ML with sklearn

```
# Necessary Imports and Setup
import numpy as np
import pandas as pd
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn import preprocessing
max_iter = 10000
```

```
np.random.seed(1234)
from google.colab import files
uploaded = files.upload()
import io
```

Browse... Auto.csv

Auto.csv(text/csv) - 17859 bytes, last modified: n/a - 100% done

```
Saving Auto.csv to Auto (3).csv
```

1. Read the Auto Data

```
df = pd.read_csv(io.BytesIO(uploaded['Auto.csv']))
print('Rows and Columns:', df.shape)
print(df.head())
```

```
Rows and Columns: (392, 9)
    mpg  cylinders  displacement  horsepower  weight  acceleration  yea
0  18.0          8         307.0         130    3504          12.0  70.
1  15.0          8         350.0         165    3693          11.5  70.
2  18.0          8         318.0         150    3436          11.0  70.
3  16.0          8         304.0         150    3433          12.0  70.
4  17.0          8         302.0         140    3449           NaN  70.

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
```

✓ 0s completed at 4:32 PM

2. Data Exploration with Code

```
# MPG     Range: 37.6 Average: 23.45
print(df.mpg.describe())

# WEIGHT  Range: 3527 Average; 2977.58
print(df.weight.describe())

#YEAR     Range: 12   Average: 76.01
print(df.year.describe())
```

```
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
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
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
```

3. Explore Data Types

```
datatypes = df.dtypes
datatypes
```

```
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')
datatypes = df.dtypes
datatypes
```

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

4. Deal with NAs

```
df.isnull().sum()
```

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

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

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

5. Modify Columns

```
df['mpg_high'] = (df['mpg'] > 23.446).astype(int)
```

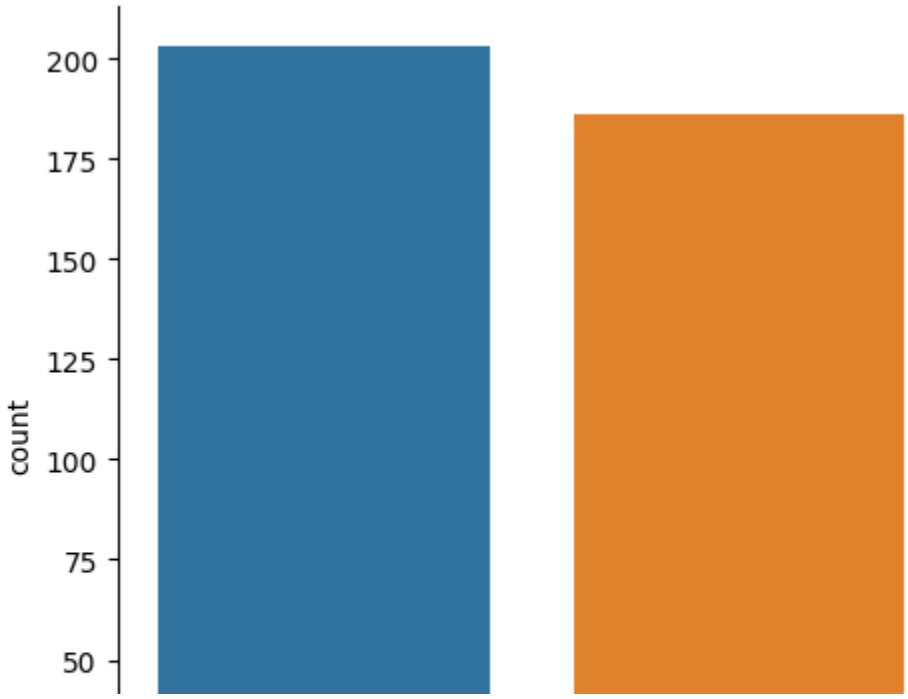
```
df = df.drop(columns=['mpg', 'name'])
print(df.head())
```

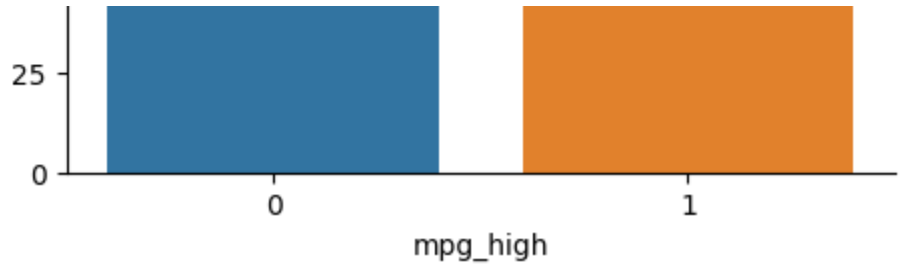
```
   cylinders  displacement  horsepower  weight  acceleration  year orig
0          4         307.0         130    3504          12.0  70.0
1          4         350.0         165    3693          11.5  70.0
2          4         318.0         150    3436          11.0  70.0
3          4         304.0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

   mpg_high
0         0
1         0
2         0
3         0
6         0
```

6. Data Exploration with Graphs

```
# There are less vehicles with high MPG
sb.catplot(x='mpg_high', kind='count', data=df)
```

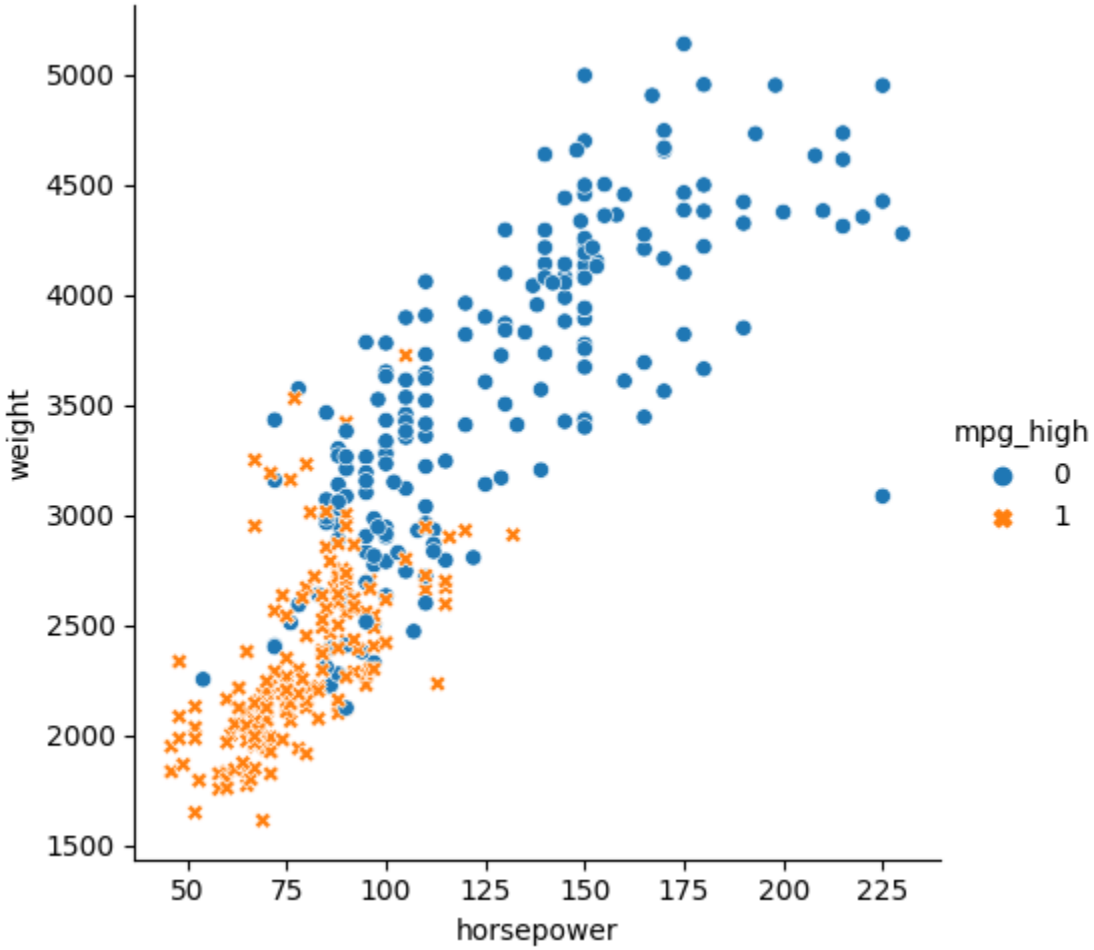
```
<seaborn.axisgrid.FacetGrid at 0×7fee18b36130>
```





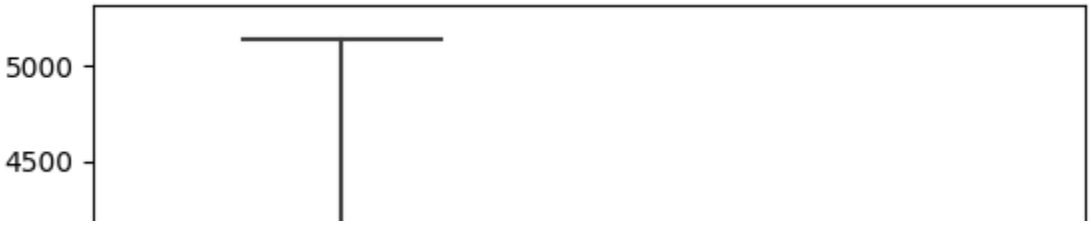

```
# Horsepower and Weight are inversely proportional to MPG
sb.relplot(x='horsepower', y='weight', data=df, hue=df.mpg_high, style=df.mp
```

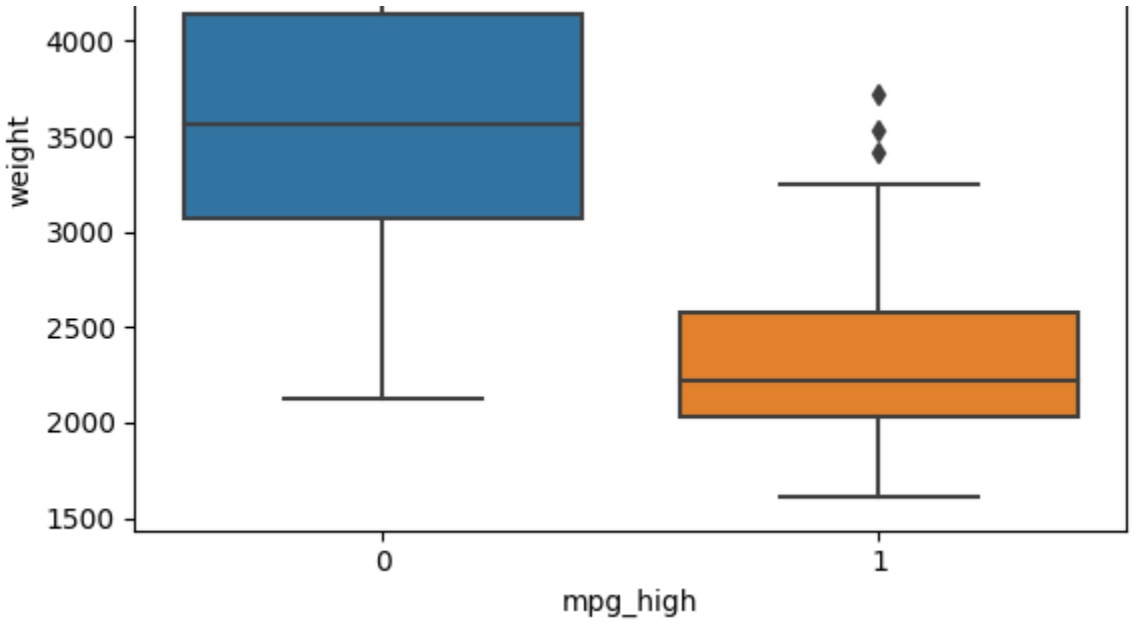
```
<seaborn.axisgrid.FacetGrid at 0×7fee18b10c40>
```




```
# Lower weight vehicles are almost always better for MPG
sb.boxplot(data=df, x='mpg_high', y='weight')
```

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





7. Train/Test

```
df_y = df.mpg_high
df_x = df.loc[:, df.columns≠'mpg_high']
X_train, X_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0.
print('train size:', X_train.shape)
print('test size:', X_test.shape)
scaler = preprocessing.StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

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

8. Logistic Regression

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

```
0.8971061093247589
```

```
pred = clf.predict(X_test)

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

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

9. Decision Tree

```
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
```

10. Neural Network

```
regr = MLPRegressor(hidden_layer_sizes=(6, 3), max_iter=500, random_state=1
regr.fit(X_train, y_train)
y_pred = regr.predict(X_test)

print('mse:', mean_squared_error(y_test, y_pred))
print('corr:', r2_score(y_test, y_pred))
```

```
    mse: 0.08188287460306254
    corr: 0.6441604220821195
```

```
clf = MLPClassifier(solver='lbfgs', hidden_layer_sizes=(5, 2), max_iter=100
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

print('mse:', mean_squared_error(y_test, y_pred))
print('corr:', r2_score(y_test, y_pred))
```

```
    mse: 0.14102564102564102
    corr: 0.3871428571428557
```

```
corr: 0.38714285714285
```

The Regression neural network is better. The scaled data is set up for a linear relationship, resulting in a more fitting model. This is supported by the lower MSE value and higher correlation metric.

11. Analysis

The decision tree classifier performed better. The precision was much higher than LogReg for class 1, and slightly lower for class 0. Recall is similarly comparable, weaker in decision tree class 1 but stronger in 0. Overall accuracy makes the better model clear. Decision trees increased the accuracy by 0.03. The scaled data could fit classification better. R, although powerful, is not my main choice for this kind of work. Sklearn allows me to carry over my python skills into data science. This makes a world of difference with understanding how to structure projects and code.