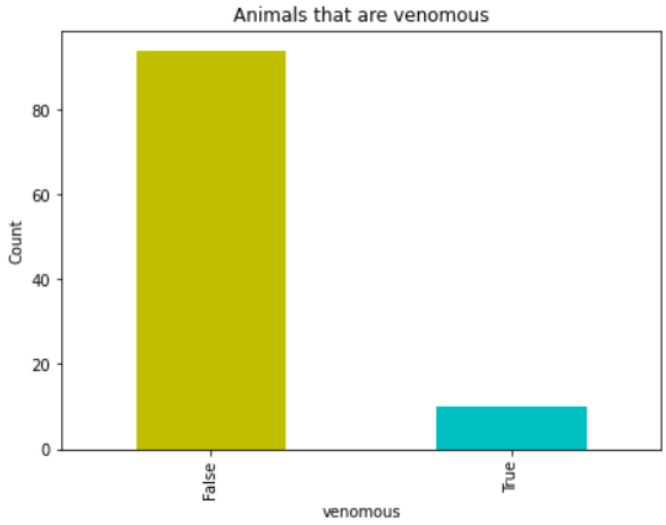
In [524]:
```
plot('backbone', xtrain)
```



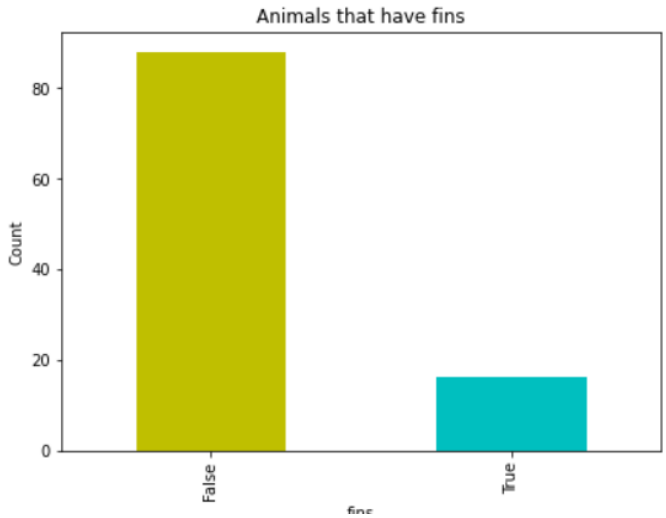
In [525]:
```
plot('breathes', xtrain)
```



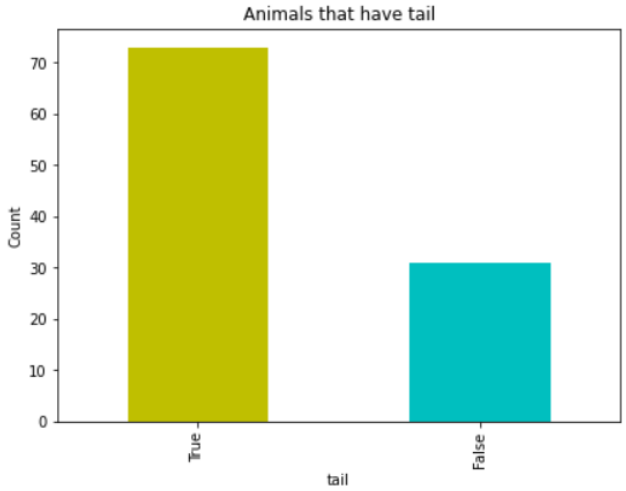
In [526]:
```
plot('venomous', xtrain)
```



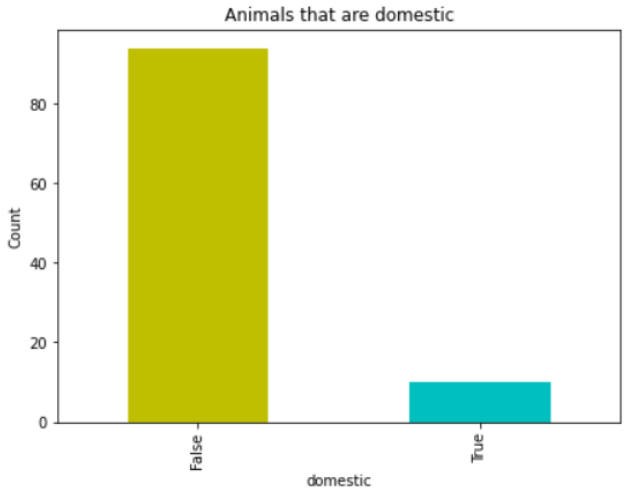
In [527]:
```
plot('fins', xtrain)
```



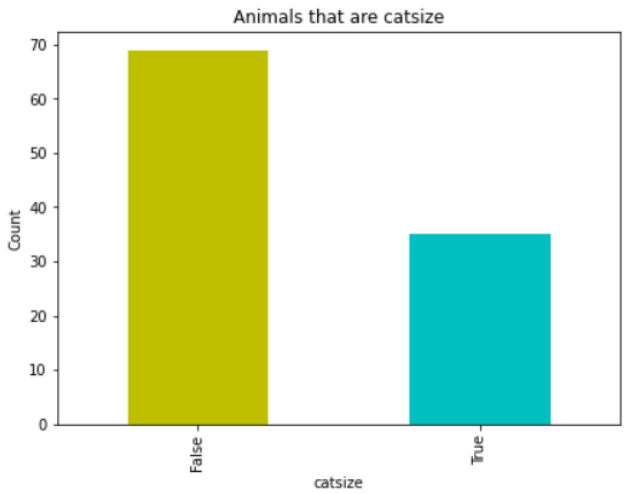
In [469]:
```
plot('tail', xtrain)
```



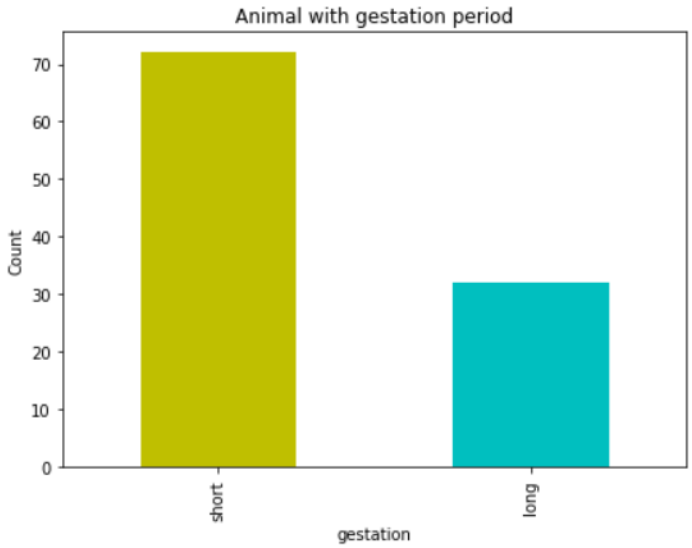
In [528]:
```
plot('domestic', xtrain)
```



In [529]:
```
plot('catsize', xtrain)
```



In [530]:
```
plot('gestation', xtrain)
```



In [473]:
```
#  xtrain['gestation'].mean()
```

In [474]:
```
# xtest['gestation'].mean()
```

In [475]:
```
# # xtrain['gestation'] = [1 if float(x) > 170  else 0 for x in xtrain['gest

# for i, d in xtrain.iterrows():
#     if pd.to_numeric(d['gestation'])  >=  xtrain['gestation'].mean():
#         xtrain.at[i,'gestation'] =  1 # for longerm gestation grear than 1
#     else:
#         xtrain.at[i,'gestation'] = 0
# #

# for i, d in xtest.iterrows():
#     if pd.to_numeric(d['gestation'])  >=  xtest['gestation'].mean():  #107
#         xtest.at[i,'gestation'] =  1 # for longerm gestation grear than 16
#     else:
#         xtest.at[i,'gestation'] = 0
```

In [476]:
```
# xtrain['gestation']
```

In [477]:
```
# xtest['gestation']
```

In [478]:
```
# sns.countplot(x="gestation", hue="gestation", data=xtrain)
# plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)
```

This graph indicates most of the the animal in the training data set has a gestation period of above the mean 123 days .

Mining or Analytics:

Cross Tab, Rule Generation, and Error Rate

In [479]:
```
def crossTab (data, attribute):
    return pd.crosstab(index = data[attribute], columns = ytrain["mammal"], margins=True)
```

In [480]:
```
def ruleAndErrorRate (trainData, name):
    RULE = pd.DataFrame()
    data  = []
    rule = []
    error= []
    totalError = 0
    for i, d in trainData.iterrows():
        if i!= 'All':
            if d['mammal'] > d['non-mammal']:
                data.append(i)
                rule.append('mammal')
                error.append(d['non-mammal'])
                totalError += d['non-mammal']
            else:
                data.append(i)
                rule.append('non-mammal')
                error.append(d['mammal'])
                totalError += d['mammal']

    RULE [name] = data
    RULE ['rule'] =  rule
    RULE ['error'] =  error

    # Error rate
    ErrorRate = totalError / trainData.loc['All','All']

    RULE.reset_index(inplace=False)
    RULE.set_index(name, inplace=True)

    return RULE, ErrorRate    #RULE.style.hide_index()
```

In [481]:
```
def dislayer (Data, attribute):
    value = crossTab (Data, attribute)
    rule, errorRate = ruleAndErrorRate (value, attribute)
    display(value.style.set_caption("Cross Table"))
    display(rule.style.set_caption("Rule set for " +attribute))
    print("\n\n"+ '\033[1m' + '\033[91m'+ "Error Rate: {:0.3f}".format(errorRate), '\033[91m', '\033[1m')
```

Legs

In [482]:
```
dislayer(xtrain, 'legs')
```

Cross Table

mammal	mammal	non-mammal	All
legs			
0	2	24	26
10	0	1	1
12	0	1	1
2	5	15	20
4	32	11	43
5	0	1	1
6	0	10	10
8	0	2	2
All	39	65	104

Rule set for legs

	rule	error
legs		
0	non-mammal	2
10	non-mammal	0
12	non-mammal	0
2	non-mammal	5
4	mammal	11
5	non-mammal	0
6	non-mammal	0
8	non-mammal	0

```
Error Rate: 0.173
```

The rule set indicates that all animal that have a leg of 4 are mammal.

Milk

In [483]:
```
dislayer(xtrain, 'milk')
```

Cross Table

mammal	mammal	non-mammal	All
milk			
False	0	65	65
True	39	0	39
All	39	65	104

Rule set for milk

	rule	error
milk		
False	non-mammal	0
True	mammal	0

```
Error Rate: 0.000
```

- Animals that have milk are mammals or else non-mammals.
- The rule set has **zero error rate**.

Hair

In [484]:
```
dislayer(xtrain, 'hair')
```

Cross Table

mammal	mammal	non-mammal	All
hair			
False	1	62	63
True	38	3	41
All	39	65	104

Rule set for hair

	rule	error
hair		
False	non-mammal	1
True	mammal	3

```
Error Rate: 0.038
```

- The rules indicates that animals with hair are mammals or else non-mammals.
- The rule set has a good error rate.

Tail

In [485]:
```
dislayer(xtrain, 'tail')
```

Cross Table

mammal	mammal	non-mammal	All
tail			
False	8	23	31
True	31	42	73
All	39	65	104

Rule set for tail

	rule	error
tail		
False	non-mammal	8
True	non-mammal	31

```
Error Rate: 0.375
```

- Most animals with tails or not are non-mammals.
- In addition to its high error rate, it won't be a good candidate for prediction since it does not capture mammal animals.

Eggs

In [486]:
```
dislayer(xtrain, 'eggs')
```

Cross Table

mammal	mammal	non-mammal	All
eggs			
False	38	5	43
True	1	60	61
All	39	65	104

Rule set for eggs

	rule	error
eggs		
False	mammal	5
True	non-mammal	1

Error Rate: 0.058

- Animals which have eggs are non-mammals, and else mammals.
- It have acceptable error rate, as well.

Backbone

In [487]:
```
dislayer(xtrain, 'backbone')
```

Cross Table

mammal	mammal	non-mammal	All
backbone			
False	0	23	23
True	39	42	81
All	39	65	104

Rule set for backbone

	rule	error
backbone		
False	non-mammal	0
True	non-mammal	39

Error Rate: 0.375

- It is not a good rule set because it only predict non-mammals and has high error rate.

Feathers

In [488]:
```
dislayer(xtrain, 'feathers')
```

Cross Table

mammal	mammal	non-mammal	All
feathers			
False	39	50	89
True	0	15	15
All	39	65	104

Rule set for feathers

	rule	error
feathers		
False	non-mammal	39
True	non-mammal	0

Error Rate: 0.375

- Similar to backbone, this attribute have high error rate and it does not capture mammal examples.

Predator

In [489]:
```
dislayer(xtrain, 'predator')
```

Cross Table

mammal	mammal	non-mammal	All
predator			
False	17	28	45
True	22	37	59
All	39	65	104

Rule set for predator

	rule	error
predator		
False	non-mammal	17
True	non-mammal	22

Error Rate: 0.375

- This rule set indicates that whether an animal is true or false, most of the animals will be predicted as non-mammals.

Aquatic

In [490]:
```
dislayer(xtrain, 'aquatic')
```

Cross Table

mammal	mammal	non-mammal	All
aquatic			
False	34	32	66
True	5	33	38
All	39	65	104

Rule set for aquatic

	rule	error
aquatic		
False	mammal	32
True	non-mammal	5

Error Rate: 0.356

- The rule set captures both categories though it has high error rate.

catsize

In [491]:
```
dislayer(xtrain, 'catsize')
```

Cross Table

mammal	mammal	non-mammal	All
catsize			
False	13	56	69
True	26	9	35
All	39	65	104

Rule set for catsize

	rule	error
catsize		
False	non-mammal	13
True	mammal	9

Error Rate: 0.212

- The rule set captures both categories with fair error rate.

fins

In [492]:
```
dislayer(xtrain, 'fins')
```

Cross Table

mammal	mammal	non-mammal	All
fins			
False	36	52	88
True	3	13	16
All	39	65	104

Rule set for fins

	rule	error
fins		
False	non-mammal	36
True	non-mammal	3

Error Rate: 0.375

- This rule can only used for non-mammals.
- The rule set has a fairly high error rate.

Venomous

In [493]:
```
dislayer(xtrain, 'venomous')
```

Cross Table

mammal	mammal	non-mammal	All
venomous			
False	39	55	94
True	0	10	10
All	39	65	104

Rule set for venomous

	rule	error
venomous		
False	non-mammal	39
True	non-mammal	0

Error Rate: 0.375

- This rule set doe not capture mammals,too.
- It also has a high error rate.

Breathes

In [494]:
```
dislayer(xtrain, 'breathes')
```

Cross Table

mammal	mammal	non-mammal	All
breathes			
False	0	28	28
True	39	37	76
All	39	65	104

Rule set for breathes

	rule	error
breathes		
False	non-mammal	0
True	mammal	37

```
Error Rate: 0.356
```

- The rule set shows if an animal breathes, it is mammals and else non-mammal with an error rate of 0.356.

Toothed

In [495]:
```
dislayer(xtrain, 'toothed')
```

Cross Table

mammal	mammal	non-mammal	All
toothed			
False	1	35	36
True	38	30	68
All	39	65	104

Rule set for toothed

	rule	error
toothed		
False	non-mammal	1
True	mammal	30

```
Error Rate: 0.298
```

- This rule set captures both classes (mammal and non-mammal).
- It has a fairly acceptable error rate.

Domestic

In [496]:
```
dislayer(xtrain, 'domestic')
```

Cross Table

mammal	mammal	non-mammal	All
domestic			
False	34	60	94
True	5	5	10
All	39	65	104

Rule set for domestic

	rule	error
domestic		
False	non-mammal	34
True	non-mammal	5

```
Error Rate: 0.375
```

- This rule set does not capture mammals.
- It does not acceptable error rate.
- Therefore, this rule set should be avoided.

Gestation

In [533]:
```
dislayer(xtrain, 'gestation')
```

Cross Table

mammal	mammal	non-mammal	All
gestation			
long	19	13	32
short	20	52	72
All	39	65	104

Rule set for gestation

	rule	error
gestation		
long	mammal	13
short	non-mammal	20

```
Error Rate: 0.317
```

- This attribute can not be a good candidate for selecting rule set since it was originally a continuous attribute.

Animal Name

In [498]:
```
dislayer(xtrain, 'animal name')
```

Cross Table

animal name	mammal	non-mammal	All
aardvark	1	0	1
anole	0	1	1
antelope	1	0	1
axolotl	0	1	1
bear	1	0	1
blue dragon	0	1	1
boar	1	0	1
buffalo	1	0	1
caecilians	0	1	1
capybara	1	0	1
carp	0	1	1
catfish	0	1	1
cavy	1	0	1
cayman	0	1	1
cheetah	1	0	1
chicken	0	1	1
chub	0	1	1
chupacabra	1	0	1
clam	0	1	1
cow	1	0	1
crab	0	1	1
crayfish	0	1	1
crow	0	1	1
cuttlefish	0	1	1
dartfrog	0	1	1
deer	1	0	1

Rule set for animal name

```
Error Rate: 0.000
```

animal name	rule	error
aardvark	mammal	0
anole	non-mammal	0
antelope	mammal	0
axolotl	non-mammal	0
bear	mammal	0
blue dragon	non-mammal	0
boar	mammal	0
buffalo	mammal	0
caecilians	non-mammal	0
capybara	mammal	0
carp	non-mammal	0
catfish	non-mammal	0
cavy	mammal	0
cayman	non-mammal	0
cheetah	mammal	0
chicken	non-mammal	0
chub	non-mammal	0
chupacabra	mammal	0
clam	non-mammal	0
cow	mammal	0
crab	non-	0
squid	non-mammal	0
starfish	non-mammal	0
stingray	non-mammal	0
swan	non-mammal	0
termite	non-mammal	0
thylacine	mammal	0
tortoise	non-mammal	0
tuatara	non-mammal	0
tuna	non-mammal	0
vole	mammal	0
wasp	non-mammal	0
whale shark	non-mammal	0
wolf	mammal	0
worm	non-mammal	0
wren	non-mammal	0

- This rule set can not be used since it closes the whole purpose of using classification technique.

Note

- Most of the rules set only predicted non-mammals, and it indicates that the number of non-mammal is greater than mammal animals.

Evaluation

Milk rule set is selected because of its minimum error rate (0 error rate).
Cross Table and milk rule set

In [499]:
```
dislayer(xtrain, 'milk')
```

Cross Table

mammal	mammal	non-mammal	All
milk			
False	0	65	65
True	39	0	39
All	39	65	104

Rule set for milk

	rule	error
milk		
False	non-mammal	0
True	mammal	0

Error Rate: 0.000

In [500]:
```
prediction = pd.DataFrame(xtest['milk'])
prediction = prediction.rename(columns={'milk': 'prediction'})

prediction[prediction['prediction'] ==  True] = 'mammal'
prediction[prediction['prediction'] ==  False] = 'non-mammal'
```

Confusion Matrix

In [501]:
```
confusionMatrix = confusion_matrix(ytest['mammal'], prediction['prediction'])
print(confusionMatrix)
```

```
[[ 0  0  0  0]
 [ 0  0  0  0]
 [ 0 12  0  0]
 [14  0  0  0]]
```

In [502]:
```
print("---------------------------")
print("|   ytest    | prediction |")
print("---------------------------")
for test, predict in zip (list(ytest['mammal']), list(prediction['prediction'])):
    print('| {:<10} | {: <10} |'.format(test, predict))
print("---------------------------")
```

```
---------------------------
|   ytest    | prediction |
---------------------------
| non-mammal | False      |
| non-mammal | False      |
| mammal     | True       |
| mammal     | True       |
| mammal     | True       |
| mammal     | True       |
| mammal     | True       |
| non-mammal | False      |
| mammal     | True       |
| mammal     | True       |
| non-mammal | False      |
| mammal     | True       |
| mammal     | True       |
| mammal     | True       |
| mammal     | True       |
| non-mammal | False      |
| non-mammal | False      |
```

Check the accuracy of the data

In [503]:
```
TruePostive = 0
TrueNegative = 0
FalsePositive = 0
FalseNegative = 0
for i, d in ytest.iterrows():
    if (prediction.at[i,'prediction'] == 'True'):
        case = True
    else:
        case = False
    if str(ytest.at[i,'mammal']) ==  'mammal'  and case ==  True:
        TruePostive += 1
    elif str(ytest.at[i,'mammal']) ==  'non-mammal'  and case ==  False:
        TrueNegative += 1
    elif str(ytest.at[i,'mammal']) ==  'mammal'  and case  ==  False:
        FalsePositive += 1
    elif str(ytest.at[i,'mammal']) ==  'non-mammal'  and case ==  True:
        FalseNegative +=1
```

In [504]:
```
value =  {'True Postive': TruePostive ,
          'True Negative': TrueNegative,
          'False Positive': FalsePositive,
           'False Negative': FalseNegative }

for key, value in value.items():
    print("\n\n"+ '\033[1m' + '\033[91m'+ str(key) + ': '+ str(value), '\033[91m', '\033[1m')
```

True Postive: 12

True Negative: 14

False Positive: 0

False Negative: 0

Accuracy

Accuracy = {number of correct prediction/ number of all prediction}

In [505]:
```
Accuracy  = (TruePostive + TrueNegative)/(TruePostive + TrueNegative + FalsePositive + FalseNegative)
Accuracy
```

Out[505]: 1.0

Error Rate

In [506]:
```
ErrorRate  = (FalsePositive + FalseNegative)/(TruePostive + TrueNegative + FalsePositive + FalseNegative)
ErrorRate
```

Out[506]: 0.0

Precsion

In [507]:
```
precsion =  (TruePostive)/(TruePostive + FalsePositive)
precsion
```

Out[507]: 1.0

Specificity

In [508]:
```
specificity = (TrueNegative)/( TrueNegative + FalsePositive)
specificity
```

Out[508]: 1.0

Negative Prediction

In [509]:
```
NegativePrediction =  TrueNegative /(TrueNegative + FalseNegative)
NegativePrediction
```

Out[509]: 1.0

Sensetivity or Recall

In [510]:
```
sensitivity = TruePostive /(TruePostive + FalseNegative)
sensitivity
```

Out[510]: 1.0

F-Measure

In [511]:
```
fmeasure = (2*sensitivity * precsion)/(sensitivity + precsion)
fmeasure
```

Out[511]: 1.0

In [512]:
```
# accuracy_score(ytest['mammal'], prediction['prediction'])
```

In [513]:
```
# print(classification_report(ytest['mammal'], prediction['prediction']))
```

Results:

As we can see above, the error rate of our model 0 which indicates milk rule set is a good rule for determinging wheather an animal is mammal or not.

In [514]:
```
dislayer(xtrain, 'milk')
```

Cross Table

mammal	mammal	non-mammal	All
milk			
False	0	65	65
True	39	0	39
All	39	65	104

Rule set for milk

	rule	error
milk		
False	non-mammal	0
True	mammal	0

```
Error Rate: 0.000
```

- If an animal has milk, then it is mammal
- If an animal does not have milk, then is is non-mammal.

References:

- Craig Nevill-Manning and Geoffrey Holmes and Ian H. Witten (?) The Development of Holte's 1R Classifier Retrived (2021, July 4) from https://www.cs.waikato.ac.nz/~ihw/papers/95NM-GH-IHW-Develop.pdf.
- Perry B. Koob (2020) koobp-classifier-1r. retrived (2021, July 1) from Missiour S&T canvas.
- Stackoverflow (2021). General python questions. Retrived (2021, July 1) from https://stackoverflow.com/