In [287]:

```
import pandas as pd
```

Dylan Kapustka

Homework 7

11/06/22

ML with Sklearn

Data Exploration

In [288]:

```
data = pd.read_csv('../Auto.csv')
data.head()
```

Out[288]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	8	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu
1	15.0	8	350.0	165	3693	11.5	70.0	1	buick skylark 320
2	18.0	8	318.0	150	3436	11.0	70.0	1	plymouth satellite
3	16.0	8	304.0	150	3433	12.0	70.0	1	amc rebel sst
4	17.0	8	302.0	140	3449	NaN	70.0	1	ford torino

In [289]:

```
data.shape
```

Out[289]:

```
(392, 9)
```

In [290]:

```
data.mpg.describe()
# Average mpg: 23.445919
# Range: 9.0-46.6
```

Out[290]:

```
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
```

In [291]:

```
data.weight.describe()
# Average weight: 2977.584184
# Range is from 1613.0-5140.0
```

Out[291]:

```
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
```

In [292]:

```
data.year.describe()
# Average year: 76.010256
# Range is from 70.0-82.0
```

Out[292]:

```
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
```

In [293]:

```
data.dtypes
```

Out[293]:

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

In [294]:

```
data.cylinders = data.cylinders.astype('category')
```

In [295]:

```
data.origin = data.origin.astype('category')
```

In [296]:

```
data.dtypes
```

Out[296]:

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

In [297]:

```
data.dropna(how='any',inplace=True)
```

In [298]:

```
data.shape
```

Out[298]:

```
(389, 9)
```

In [299]:

```
import numpy as np
```

In [300]:

```
data['mpg_high'] = np.where(data.mpg > 23,1,0)
```

In [301]:

```
data.drop('mpg',inplace=True,axis=1)
```

In [302]:

```
data.drop('name',inplace=True,axis=1)
```

In [303]:

```
data.head()
```

Out[303]:

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

Graphs

In [304]:

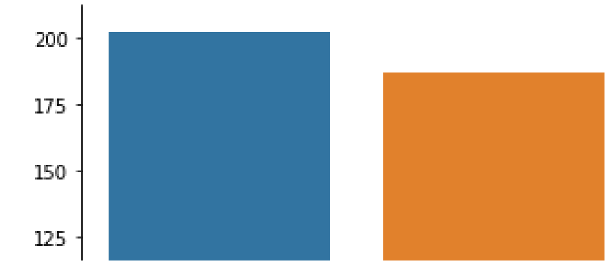
```
import seaborn as sb
```

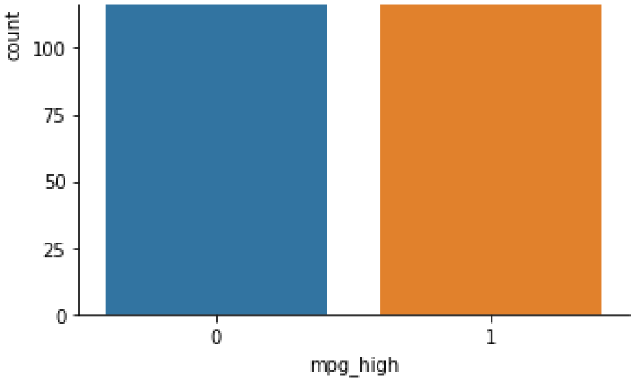
In [305]:

```
sb.catplot(x="mpg_high",kind='count',data=data)
# About half the cars have high mpg, while the other half has low mpg. Low is slightly higher.
```

Out[305]:

```
<seaborn.axisgrid.FacetGrid at 0x7f5c6830c310>
```



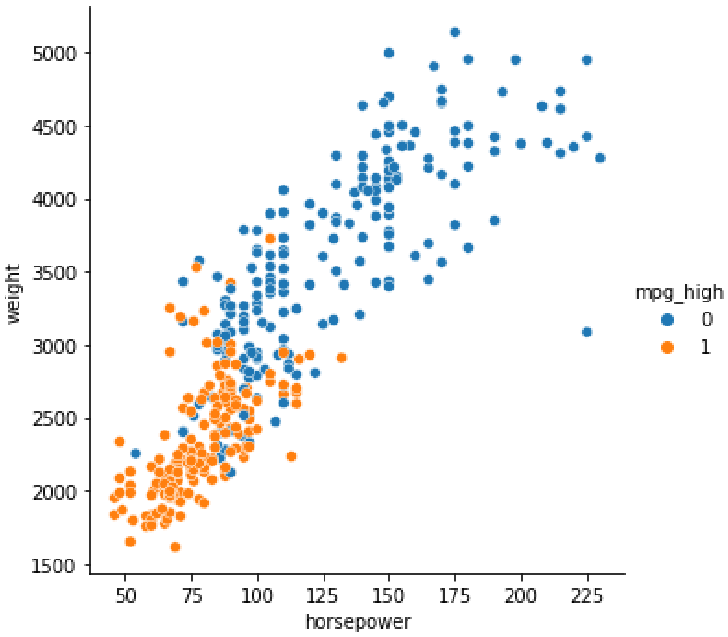


In [306]:

```
sb.relplot(x='horsepower',y='weight',data=data,hue=data.mpg_high)
# As the weight of the car increases the horsepower increases
```

Out[306]:

```
<seaborn.axisgrid.FacetGrid at 0x7f5c683dbbd0>
```



In [306]:

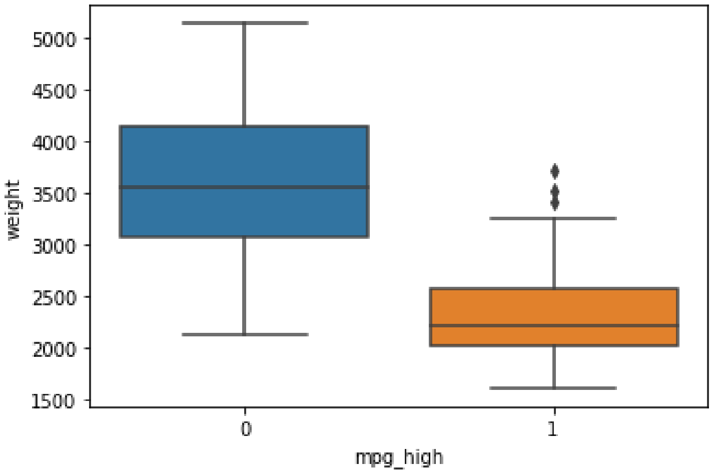
```

```

In [307]:

```
sb.boxplot(x='mpg_high',y='weight',data=data)
# The lighter the car is the the higher the mpg
```

Out[307]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c68299690>
```



In [308]:

```
from sklearn.model_selection import train_test_split
```

In [309]:

```
X = data.loc[:,data.columns != 'mpg_high']
```

In [310]:

```
y = data.mpg_high
```

In [311]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12
34)
```

In [312]:

```
X_train.shape
```

Out[312]:

```
(311, 7)
```

In [313]:

```
X_test.shape
```

Out[313]:

```
(78, 7)
```

In [314]:

```
from sklearn.linear_model import LogisticRegression
```

Logistic Regression

In [315]:

```
X_train
```

Out[315]:

	cylinders	displacement	horsepower	weight	acceleration	year	origin
184	4	101.0	83	2202	15.3	76.0	2
355	6	145.0	76	3160	19.6	81.0	2
57	4	97.5	80	2126	17.0	72.0	1
170	4	90.0	71	2223	16.5	75.0	2
210	8	350.0	180	4380	12.1	76.0	1
...	...	...	...	...	...	...	...
207	4	120.0	88	3270	21.9	76.0	2
56	4	113.0	95	2278	15.5	72.0	3
297	4	141.0	71	3190	24.8	79.0	2
214	4	98.0	68	2045	18.5	77.0	3
306	4	151.0	90	2556	13.2	79.0	1

311 rows × 7 columns

In [316]:

```
model = LogisticRegression(solver='lbfgs', max_iter=1000)
```

In [317]:

```
model.fit(X_train, y_train)
```

Out[317]:

```
LogisticRegression(max_iter=1000)
```

In [318]:

```
model.score(X_train, y_train)
```

Out[318]:

```
0.9003215434083601
```

In [319]:

```
pred = model.predict(X_test)
```

In [320]:

```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred))
```

```
              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

```

In [321]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12
34)
```

Decision trees

In [322]:

```
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
```

Out[322]:

```
DecisionTreeClassifier()
```

In [323]:

```
pred = model.predict(X_test)
```

In [324]:

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

```
              precision    recall  f1-score   support

           0       0.94      0.90      0.92        50
           1       0.83      0.89      0.86        28

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

In [325]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=12
34)
```

In [326]:

```
from sklearn import preprocessing

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

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

Neural Network 1

In [327]:

```
from sklearn.neural_network import MLPClassifier

model = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(5, 2), max_iter=500, random_st
ate=1234)
model.fit(X_train_scaled, y_train)
```

Out[327]:

```
MLPClassifier(hidden_layer_sizes=(5, 2), max_iter=500, random_state=1234,
              solver='lbfgs')
```

In [328]:

```
pred = model.predict(X_test_scaled)
```

In [329]:

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

```
              precision    recall  f1-score   support

           0       0.91      0.86      0.89        50
           1       0.77      0.86      0.81        28

    accuracy                           0.86        78
   macro avg       0.84      0.86      0.85        78
weighted avg       0.86      0.86      0.86        78
```

Neural Network 2

In [333]:

```
model = MLPClassifier(solver='adam',hidden_layer_sizes=(5, 2), max_iter=2000, random_sta
te=1234)
model.fit(X_train_scaled, y_train)
```

Out[333]:

```
MLPClassifier(hidden_layer_sizes=(5, 2), max_iter=2000, random_state=1234)
```

In [334]:

```
pred = model.predict(X_test_scaled)
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       1.00      0.82      0.90        50
           1       0.76      1.00      0.86        28
           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
```

Model Analysis

The model that performed **best** overall was the first NN Model. **Second place** was Logistic Regression, **third** was Decision Trees, and **fourth** was NN2.

For starters, they all did extremly well. This is due to the fact that the data exhibited a fantastic linear corellation between all the various aspects of it.

Neural Networks performed best and Logistic regression was a close second. This is due to the fact that we can think of LR and a one layer NN. Thus, a multilayered LR can be much more sophistacted, precise, and accurate.

Python vs R

Python has definitely surprised me as far as the sophistication of its Machine Learning libraries. There is such immense support for every little thing, it was such a smooth transition from R, which is a language specifically made for data science.

As a general purpose language, A++++++ for Python in regards to ML

```
In [ ]:
```