In [160]:

```
import pandas as pd
```

Abed Ahmed
Homework 67
November 6 2022
ML with Sklearn

Data Exploration

In [232]:

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

Out[232]:

	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 [233]:

```
df.size
```

Out[233]:

```
3528
```

In [234]:

```
df.shape
```

Out[234]:

```
(392, 9)
```

In [235]:

```
df.mpg.describe()
# The average mpg is 23.4
# The  range is from 9-46
```

Out[235]:

```
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 [236]:

```
df.weight.describe()
```

```
# The average weight 2977.58
# The  range is from 1613-5140
```

Out[236]:

```
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 [237]:

```
df.year.describe()
# The average year 76
# The  range is from 70-82
```

Out[237]:

```
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 [238]:

```
df.dtypes
```

Out[238]:

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

In [239]:

```
df.cylinders = df.cylinders.astype('category')
# Doesn't seem to be working with cat.codes
```

In [240]:

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

In [241]:

```
df.dtypes
```

Out[241]:

```
mpg              float64
cylinders       category
displacement     float64
horsepower         int64
weight             int64
acceleration     float64
year             float64
origin          category
```

```
name                object
dtype: object
```

In [242]:

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

In [243]:

```
df.size
```

Out[243]:

```
3501
```

In [244]:

```
df.shape
```

Out[244]:

```
(389, 9)
```

In [245]:

```
import numpy as np
```

In [246]:

```
df['mpg_high'] = np.where(df.mpg > 23.445918,1,0)
```

In [247]:

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

In [248]:

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

In [249]:

```
df.head()
```

Out[249]:

	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

Plotting

In [250]:

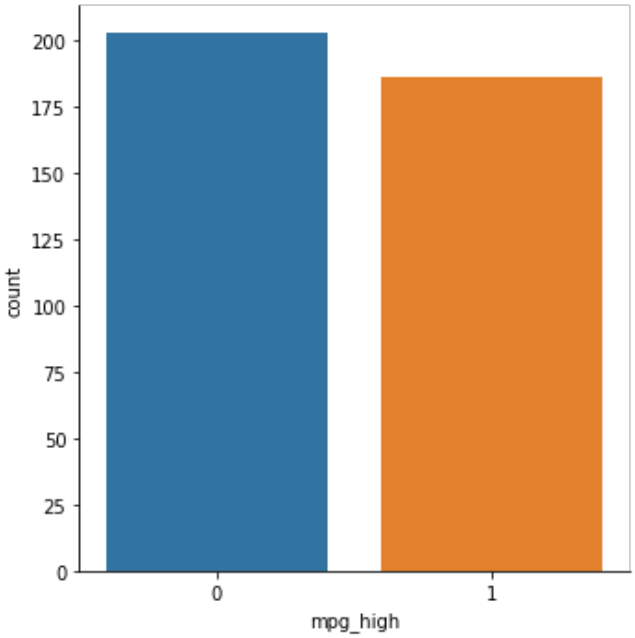
```
import seaborn as sb
```

In [251]:

```
sb.catplot(x="mpg_high",kind='count',data=df)
# Seems as though the cars are split about even in high and low mpg
```

Out[251]:

```
<seaborn.axisgrid.FacetGrid at 0x7f5c6898c210>
```

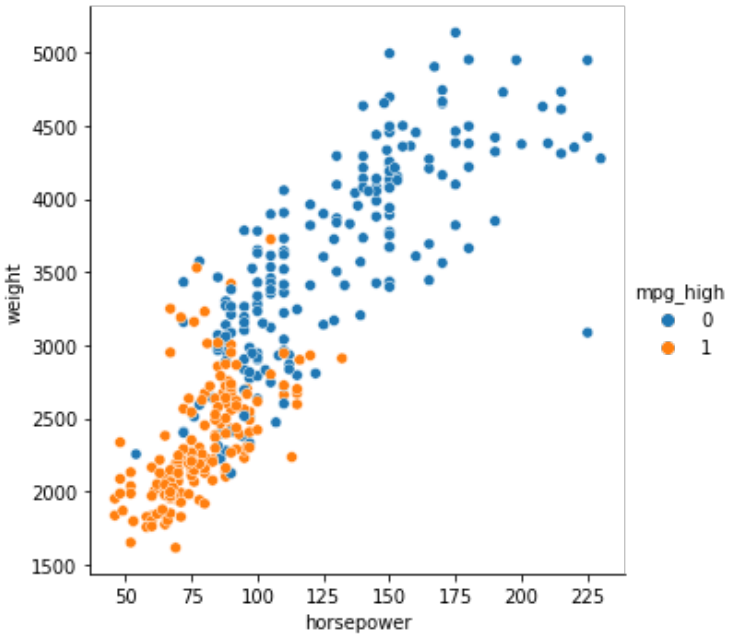


In [252]:

```
sb.relplot(x='horsepower',y='weight',data=df,hue=df.mpg_high)
# there is a direct linear correlation where as horsepower increase, weight also increase
s. This means that heavier cars, obviously need more horsepower
```

Out[252]:

```
<seaborn.axisgrid.FacetGrid at 0x7f5c689e1e90>
```

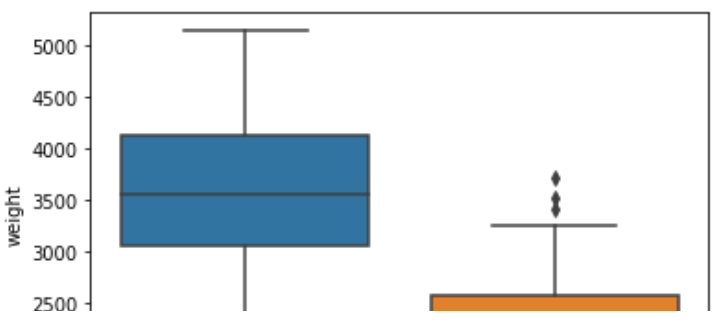


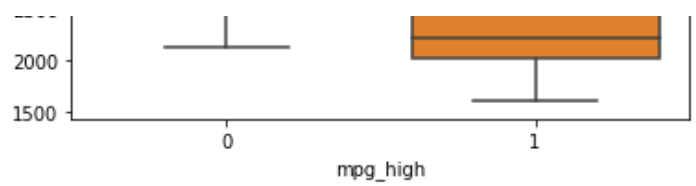
In [253]:

```
sb.boxplot(x='mpg_high',y='weight',data=df)
# This plot shows that heavier cars tend to have low mpg
```

Out[253]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f5c683e7110>
```





In [254]:

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

In [255]:

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

In [256]:

```
y = df.mpg_high
```

In [257]:

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

In [258]:

```
X_train.shape
```

Out[258]:

```
(311, 7)
```

In [259]:

```
X_test.shape
```

Out[259]:

```
(78, 7)
```

In [260]:

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

Logistic Regression

In [261]:

```
X_train
```

Out[261]:

	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

	cylinders	displacement	horsepower	weight	acceleration	year	origin
306	4	151.0	90	2556	13.2	79.0	1

311 rows × 7 columns

In [262]:

```
clf = LogisticRegression(solver='lbfgs', max_iter=900)
```

In [263]:

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

Out[263]:

```
LogisticRegression(max_iter=900)
```

In [264]:

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

Out[264]:

```
0.9067524115755627
```

In [265]:

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

Logistic regression results

In [266]:

```
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 [268]:

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

Decision trees

In [269]:

```
from sklearn.tree import DecisionTreeClassifier

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

Out[269]:

```
DecisionTreeClassifier()
```

In [270]:

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

Decision Tree Results

In [271]:

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

```
              precision    recall  f1-score   support

           0       0.93      0.82      0.87        50
           1       0.74      0.89      0.81        28

    accuracy                           0.85        78
   macro avg       0.83      0.86      0.84        78
weighted avg       0.86      0.85      0.85        78
```

In [273]:

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

In [274]:

```
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 Model 1link text

In [275]:

```
from sklearn.neural_network import MLPClassifier

clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(5, 2), max_iter=500, random_stat
e=1234)
clf.fit(X_train_scaled, y_train)
```

Out[275]:

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

In [276]:

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

Neural Network Model 1 - Results

In [277]:

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

```
              precision    recall  f1-score   support

           0       0.94      0.88      0.91        50
           1       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 Model 2

In [279]:

```
clf = MLPClassifier(solver='sgd',hidden_layer_sizes=(5, 2), max_iter=900, random_state=1
234)
clf.fit(X_train_scaled, y_train)
```

Out[279]:

```
MLPClassifier(hidden_layer_sizes=(5, 2), max_iter=900, random_state=1234,
              solver='sgd')
```

Neural Network Model 2 - Results

In [280]:

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

```
              precision    recall  f1-score   support

           0       0.93      0.82      0.87        50
           1       0.74      0.89      0.81        28

    accuracy                           0.85        78
   macro avg       0.83      0.86      0.84        78
weighted avg       0.86      0.85      0.85        78
```

Analysis

It seems that the best and most balanced model across all verticals (precision, accuracy, adn recall was the Neural Network Model 1. Next was Good Old Logistic regression.

There was a very strong linear Trend for in this dataset. In the real world, many of the columns are direct causations of the decisions that result in high or low mpg. This explains why Logistic regression did so well (and why all the models did well generally).

My guess as to why Neural Network did best based on some research is that because Logistic regression is actually a subset of NNs, and in theory NN is much more thorough than Logistic regression provided it is trained well. Therefore, theoretically, a neural network is always better than logistic regression, or more precisely, a neural network can do no worse than logistic regression.

Python vs R

Honestly, I did not expect python to be this great for machine learning. I was under the impression that R is the go to ML language,but python is just as good ands still maintains the "programmer" aspect to it. Both are pretty much the same from a practical standpoint for me, but I will say I prefer python.

In [281]:

```
```