```
                different ways:

                > jupyter nbconvert notebook*.ipynb
                > jupyter nbconvert notebook1.ipynb notebook2.ipynb

                or you can specify the notebooks list in a config file, containin
g::

                    c.NbConvertApp.notebooks = ["my_notebook.ipynb"]

                > jupyter nbconvert --config mycfg.py

To see all available configurables, use `--help-all`.
```

Author:Atmin Sheth

Data: Auto.CSV

Target:mpg_high

the target derived from knowing of the average is less or high then mpg

In [ ]:
```
from google.colab import drive
drive.mount('/drive')
```

```
Mounted at /drive
```

In [ ]:
```
import pandas as pd
df = pd.read_csv('/drive/My Drive/UTD/ML portfolio/Assignment 7/Auto.csv')
df.head()
```

Out[ ]:

	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 [ ]:
```
print(df.shape)
```

```
(392, 9)
```

In [ ]:
```
print("descripion of mpg, weight, and year")
print(df[['mpg',"weight","year"]].describe(include="all"))
# averag: mpg- 23.45 weight - 2977.58 year - 76.01
#rang: mpg-[9,46]  weight-[1613,5140] year- [70,82]
```

```
descripion of mpg, weight, and year
              mpg       weight        year
count  392.000000   392.000000  390.000000
mean    23.445918  2977.584184   76.010256
std      7.805007   849.402560    3.668093
min      9.000000  1613.000000   70.000000
25%     17.000000  2225.250000   73.000000
50%     22.750000  2803.500000   76.000000
75%     29.000000  3614.750000   79.000000
max     46.600000  5140.000000   82.000000
```

In [ ]:
```
for col in df.columns:
  print(col,type(df[col][0]))
```

```
mpg <class 'numpy.float64'>
cylinders <class 'numpy.int64'>
displacement <class 'numpy.float64'>
horsepower <class 'numpy.int64'>
weight <class 'numpy.int64'>
acceleration <class 'numpy.float64'>
year <class 'numpy.float64'>
origin <class 'numpy.int64'>
name <class 'str'>
```

In [ ]:
```
import numpy as np
```

In [ ]:
```
df["cylinders"] = df['cylinders'].astype('category').cat.codes
print(np.dtype(df.cylinders))
df.origin = df.origin.astype('category')
```

```
int8
```

In [ ]:
```
#fiindign the missinng values
df.isnull().sum()
```

Out[ ]:
```
mpg             0
cylinders       0
displacement    0
horsepower      0
weight          0
acceleration    1
year            2
origin          0
name            0
dtype: int64
```

In [ ]:
```
df.drop(['year','acceleration'],axis=1,inplace=True)
print(df.shape)
```

```
(392, 7)
```

In [ ]:
```
mpg_avg = np.mean(df.mpg)
df = df.assign(mpg_high=lambda x:  df.mpg>mpg_avg  )
df["mpg_high"] = df['mpg_high'].astype(int)
df.head()
```

Out[ ]:

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

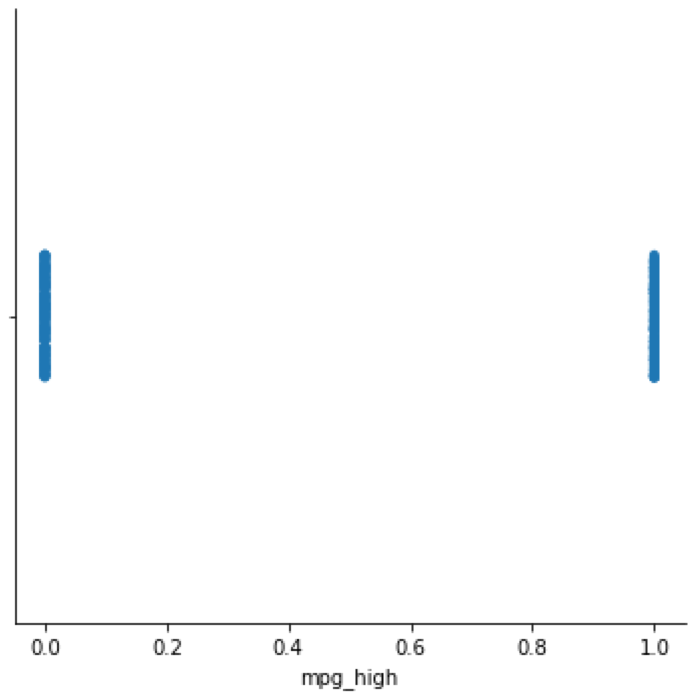
In [ ]:
```
df=df.drop("mpg", True)
df = df.drop("name",True)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: FutureWarnin
g: In a future version of pandas all arguments of DataFrame.drop except for t
he argument 'labels' will be keyword-only
  """Entry point for launching an IPython kernel.
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: FutureWarnin
g: In a future version of pandas all arguments of DataFrame.drop except for t
he argument 'labels' will be keyword-only
```

In [ ]:
```
import seaborn as sb
```

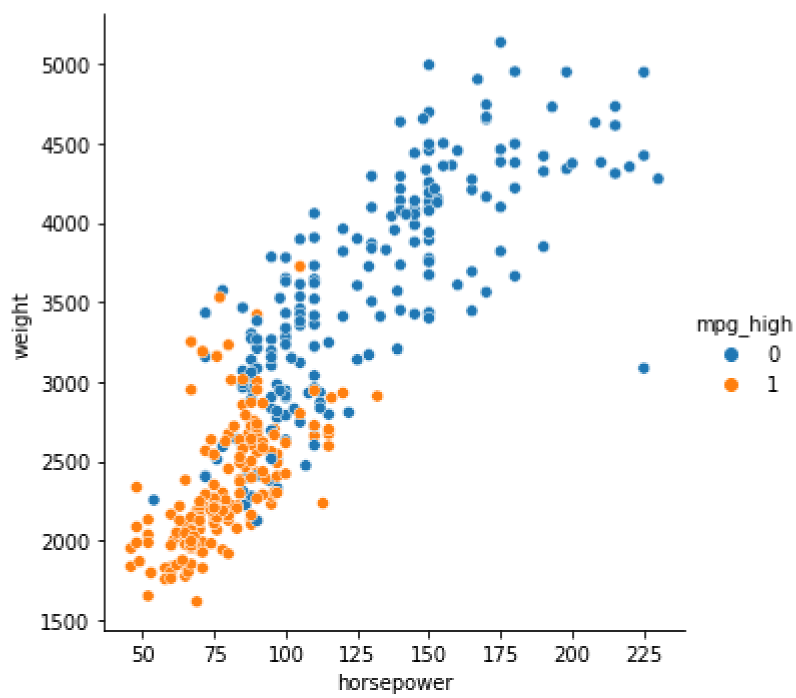
In [ ]:
```
sb.catplot(data=df,x="mpg_high")
#the raange of mpg_high
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7f1a8cb9cb50>



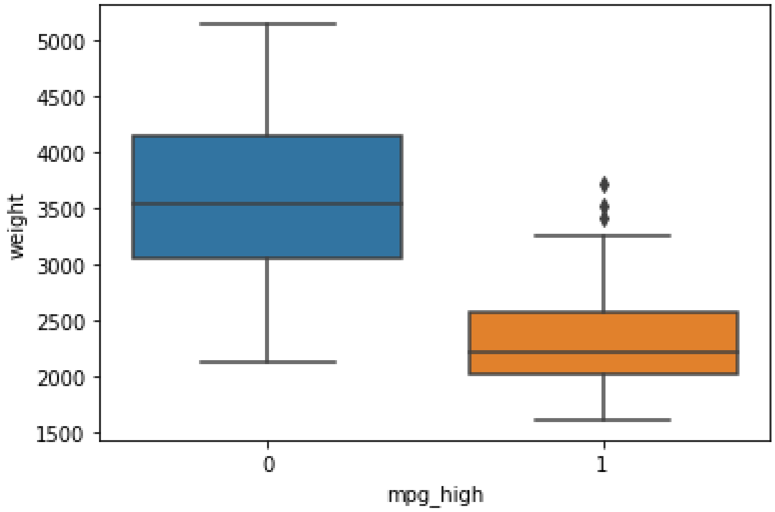
In [ ]:
```
sb.relplot(data=df, x='horsepower',y='weight',hue='mpg_high')
#the horsepower being more less then the average for mpg weight and being clus
ter at lighter weight
```

Out[ ]: <seaborn.axisgrid.FacetGrid at 0x7f1a8cafcb90>



In [ ]:
```
sb.boxplot(data=df,x='mpg_high', y='weight')
# the rang of weight in mpg being high(1) or below(0) average
```

Out[ ]: <matplotlib.axes._subplots.AxesSubplot at 0x7f1a894b4350>



In [ ]:
```
#split into train and test
from sklearn.model_selection import train_test_split
x=df. drop("mpg_high", axis=1)
y=df['mpg_high']
x_train,x_test, y_train,y_test = train_test_split(x,y,test_size=.2,random_stat
e=1234)
print("train_size",x_train.shape)
print("test_shape",x_test.shape)
```

```
train_size (313, 5)
test_shape (79, 5)
```

In [ ]:
```
#training and evaluating linear regression
from sklearn.linear_model import LogisticRegression
clf= LogisticRegression(max_iter=400)
clf.fit(x_train,y_train)
clf.score(x_train,y_train)
```

Out[ ]: 0.8945686900958466

In [ ]:
```
pred= clf.predict(x_test)
```

In [ ]:
```
# evaluate
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
score

print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred))
print('recall score: ', recall_score(y_test, pred))
print('f1 score: ', f1_score(y_test, pred))
```

```
accuracy score:  0.8860759493670886
precision score:  0.9
recall score:  0.8780487804878049
f1 score:  0.888888888888889
```

In [ ]:
```
# confusion matrix
from sklearn.metrics import confusion_matrix

confusion_matrix(y_test, pred)
```

Out[ ]:
```
array([[34,  4],
       [ 5, 36]])
```

In [ ]:
```
from sklearn.tree import DecisionTreeClassifier

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

Out[ ]:
```
DecisionTreeClassifier()
```

In [ ]:
```
#pred
pred = clf.predict(x_test)
```

In [ ]:
```
# evaluate
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_
score

print('accuracy score: ', accuracy_score(y_test, pred))
print('precision score: ', precision_score(y_test, pred))
print('recall score: ', recall_score(y_test, pred))
print('f1 score: ', f1_score(y_test, pred))
```

```
accuracy score:  0.8860759493670886
precision score:  0.9210526315789473
recall score:  0.8536585365853658
f1 score:  0.8860759493670887
```

In [ ]:
```
# Nueral Network on Classification
from sklearn import preprocessing

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

X_train_scaled = scaler.transform(x_train)
X_test_scaled = scaler.transform(x_test)
```

In [ ]:
```
# train
from sklearn.neural_network import MLPClassifier

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

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

In [ ]:
```
pred = clf.predict(X_test_scaled)
```

In [ ]:
```
# output results

print('accuracy = ', accuracy_score(y_test, pred))

confusion_matrix(y_test, pred)
```

```
accuracy =  0.8607594936708861
```

Out[ ]:
```
array([[32,  6],
       [ 5, 36]])
```

In [ ]:
```
from sklearn.metrics import classification_report
print(classification_report(y_test, pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.84      0.85        38
           1       0.86      0.88      0.87        41

    accuracy                           0.86        79
   macro avg       0.86      0.86      0.86        79
weighted avg       0.86      0.86      0.86        79
```

a. which algorithm performed better?

```
It seems Decision tree has perform the best.
```

b. compare accuracy, recall and precision metrics by class

```
The accuracy of DecisionTree and logistic regression is the same .886 and the Neral
network was .86. Nuero network had  recall of .85, the logistic had of .87 and DT h
ad of .85. The persicion of Neoro networek was of presision of .86, DT had .85 and
 Logistic had .9
```

c. give your analysis of why the better-performing algorithm might have outperformed the other

```
I think Decision Tree had out performed, others, There is limit at logistic of itte
ration, Nueronetwork worked well but had less accuracy
```

d. write a couple of sentences comparing your experiences using R versus sklearn. Feel free to express strong preferences.

I do think sklear is more helpful in studying and learning the model. R is seems to be more for analysis more then making a model to apply on a full flege AI projec. sklearn also give more details on teh data and model