ML_with_sklearn

November 7, 2022

1 ML with sklearn

[183]:
```
import seaborn as sb
import pandas as pd
```

1. Read the Auto data

[184]:
```
# a. use pandas to read the data
df = pd.read_csv('Auto.csv')

# b. output the first few rows
print(df.head())

# c. output the dimensions of the data
print('\nDimensions of the data: ', df.shape)
```

```
    mpg  cylinders  displacement  horsepower  weight  acceleration  year  \
0  18.0          8         307.0         130    3504          12.0  70.0
1  15.0          8         350.0         165    3693          11.5  70.0
2  18.0          8         318.0         150    3436          11.0  70.0
3  16.0          8         304.0         150    3433          12.0  70.0
4  17.0          8         302.0         140    3449           NaN  70.0

   origin                       name
0       1  chevrolet chevelle malibu
1       1          buick skylark 320
2       1         plymouth satellite
3       1              amc rebel sst
4       1                ford torino

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

2. Data exploration with code

[185]:
```
# a. use describe() on the mpg, weight, and year columns
df[['mpg', 'weight', 'year']].describe()

# b. write comments indicating the range and average of each column
```

```python
# mpg range: 37.6 (9 to 46.6)
# mpg average: 23.446

# weight range: 3527 (1613 to 5140)
# weight average: 2977.584

# year range: 12 (70 to 82)
# year average: 76.010
```

[185]:

	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

3. Explore data types

[186]:

```python
# a. check the data types of all columns
print(df.dtypes)

# b. change the cylinders column to categorical (use cat.codes)
df['cylinders'] = df['cylinders'].astype('category').cat.codes

# c. change the origin column to categorical (don't use cat.codes)
df['origin'] = df['origin'].astype('category')

# d. verify the changes with the dtypes attribute
print(df.dtypes)
```

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

```
acceleration     float64
year             float64
origin          category
name              object
dtype: object
```

4. Deal with NAs

[187]:
```python
# a. delete rows with NAs
df = df.dropna()

# b. output the new dimensions
print('New dimensions of the data: ', df.shape)
```

```
New dimensions of the data:  (389, 9)
```

5. Modify columns

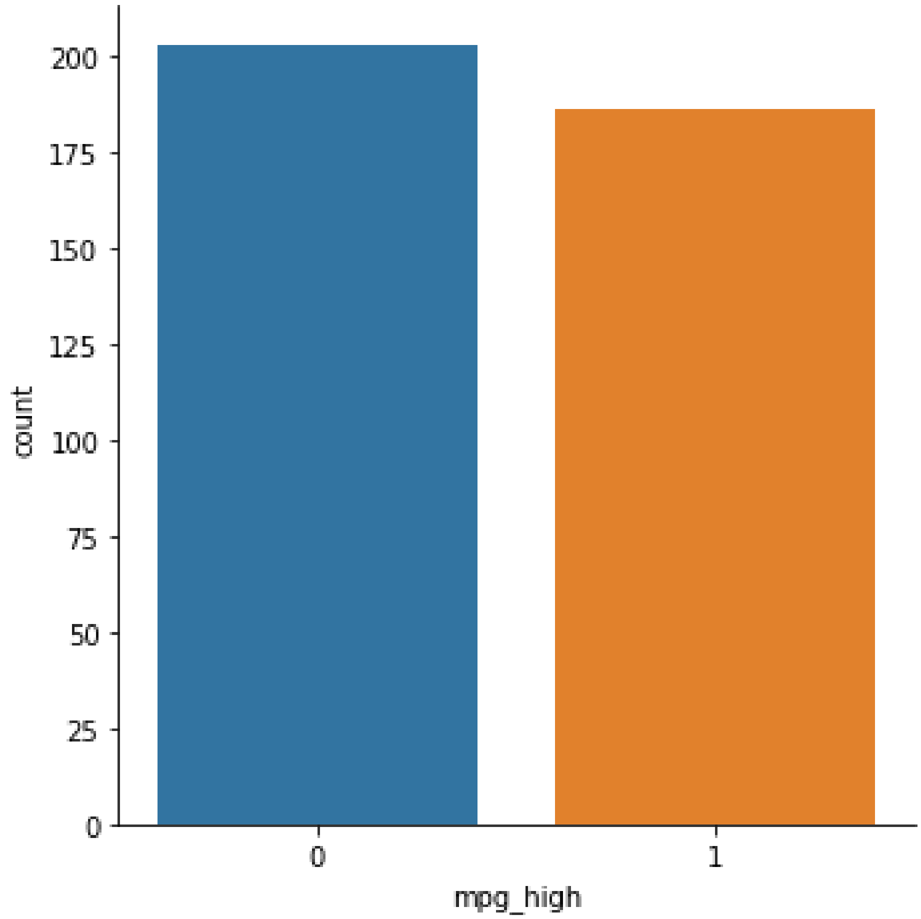
[188]:
```python
# a. make a new column, mpg_high, and make it categorical:
  # i. the column == 1 if mpg > average mpg, else == 0
mpgavg = df['mpg'].mean()
df['mpg_high'] = [0 if x < mpgavg else 1 for x in df['mpg']]
df.mpg_high = df.mpg_high.astype('category')

#b. delete the mpg and name columns (delete mpg so the algorithm doesn't just␣
↪learn to predict mpg_high from mpg)
df = df.drop(columns = ['mpg', 'name'])

# c. output the first few rows of the modified data frame
df.head()
```

[188]:
```
   cylinders  displacement  horsepower  weight  acceleration  year origin  \
0          4         307.0         130    3504          12.0  70.0      1
1          4         350.0         165    3693          11.5  70.0      1
2          4         318.0         150    3436          11.0  70.0      1
3          4         304.0         150    3433          12.0  70.0      1
6          4         454.0         220    4354           9.0  70.0      1

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

6. Data exploration with graphs

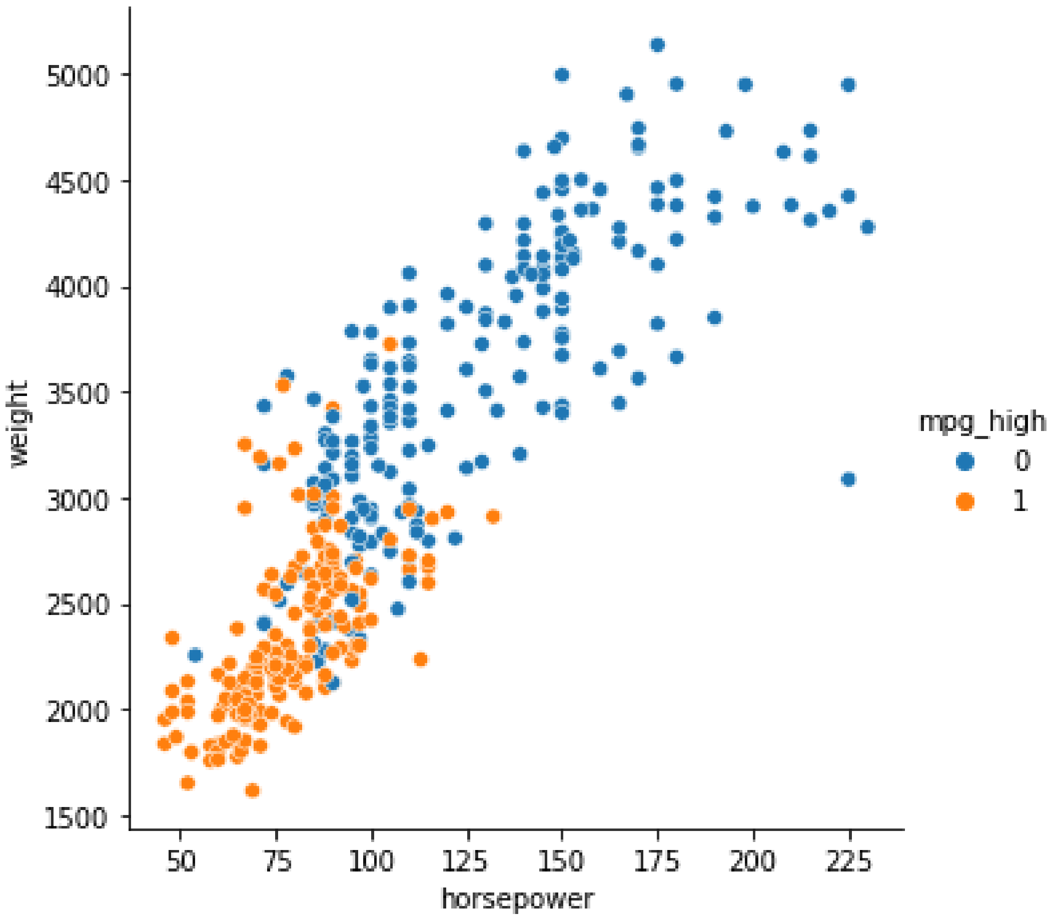
[189]:
```python
# a. seaborn catplot on the mpg_high column
sb.catplot(x = "mpg_high", kind = 'count', data = df)
```

[189]: <seaborn.axisgrid.FacetGrid at 0x7f638b6be710>



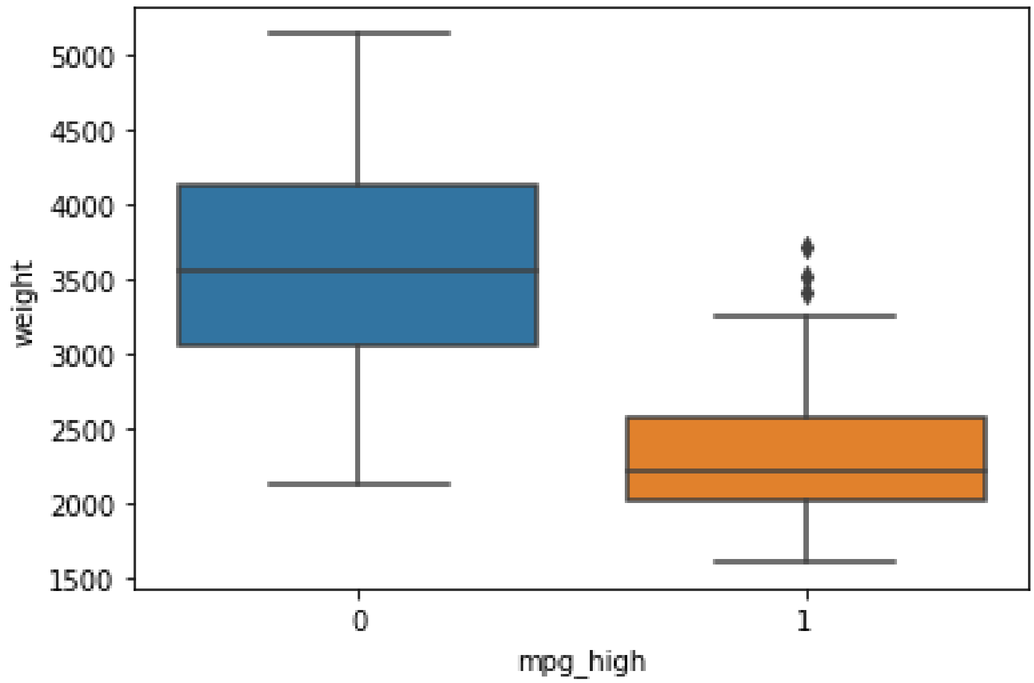
[190]:
```
# b. seaborn relplot with horsepower on the x axis, weight on the y axis,␣
↪setting hue or style to mpg_high
sb.relplot(x = 'horsepower', y = 'weight', data = df, hue = df.mpg_high)
```

[190]: <seaborn.axisgrid.FacetGrid at 0x7f638b8f7dd0>




```
[191]: # c. seaborn boxplot with mpg_high on the x axis and weight on the y axis
       sb.boxplot(x = 'mpg_high', y = 'weight', data = df)
```

```
[191]: <matplotlib.axes._subplots.AxesSubplot at 0x7f638bf09fd0>
```



[192]:
```
# d. for each graph, write a comment indicating one thing you learned about the
↪data from the graph
# catplot: there are slightly more mpg_lows but overall the distribution of
↪high and low is fairly even
# relplot: cars with high mpg are on the lighter and less horsepower ends of
↪the scale
# boxplot: there are some outliers for the cars with high mpg whereas there are
↪none for the low mpg
```

7. Train/test split

[193]:
```
# a. 80/20
# b. use seed 1234 so we all get the same results
# c. train /test X data frames consists of all remaining columns except mpg_high
# d. output the dimensions of train and test
from sklearn.model_selection import train_test_split

X = df.loc[:, df.columns != 'mpg_high']
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('Dimensions of train: ', X_train.shape)
print('Dimensions of test: ', X_test.shape)
```

```
Dimensions of train:  (311, 7)
Dimensions of test:  (78, 7)
```

8. Logistic Regression

[194]:
```python
# a. train a logistic regression model using solver lbfgs
# b. test and evaluate
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression(solver = 'lbfgs', max_iter = 270)
clf.fit(X_train, y_train)
clf.score(X_train, y_train)

pred = clf.predict(X_test)

# c. print metrics using the classification report
from sklearn.metrics import classification_report

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

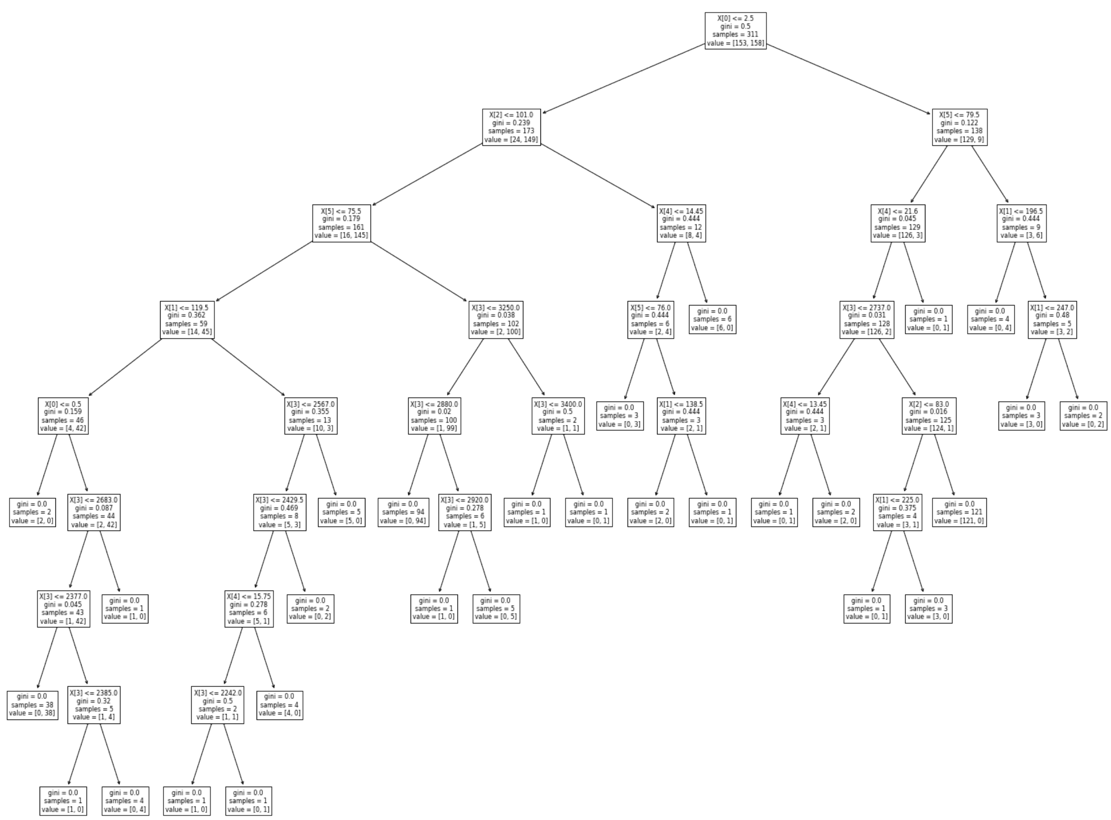
```
              precision    recall  f1-score   support

           0       1.00      0.84      0.91        50
           1       0.78      1.00      0.88        28

    accuracy                           0.90        78
   macro avg       0.89      0.92      0.89        78
weighted avg       0.92      0.90      0.90        78
```

9. Decision Tree

[195]:
```python
# a. train a decision tree
# b. test and evaluate
from sklearn.tree import DecisionTreeClassifier

clf1 = DecisionTreeClassifier()
clf1.fit(X_train, y_train)
clf1.score(X_train, y_train)

pred1 = clf1.predict(X_test)

# c. print the classification report metrics
print(classification_report(y_test, pred1))

# d. plot the tree (optional, see: https://scikit-learn.org/stable/modules/tree.
 ↪html)
from sklearn import tree
from matplotlib import pyplot as plt
```

```
fig = plt.figure(figsize = (25,20))
tree.plot_tree(clf1)
plt.show()
```

```
              precision    recall  f1-score   support

           0       0.98      0.86      0.91        50
           1       0.79      0.96      0.87        28

    accuracy                           0.90        78
   macro avg       0.89      0.91      0.89        78
weighted avg       0.91      0.90      0.90        78
```



10. Neural Network

[202]:

```
# a. train a neural network, choosing a network topology of your choice
# b. test and evaluate
from sklearn import preprocessing
```

```python
from sklearn.neural_network import MLPClassifier

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

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

clf2 = MLPClassifier(solver = 'lbfgs', hidden_layer_sizes = (5), max_iter =␣
 ↪1000, random_state = 1234)
clf2.fit(X_train_scaled, y_train)
clf2.score(X_train_scaled, y_train)

pred2 = clf2.predict(X_test_scaled)

print(classification_report(y_test, pred2))

# c. train a second network with a different topology and different settings
# d. test and evaluate
clf3 = MLPClassifier(solver = 'sgd', hidden_layer_sizes = (3,2), max_iter =␣
 ↪1000, random_state = 1234)
clf3.fit(X_train_scaled, y_train)
clf3.score(X_train_scaled, y_train)

pred3 = clf3.predict(X_test_scaled)

print(classification_report(y_test, pred3))

# e. compare the two models and why you think the performance was same/different
# The performance was very similar, but the first model with one hidden layer␣
 ↪performed slightly better.
# Having one layer is best for representing linear relationships in the data,␣
 ↪which makes sense for this data set.
```

```
              precision    recall  f1-score   support

           0       0.93      0.84      0.88        50
           1       0.76      0.89      0.82        28

    accuracy                           0.86        78
   macro avg       0.85      0.87      0.85        78
weighted avg       0.87      0.86      0.86        78

              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
```

11. Analysis

a. which algorithm performed better?

The logistic regression and decision tree had roughly the same results, so both performed better than the neural network.

b. compare accuracy, recall and precision metrics by class

Logistic regression:

Accuracy: 90%

Recall (mpg_low): 84%

Recall (mpg_high): 100%

Precision (mpg_low): 100%

Precision (mpg_high): 78%

Decision Tree:

Accuracy: 90%

Recall (mpg_low): 86%

Recall (mpg_high): 96%

Precision (mpg_low): 98%

Precision (mpg_high): 79%

Neural Network 1:

Accuracy: 86%

Recall (mpg_low): 84%

Recall (mpg_high): 89%

Precision (mpg_low): 93%

Precision (mpg_high): 76%

Neural Network 2:

Accuracy: 85%

Recall (mpg_low): 82%

Recall (mpg_high): 89%

Precision (mpg_low): 93%

Precision (mpg_high): 74%

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

The logistic regression and decision tree algorithms may have outperformed the neural network since the data is not complex enough to warrant the neural network algorithm.

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

Due to spending lots of time in this class understanding and coding in R, I currently prefer using it. However, this may be due to my dislike of Google Colab since I struggled with locating the proper tools to help me in this assignment. In the future, more practice with sklearn may change my mind.