# Read in the Auto data file.

```
import pandas as pd
df = pd.read_csv('Auto.csv')

print(df.head())
print('\nDimensions of data frame:', df.shape)
```

```
    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
3  16.0          8         304.0         150    3433          12.0  70.0
4  17.0          8         302.0         140    3449           NaN  70.0

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino

Dimensions of data frame: (392, 9)
```

# Data Exploration of the Auto Data file

```
print(df.describe())
```

```
              mpg   cylinders  displacement  horsepower       weight  \
count  392.000000  392.000000    392.000000  392.000000   392.000000
mean    23.445918    5.471939    194.411990  104.469388  2977.584184
std      7.805007    1.705783    104.644004   38.491160   849.402560
min      9.000000    3.000000     68.000000   46.000000  1613.000000
25%     17.000000    4.000000    105.000000   75.000000  2225.250000
50%     22.750000    4.000000    151.000000   93.500000  2803.500000
75%     29.000000    8.000000    275.750000  126.000000  3614.750000
max     46.600000    8.000000    455.000000  230.000000  5140.000000

       acceleration        year      origin
count    391.000000  390.000000  392.000000
mean      15.554220   76.010256    1.576531
std        2.750548    3.668093    0.805518
min        8.000000   70.000000    1.000000
25%       13.800000   73.000000    1.000000
50%       15.500000   76.000000    1.000000
75%       17.050000   79.000000    2.000000
max       24.800000   82.000000    3.000000
```

- The range for mpg is from 9 to 46.6 with a mean of 23.45
- The range for cylinders is from 3 to 8 with a mean of 5.47
- The range for displacement is from 68 to 455 with a mean of 194.41
- The range for horsepower is from 46 to 230 with a mean of 104.47
- The range for weight is from 1613 to 5140 with a mean of 2977.58
- The range for acceleration is from 8 to 24.8 with a mean of 15.55
- The range for year is from 70 to 82 with a mean of 76.01
- The range for origin is from 1 to 3 with a mean of 1.58

## Explore Data Types

```
print(df.dtypes)
print('\n')
df.cylinders = df.cylinders.astype('category').cat.codes
df.origin = df.origin.astype('category')

print(df.dtypes)
```

```
mpg             float64
cylinders         int64
displacement    float64
horsepower        int64
weight            int64
acceleration    float64
year            float64
origin            int64
name             object
dtype: object


mpg              float64
cylinders           int8
displacement     float64
horsepower         int64
weight             int64
acceleration     float64
year             float64
origin          category
name              object
dtype: object
```

## Remove NAs

```
df = df.dropna()
print('\nDimensions of data frame:', df.shape)
```

```
Dimensions of data frame: (389, 9)
```

## Modify Columns

Create the mph_high column and remove the mpg and name columns

```
avg = df['mpg'].mean()
new_df_copy = df.loc[df.mpg>1].copy()
new_df_copy.loc[:, 'mpg_high'] = [0 if x < avg else 1 for x in
                                  new_df_copy['mpg']]
new_df_copy.head()
df = new_df_copy
df.mpg_high = df.mpg_high.astype('category')
df = df.drop(columns=['mpg', 'name'])
print(df.head())
```

```
   cylinders  displacement  horsepower  weight  acceleration  year origin  \
0          4         307.0         130    3504          12.0  70.0      1
1          4         350.0         165    3693          11.5  70.0      1
2          4         318.0         150    3436          11.0  70.0      1
3          4         304.0         150    3433          12.0  70.0      1
6          4         454.0         220    4354           9.0  70.0      1

  mpg_high
0        0
1        0
2        0
3        0
6        0
```

## Data exploration with graphs

```
import seaborn as sb

sb.catplot(x="mpg_high", kind='count', data=df)
```

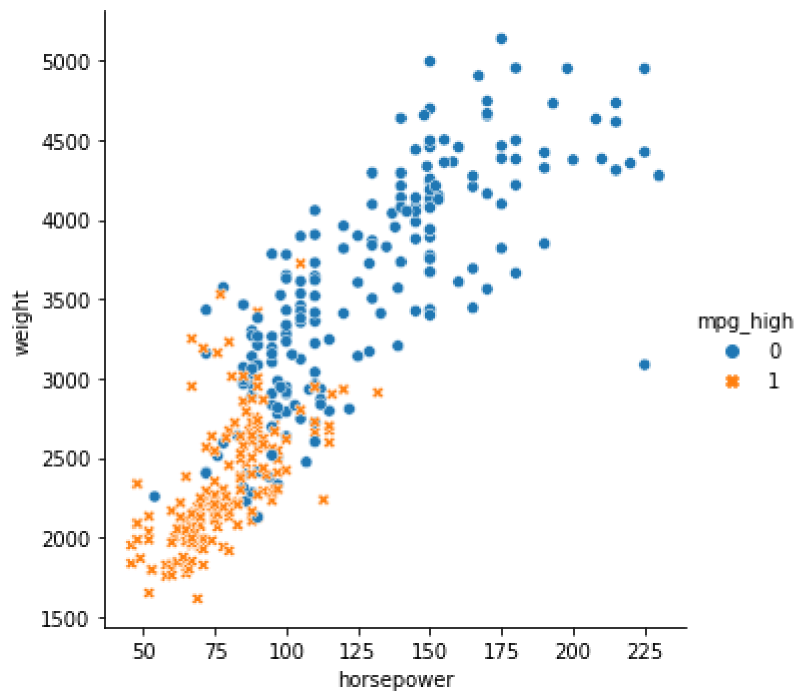
```
<seaborn.axisgrid.FacetGrid at 0x7f4283d57650>
```


seaborn catplot on the mpg_high column

- I learned that there are more instances with lower miles per gallon than there are higher miles per gallon.
- addtionally the split between low and high miles per gallon in the dataframe is relativly even.

```
sb.relplot(x="horsepower", y='weight', data=df, hue=df.mpg_high,
           style=df.mpg_high)
```

```
<seaborn.axisgrid.FacetGrid at 0x7f427a1a6e50>
```

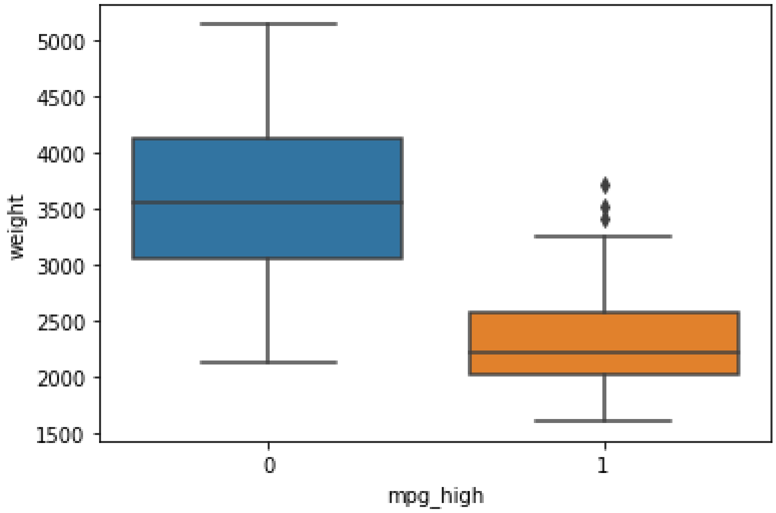



seaborn relplot with horsepower on the x axis, weight on the y axis, setting hue or style to mpg_high

- I learned that the lighter the car and the less horsepower it has the more likely it is to have a higher miles per gallon.
- So a heavy car with a lot of horsepower will rapidly consume gas.

```
sb.boxplot(x="mpg_high", y="weight", data=df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f426b004110>
```




Seaborn boxplot with mpg_high on the x axis and weight on the y axis

- I learned that on average a high miles per gallon car will weigh about 1000 pounds less than a low miles per gallon car.
- Additionally only a few outlier high mpg cars weighed more than the average weight of low mpg cars.

# Train/Test 80/20 split

```
from sklearn.model_selection import train_test_split

X = df.loc[:, ['year', 'origin', 'horsepower', 'weight', 'displacement',
               'acceleration', 'cylinders']]
y = df.mpg_high

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
                                                    random_state=1234)

print('X_train size:', X_train.shape)
print('X_test size:', X_test.shape)
print('Y_train size:', y_train.shape)
print('Y_test size:', y_test.shape)
```

```
X_train size: (311, 7)
X_test size: (78, 7)
Y_train size: (311,)
Y_test size: (78,)
```

# Logistic Regression

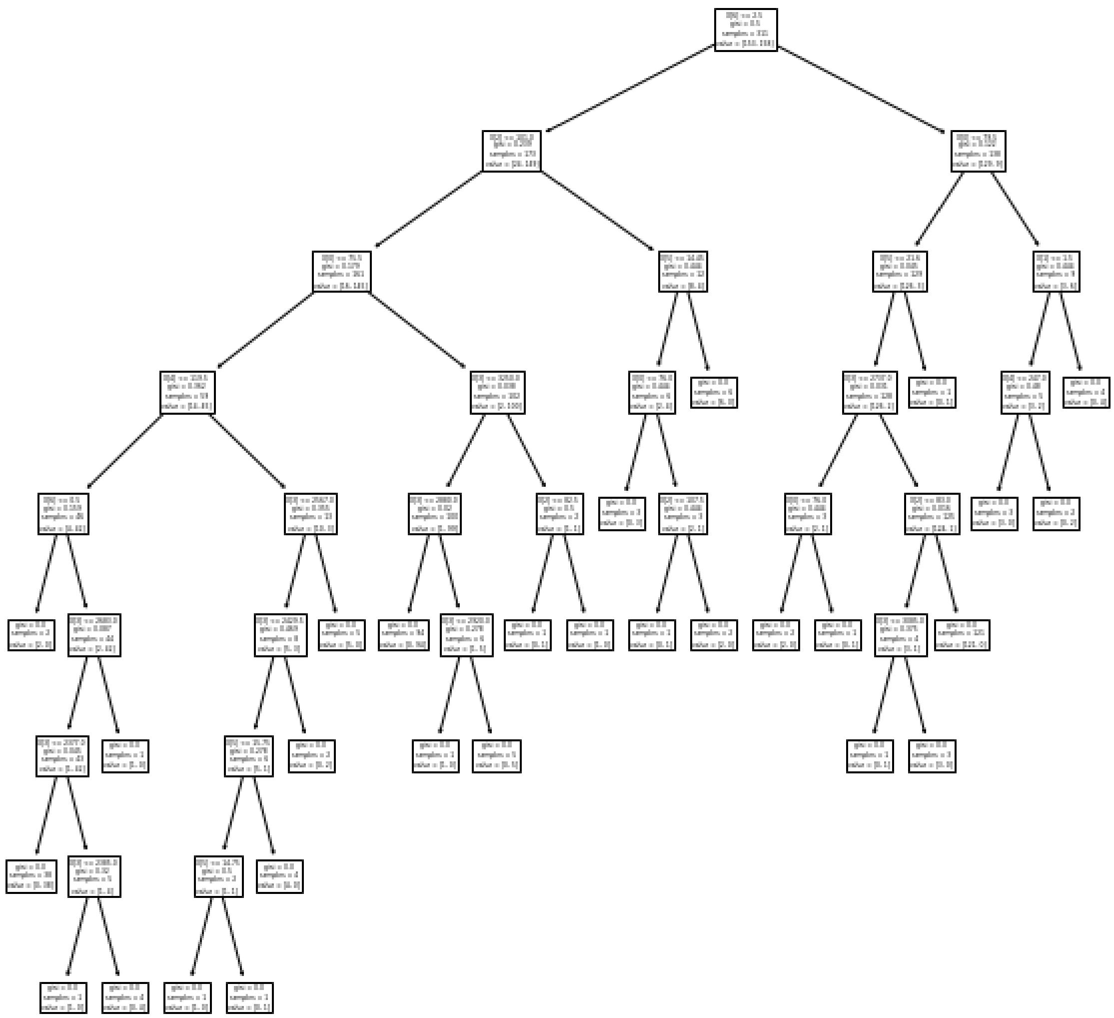
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

clf = LogisticRegression(solver = 'lbfgs', max_iter = 400)
clf.fit(X_train, y_train)

pred = clf.predict(X_test)

target_names = ['low_mpg', 'high_mpg']

print(classification_report(y_test, pred, target_names=target_names))
```

```
              precision    recall  f1-score   support

     low_mpg       0.98      0.80      0.88        50
    high_mpg       0.73      0.96      0.83        28

    accuracy                           0.86        78
   macro avg       0.85      0.88      0.85        78
weighted avg       0.89      0.86      0.86        78
```

# Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import tree
from matplotlib import pyplot as plt

clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)

pred = clf.predict(X_test)

target_names = ['low_mpg', 'high_mpg']

print(classification_report(y_test, pred, target_names=target_names))

plt.figure(figsize=(10, 10))
tree.plot_tree(clf)
plt.show()
```

```
              precision    recall  f1-score   support

     low_mpg       0.94      0.88      0.91        50
    high_mpg       0.81      0.89      0.85        28

    accuracy                           0.88        78
   macro avg       0.87      0.89      0.88        78
weighted avg       0.89      0.88      0.89        78
```



## Neural Network

```
from sklearn import preprocessing

scaler = preprocessing.StandardScaler().fit(X_train)

X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

from sklearn.neural_network import MLPClassifier
```

```
# since the input size will be 7, we will have 6 nodes

clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(6), max_iter=500,
                    random_state=1234)
clf.fit(X_train_scaled, y_train)

pred = clf.predict(X_test_scaled)

print(classification_report(y_test, pred))

clf2 = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(3, 3), max_iter=500,
                     random_state=1234)
clf2.fit(X_train_scaled, y_train)

pred2 = clf2.predict(X_test_scaled)

print(classification_report(y_test, pred2))
```

```
              precision    recall  f1-score   support

           0       0.92      0.90      0.91        50
           1       0.83      0.86      0.84        28

    accuracy                           0.88        78
   macro avg       0.87      0.88      0.88        78
weighted avg       0.89      0.88      0.89        78
              precision    recall  f1-score   support

           0       0.96      0.88      0.92        50
           1       0.81      0.93      0.87        28

    accuracy                           0.90        78
   macro avg       0.88      0.90      0.89        78
weighted avg       0.90      0.90      0.90        78
```

- The performance of the two configuations was very similar with the second configuration having a slightly higher accuracy than the second configuration. The likely reason for the similar performance of the two algorithms is that the data is relativly linear minimizing the benefit of adding more hidden layers to the neural network.

# Analysis

- Which algorithm performed better?

    - Of the three algorithms the neural network algorithm performed the best and the logistic regression algorithm performed the worst.
- Compare accuracy, recall and precision metrics by class
    - Logistic regression had an accuracy of 86%, recall of 80% for low mpg and 96% for high mpg, and precision of 98% for low mpg and 81% for high mpg.
    - Decision tree had an accuracy of 88%, recall of 88% for low mpg and 89% for high mpg, and precision of 94% for low mpg and 81% for high mpg.
    - Neural network had an accuracy of 90%, recall of 88% for low mpg and 93% for high mpg, and precision of 96% for low mpg and 81% for high mpg.
    - In general the low mpg class has a higher precision in all algorithms and the high mpg class has a higher recall in all algorithms.
- Give your analysis of why the better-performing algorithm might have outperformed the other
    - The likely reason why the neural network outperformed the other algorithms is that I tuned the parameters of the neural network algorithm by changing the number of hidden layers for the neural network and the number of neurons in each layer.
- Write a couple of sentences comparing your experiences using R versus sklearn. Feel free to express strong preferences.
    - I feel that in R the creation an manipulation of a dataframe came much more naturally to me and in python I needed to do a lot of research about how to make a new column with the values that I wanted. I prefer the way that R handles the seperation of labels and the rest of the data while in Python we need a test and train for both the data and the labels. Other than these differences the preformance of both R and Python have been similar to me.

✓ 0s completed at 11:47 PM