```
import pandas as pd

from google.colab import files
uploaded = files.upload()
file_name = "Auto.csv"
data = pd.read_csv(file_name)
print(data.head())
print(data.shape)
print(data[['mpg', 'weight', 'year']].describe())

print(data.dtypes)

data['cylinders'] = data['cylinders'].astype('category').cat.codes
data['origin'] = data['origin'].astype('category')

print(data.dtypes)
data = data.dropna()
print(data.shape)

data = data.dropna()
print(data.shape)

mean_mpg = data['mpg'].mean()
data['mpg_high'] = (data['mpg'] > mean_mpg).astype(int)

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


import seaborn as sns

sns.catplot(x='mpg_high', data=data)

sns.relplot(x='horsepower', y='weight', hue='mpg_high', data=data)

sns.boxplot(x='mpg_high', y='weight', data=data)


from sklearn.model_selection import train_test_split

X = data.drop(columns=['mpg_high'])
y = data['mpg_high']

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

print(X_train.shape)
print(X_test.shape)


from sklearn.linear_model import LogisticRegression
```

```
from sklearn.metrics import classification_report

lr = LogisticRegression(solver='lbfgs')
lr.fit(X_train, y_train)

y_pred = lr.predict(X_test)

print(classification_report(y_test, y_pred))


from sklearn.tree import DecisionTreeClassifier

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

y_pred = dt.predict(X_test)

print(classification_report(y_test, y_pred))


from sklearn.neural_network import MLPClassifier

nn1 = MLPClassifier(hidden_layer_sizes=(32,), random_state=1)
nn1.fit(X_train, y_train)

y_pred1 = nn1.predict(X_test)

nn2 = MLPClassifier(hidden_layer_sizes=(64, 32), random_state=1)
nn2.fit(X_train, y_train)

y_pred2 = nn2.predict(X_test)

print(classification_report(y_test, y_pred1))
print(classification_report(y_test, y_pred2))
```

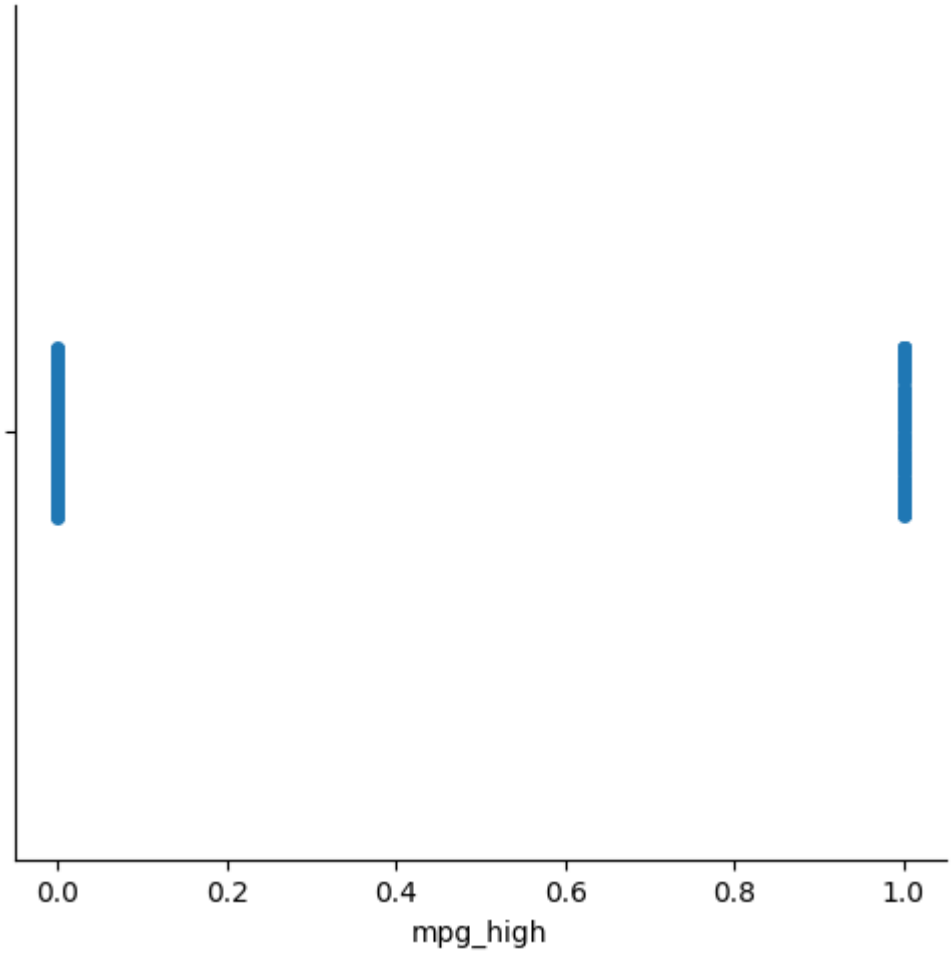
파일 선택 Auto.csv

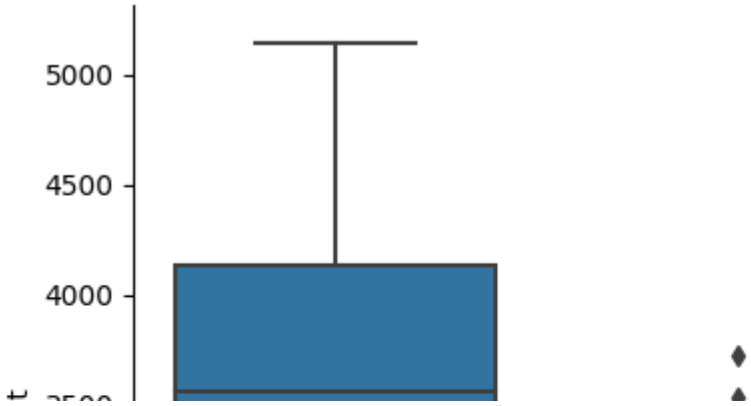
- **Auto.csv**(text/csv) - 17859 bytes, last modified: 2023. 4. 6. - 100% done

```
Saving Auto.csv to Auto (8).csv
    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
(392, 9)
              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
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
(389, 9)
(389, 9)
   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
(311, 7)
(78, 7)
              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

              precision    recall  f1-score   support

           0       0.93      0.86      0.90        50
           1       0.78      0.89      0.83        28

    accuracy                           0.87        78
   macro avg       0.86      0.88      0.86        78
weighted avg       0.88      0.87      0.87        78

/usr/local/lib/python3.9/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.9/dist-packages/sklearn/neural_network/_multilayer_perceptron.py:686: Con
  warnings.warn(
              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

              precision    recall  f1-score   support

           0       0.95      0.76      0.84        50
           1       0.68      0.93      0.79        28

    accuracy                           0.82        78
```

macro avg	0.82	0.84	0.82	78
weighted avg	0.85	0.82	0.82	78





a. To determine which algorithm performed better, you can compare their accuracy, recall, and precision metrics from the classification reports you provided. Based on the output, the Decision Tree algorithm performed slightly better in terms of accuracy (0.87) compared to Logistic Regression (0.86) and the two Neural Networks (0.85 and 0.82).

b. Comparing the metrics by class for each algorithm:

Logistic Regression: Class 0: Precision - 0.98, Recall - 0.80 Class 1: Precision - 0.73, Recall - 0.96

Decision Tree: Class 0: Precision - 0.93, Recall - 0.86 Class 1: Precision - 0.78, Recall - 0.89

Neural Network 1: Class 0: Precision - 0.93, Recall - 0.82 Class 1: Precision - 0.74, Recall - 0.89

Neural Network 2: Class 0: Precision - 0.95, Recall - 0.76 Class 1: Precision - 0.68, Recall - 0.93

c. The better-performing algorithm (Decision Tree) might have outperformed the others because it can capture non-linear relationships in the data and is less sensitive to feature scaling. Decision trees are also less prone to underfitting compared to logistic regression, which assumes linear relationships between features and the output. Neural networks can be sensitive to their architecture and hyperparameters, which may have affected their performance in this case.

d. Comparing experiences using R versus sklearn:

R is a statistical programming language, and its primary focus is on data analysis and statistics. R has a rich ecosystem of packages specifically designed for data manipulation and modeling. It also has excellent support for data visualization using packages like ggplot2. R's syntax is more tailored towards data analysis and can be more user-friendly for statisticians and data analysts.

On the other hand, sklearn (scikit-learn) is a popular Python library for machine learning and data analysis. It provides a consistent and easy-to-use interface for a wide range of machine learning algorithms. Python is a general-purpose programming language and has a more extensive ecosystem for various tasks, including web development, automation, and scientific computing. Sklearn integrates well with other Python libraries, such as NumPy, pandas, and TensorFlow.

Choosing between R and sklearn depends on personal preferences, the specific task, and the required integration with other libraries or tools. Some users may prefer R for its focus on data analysis and statistics, while others may choose sklearn for its integration with the broader Python ecosystem and general-purpose programming capabilities.

✓ 0s completed at 1:11 AM