In [48]:

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import seaborn as sns
%matplotlib inline
```

Machine Learning Models

1. Linear Regression

In [49]:

```
from sklearn import linear_model
```

In [50]:

```
data = pd.read_csv("FuelConsumption.csv")
data.head()
```

Out[50]:

	MODELYEAR	MAKE	MODEL	VEHICLECLASS	ENGINESIZE	CYLINDERS	TRANSMISSION	FUELTYPE	FUELCONSUMPTION_CITY
0	2014	ACURA	ILX	COMPACT	2.0	4	AS5	Z	9.9
1	2014	ACURA	ILX	COMPACT	2.4	4	M6	Z	11.2
2	2014	ACURA	ILX HYBRID	COMPACT	1.5	4	AV7	Z	6.0
3	2014	ACURA	MDX 4WD	SUV - SMALL	3.5	6	AS6	Z	12.7
4	2014	ACURA	RDX AWD	SUV - SMALL	3.5	6	AS6	Z	12.1

In [51]:

```
data.describe()
```

Out[51]:

	MODELYEAR	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_CITY	FUELCONSUMPTION_HWY	FUELCONSUMPTION_COMB	FUE
count	1067.0	1067.000000	1067.000000	1067.000000	1067.000000	1067.000000	
mean	2014.0	3.346298	5.794752	13.296532	9.474602	11.580881	
std	0.0	1.415895	1.797447	4.101253	2.794510	3.485595	
min	2014.0	1.000000	3.000000	4.600000	4.900000	4.700000	
25%	2014.0	2.000000	4.000000	10.250000	7.500000	9.000000	
50%	2014.0	3.400000	6.000000	12.600000	8.800000	10.900000	
75%	2014.0	4.300000	8.000000	15.550000	10.850000	13.350000	
max	2014.0	8.400000	12.000000	30.200000	20.500000	25.800000	

In [52]:

```
#Filter the data set to just "CYLINDERS", "ENGINESIZE, "FUELCONSUMPTION_COMB" and "CO2EMISSIONS".

cdf = data[['ENGINESIZE', 'CYLINDERS', 'FUELCONSUMPTION_COMB', 'CO2EMISSIONS']]

cdf
```

Out[52]:

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
...	...	...	...	...
1062	3.0	6	11.8	271
1063	3.2	6	11.5	264
1064	3.0	6	11.8	271
1065	3.2	6	11.3	260
1066	3.2	6	12.8	294

1067 rows × 4 columns

Plot each of these features saparately:

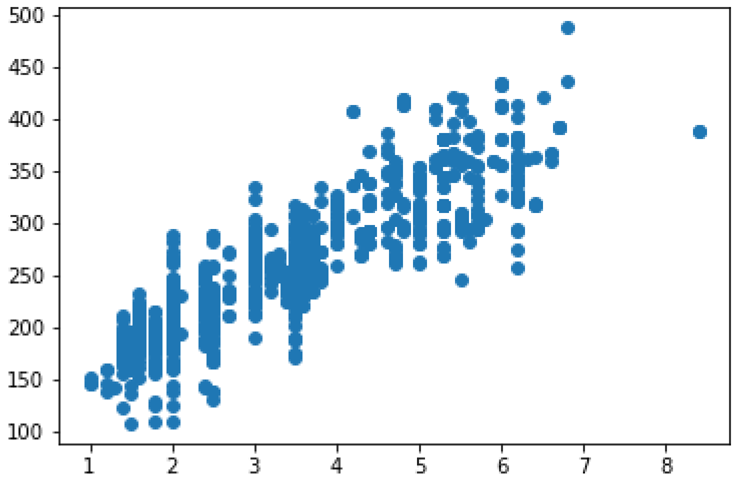
Now, lets plot each of these features vs the Emission, to see how linear is their relation:

In [53]:

```
#ENGINESIZE vs EMISSION
plt.scatter(cdf.ENGINESIZE,cdf.CO2EMISSIONS)
plt.show
```

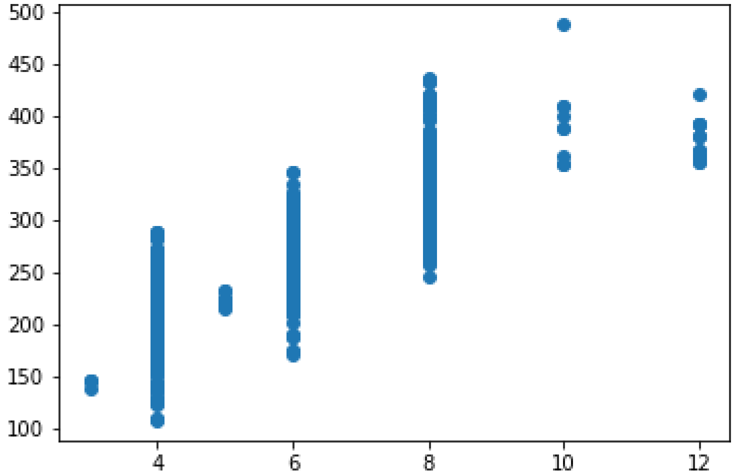
Out[53]:

```
<function matplotlib.pyplot.show(close=None, block=None)>
```



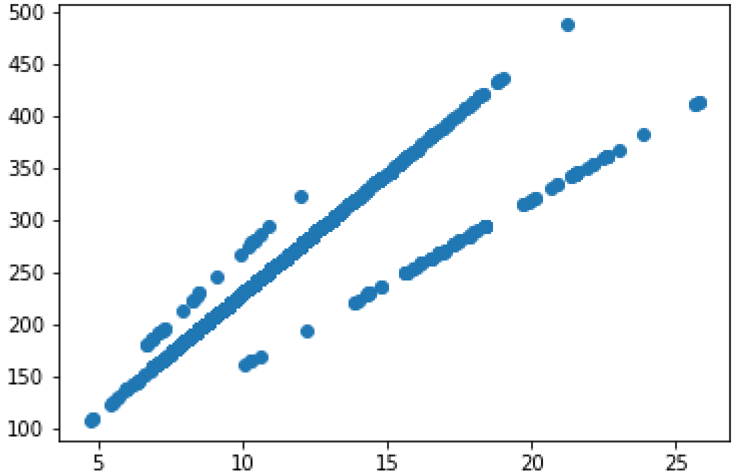
In [54]:

```
#CYLINDERS against CO2EMISSIONS
plt.scatter(cdf.CYLINDERS,cdf.CO2EMISSIONS)
plt.show()
```



In [55]:

```
#FuelCONSUMPTION_COMB against CO2EMISSIONS
plt.scatter(cdf.FUELCONSUMPTION_COMB,cdf.CO2EMISSIONS)
plt.show()
```



Now, lets plot each of these features vs the Emission, to see how linear is their relation:

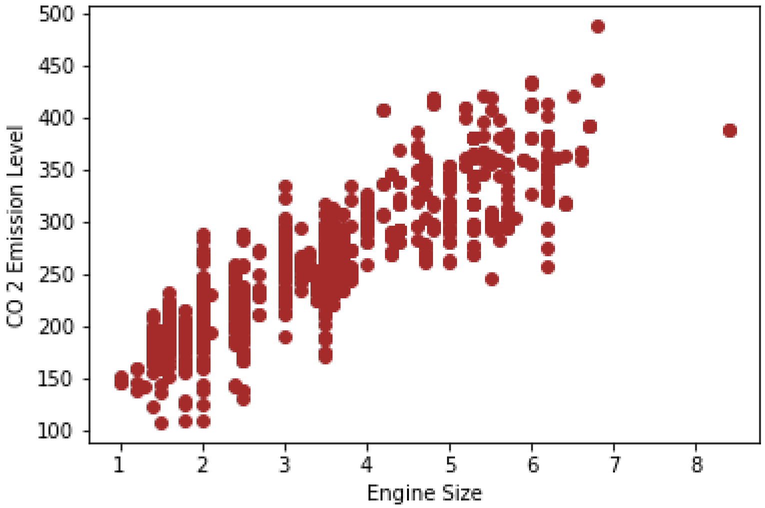
Simple Regression Model

Linear Regression fits a linear model with coefficients $\theta = (\theta_1, ..., \theta_n)$ to minimize the 'residual sum of squares' between the independent x in the dataset, and the dependent y by the linear approximation.

In [56]:

```
from sklearn.linear_model import LinearRegression

plt.scatter(cdf.ENGINESIZE, cdf.CO2EMISSIONS, color='brown')
plt.xlabel('Engine Size')
plt.ylabel('CO 2 Emission Level')
plt.show()
```



In [57]:

```
cdf.head(20)
```

Out[57]:

	ENGINESIZE	CYLINDERS	FUELCONSUMPTION_COMB	CO2EMISSIONS
0	2.0	4	8.5	196
1	2.4	4	9.6	221
2	1.5	4	5.9	136
3	3.5	6	11.1	255
4	3.5	6	10.6	244
5	3.5	6	10.0	230
6	3.5	6	10.1	232
7	3.7	6	11.1	255
8	3.7	6	11.6	267
9	2.4	4	9.2	212
10	2.4	4	9.8	225
11	3.5	6	10.4	239
12	5.9	12	15.6	359
13	5.9	12	15.6	359
14	4.7	8	14.7	338
15	4.7	8	15.4	354
16	4.7	8	14.7	338
17	4.7	8	15.4	354
18	5.9	12	15.6	359
19	2.0	4	8.8	202

In [58]:

```
lm = LinearRegression()
lm.fit(cdf[['ENGINESIZE']],cdf.CO2EMISSIONS)
```

Out[58]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

We can now plot the fit line over the data:

In [59]:

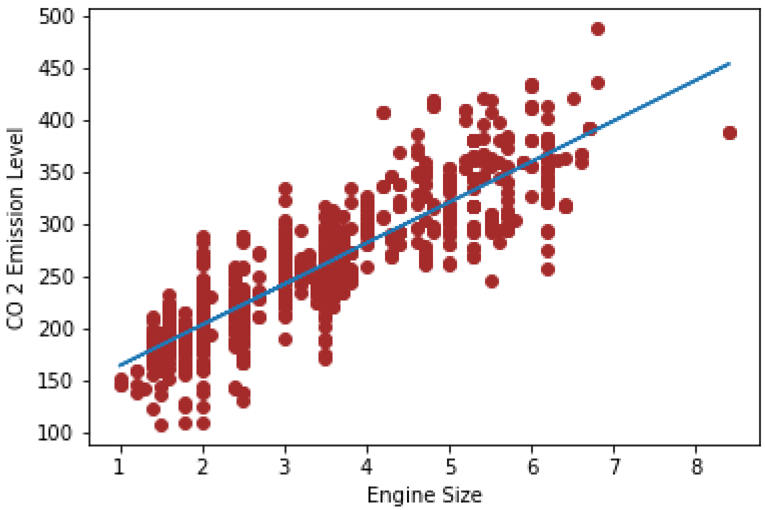
```
lm.predict([[5]])
```

Out[59]:

```
array([320.93009843])
```

In [60]:

```
#Now lets make the scatterplot with the fit line
plt.scatter(cdf.ENGINESIZE, cdf.CO2EMISSIONS, color='brown')
plt.xlabel('Engine Size')
plt.ylabel('CO 2 Emission Level')
plt.plot(cdf.ENGINESIZE, lm.predict(cdf[['ENGINESIZE']]))
plt.show()
```



Evaluation

we compare the actual values and predicted values to calculate the accuracy of a regression model. Evaluation metrics provide a key role in the development of a model, as it provides insight to areas that require improvement.

There are different model evaluation metrics, lets use MSE here to calculate the accuracy of our model based on the test set:

- Mean absolute error: It is the mean of the absolute value of the errors. This is the easiest of the metrics to understand since it's just average error.
- Mean Squared Error (MSE): Mean Squared Error (MSE) is the mean of the squared error. It's more popular than Mean absolute error because the focus is geared more towards large errors. This is due to the squared term exponentially increasing larger errors in comparison to smaller ones.
- Root Mean Squared Error (RMSE): This is the square root of the Mean Square Error.
- R-squared is not error, but is a popular metric for accuracy of your model. It represents how close the data are to the fitted regression line. The higher the R-squared, the better the model fits your data. Best possible score is 1.0 and it can be negative (because the model can be arbitrarily worse).

In [61]:

```
#Now let test the accuracy of the model:

x = np.array(cdf['ENGINESIZE']).reshape((-1, 1))
y = cdf['CO2EMISSIONS']

lm = LinearRegression()
lm.fit(x,y)
accuracy = lm.score(x,y)
accuracy
```

Out[61]:

```
0.7641458597854816
```

Now let's try using multiple x values as 'Multiple Linear Regression' to make a prediction:

In [62]:

```
lm.fit(cdf[['ENGINESIZE', 'FUELCONSUMPTION_COMB', 'CYLINDERS']],cdf.CO2EMISSIONS)
```

Out[62]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
         normalize=False)
```

In [63]:

```
lm.predict([[1,2,9]])
```

Out[63]:

```
array([162.91581285])
```

2. Using DescisionTree Method

In [71]:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from IPython.display import Image
from sklearn import tree
import pydotplus
```

In [72]:

```
#Let's first produce the dataset.
data = pd.read_csv('Monster Matcher.csv')
data
```

Out[72]:

	Monster	Has Claws	Has Scales	Long Tail	Short Tail	Has Wings	Feathers	Has Fur	Has Beak	Sharp Teeth	...	Has Fins/Gills	Has Horns	Has Hooves	Two Legs	Four Legs	Has Whiskers
0	Dragon	1	1	1	0	1	0	0	0	1	...	0	0	0	1	1	0
1	Griffon	1	0	1	1	1	1	1	1	0	...	0	0	0	0	1	0
2	Minotaur	0	0	1	0	0	0	1	0	0	...	0	1	1	1	0	0
3	Kraken	1	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0
4	Crocodile	1	1	1	0	0	0	0	0	1	...	0	0	0	0	1	0
5	Dragon	1	1	1	0	1	0	0	0	1	...	0	0	0	1	1	0
6	Griffon	1	0	1	1	1	1	1	1	0	...	0	0	0	0	1	0
7	Minotaur	0	0	1	0	0	0	1	0	0	...	0	1	1	1	0	0
8	Kraken	1	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0
9	Crocodile	1	1	1	0	0	0	0	0	1	...	0	0	0	0	1	0
10	Chimera	1	1	1	0	0	1	1	0	1	...	0	1	1	0	1	1
11	Giant Fish	0	1	0	0	0	0	0	0	1	...	1	0	0	0	0	0
12	Big Cat	1	0	1	0	0	0	1	0	1	...	0	0	0	0	1	1
13	Dragon	1	1	1	0	1	0	0	0	1	...	0	0	0	1	1	0
14	Griffon	1	0	1	1	1	1	1	1	0	...	0	0	0	0	1	0
15	Minotaur	0	0	1	0	0	0	1	0	0	...	0	1	1	1	0	0
16	Kraken	1	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0
17	Crocodile	1	1	1	0	0	0	0	0	1	...	0	0	0	0	1	0
18	Dragon	1	1	1	0	1	0	0	0	1	...	0	0	0	1	1	0
19	Griffon	1	0	1	1	1	1	1	1	0	...	0	0	0	0	1	0
20	Minotaur	0	0	1	0	0	0	1	0	0	...	0	1	1	1	0	0
21	Kraken	1	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0
22	Crocodile	1	1	1	0	0	0	0	0	1	...	0	0	0	0	1	0
23	Chimera	1	1	1	0	0	1	1	0	1	...	0	1	1	0	1	1

	Monster	Has Claws	Has Scales	Long Tail	Short Tail	Has Wings	Feathers	Has Fur	Has Beak	Sharp Teeth	...	Has Fins/Gills	Has Horns	Has Hooves	Two Legs	Four Legs	Has Whiskers
24	Giant Fish	0	1	0	0	0	0	0	0	1	...	1	0	0	0	0	0
25	Big Cat	1	0	1	0	0	0	1	0	1	...	0	0	0	0	1	1
26	Dragon	1	1	1	0	1	0	0	0	1	...	0	0	0	1	1	0
27	Griffon	1	0	1	1	1	1	1	1	0	...	0	0	0	0	1	0
28	Minotaur	0	0	1	0	0	0	1	0	0	...	0	1	1	1	0	0
29	Giant Squid	1	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0
30	Crocodile	1	1	1	0	0	0	0	0	1	...	0	0	0	0	1	0
31	Dragon	1	1	1	0	1	0	0	0	1	...	0	0	0	1	1	0
32	Griffon	1	0	1	1	1	1	1	1	0	...	0	0	0	0	1	0
33	Minotaur	0	0	1	0	0	0	1	0	0	...	0	1	1	1	0	0
34	Kraken	1	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0
35	Crocodile	1	1	1	0	0	0	0	0	1	...	0	0	0	0	1	0
36	Chimera	1	1	1	0	0	1	1	0	1	...	0	1	1	0	1	1
37	Giant Fish	0	1	0	0	0	0	0	0	1	...	1	0	0	0	0	0
38	Big Cat	1	0	1	0	0	0	1	0	1	...	0	0	0	0	1	1
39	Dragon	1	1	1	0	1	0	0	0	1	...	0	0	0	1	1	0
40	Griffon	1	0	1	1	1	1	1	1	0	...	0	0	0	0	1	0
41	Minotaur	0	0	1	0	0	0	1	0	0	...	0	1	1	1	0	0
42	Kraken	1	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0
43	Crocodile	1	1	1	0	0	0	0	0	1	...	0	0	0	0	1	0
44	Dragon	1	1	1	0	1	0	0	0	1	...	0	0	0	1	1	0
45	Griffon	1	0	1	1	1	1	1	1	0	...	0	0	0	0	1	0
46	Minotaur	0	0	1	0	0	0	1	0	0	...	0	1	1	1	0	0
47	Kraken	1	0	0	0	0	0	0	1	0	...	1	0	0	0	0	0

	Monster	Has Claws	Has Scales	Long Tail	Short Tail	Has Wings	Feathers	Has Fur	Has Beak	Sharp Teeth	...	Has Fins/Gills	Has Horns	Has Hooves	Two Legs	Four Legs	Has Whiskers
48	Crocodile	1	1	1	0	0	0	0	0	1	...	0	0	0	0	1	0
49	Chimera	1	1	1	0	0	1	1	0	1	...	0	1	1	0	1	1
50	Giant Fish	0	1	0	0	0	0	0	0	1	...	1	0	0	0	0	0
51	Big Cat	1	0	1	0	0	0	1	0	1	...	0	0	0	0	1	1

52 rows × 23 columns

The above dataset consists of different mythical monsters and animals as well as the characteristics(traits) that are associated with them. These traits have numerical values, "1" for yes and "0" for no, which will help tell if a animal have them. An example is how whiskers, short ears, and retractable claws are all "traits" of a "cat". Now lets seperate the data into the predictors and outcome.

In [73]:

```
#Now lets get the values for 'X' and 'Y':

x = data.drop(columns='Monster')
y = data['Monster']

x.columns
```

Out[73]:

```
Index(['Has Claws', 'Has Scales', 'Long Tail', 'Short Tail', 'Has Wings',
       'Feathers', 'Has Fur', 'Has Beak', 'Sharp Teeth', 'Height(Ft)',
       'Length(Ft)', 'Breathes Fire', 'Has Fins/Gills', 'Has Horns',
       'Has Hooves', 'Two Legs', 'Four Legs', 'Has Whiskers', 'Spikey Tongue',
       'Tentacles', 'Two Arms', 'Two Arms +'],
      dtype='object')
```

In [74]:

```
model = DecisionTreeClassifier()
model.fit(x, y)
prediction = model.predict([[1,0,0,0,0,0,0,1,0,20,35,0,0,0,0,0,0,0,0,1,0,0]])
prediction
```

Out[74]:

```
array(['Kraken'], dtype=object)
```

We can also visually plot the information as well:

In [75]:

```
clf = DecisionTreeClassifier(random_state=0)
model = clf.fit(x, y)
```

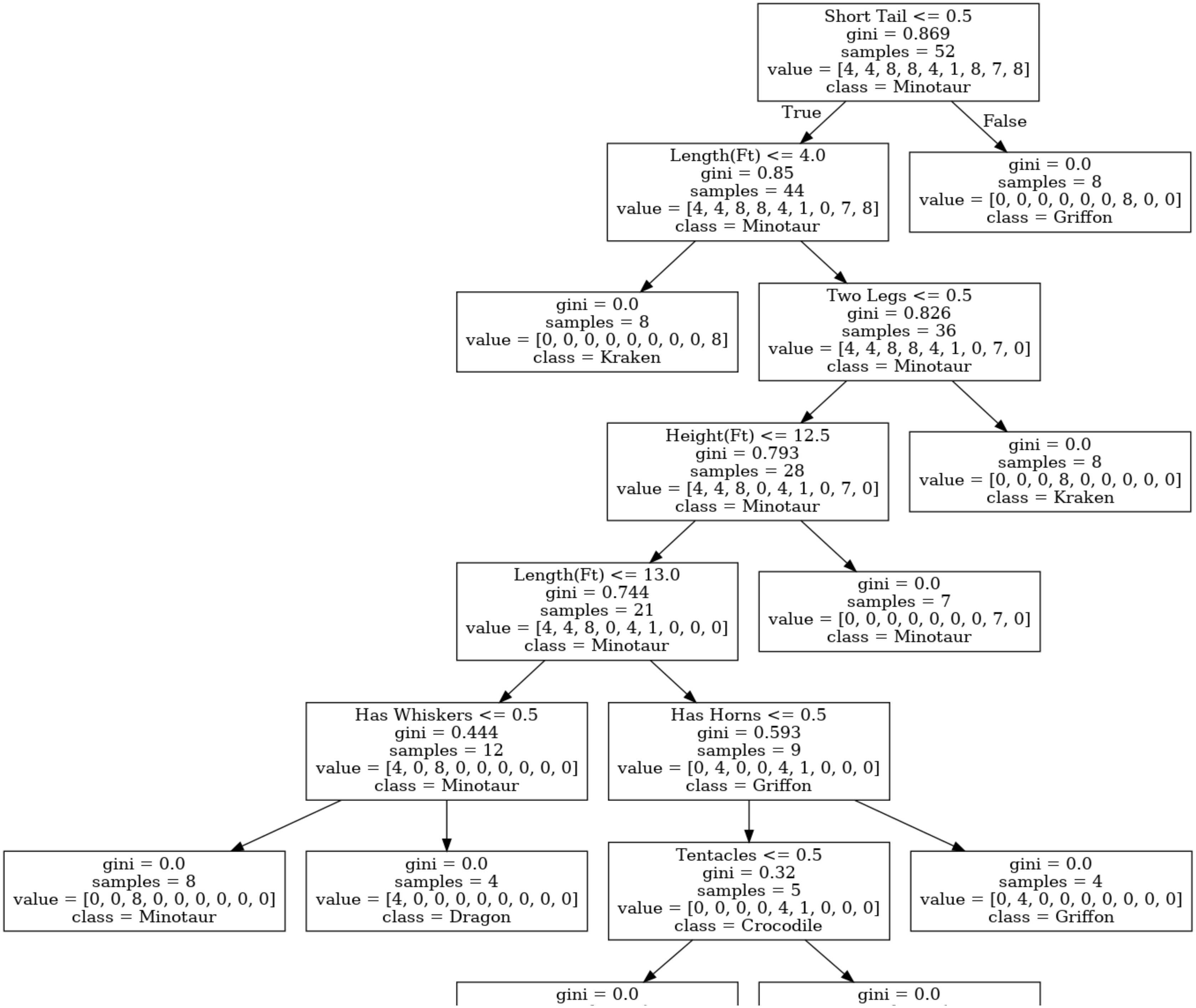
In [76]:

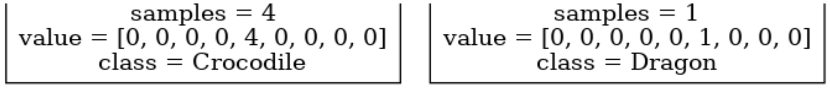
```
#Prepare dot data, where the fit the imformation for the graph

dot_data = tree.export_graphviz(model, out_file=None,
                                feature_names=['Has Claws', 'Has Scales', 'Long Tail', 'Short Tail', 'Has Wings',
                                               'Feathers', 'Has Fur', 'Has Beak', 'Sharp Teeth', 'Height(Ft)',
                                               'Length(Ft)', 'Breathes Fire', 'Has Fins/Gills', 'Has Horns',
                                               'Has Hooves', 'Two Legs', 'Four Legs', 'Has Whiskers', 'Spikey Tongue',
                                               'Tentacles', 'Two Arms', 'Two Arms +'],
                                class_names=y)
#Draw Graph
graph = pydotplus.graph_from_dot_data(dot_data)

#Show grapjh
Image(graph.create_png())
```

Out[76]:


Short Tail <= 0.5
gini = 0.869
samples = 52
value = [4, 4, 8, 8, 4, 1, 8, 7, 8]
class = Minotaur
True
False
Length(Ft) <= 4.0
gini = 0.85
samples = 44
value = [4, 4, 8, 8, 4, 1, 0, 7, 8]
class = Minotaur
gini = 0.0
samples = 8
value = [0, 0, 0, 0, 0, 0, 8, 0, 0]
class = Griffon
gini = 0.0
samples = 8
value = [0, 0, 0, 0, 0, 0, 0, 0, 8]
class = Kraken
Two Legs <= 0.5
gini = 0.826
samples = 36
value = [4, 4, 8, 8, 4, 1, 0, 7, 0]
class = Minotaur
Height(Ft) <= 12.5
gini = 0.793
samples = 28
value = [4, 4, 8, 0, 4, 1, 0, 7, 0]
class = Minotaur
gini = 0.0
samples = 8
value = [0, 0, 0, 8, 0, 0, 0, 0, 0]
class = Kraken
Length(Ft) <= 13.0
gini = 0.744
samples = 21
value = [4, 4, 8, 0, 4, 1, 0, 0, 0]
class = Minotaur
gini = 0.0
samples = 7
value = [0, 0, 0, 0, 0, 0, 0, 7, 0]
class = Minotaur
Has Whiskers <= 0.5
gini = 0.444
samples = 12
value = [4, 0, 8, 0, 0, 0, 0, 0, 0]
class = Minotaur
Has Horns <= 0.5
gini = 0.593
samples = 9
value = [0, 4, 0, 0, 4, 1, 0, 0, 0]
class = Griffon
gini = 0.0
samples = 8
value = [0, 0, 8, 0, 0, 0, 0, 0, 0]
class = Minotaur
gini = 0.0
samples = 4
value = [4, 0, 0, 0, 0, 0, 0, 0, 0]
class = Dragon
Tentacles <= 0.5
gini = 0.32
samples = 5
value = [0, 0, 0, 0, 4, 1, 0, 0, 0]
class = Crocodile
gini = 0.0
samples = 4
value = [0, 4, 0, 0, 0, 0, 0, 0, 0]
class = Griffon
gini = 0.0
gini = 0.0



Now let's use the "accuracy_score" with "train_test_split" method to predict the accuracy of the model, where "1" is a hundred percent accurate, and gets less accurate the lower it goes:

In [77]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
y_prediction = model.predict(x_test)
score = accuracy_score(y_test, y_prediction)
score
```

Out[77]:

```
1.0
```

Fin