1a

```
import pandas as pd
import numpy as np
import seaborn as sb
from google.colab import files
auto = pd.read_csv('Auto.csv')
```

1b

```
auto.head()
```

	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

1c

```
print(auto.shape)
```

```
(392, 9)
```

2a

```
auto['mpg'].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
```

```
auto['weight'].describe()
```

```
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
```

```
auto['year'].describe()
```

```
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
```

2b

mpg---

range : 37.000000

mean : 23.445918

weight---

range : 3527.000000

mean : 2977.584184

year---

range : 12.000000

mean : 76.010256

3a

```
auto.dtypes
```

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

3b

```
auto['cylinders'] = auto['cylinders'].astype('category').cat.codes
```

3c

```
auto['origin'] = auto['origin'].astype('category')
```

3d

```
auto.dtypes
```

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

4a

```
auto.dropna(axis=0)
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name
0	18.0	4	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu
1	15.0	4	350.0	165	3693	11.5	70.0	1	buick skylark 320
2	18.0	4	318.0	150	3436	11.0	70.0	1	plymouth satellite

4b

```
auto.shape
```

```
(392, 9)
```

Double-click (or enter) to edit

390	28.0	1	120.0	79	2625	18.6	82.0	1	ford ranger

5a

389 rows × 9 columns

```
auto['mpg_high'] = auto['mpg'].map(lambda x: x > 23.445918)
auto['mpg_high'] = auto['mpg_high'].astype('int')
auto['mpg_high'] = auto['mpg_high'].astype('category')
```

```
auto.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	mpg_high
0	18.0	8	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu	0
1	15.0	8	350.0	165	3693	11.5	70.0	1	buick skylark 320	0
2	18.0	8	318.0	150	3436	11.0	70.0	1	plymouth satellite	0
3	16.0	8	304.0	150	3433	12.0	70.0	1	amc rebel sst	0
4	17.0	8	302.0	140	3449	NaN	70.0	1	ford torino	0

5b

```
auto.drop(labels = 'mpg', axis = 1)
```

	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	mpg_high
0	8	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu	0
1	8	350.0	165	3693	11.5	70.0	1	buick skylark 320	0
2	8	318.0	150	3436	11.0	70.0	1	plymouth satellite	0
3	8	304.0	150	3433	12.0	70.0	1	amc rebel sst	0
4	8	302.0	140	3449	NaN	70.0	1	ford torino	0
...	...	...	...	...	...	...	...	...	...
387	4	140.0	86	2790	15.6	82.0	1	ford mustang gl	1
388	4	97.0	52	2130	24.6	82.0	2	vw pickup	1
389	4	135.0	84	2295	11.6	82.0	1	dodge rampage	1
390	4	120.0	79	2625	18.6	82.0	1	ford ranger	1
391	4	119.0	82	2720	19.4	82.0	1	chevy s-10	1

392 rows × 9 columns

5c

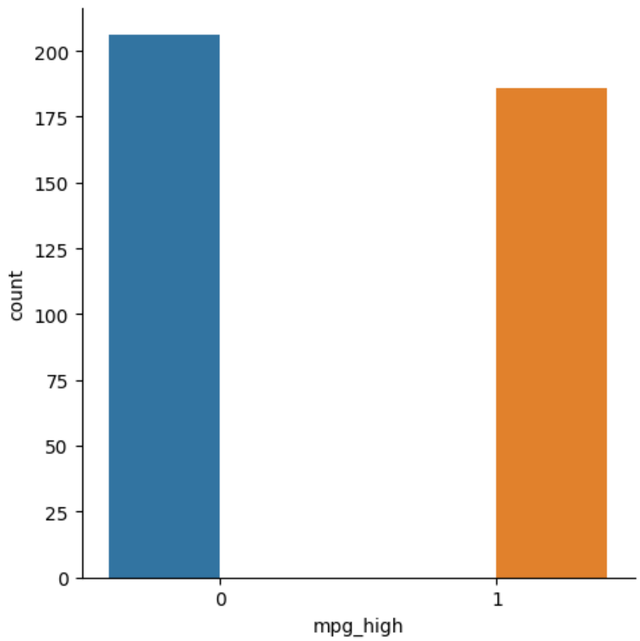
```
auto.head()
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	year	origin	name	mpg_high
0	18.0	8	307.0	130	3504	12.0	70.0	1	chevrolet chevelle malibu	0
1	15.0	8	350.0	165	3693	11.5	70.0	1	buick skylark 320	0
2	18.0	8	318.0	150	3436	11.0	70.0	1	plymouth satellite	0

6a

```
sb.catplot(data = auto , x= 'mpg_high', hue = 'mpg_high', kind = 'count')
```

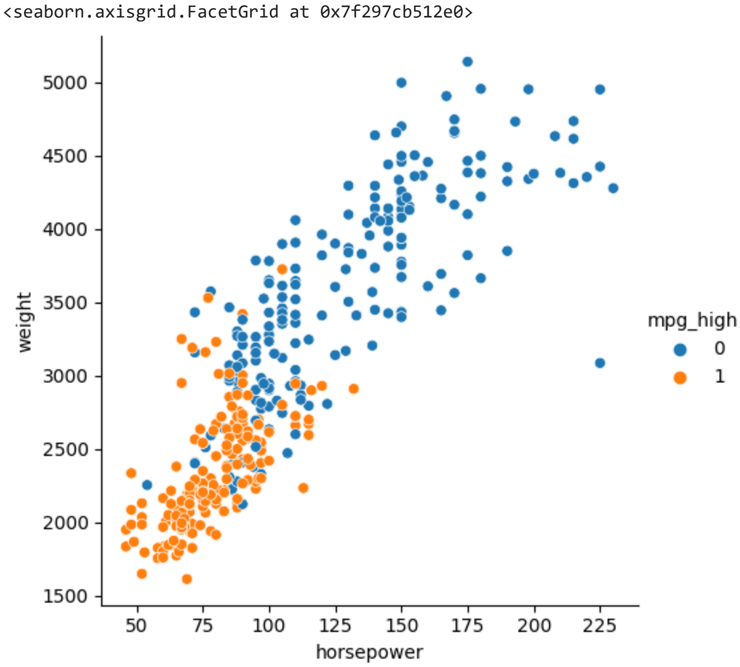
```
<seaborn.axisgrid.FacetGrid at 0x7f297cc631c0>
```



6b

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

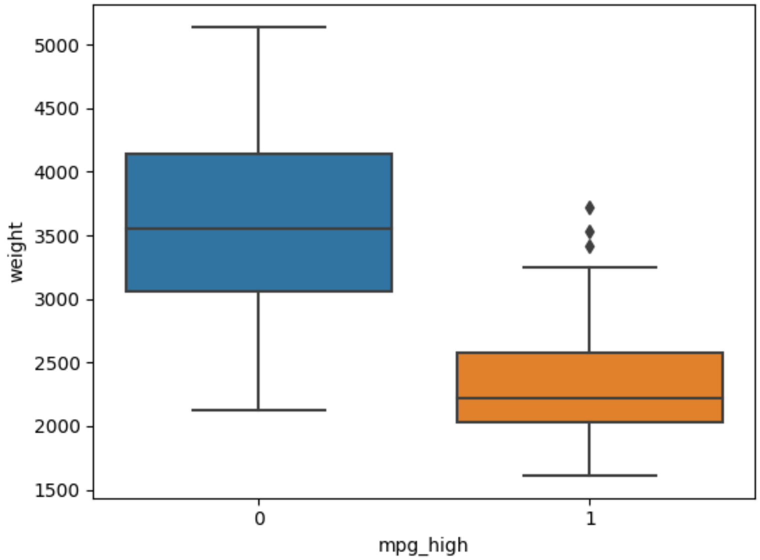
```
<seaborn.axisgrid.FacetGrid at 0x7f297cb512e0>
```



6c

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

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



6d

catplot : there are more cars with a lower than average mpg

relplot : there ia a strong positive correlation between weight and horsepower and as they increase, it is more likely that mpg_high is false

boxplot : cars that have a lower weight typically have a higher than average mpg

7

```
from sklearn.model_selection import train_test_split
X = auto.values[:,1:8]
Y = auto.values[:,0]

X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = .2, random_state = 1234)
```

```
X_train.shape
```

```
(313, 7)
```

```
X_test.shape
```

```
(79, 7)
```

```
y_train.shape
```

```
(313,)
```

```
y_test.shape
```

```
(79,)
```

8

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
classifier_tree = DecisionTreeClassifier()
class_names = Y
y_predict = classifier_tree.fit(X_train,y_train).predict(X_test)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-174-0d4c940a7357> in <cell line: 6>()
      4 classifier_tree = DecisionTreeClassifier()
      5 class_names = Y
----> 6 y_predict = classifier_tree.fit(X_train,y_train).predict(X_test)
      7

4 frames

/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py in _assert_all_finite(X, allow_nan, msg_dtype, estimator_name,
input_name)
    159                 "#estimators-that-handle-nan-values"
    160             )
--> 161         raise ValueError(msg_err)
    162
    163

ValueError: Input X contains NaN.
DecisionTreeClassifier does not accept missing values encoded as NaN natively. For supervised learning, you might want to consider
sklearn.ensemble.HistGradientBoostingClassifier and Regressor which accept missing values encoded as NaNs natively. Alternatively, it
is possible to preprocess the data, for instance by using an imputer transformer in a pipeline or drop samples with missing values.
See https://scikit-learn.org/stable/modules/impute.html You can find a list of all estimators that handle NaN values at the following
page: https://scikit-learn.org/stable/modules/impute.html#estimators-that-handle-nan-values
```

```
print(classification_report(y_test, y_predict, target_names = class_names))
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-160-3cfc27bd19bf> in <cell line: 1>()
----> 1 print(classification_report(y_test, y_predict, target_names = class_names))

NameError: name 'y_predict' is not defined
```

SEARCH STACK OVERFLOW

```
import tree
clf_auto = DecisionTreeClassifier(criterion='entropy', random_state = 1234, max_depth = 3, min_samples_leaf=5)
clf_auto.fit(X_train, y_train)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-167-bbff0add28de> in <cell line: 3>()
      1 import tree
      2 clf_auto = DecisionTreeClassifier(criterion='entropy', random_state = 1234, max_depth = 3, min_samples_leaf=5)
----> 3 clf_auto.fit(X_train, y_train)

3 frames

/usr/local/lib/python3.9/dist-packages/sklearn/utils/validation.py in check_array(array, accept_sparse, accept_large_sparse, dtype, ord
ensure_min_samples, ensure_min_features, estimator, input_name)
    900             # If input is 1D raise error
    901             if array.ndim == 1:
--> 902                 raise ValueError(
    903                     "Expected 2D array, got 1D array instead:\narray={}.\n"
    904                     "Reshape your data either using array.reshape(-1, 1) if "

ValueError: Expected 2D array, got 1D array instead:
array=[0. 1. 0. 0. 0. 1. 1. 0. 1. 0. 1. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 1.
 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 1.
 1. 0. 1. 0. 0. 0. 1. 1. 1. 0. 0. 1. 0. 0. 1. 0. 1. 0. 0. 1. 0. 1. 0. 1.
 1. 0. 0. 1. 0. 1. 0. 1. 0. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0.
 1. 0. 1. 1. 1. 1. 0. 0. 1. 0. 0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 1. 0. 1. 1.
 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 0. 1. 1. 1. 1. 1. 0. 1. 0. 1. 0. 1.
 1. 0. 0. 0. 0. 0. 0. 1. 1. 1. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1. 1. 0. 1. 0.
 1. 0. 0. 0. 1. 1. 0. 0. 0. 0. 1. 0. 0. 0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 1.
 1. 0. 0. 1. 0. 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.
 1. 0. 0. 0. 0. 1. 1. 1. 1. 0. 0. 1. 0. 1. 0. 1. 0. 1. 0. 0. 0. 0. 0. 1.
 0. 1. 1. 0. 1. 1. 0. 0. 1. 0. 1. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.
 0. 1. 0. 1. 0. 1. 1. 1. 1. 0. 0. 0. 0. 1. 1. 1. 1. 1. 1. 1. 0. 1. 0. 0.
 0. 1. 1. 1. 1. 1. 0. 1. 0. 1. 0. 1. 1. 0. 1. 0. 0. 1. 0. 1. 1. 1. 1. 0.
 1.].
Reshape your data either using array.reshape(-1, 1) if your data has a single feature or array.reshape(1, -1) if it contains a single s
```

SEARCH STACK OVERFLOW

11

Between R and sklearn I had an easier time with sklearn. It feels more user friendly in terms of data manipulation, but I had a hard time with creating train test splits. Overall I think I still prefer R since I've spent more time with it but I can easily see myself transitioning to sklearn as I advance in data science.