ML with Sklearn

Step 1: Read the data

```
import pandas as pd
df = pd.read_csv('Auto.csv')
df.head(6)
```

	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
5	15.0	8	429.0	198	4341	10.0	NaN	1	ford galaxie 500

```
print('\nDimensions of data frame:', df.shape)
```

```
Dimensions of data frame: (392, 9)
```

Step 2: Describe the data

Explanation:

MPG is in the range [9, 46.6], with an average of 23.4.
Weight is in the range [1613, 5140], with an average of 2977.6.
Year is in the range [70, 82], with an average of 76.

```
print('\nDescribe mpg, weight, and year:\n', df.loc[:, ['mpg', 'weight', 'year']].describe())
```

```
Describe 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
```

Step 3: Explore data types

```
df.dtypes
```

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

```
df.cylinders = df.cylinders.astype('category').cat.codes
df.origin = df.origin.astype('category')
df.dtypes
```

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

Step 4: Deal with NAs

```
df = df.dropna()
print('\nDimensions of data frame:', df.shape)
```

```
Dimensions of data frame: (389, 9)
```

Step 5: Modify columns

```
average_mean = df.mpg.mean()
mpg_high = []
for mpg in df.mpg:
  if mpg > average_mean:
    mpg_high.append(1)
  else:
    mpg_high.append(0)

df['mpg_high'] = mpg_high
df.mpg_high = df.mpg_high.astype('category')
df = df.drop(columns=['mpg', 'name'])
df.head()
```

	cylinders	displacement	horsepower	weight	acceleration	year	origin	mpg_high
0	4	307.0	130	3504	12.0	70.0	1	0
1	4	350.0	165	3693	11.5	70.0	1	0
2	4	318.0	150	3436	11.0	70.0	1	0
3	4	304.0	150	3433	12.0	70.0	1	0
6	4	454.0	220	4354	9.0	70.0	1	0

Step 6: Data Exploration with graphs

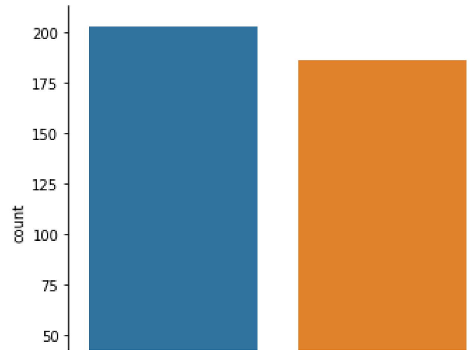
CatPlot:

Explanation:

The data is almost equally distributed betweeen the two categories, with around 180 of them having a high mpg and around 200 of them having low mpg.

```
import seaborn as sb
sb.catplot(x = "mpg_high", kind = "count", data = df)
```

<seaborn.axisgrid.FacetGrid at 0x7f4376e12e10>



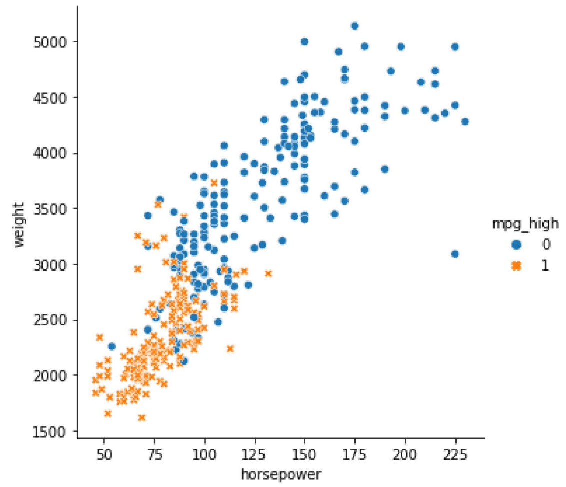
Relplot:

Explanation:

In general, less horsepower and less weight results in high mpg; similarly, more horsepower and more weight results in low mpg.

```
sb.relplot(x='horsepower', y='weight', data=df, hue=df.mpg_high, style=df.mpg_high)
```

<seaborn.axisgrid.FacetGrid at 0x7f4376f5bd50>



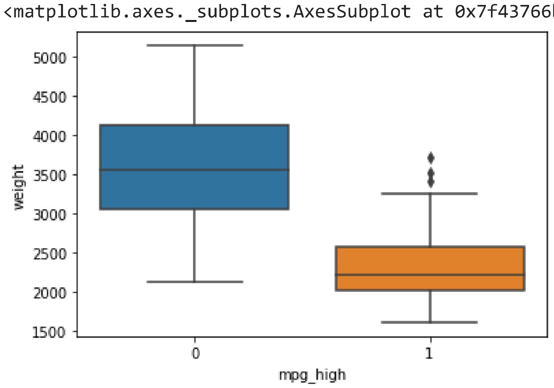
Boxplot:

Explanation:

More weight leads to less mpg, and less weight leads to high mpg.

```
sb.boxplot(x='mpg_high', y='weight', data=df)
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f43766bb810>



Step 7: Train/Test split

```
from sklearn.model_selection import train_test_split

X = df.loc[:, ['cylinders', 'displacement', 'horsepower', 'weight', 'acceleration', 'year', 'origin']]
y = df.mpg_high

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1234)

print('train size:', X_train.shape)
print('test size:', X_test.shape)
```

```
    train size: (311, 7)
    test size: (78, 7)
```

Step 8: Logistic Regression

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


clf = LogisticRegression()
clf.fit(X_train, y_train)
clf.score(X_train, y_train)

pred = clf.predict(X_test)

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

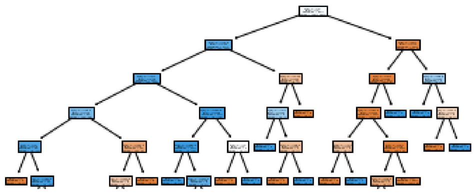
```
                  precision    recall  f1-score   support

               0       0.98      0.80      0.88        50
               1       0.73      0.96      0.83        28

        accuracy                           0.86        78
       macro avg       0.85      0.88      0.85        78
    weighted avg       0.89      0.86      0.86        78
```

Step 9: Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report

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

pred = clf.predict(X_test)

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

```
                  precision    recall  f1-score   support

               0       0.98      0.88      0.93        50
               1       0.82      0.96      0.89        28

        accuracy                           0.91        78
       macro avg       0.90      0.92      0.91        78
    weighted avg       0.92      0.91      0.91        78
```

```
from sklearn import tree
_ = tree.plot_tree(clf, filled = True)
```



Step 10: Neural Network

Comparing two models:

The first model has 1 hidden layer with 7 nodes, and the second model has 2 hidden layers with first layer having 4 nodes and second layer having 3 nodes.

The second model performed better than the first one because the mapping of the inputs to the output was not smooth enough to be covered with 1 hidden layer. This means that the data contained a more complex relationshiip, which required 2 hidden layers in order to be captured.

```
from sklearn import preprocessing
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import classification_report

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

clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(7,))
clf.fit(X_train, y_train)

pred = clf.predict(X_test)

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

```
              precision    recall  f1-score   support

           0       0.92      0.88      0.90        50
           1       0.80      0.86      0.83        28

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

```
clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(4,3))
clf.fit(X_train, y_train)

pred = clf.predict(X_test)

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

```
              precision    recall  f1-score   support

           0       0.98      0.90      0.94        50
           1       0.84      0.96      0.90        28

    accuracy                           0.92        78
   macro avg       0.91      0.93      0.92        78
weighted avg       0.93      0.92      0.92        78
```

Step 11: Analysis

The second neural network performed better than all the other models. The decision tree was a close second.

For accuracy, the second neural network model was 5%, 1%, and 6% more accurate than the first neural network model, decision tree, and logistic regressiom model, respectively.

For recall value of Class 0, the second neural network model was 2% higher than the first neural network model and the decision tree, but it was 10% higher than the logistic regression model.

For recall value of Class 1, the second neural network model had the same value as the logistic regression and decision tree models, but it was 10% higher than the first neural network model.

For precision value of Class 0, the second neural network model had the same value as the logistic regression and decision tree models, but it was 6% higher than the first neural network model.

For precision value of Class 1, the second neural network model was 4%, 2%, and 11% higher than the first neural network model, decision tree, and logistic regressiom model, respectively.

I believe that the second neural network model was able to outperform the other models because our data does not have linear relationship and it does not have smooth mapping from the its inputs to the outputs. The outperforming model had two hidden layers, which was able to capture the complex relationship between our data better than all the other models.

Now, if I compare running these algorithms in R versus sklearn, I would say that both platforms provided libraries that were easy to learn and easy to use. But if I have to choose one of them, I would choose R because it gave better statistical representaion of the training model, and it gave better metrics to evaluate the test predictions.