1.

In [ ]:
```
import pandas as pd
import numpy as np
df = pd.read_csv("Auto.csv")
print(df.head())
print("Size {}".format(df.size))
print("Shape {}".format(df.shape))
print("nDim {}".format(df.ndim))
```

```
    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
Size 3528
Shape (392, 9)
nDim 2
```

2.

In [ ]:
```
df['mpg'].describe()
```

Out[ ]:
```
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
```

3.

In [ ]:
```
print(df.dtypes)
df['cylinders'] = df['cylinders'].astype('category')
df['cylinders_codes'] = df['cylinders'].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         category
displacement       float64
horsepower           int64
weight               int64
acceleration       float64
year               float64
origin            category
name                object
cylinders_codes       int8
dtype: object
```

4.

In [ ]:
```
df = df.dropna()
print("Shape {}".format(df.shape))
```

```
Shape (389, 10)
```

5.

In [ ]:
```
df['mpg_high'] = df.apply(lambda row: 1 if row['mpg'] > df['mpg'].mean() else 0, ax
df = df.drop(columns=['mpg','name'])
print(df.head())
```

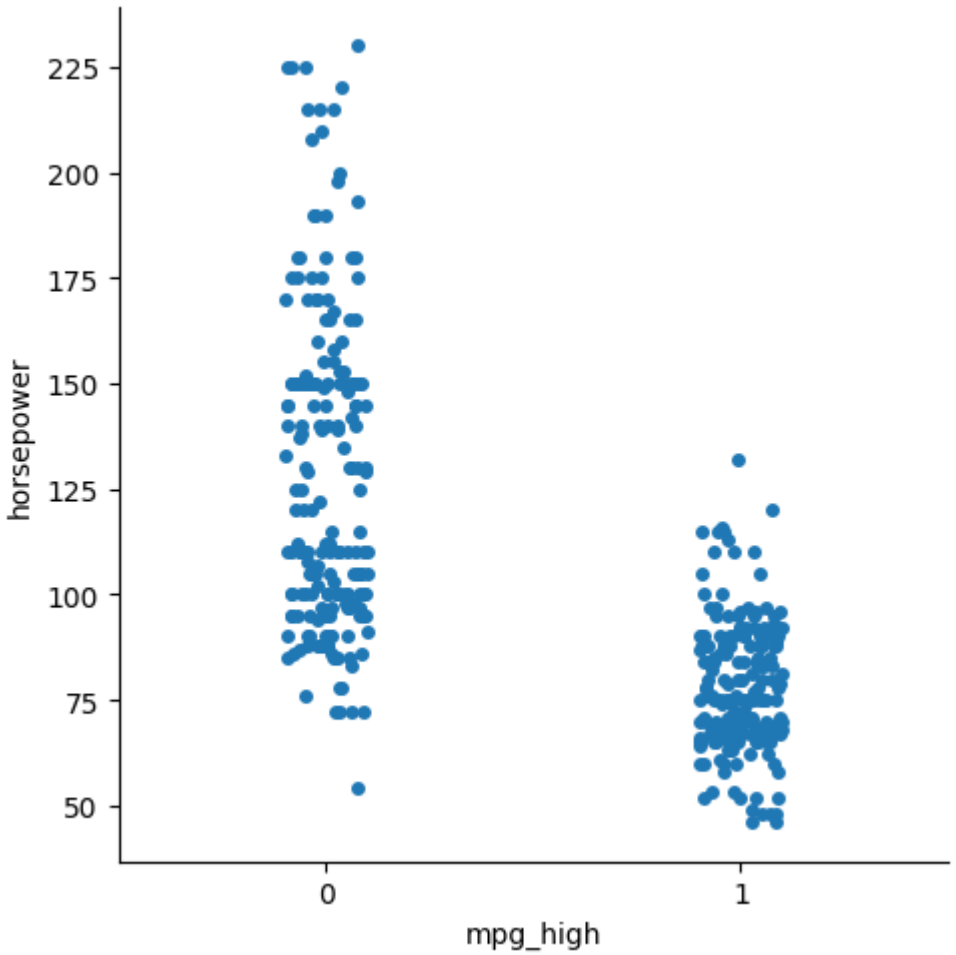
```
   cylinders  displacement  horsepower  weight  acceleration  year origin
0          8         307.0         130    3504          12.0  70.0      1  \
1          8         350.0         165    3693          11.5  70.0      1
2          8         318.0         150    3436          11.0  70.0     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

   cylinders_codes  mpg_high
0                4         0
1                4         0
2                4         0
3                4         0
6                4         0
```

6. Graphs Data Exploration

In [ ]:
```
import seaborn as sb
sb.catplot(data=df, x="mpg_high", y="horsepower")
```

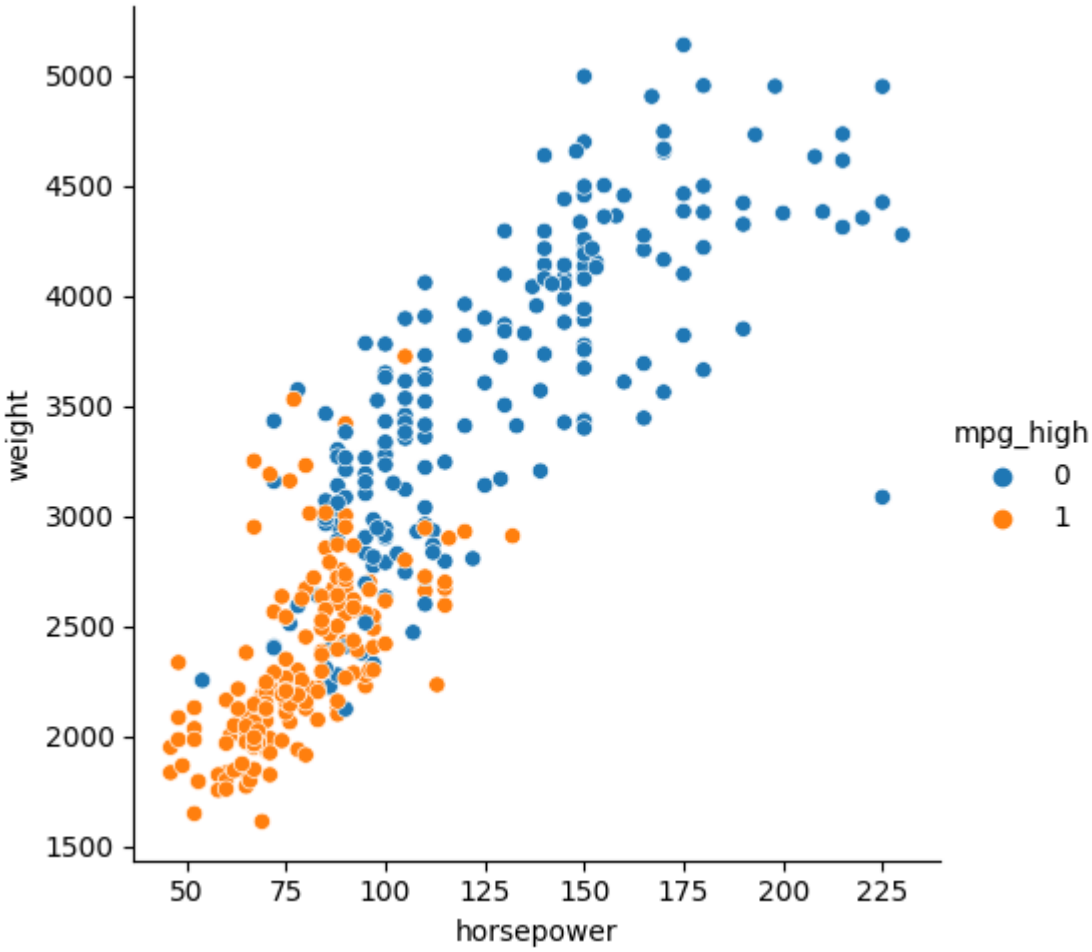
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x2113eccf490>



This graph shows that vehicles that have a high mpg tend to have a lower horse power.

In [ ]:
```
sb.relplot(data=df, x="horsepower", y="weight",hue="mpg_high")
```

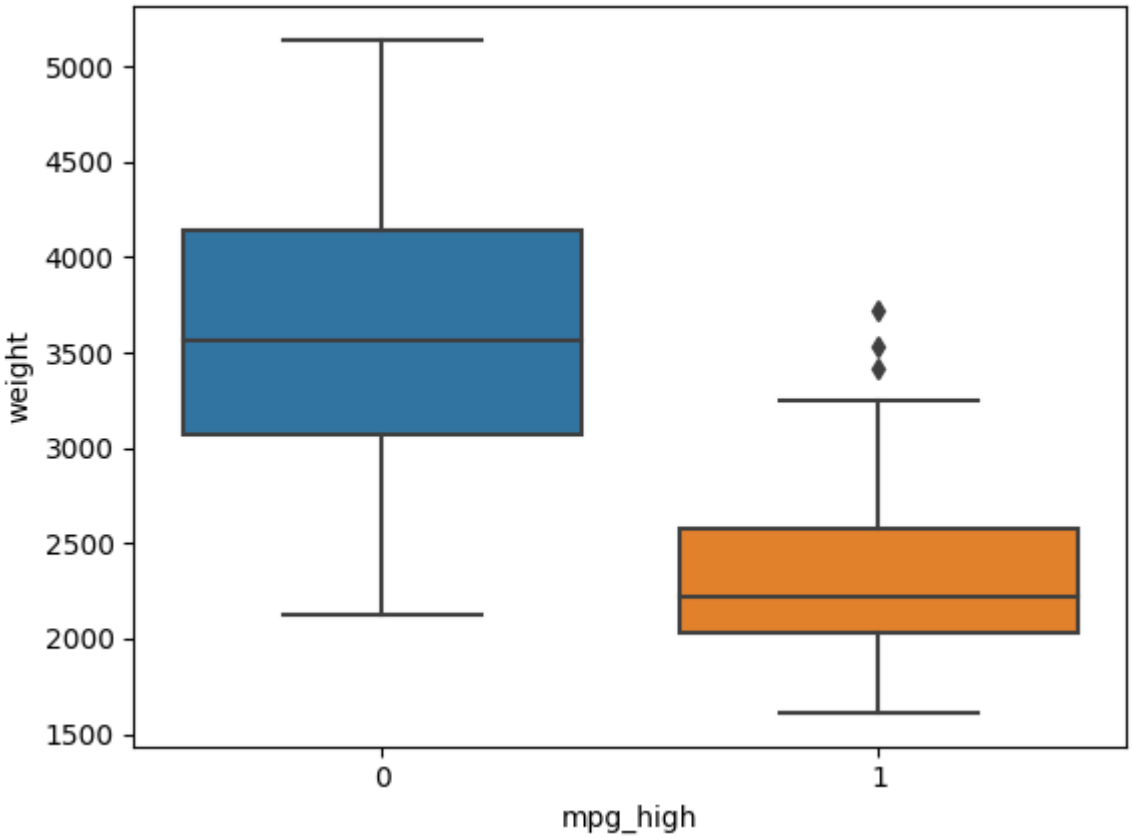
Out[ ]: <seaborn.axisgrid.FacetGrid at 0x2113c2405d0>



This graph shows that weight and horse power are good predictors of mpg_high. Vehicles that have a high mpg tend to be lighter and have less horsepower.

In [ ]:
```
sb.boxplot(data=df, x="mpg_high", y="weight")
```

Out[ ]:
```
<Axes: xlabel='mpg_high', ylabel='weight'>
```



This graph further shows the correlation between a low weight and a high mpg.

7. Train/Test Split

In [ ]:
```
import sklearn.model_selection as skms
train,test,train_label,test_label = skms.train_test_split(df.drop(['mpg_high'], axi
print("Train Shape {}".format(train.shape))
print("Test Shape {}".format(test.shape))
```

```
Train Shape (311, 8)
Test Shape (78, 8)
```

8. Logistic Regression

In [ ]:
```
import sklearn.linear_model as sklm
import sklearn.metrics as skm
logreg = sklm.LogisticRegression(max_iter=200, solver='lbfgs', random_state=1234)
logreg.fit(train, train_label)
logPred = logreg.predict(test)
logAcc = logreg.score(test, test_label)
print(skm.classification_report(test_label,logPred))
print(f"Logistic Regression Accuracy: {logAcc}")
```

```
              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

Logistic Regression Accuracy: 0.8717948717948718
```

9. Decision Tree

In [ ]:
```
import sklearn.tree as skt
tree = skt.DecisionTreeClassifier(random_state=1234)
tree.fit(train, train_label)
treePred = tree.predict(test)
treeAcc = skm.accuracy_score(test_label, treePred)
print(skm.classification_report(test_label, treePred))
print(f"Decision Tree Accuracy: {treeAcc}")
print(skt.export_text(tree, feature_names=['cylinders', 'displacement', 'horsepower
```

```
              precision    recall  f1-score   support

           0       0.90      0.90      0.90        50
           1       0.82      0.82      0.82        28

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

Decision Tree Accuracy: 0.8717948717948718
|--- cylinders <= 5.50
|   |--- horsepower <= 101.00
|   |   |---  year <= 75.50
|   |   |   |--- displacement <= 119.50
|   |   |   |   |--- cylinders <= 3.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- cylinders >  3.50
|   |   |   |   |   |--- weight <= 2683.00
|   |   |   |   |   |   |--- weight <= 2377.00
|   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |--- weight >  2377.00
|   |   |   |   |   |   |   |--- weight <= 2385.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |   |--- weight >  2385.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- weight >  2683.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |--- displacement >  119.50
|   |   |   |   |---  acceleration <= 17.75
|   |   |   |   |   |--- horsepower <= 81.50
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- horsepower >  81.50
|   |   |   |   |   |   |--- horsepower <= 89.00
|   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |   |--- horsepower >  89.00
|   |   |   |   |   |   |   |---  year <= 73.00
|   |   |   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |   |   |---  year >  73.00
|   |   |   |   |   |   |   |   |--- class: 0
|   |   |   |   |---  acceleration >  17.75
|   |   |   |   |   |--- class: 0
|   |   |---  year >  75.50
|   |   |   |--- weight <= 3250.00
|   |   |   |   |--- weight <= 2880.00
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- weight >  2880.00
|   |   |   |   |   |--- weight <= 2920.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |   |--- weight >  2920.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |--- weight >  3250.00
|   |   |   |   |--- displacement <= 151.50
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- displacement >  151.50
|   |   |   |   |   |--- class: 1
|   |--- horsepower >  101.00
```

```
|   |   |---  acceleration <= 14.45
|   |   |   |---  year <= 76.00
|   |   |   |   |--- class: 1
|   |   |   |---  year >  76.00
|   |   |   |   |--- origin <= 1.50
|   |   |   |   |   |--- class: 1
|   |   |   |   |--- origin >  1.50
|   |   |   |   |   |--- class: 0
|   |   |---  acceleration >  14.45
|   |   |   |--- class: 0
|--- cylinders >  5.50
|   |---  year <= 79.50
|   |   |---  acceleration <= 21.60
|   |   |   |--- weight <= 2737.00
|   |   |   |   |--- weight <= 2674.00
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- weight >  2674.00
|   |   |   |   |   |--- class: 1
|   |   |   |--- weight >  2737.00
|   |   |   |   |--- horsepower <= 83.00
|   |   |   |   |   |--- weight <= 3085.00
|   |   |   |   |   |   |--- class: 1
|   |   |   |   |   |--- weight >  3085.00
|   |   |   |   |   |   |--- class: 0
|   |   |   |   |--- horsepower >  83.00
|   |   |   |   |   |--- class: 0
|   |   |---  acceleration >  21.60
|   |   |   |--- class: 1
|   |---  year >  79.50
|   |   |--- origin <= 1.50
|   |   |   |--- displacement <= 247.00
|   |   |   |   |--- class: 0
|   |   |   |--- displacement >  247.00
|   |   |   |   |--- class: 1
|   |   |--- origin >  1.50
|   |   |   |--- class: 1
```

10. Neural Network

In [ ]:
```
import sklearn.neural_network as sknn
nnModel1 = sknn.MLPClassifier(hidden_layer_sizes=(5,7), max_iter=1000, solver='adam
nnModel2 = sknn.MLPClassifier(hidden_layer_sizes=(4,6), max_iter=1000, solver='lbfg
nnModel1.fit(train,train_label)
nnModel2.fit(train,train_label)
nnPred1 = nnModel1.predict(test)
nnPred2 = nnModel2.predict(test)
nnAcc1 = skm.accuracy_score(test_label, nnPred1)
nnAcc2 = skm.accuracy_score(test_label, nnPred2)
print(skm.classification_report(test_label, nnPred1))
print(f"Neural Network 1 Accuracy: {nnAcc1}")
print(skm.classification_report(test_label, nnPred2))
print(f"Neural Network 2 Accuracy: {nnAcc2}")
```

```
              precision    recall  f1-score   support

           0       1.00      0.82      0.90        50
           1       0.76      1.00      0.86        28

    accuracy                           0.88        78
   macro avg       0.88      0.91      0.88        78
weighted avg       0.91      0.88      0.89        78

Neural Network 1 Accuracy: 0.8846153846153846
              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

Neural Network 2 Accuracy: 0.8717948717948718
```

These were the best 2 models I was able to make completly without over fitting the data. Interestingly both models perfored very similarly and I think this is due to the size of the data. There are only at most 8 predictors and only so much information that can be fulled from them.

11. Analysis

All algorithms performed very similary. The logistic regression, decision tree and one of the neural networks all got the same over all accuracy and they only differ in their other metrics. The other neural network technically has a higher accuracy but it seems to have only gotten 1 more correct prediction than the other models.

Logistic Regresion P R 0 0.98 0.82
1 0.75 0.96 Accuracy: 87%

Decision Tree P R 0 0.90 0.90 1 0.82 0.82 Accuracy: 87%

Nerual Network 1 P R 0 1.00 0.82 1 0.76 1.00 Accuracy: 88%

Nerual Network 2 P R 0 0.98 0.82 1 0.75 0.96 Accuracy: 87%

The first interesting point is that the logistic regression model and the 2nd neural network performed almost identically. They are both good at correctly identifing the positive class but seem to be more biased to it. The decision tree is interesting because it scored the same for the positive and negative classes, indicating it is good at identifing either with similar accuracy. The 1st neural network have perfect percision for the negative class, meaning that for all its predictions it made in the negative class were correct and it didnt misidentify them has positive, but it didnt predict all of them. Conversly, the positive class had perfect recall, meaning it correclty predicted all of the positive class, but also incorrectly guess the negative

class as positive. This shows that the model seems to be bias to the positive class and is much more likely to guess it than the negative class.

Personally I am more comfortable with Python as even though I dont have much experience using it, it is much much closer to languages I do have experience in than R is. But, once i learned a bit more of how R works it was not bad. I still think I would perfer python though.