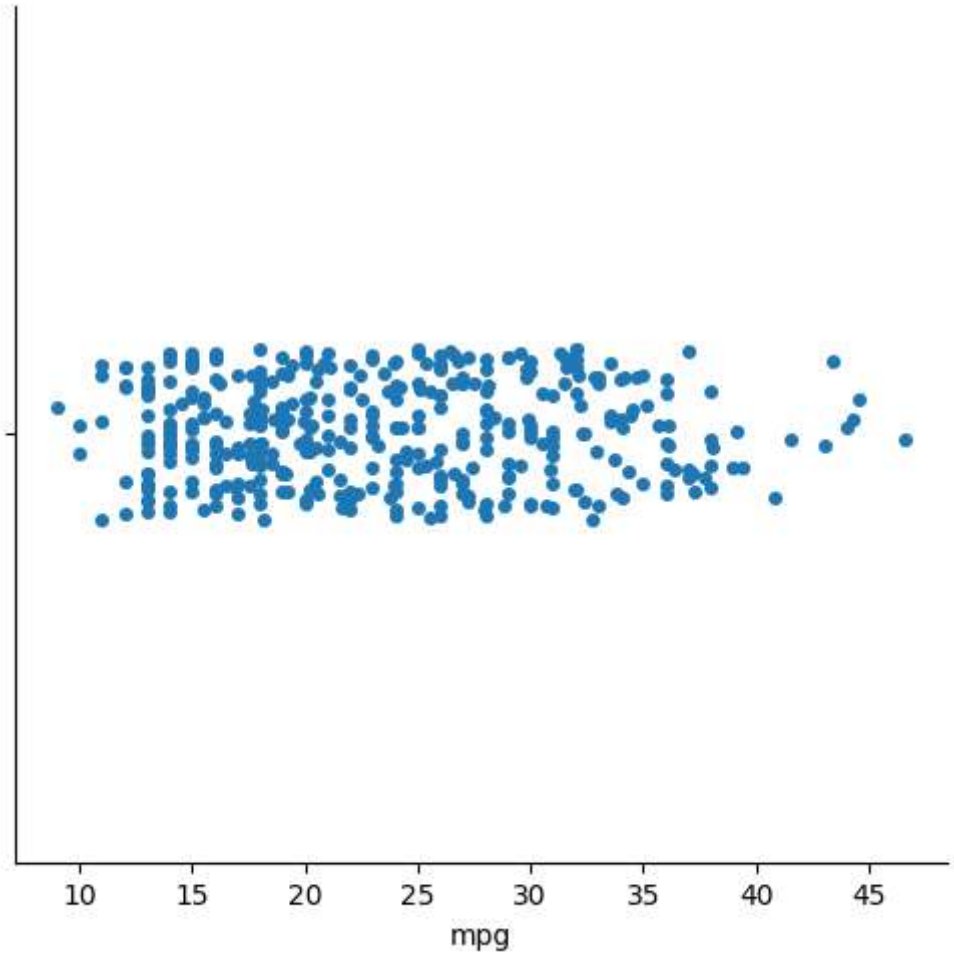
In [14]:
```
# This is a sample Python script.

# Press Shift+F10 to execute it or replace it with your code.
# Press Double Shift to search everywhere for classes, files, tool windows, actions
import pandas as pd
import numpy as np
import seaborn
import seaborn as sns
import matplotlib.pyplot as plt
import jupyter as j
import sklearn as sks
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn import tree
from sklearn.neural_network import MLPClassifier

#read in auto using pandas
dataFrame = pd.read_csv("Auto.csv")

#prints out first few rows and dimensions of data
print("--------------First few rows of data--------------")
print(dataFrame.head(3))
print("Dimensions are", len(dataFrame), "rows x 9 columns")

#uses describe on mpg, weight, and year. Caculates average and range
print("--------------MPG stats--------------")
print(dataFrame['mpg'].describe())
print("Average = 23.445918")
print("Range = 28.6")

print("--------------WEIGHT stats--------------")
print(dataFrame['weight'].describe())
print("Average = 2,977.584184")
print("Range = 3,527")

print("--------------YEAR stats--------------")
print(dataFrame['year'].describe())
print("Average = 76.010256")
print("Range = 12")

#Checks type of each column
print("--------------TYPES--------------")
print("         mpg type = ", dataFrame['mpg'].dtypes)
print("   cylinders type = ", dataFrame['cylinders'].dtypes)
print("displacement type = ", dataFrame['displacement'].dtypes)
print("  horsepower type = ", dataFrame['horsepower'].dtypes)
print("      weight type = ", dataFrame['weight'].dtypes)
print("acceleration type = ", dataFrame['acceleration'].dtypes)
print("        year type = ", dataFrame['year'].dtypes)
print("      origin type = ", dataFrame['origin'].dtypes)
print("        name type = ", dataFrame['name'].dtypes)

#Changes cylinders and origin datatypes to categorical and then checks to make sure
```

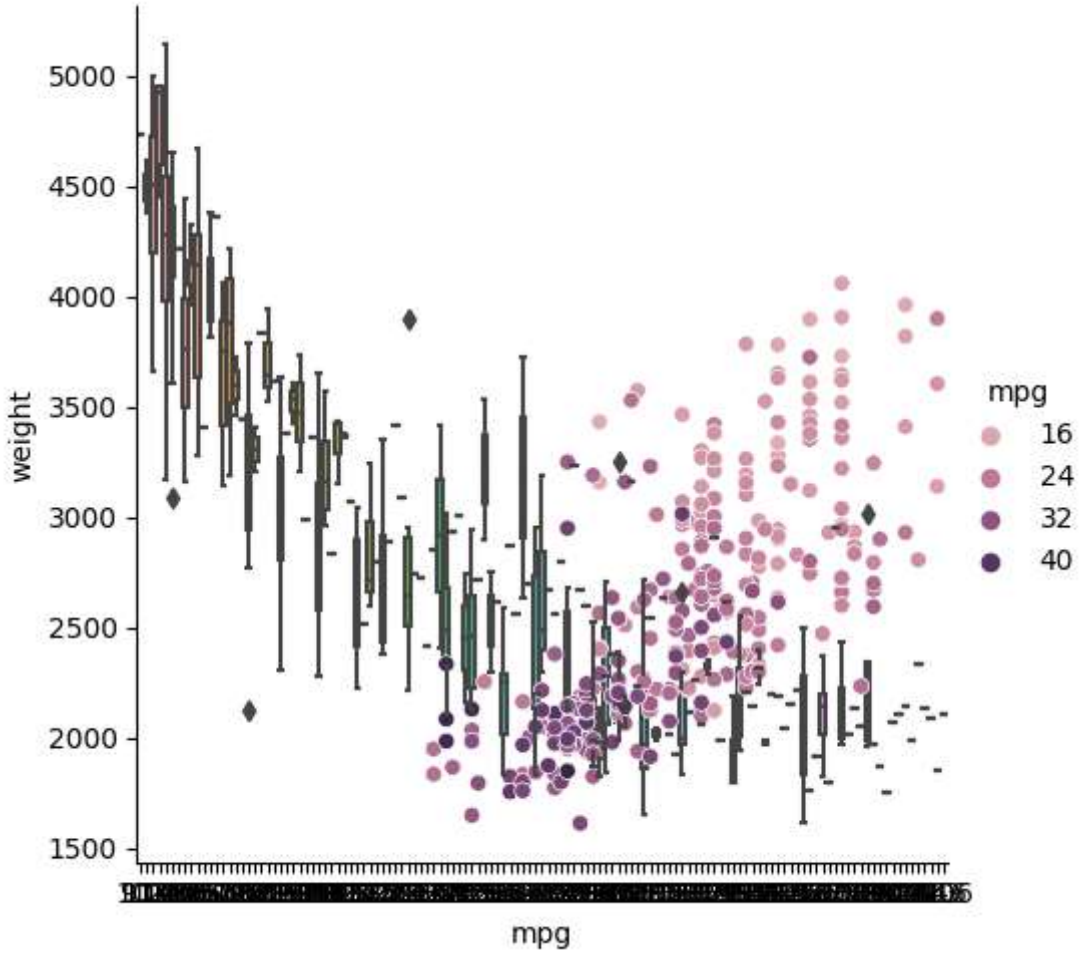
```
print("--------------CHANGING TYPES--------------")
####NOT SURE IF I DID ONE AS CAT.CODES AND ONE NOT AS CAT.CODES
dataFrame['cylinders'] = dataFrame['cylinders'].astype("category")
dataFrame['origin'] = dataFrame['origin'].astype("category")
print("cylinders type = ", dataFrame['cylinders'].dtypes)
print("   origin type = ", dataFrame['origin'].dtypes)

#Drops rows that have NA values and prints out new dimensions
print("--------------DROPPING ROWS THAT HAVE NA--------------")
dataFrame = dataFrame.dropna()
###CONFIRM THIS IS WHAT OTHERS GOT
print("New Dimensions are", len(dataFrame), "rows x 9 columns")

#print("--------------MODIFYING COLUMNS--------------")
#dataFrame["mpg_high"] = "10"
#dataFrame['mpg_high'] = dataFrame['mpg_high'].astype("category")
#dataFrame['mpg'] = np.where(dataFrame['mpg'])
#dataFrame.loc[dataFrame['mpg'] > 103, 'mpg_high'] = '1'
####FOR SOME REASON THERE IS 3 ROWS AT THE VERY BOTTOM WITH NaN VALUES.
#for i in range(0, 392):
   # if dataFrame.at[i, 'mpg'] > 23.445918:
      #  dataFrame.at[i, 'mpg_high'] = '1'
    #else:
       # dataFrame.at[i, 'mpg_high'] = '0'

#print(dataFrame.head())
#print(dataFrame.tail(20))
#print("mpg_high column created")
#print("mpg and name columns deleted")
#print(dataFrame.head())

print("--------------PLOTTING DATA--------------")
#g = seaborn.catplot(x="mpg", y="horsepower", hue="weight", data=dataFrame)
seaborn.catplot(x="mpg", data=dataFrame)
print("This graph shows the amount of 0 to 1's")
seaborn.relplot(x="horsepower", y="weight", hue="mpg", data=dataFrame)
print("This graph shows the trend of worse mpg to higher weight")
seaborn.boxplot(x="mpg", y="weight", data=dataFrame)
print("This graph shows also the trend of weight to mpg.")
#Plot.show()
#sns.distplot(dataFrame["mpg"], kde=True, rug=True)
#print(g)
#plt.show(g)

print("--------------SPLITTING DATA--------------")

x = dataFrame[['weight', 'cylinders']].head(320)
y = dataFrame['origin'].head(320)
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=1234, test_s

print("--------------LOGISTIC REGRESSION--------------")
clf = LogisticRegression(random_state=0).fit(x, y)
print(clf.predict(x_test))

print("--------------DECISION TREE--------------")
#clftree = tree.DecisionTreeClassifier
```

```
#clftree = clftree.fit(x, y_train)
#clftree.predict(y_test)

print("--------------NEURAL NETWORK--------------")
clfNN = MLPClassifier(solver='lbfgs', alpha= 1e-5, hidden_layer_sizes=(5-2), random
clfNN.fit(x, y)
print(clfNN.predict(x_test))

print("--------------ANALYSIS--------------")
print("Since I couldnt get the decision tree working, between the neural network an
      "The logistic regression model performed better.")
print("For the accuracy and metrics of each model. The neural network predicted all
      "prediction. For the logistic regression the predictions are closer to the ac
print("I think logistic regression performed better since the data is a smaller dat
      "the neural network would have performed better")
print("I much prefer to use R instead of sklearn, in R a lot of things are simplier
      "frames is a lot easier. I did not like coding this in python at all and only
```

```
--------------First few rows of data--------------
    mpg  cylinders  displacement  horsepower  weight  acceleration  year  \
0  18.0          8         307.0         130    3504          12.0  70.0  \
1  15.0          8         350.0         165    3693          11.5  70.0
2  18.0          8         318.0         150    3436          11.0  70.0

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
Dimensions are 392 rows x 9 columns
--------------MPG stats--------------
count    392.000000
mean      23.445918
std        7.805007
min        9.000000
25%       17.000000
50%       22.750000
75%       29.000000
max       46.600000
Name: mpg, dtype: float64
Average = 23.445918
Range = 28.6
--------------WEIGHT stats--------------
count     392.000000
mean     2977.584184
std       849.402560
min      1613.000000
25%      2225.250000
50%      2803.500000
75%      3614.750000
max      5140.000000
Name: weight, dtype: float64
Average = 2,977.584184
Range = 3,527
--------------YEAR stats--------------
count    390.000000
mean      76.010256
std        3.668093
min       70.000000
25%       73.000000
50%       76.000000
75%       79.000000
max       82.000000
Name: year, dtype: float64
Average = 76.010256
Range = 12
--------------TYPES--------------
         mpg type =  float64
   cylinders type =  int64
displacement type =  float64
  horsepower type =  int64
      weight type =  int64
acceleration type =  float64
        year type =  float64
      origin type =  int64
```

```
        name type =  object
--------------CHANGING TYPES--------------
cylinders type =  category
   origin type =  category
--------------DROPPING ROWS THAT HAVE NA--------------
New Dimensions are 389 rows x 9 columns
--------------PLOTTING DATA--------------
This graph shows the amount of 0 to 1's
This graph shows the trend of worse mpg to higher weight
This graph shows also the trend of weight to mpg.
--------------SPLITTING DATA--------------
--------------LOGISTIC REGRESSION--------------
[2 1 2 3 1 2 1 1 1 2 1 1 1 1 1 1 2 2 3 3 1 3 1 1 1 2 2 3 2 1 1 2 1 1 1 3 1
 1 2 2 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 2 2 1 3 1 2 3 1 1 1]
--------------DECISION TREE--------------
--------------NEURAL NETWORK--------------
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
--------------ANALYSIS--------------
Since I couldnt get the decision tree working, between the neural network and logist
ic regression.The logistic regression model performed better.
For the accuracy and metrics of each model. The neural network predicted all values
to be 1. Which is not a realisticprediction. For the logistic regression the predict
ions are closer to the actual data.
I think logistic regression performed better since the data is a smaller data set an
d maybe if we had a very large data setthe neural network would have performed bette
r
I much prefer to use R instead of sklearn, in R a lot of things are simplier. Handli
ng data and messing around with dataframes is a lot easier. I did not like coding th
is in python at all and only ran into trouble
```


10
15
20
25
30
35
40
45
mpg


5000
4500
4000
3500
3000
2500
2000
1500
weight
mpg
mpg
16
24
32
40

In [ ]: