```
import pandas as pd

#read CSV file
data = pd.read_csv("Auto.csv")

#output information
print("dimension of data: [", data.shape[0], "x", data.shape[1],"]")
print(data.head())
```

```
dimension of data: [ 392 x 9 ]
    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
```

```
#Avg:   23.4
#Range: 37.6
print(data["mpg"].describe(),"\n")

#Avg:   2977.58
#Range: 3527
print(data["weight"].describe(),"\n")

#Avg:   76.010256
#Range: 12
print(data["year"].describe(),"\n")
```

```
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

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

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
```

```
print(data.dtypes, "\n")
#if you re-run this code the 1st and 2nd print will be the same

#convert columns to categorical data type
data["cylinders"] = data["cylinders"].astype('category')
data["origin"] = data["origin"].astype('category')

#convert categorical data to integer codes
data["cylindersCodes"] = data["cylinders"].cat.codes
data["originCodes"] = data["origin"].cat.codes
```

```
print(data.dtypes)
```

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

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

```
#delete all rows with NAs
data = data.dropna()
print("dimension of data: [", data.shape[0], "x", data.shape[1],"]")
```

```
dimension of data: [ 389 x 11 ]
```

```
data["mpgHigh"] = data["mpg"].apply(lambda m: 1 if m > 23.4 else 0)
data = data.drop(["mpg", "name"], axis=1)
print(data.head())
```

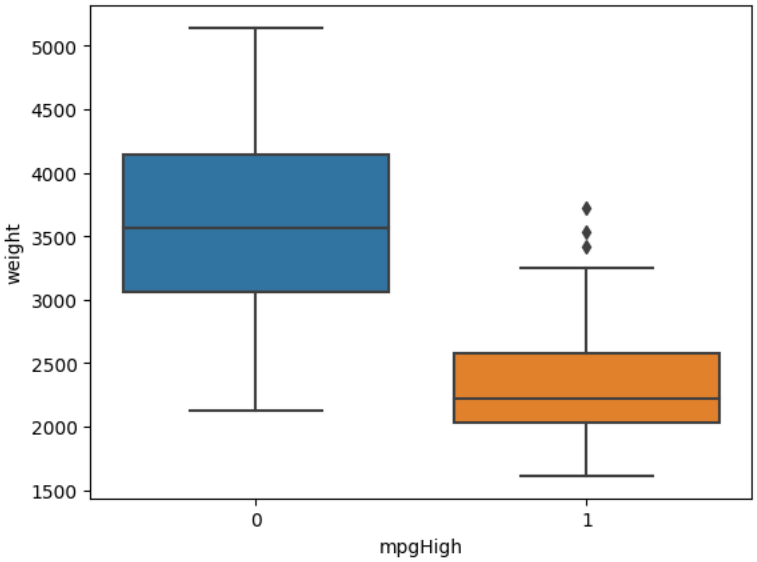
```
  cylinders  displacement  horsepower  weight  acceleration  year origin  \
0         8         307.0         130    3504          12.0  70.0      1
1         8         350.0         165    3693          11.5  70.0      1
2         8         318.0         150    3436          11.0  70.0      1
3         8         304.0         150    3433          12.0  70.0      1
6         8         454.0         220    4354           9.0  70.0      1

   cylindersCodes  originCodes  mpgHigh
0               4            0        0
1               4            0        0
2               4            0        0
3               4            0        0
6               4            0        0
```

```
import seaborn as sns
sns.catplot(x="mpgHigh", kind="count", data=data) #There is roughly a 50/50 split with mpg high/low in the data set with there being slightly
```

```
<seaborn.axisgrid.FacetGrid at 0x7f2a19076b20>
```

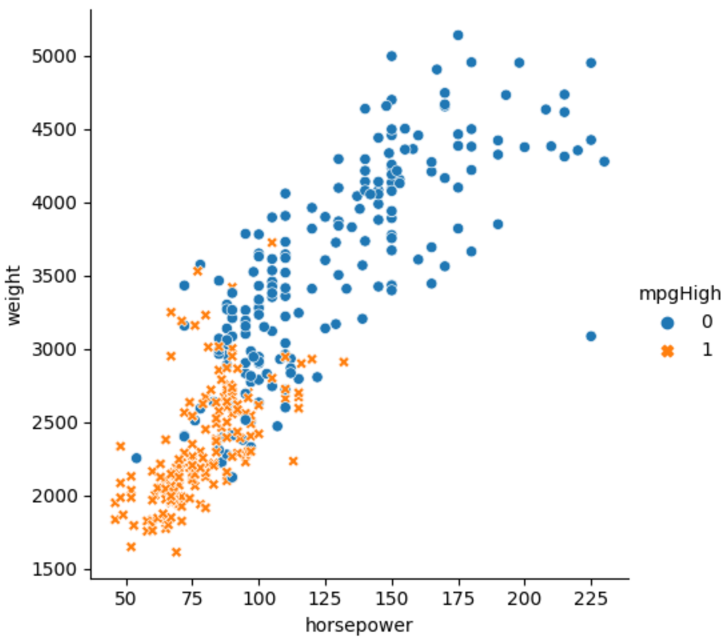
```
sns.boxplot(x="mpgHigh", y="weight", data=data) #MPG high cars typically weigh less than MGP low cars
```

```
<Axes: xlabel='mpgHigh', ylabel='weight'>
```




```
sns.relplot(x="horsepower", y="weight", hue="mpgHigh", style="mpgHigh", data=data) #As the weight of the car increases so does the horsepower
```

```
<seaborn.axisgrid.FacetGrid at 0x7f2a4888aee0>
```




```
from sklearn.model_selection import train_test_split

#split data into X and y
dataY = data["mpgHigh"]
dataX = data.drop("mpgHigh", axis=1)

#split data into train and test sets
XTrain, XTest, YTrain, YTest = train_test_split(dataX, dataY, test_size=0.2, random_state=1234)

print("Dimensions of train data: ", XTrain.shape, YTrain.shape)
print("Dimensions of test data: ", XTest.shape, YTest.shape)
```

```
Dimensions of train data:  (311, 9) (311,)
Dimensions of test data:  (78, 9) (78,)
```

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

#logistic regression model
lrModel = LogisticRegression(solver='lbfgs', max_iter=1000)
lrModel.fit(XTrain, YTrain)

#test and evaluate model
predModel = lrModel.predict(XTest)
print(classification_report(YTest, predModel))
```

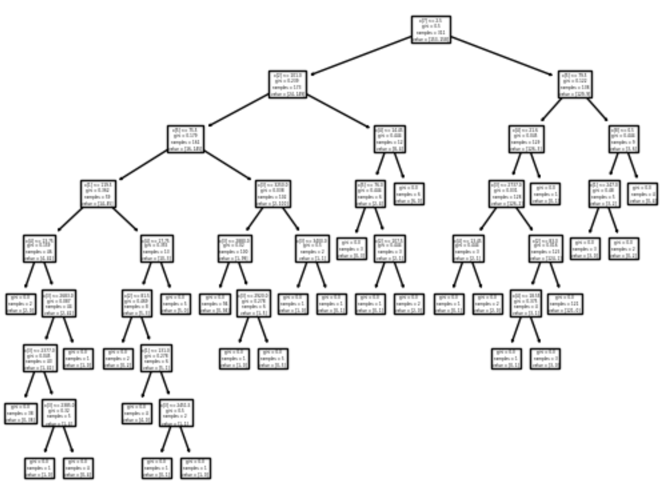
```
              precision    recall  f1-score   support

           0       0.98      0.82      0.89        50
           1       0.75      0.96      0.84        28

    accuracy                           0.87        78
   macro avg       0.86      0.89      0.87        78
weighted avg       0.89      0.87      0.87        78
```

```
from sklearn.tree import DecisionTreeClassifier

#train decision tree model
dtModel = DecisionTreeClassifier(random_state=1234).fit(XTrain, YTrain)

#test and evaluate model
y_pred = dtModel.predict(XTest)
print(classification_report(YTest, y_pred))
```

```
              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
```

```
from sklearn import tree
import graphviz

#print tree
tree.plot_tree(dtModel)

#create file
dotData = tree.export_graphviz(dtModel, out_file=None)
graph = graphviz.Source(dotData)
graph.render("cars")
```

```
'cars.pdf'
```



```
from sklearn.neural_network import MLPClassifier
```

```
nn1 = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(8, 6), random_state=1234, max_iter=1000)
nn1.fit(XTrain, YTrain)

#test and evaluate first model
pred1 = nn1.predict(XTest).round().astype(int)
print(classification_report(YTest, pred1))

nn2 = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(32, 16, 8), random_state=1234, max_iter=1000)
nn2.fit(XTrain, YTrain)

#test and evaluate first model
pred2 = nn2.predict(XTest).round().astype(int)
print(classification_report(YTest, pred2))
```

```
              precision    recall  f1-score   support

           0       0.98      0.82      0.89        50
           1       0.75      0.96      0.84        28

    accuracy                           0.87        78
   macro avg       0.86      0.89      0.87        78
weighted avg       0.89      0.87      0.87        78

              precision    recall  f1-score   support

           0       0.93      0.84      0.88        50
           1       0.76      0.89      0.82        28

    accuracy                           0.86        78
   macro avg       0.85      0.87      0.85        78
weighted avg       0.87      0.86      0.86        78
```

Neural Netwoks Analysis

For the **Neural Netwoks** I tried many different sizes and layers of hidden nodes but the results were usually the same. I noticed that the more extensive the neural network was, the less accurate it became but only by about a couple of hundredths, likely due to overfitting. The best model I found is the 1st one, which I found by trying to create the smallest model. My theory is that the data has a strong correlation but because of the many cases that overlap the model is bound to overfit.

Model Analysis

Ranked based on **precision (FALSE)**

1. Linear Model/Neural Network (8, 6)
2. Neural Network (32, 16, 8)
3. Decision Tree

Ranked based on **precision (TRUE)**

1. Decision Tree
2. Neural Network (32, 16, 8)
3. Linear Model/Neural Network (8, 6)

Ranked based on **recall**

1. Decision Tree
2. Neural Network (32, 16, 8)
3. Linear Model/Neural Network (8, 6)

Ranked based on **accuracy**

1. Decision Tree
2. Linear Model/Neural Network (8, 6)
3. Neural Network (32, 16, 8)

Based on the earlier graphs the data seems to be divided into 3 groups certainly high MPG, certainly low MPG, and the in-between. Because of that division, I believe that is why the decision tree performed the best out of each of the models. I can imagine the data as a 9th-dimensional box and the decision tree dividing up the sections in the smaller overlapping spaces which would explain its accuracy. The neural networks most likely do the same thing but end up overfitting. Lastly, the linear model can't has a hard time differentiating the overlap areas.

R vs Python

When I started this assignment I didn't think that I would enjoy python but I had an easier time with python than I did with R. I think I attribute this to how similar Python is to other languages than R,

✓ 2s completed at 10:02 PM