Portfolio Component: ML with SKLearn

Neo Zhao

CS 4375 - Introduction to Machine Learning

1) Read the Auto Data

```
import pandas as pd
import seaborn as sb

# 1.a.
autoData = pd.read_csv('Auto.csv', sep = ',')

# 1.b.
autoData.head()

# 1.c.
autoData.shape
```

```
(392, 9)
```

2) Data Exploration

```
# 2.a.
autoData.describe()
```

mpg cylinders displacement horsepower weight acceleration

2.b.

MPG

- Range = 9 to 46.6
- Mean = 23.45

Cylinders

- Range = 3 to 8
- Mean = 5.47

Displacement

- Range = 68 to 455
- Mean = 194.41

Horsepower

- Range = 46 to 230
- Mean = 104.47

Weight

- Range = 1613 to 5140
- Mean = 2977.58

Acceleration

- Range = 8 to 24.8
- Mean = 15.55

Year

- Range = 70 to 82
- Mean = 76.01

Origin

- Range = 1 to 3
- Mean = 1.58

3) Explore Data Types

```
# 3.a.
print("Data Types before Modification")
print(autoData.dtypes)
```

```
# 3.b.
autoData.cylinders = autoData.cylinders.astype('category').cat.codes

# 3.c.
autoData.origin = autoData.origin.astype('category')

# 3.d.
print("\nData Types after Modification")
print(autoData.dtypes)
```

```
Data Types before Modification
cylinders           int8
displacement     float64
horsepower         int64
weight             int64
acceleration     float64
year             float64
origin          category
mpg_high        category
dtype: object

Data Types after Modification
cylinders           int8
displacement     float64
horsepower         int64
weight             int64
acceleration     float64
year             float64
origin          category
mpg_high        category
dtype: object
```

4) Deal with NAs

```
# 4.a.
autoData = autoData.dropna()

# 4.b.
autoData.shape
```

```
(389, 9)
```

5) Modify Columns

```
# 5.a.
```

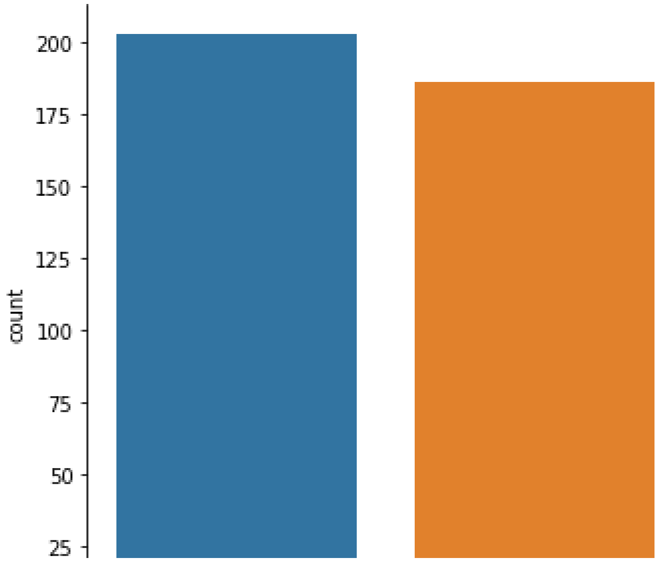
```
avg = autoData['mpg'].mean()
autoDataNew = autoData.loc[autoData.mpg > 1].copy()

autoDataNew.loc[:, 'mpg_high'] = [0 if x < avg else 1 for x in autoDataNew['mpg']]
autoData = autoDataNew
autoData.mpg_high = autoData.mpg_high.astype('category')

# 5.b.
autoData = autoData.drop (columns = ['mpg', 'name'])

# 5.c.
print(autoData.head())

print(autoDataNew.head())
```

```
   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
    mpg  cylinders  displacement  horsepower  weight  acceleration  year  \
0  18.0          4         307.0         130    3504          12.0  70.0
1  15.0          4         350.0         165    3693          11.5  70.0
2  18.0          4         318.0         150    3436          11.0  70.0
3  16.0          4         304.0         150    3433          12.0  70.0
6  14.0          4         454.0         220    4354           9.0  70.0

  origin                       name mpg_high
0      1  chevrolet chevelle malibu        0
1      1          buick skylark 320        0
2      1         plymouth satellite        0
3      1              amc rebel sst        0
6      1           chevrolet impala        0
```

6) Data Exploration (Graphs)

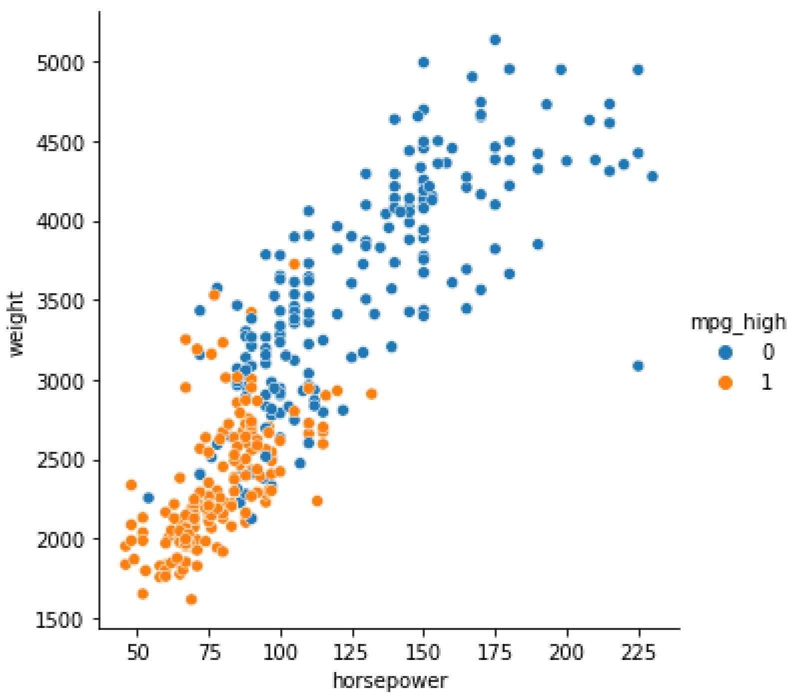
```
# 6.a.
sb.catplot(x = 'mpg_high', kind = 'count', data = autoData)
```

<seaborn.axisgrid.FacetGrid at 0x7f6292cebb90>



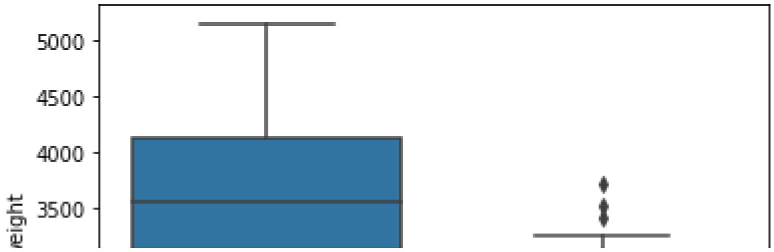

```
# 6.b.
sb.relplot(x = 'horsepower', y = 'weight', data = autoData, hue = autoData.mpg_high)
```

<seaborn.axisgrid.FacetGrid at 0x7f6292850990>




```
# 6.c.
sb.boxplot(x = 'mpg_high', y = 'weight', data = autoData)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f62927abed0>
```



6.d.

6.a.

- There are more samples of "low_mpg" than "high_mpg"; however, they are almost evenly split

6.b.

- A lighter car has less horsepower than any heavier car. Therefore, heavier cars are likely to comsume gas faster than lighter cars.

#6.c.

- "high_mpg seems to have several outliers while "low_mpg" doesn't have any.

7) Train/Test Split

```
# 7.a. & 7.b. & 7.c. & 7.d.
from sklearn.model_selection import train_test_split

X = autoData.loc[:, autoData.columns != 'mpg_high']
y = autoData.mpg_high

# Split into 80/20 with seed = 1234
xTrain, xTest, yTrain, yTest = train_test_split(X, y, test_size = 0.2, random_state = 1234)

print('Train Size:', xTrain.shape)
print('Test Size:', xTest.shape)
```

```
Train Size: (311, 7)
Test Size: (78, 7)
```

8) Logistic Regression

```
# 8.a.
```

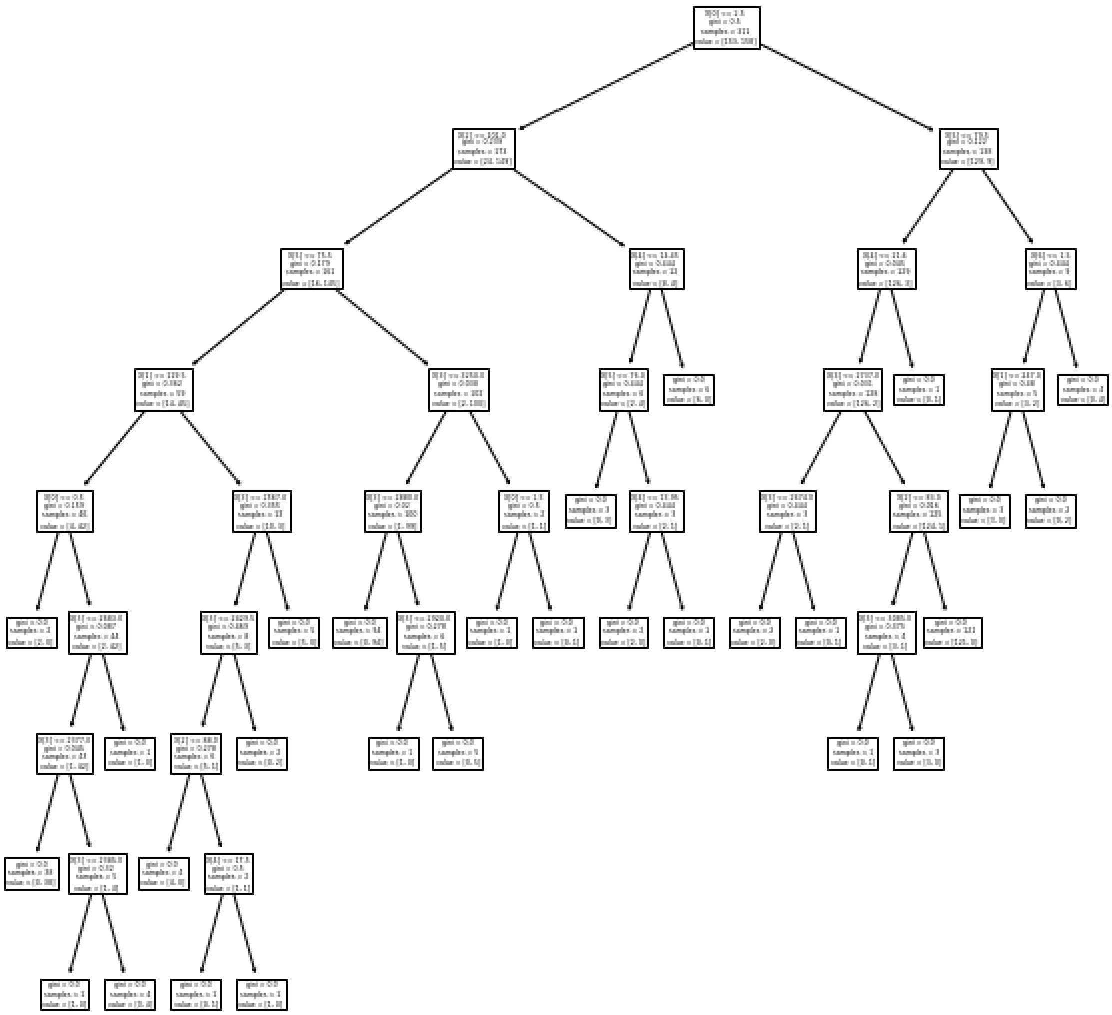
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report

clf = LogisticRegression(solver = 'lbfgs', max_iter = 300)
clf.fit(xTrain, yTrain)
clf.score (xTrain, yTrain)

# 8.b.
pred = clf.predict(xTest)

# 8.c.
# 0 = "low_mpg"
# 1 = "high_mpg"

print(classification_report(yTest, 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 Trees

```
# 9.a.
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn import tree
from matplotlib import pyplot as plt

clf2 = DecisionTreeClassifier()
clf2.fit(xTrain, yTrain)
clf2.score(xTrain, yTrain)

# 9.b.
pred2 = clf2.predict(xTest)

# 9.c.
print(classification_report(yTest, pred))

# 9.d.
plt.figure(figsize = (10, 10))
tree.plot_tree(clf2)
plt.show()
```

```
              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
```



10) Neural Network

```
# 10.a.
from sklearn import preprocessing
from sklearn.neural_network import MLPClassifier

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

xTrainScale = scaler.transform(xTrain)
```

```
xTestScale = scaler.transform(xTest)

clf3 = MLPClassifier (solver = 'lbfgs', hidden_layer_sizes = (6), max_iter = 1500, random_sta
clf3.fit(xTrainScale, yTrain)
clf3.score (xTrainScale, yTrain)

# 10.b.
pred3 = clf3.predict(xTestScale)

print(classification_report(yTest, pred3))

# 10.c.
clf4 = MLPClassifier(solver = 'sgd', hidden_layer_sizes = (3,), max_iter = 1500, random_state
clf4.fit(xTrainScale, yTrain)
clf4.score (xTrainScale, yTrain)

# 10.d.
pred4 = clf4.predict(xTestScale)

print(classification_report(yTest, pred4))
```

```
              precision    recall  f1-score   support

           0       0.94      0.90      0.92        50
           1       0.83      0.89      0.86        28

    accuracy                           0.90        78
   macro avg       0.89      0.90      0.89        78
weighted avg       0.90      0.90      0.90        78

              precision    recall  f1-score   support

           0       0.93      0.80      0.86        50
           1       0.71      0.89      0.79        28

    accuracy                           0.83        78
   macro avg       0.82      0.85      0.83        78
weighted avg       0.85      0.83      0.84        78
```

10.e.

- The first Neural Network performed better by about 7% compared to the second one. I used to same amount of interations for both networks, but the hidden layers are different for each of the networks. Logistic Regression and Decision Tree performed the same, which might mean there's a mistake somewhere.

11) Analysis

11.a. Which algorithm performed better?

- I think the Neural Networks performed the best

11.b. Compare Accuracy, Recall, and Precision metrics by class

- Logistic Regression
  - Accuracy = 90%
  - "low_mpg" Recall = 84%
  - "high_mpg" Recall = 100%
  - "low_mpg" Precision = 100%
  - "high_mpg" Precision = 78%
- Decision Tree
  - Accuracy = 90%
  - "low_mpg" Recall = 84%
  - "high_mpg" Recall = 100%
  - "low_mpg" Precision = 100%
  - "high_mpg" Precision = 78%
- Neural Network #1
  - Accuracy = 90%
  - "low_mpg" Recall = 90%
  - "high_mpg" Recall = 89%
  - "low_mpg" Precision = 94%
  - "high_mpg" Precision = 83%
- Neural Network #2
  - Accuracy = 83%
  - "low_mpg" Recall = 80%
  - "high_mpg" Recall = 89%
  - "low_mpg" Precision = 93%
  - "high_mpg" Precision = 71%

#11.c. Give your analysis of why the better-performing algorithm might have outperformed the other

- I believe Neural Networks perform more efficiently when working with complex data, allowing the model to outperform Logistic Regression as well as Decision Trees.

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

- I definitely prefer R at this point, since I have worked with it more in the past and I am more comfortable with this language; however, after this assignment, I am ready to dive deeper into learning Machine Learning with Python as it is always useful to know more.