https://colab.research.google.com/drive/1IBQqoM0ZF22CFua9O4N0KgYrnjLVwg (https://colab.research.google.com/drive/1IBQqoM0ZF22CFua9O4N0KgYrnjLVw

표준편차

https://m.blog.naver.com/galaxyenergy/221400040231 (https://m.blog.naver.com/galaxyenergy/221400040231)

data 가져오기
바꿔주기 ->dataframe으로 가능케
분석에 적합하도록 분석용 dataset 만들기 -> 전처리 과정

- describe
  - info: type, null값은 있는지 등
  - 문자 -> 숫자로 변경
  - 나이 등은 카테고리, 범주화하는 등
  - histogram, scatter 등으로 시각화 해서 보는 등

DATA 읽기

In [ ]:
```
1 %pwd
```

'C:₩₩hmkd1₩₩m2_분석라이브러리₩₩개인 정리₩₩pandas'

In [ ]:

```
# read_csv()
# CSV는 Comma Separated Values의 약어로, 쉼표로 구분된 값들을
# 'C:₩₩hmkd1₩₩m2_분석라이브러리₩dataset₩mtcars.csv'
# 운영체제 공통이므로 '/'를 써라
# '₩'는 윈도우 운영체제에서 경로 설정하는 것임


import pandas as pd

# df_auto = pd.read_csv('../../dataset/mtcars.csv')
df_auto = pd.read_csv('../../dataset/mtcars.csv', index_col=
df_auto.head()
```

	mpg	cyl	disp	hp	drat	wt	qsec	v
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0

In [ ]:

```
import pandas as pd
# 한글 Windows 운영체제에서 사용되는 문자 인코딩 방식 중 하나
# CP949로 인코딩된 파일을 읽어들일 때 사용하는 옵션
# CP949는 EUC-KR과 유사한 문자 인코딩 방식으로, 한글 외에도
# 이 옵션을 사용하지 않으면, 한글이 깨져서 나타나거나, Unico


df = pd.read_csv('../../dataset/X_train.csv', encoding='cp94
df
```

	cust_id	총구매액	최대구매액	환불금액	주구매상품	주구매지점
0	0	68282840	11264000	6860000.0	기타	강남점
1	1	2136000	2136000	300000.0	스포츠	잠실점
2	2	3197000	1639000	NaN	남성캐주얼	관악점
3	3	16077620	4935000	NaN	기타	광주점
4	4	29050000	24000000	NaN	보석	본점
...	...	...	...	...	...	...
3495	3495	3175200	3042900	NaN	골프	본점
3496	3496	29628600	7200000	6049600.0	시티웨어	부산본점
3497	3497	75000	75000	NaN	주방용품	창원점
3498	3498	1875000	1000000	NaN	화장품	본점
3499	3499	263101550	34632000	5973000.0	기타	본점

3500 rows × 10 columns

In [ ]:

```
fifa = pd.read_csv('../../dataset/FIFA.csv', encoding='cp949
fifa.head()
```

	ID	Name	Age	Nationality	Overall	Club
0	158023	L. Messi	31	Argentina	94	FC Barce
1	20801	Cristiano Ronaldo	33	Portugal	94	Juven
2	190871	Neymar Jr	26	Brazil	92	Paris Germ
3	193080	De Gea	27	Spain	91	Manc Unite
4	192985	K. De Bruyne	27	Belgium	91	Manc City

파생 변수 만드는 법

In [ ]:

```
# Seaborn은 파이썬 데이터 시각화 라이브러리 중 하나
# Seaborn은 matplotlib을 기반으로 만들어졌으며,
# matplotlib으로는 그리기 어려운 복잡하고 고급스러운 그래프를
# Matplotlib은 파이썬에서 가장 많이 사용되는 데이터 시각화 라
# 또한 Seaborn에서는 기본적으로 pandas DataFrame 형태의 데이
# 따라서 데이터를 사용하기 전에 pandas를 사용하여 데이터를 전
# Seaborn에서 제공하는 함수를 이용하여 그래프를 그리는 것이
# titanic- seaborn에서 미리 제공하는 sample dataset임 (이건

# head()는 데이터프레임이나 시리즈에서 처음 몇 개의 행을 반환

import seaborn as sns

df = sns.load_dataset('titanic')
df.head(3)

# df.tail()
```

	survived	pclass	sex	age	sibsp	parch	fare
0	0	3	male	22.0	1	0	7.250
1	1	1	female	38.0	1	0	71.28
2	1	3	female	26.0	0	0	7.925

In [ ]:

```
df_org = df.copy()
```

In [ ]:

```
# unique() - 배열이나 리스트, 데이터프레임 등에서 중복되지 않
# Series에 사용하는 pandas으 unique() (cf> np.unique()도 있음
# 891 -> 89개임 (중복 제거)

df.age.unique()
```

```
array([22.  , 38.  , 26.  , 35.  ,  nan, 54.  ,
2.  , 27.  , 14.  ,
       4.  , 58.  , 20.  , 39.  , 55.  , 31.  , 3
4.  , 15.  , 28.  ,
       8.  , 19.  , 40.  , 66.  , 42.  , 21.  , 1
8.  ,  3.  ,  7.  ,
      49.  , 29.  , 65.  , 28.5 ,  5.  , 11.  , 4
5.  , 17.  , 32.  ,
      16.  , 25.  ,  0.83, 30.  , 33.  , 23.  , 2
4.  , 46.  , 59.  ,
      71.  , 37.  , 47.  , 14.5 , 70.5 , 32.5 , 1
2.  ,  9.  , 36.5 ,
      51.  , 55.5 , 40.5 , 44.  ,  1.  , 61.  , 5
6.  , 50.  , 36.  ,
      45.5 , 20.5 , 62.  , 41.  , 52.  , 63.  , 2
3.5 ,  0.92, 43.  ,
      60.  , 10.  , 64.  , 13.  , 48.  ,  0.75, 5
3.  , 57.  , 80.  ,
      70.  , 24.5 ,  6.  ,  0.67, 30.5 ,  0.42, 3
4.5 , 74.  ])
```

In [ ]:

```
def get_category(x):
    cat = ''
    if x <= 10:
        cat = 'Child'
    elif x <= 30:
        cat = 'Young Adult'
    elif x <= 60:
        cat = 'Adult'
    else:
        'Elderly'
    return cat

df['age_class'] = df.age.apply(lambda x:get_category(x))

# df.loc[:,['age','age_class']].head()
df[['age','age_class']].head()
```

	age	age_class
0	22.0	Young Adult
1	38.0	Adult
2	26.0	Young Adult
3	35.0	Adult
4	35.0	Adult

In [ ]:

```
# Q. df에서 sibsp + parch -> family라는 파생변수를 apply() 혹
# 출력은 sibsp, parch, family 세개 컬럼만
# sibsp = Sibling(형제자매)와 Spouse(배우자)의 수
# parch = Parents and their Children

import seaborn as sns

df_org = df.copy()

def add(x,y):
    return x+y

df['family'] = df[['sibsp', 'parch']].apply(lambda x: add(x[
df.tail()
```

	survived	pclass	sex	age	sibsp	parch	fa
886	0	2	male	27.0	0	0	13
887	1	1	female	19.0	0	0	30
888	0	3	female	NaN	1	2	23
889	1	1	male	26.0	0	0	30
890	0	3	male	32.0	0	0	7.7

In [ ]:

```
# 간단한 방법

df['family'] = df['sibsp'] + df.loc[:,'parch']
df
```

	survived	pclass	sex	age	sibsp	parch	fa
0	0	3	male	22.0	1	0	7.2
1	1	1	female	38.0	1	0	71
2	1	3	female	26.0	0	0	7.9
3	1	1	female	35.0	1	0	53
4	0	3	male	35.0	0	0	8.(
...	...	...	...	...	...	...	...
886	0	2	male	27.0	0	0	13
887	1	1	female	19.0	0	0	30
888	0	3	female	NaN	1	2	23
889	1	1	male	26.0	0	0	30
890	0	3	male	32.0	0	0	7.7

891 rows × 17 columns

In [ ]:

```
df = df_org.copy()

def add(x,y):
    return x+y
df['family'] = df[['sibsp','parch']].apply(lambda row: add(r
df[['sibsp','parch','family']].head()
```

	sibsp	parch	family
0	1	0	1
1	1	0	1
2	0	0	0
3	1	0	1
4	0	0	0

In [ ]:

```
1
```

```
---------------------------------------------
-------------------------------
TypeError                                 Traceba
ck (most recent call last)
~WAppDataWLocalWTempWipykernel_8480W29533161
1.py in <module>
      1 a = lambda x: x+y
----> 2 a(1,2)

TypeError: <lambda>() takes 1 positional argument
but 2 were given
```

DATA 탐색 및 선처리 개요

In [ ]:

```
# df_auto = pd.read_csv('../../dataset/auto-mpg.csv')
df_auto = pd.read_csv('auto-mpg.csv')
df_auto.head()
```

	mpg	cylinders	displacement	horsepower	wei
0	18.0	8	307.0	130	3504
1	15.0	8	350.0	165	3693
2	18.0	8	318.0	150	3436
3	16.0	8	304.0	150	3433
4	17.0	8	302.0	140	3449

- displacement: 자동차 엔진에서 한 번의 싸이클에 실린 연료와 공기의 양으로 엔진이 발생시키는 피스톤의 움직임 거리를 말합니다. 일반적으로는 엔진의 크기와 성능을 나타내는 지표 중 하나로 사용됩니다. 단위는 주로 미터(m)나 인치(in)로 표시됩니다.
- acceleration은 물리학에서 가속도를 의미합니다. 자동차에서의 acceleration은 자동차가 달리는 속도를 빠르게 하는 정도를 나타내며, 보통은 km/h 당 초(second) 단위로 측정됩니다.

In [ ]:

```
# info() - DataFrame의 요약 정보
# object 자료형은 문자열 또는 다른 파이썬 객체가 포함된 열(c
# 일반적으로 문자열을 의미하지만, 다른 데이터 유형이 섞여 있
# 따라서, object 자료형 열이 문자열을 포함한다는 것은 보장됩

df_auto.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(3), int64(4), object(2)
memory usage: 28.1+ KB
```

In [ ]:

```
# describe() - DataFrame의 각 열에 대한 요약 통계를 계산하여
    # 기본적으로 수치 데이터에 대해서만 요약 통계를 계산함
    # standard deviation
    # quartiles - 데이터셋을 사분위로 나누어 분포를 측정하는

df_auto.describe()
```

	mpg	cylinders	displacement	weight
count	398.000000	398.000000	398.000000	398.0000(
mean	23.514573	5.454774	193.425879	2970.424(
std	7.815984	1.701004	104.269838	846.8417
min	9.000000	3.000000	68.000000	1613.000(
25%	17.500000	4.000000	104.250000	2223.750(
50%	23.000000	4.000000	148.500000	2803.500(
75%	29.000000	8.000000	262.000000	3608.000(
max	46.600000	8.000000	455.000000	5140.000(

In [ ]:

```
# unique() -  어떤 Series나 DataFrame 컬럼에서 중복을 제거하
# 하나라도 문자가 있으면 문자열로 인식한다 - object
# df_auto = pd.read_csv('../../dataset/auto-mpg.csv')

import pandas as pd
import numpy as np

print(df_auto.horsepower.unique())
```

```
['130' '165' '150' '140' '198' '220' '215' '225'
'190' '170' '160' '95'
 '97' '85' '88' '46' '87' '90' '113' '200' '210'
'193' '?' '100' '105'
 '175' '153' '180' '110' '72' '86' '70' '76' '65'
'69' '60' '80' '54'
 '208' '155' '112' '92' '145' '137' '158' '167' '9
4' '107' '230' '49' '75'
 '91' '122' '67' '83' '78' '52' '61' '93' '148' '1
29' '96' '71' '98' '115'
 '53' '81' '79' '120' '152' '102' '108' '68' '58'
'149' '89' '63' '48'
 '66' '139' '103' '125' '133' '138' '135' '142' '7
7' '62' '132' '84' '64'
 '74' '116' '82']
```

In [ ]:

```
# 문자도 있고하니, 숫자로된 것, 의미 있는 것만 뽑아냄

# df_auto[:, 'mpg','cylinders','displacement','weight','acce
df_auto[['mpg','cylinders','displacement','weight','accelera
```

	mpg	cylinders	displacement	weight	acceler
0	18.0	8	307.0	3504	12.0
1	15.0	8	350.0	3693	11.5
2	18.0	8	318.0	3436	11.0
3	16.0	8	304.0	3433	12.0
4	17.0	8	302.0	3449	10.5
...	...	...	...	...	...
393	27.0	4	140.0	2790	15.6
394	44.0	4	97.0	2130	24.6
395	32.0	4	135.0	2295	11.6
396	28.0	4	120.0	2625	18.6
397	31.0	4	119.0	2720	19.4

398 rows × 5 columns

In [ ]:

```
df_auto[['mpg','cylinders','displacement','weight','accelera
```

	mpg	cylinders	displacement	weight
count	398.000000	398.000000	398.000000	398.0000
mean	23.514573	5.454774	193.425879	2970.424
std	7.815984	1.701004	104.269838	846.8417
min	9.000000	3.000000	68.000000	1613.000
25%	17.500000	4.000000	104.250000	2223.750
50%	23.000000	4.000000	148.500000	2803.500
75%	29.000000	8.000000	262.000000	3608.000
max	46.600000	8.000000	455.000000	5140.000

In [ ]:

```
df_s = df_auto[['mpg','cylinders','displacement','weight','a
df_s.head()
```

	mpg	cylinders	displacement	weight	accelerat
0	18.0	8	307.0	3504	12.0
1	15.0	8	350.0	3693	11.5
2	18.0	8	318.0	3436	11.0
3	16.0	8	304.0	3433	12.0
4	17.0	8	302.0	3449	10.5

scikit-learn 또는 sklearn

- 파이썬에서 사용하는 머신러닝 라이브러리 중 하나입니다. 데이터 전처리, 특징 추출, 지도 학습 및 비지도 학습 알고리즘, 모델 평가 등의 기능을 제공합니다. scikit-learn은 다양한 분야에서 사용되며, 예를 들어, 영상 인식, 자연어 처리, 음성 인식 등 다양한 분야에서 활용됩니다.

preprocessing

- 데이터 전처리를 위한 모듈로, 다양한 스케일링, 변환, 인코딩 등의 기능을 제공합니다. 예를 들어, MinMaxScaler, StandardScaler, OneHotEncoder 등

의 함수를 제공합니다. 이러한 함수들은 머신러닝 모델에 데이터를 입력하기 전에 데이터를 전처리하는데 사용됩니다. 전처리는 데이터 분석과 머신러닝 모델링에서 중요한 단계 중 하나입니다. 데이터의 크기, 단위, 범위 등을 일정하게 맞춰줌으로써 머신러닝 모델의 성능을 향상시키고, 모델의 안정성을 높일 수 있습니다. preprocessing 모듈은 이러한 전처리 과정을 쉽게 수행할 수 있도록 도와줍니다.

MinMaxScaler

- 데이터를 0과 1 사이의 범위로 스케일링하는데 사용되는 Scikit-learn 라이브러리의 클래스입니다. 이를 사용하면 데이터의 최소값과 최대값을 찾아서 모든 데이터를 해당 범위로 변환할 수 있습니다. 이렇게 데이터를 스케일링하면 모델의 학습 속도를 높일 수 있고, 동시에 이상치(outlier)가 있는 경우에도 적절한 결과를 얻을 수 있습니다.

스케일링(scaling)

- 데이터의 값 범위(range)를 조정하는 작업 -> 분석에 적합한 형태로 만드는 데이터 변환 기법 중 하나
- 데이터의 값 범위를 조정하는 전처리 기법 중 하나로, 모든 feature(변수)가 동일한 범위를 가지도록 만들어 주는 과정을 말합니다. 스케일링은 특히, feature 간의 값의 차이가 매우 크고, 각 feature의 중요도가 동일하다고 가정할 때, 데이터를 모델에 적용하기 전에 반드시 수행되어야 합니다.
- 목적
  - 데이터의 범위를 조정하여, 모델의 학습을 더 잘 일반화시킵니다.
  - 독립 변수(feature) 간의 값의 차이가 크게 나는 경우, 학습 속도가 느려지거나 발산할 수 있습니다. 이를 방지하기 위해 독립 변수의 스케일을 맞춥니다.
  - 일부 머신 러닝 알고리즘에서는 스케일링이 모델 성능에 직접적인 영향을 미칩니다. 예를 들어, 거리 기반 알고리즘(KNN, K-means 등)에서는 스케일링을 하지 않으면, 변수 간의 차이가 클 경우 더 큰 영향력을 가진 변수로만 거리를 계산하게 됩니다.

- 스케일링에는 여러가지 종류가 있습니다. 예를 들어,
  - MinMax 스케일링은 데이터의 최솟값을 0, 최댓값을 1로 조정하는 방법 (조정하여 변환)
  - Standard 스케일링은 평균을 0, 표준편차를 1로 만드는 방법 (데이터의 평균과 표준편차를 이용하여 변환)
  - Robust Scaler: 중앙값과 사분위수를 이용하여, 중앙값이 0이고 사분위수 범위(IQR)가 1인 값으로 변환합니다.

둘 다 데이터의 범위를 일정하게 만들지만 -> 데이터의 분포를 고려하여 변환하기 때문에 Standard Scaling이 더 많이 쓰임

In [ ]:

```
# 스케일링
# fit_transform() - fit()과 transform() 메소드를 한 번에 실행

from sklearn.preprocessing import MinMaxScaler # 라이브러리-

scaler= MinMaxScaler()
df_scaler = scaler.fit_transform(df_s) # 0과 1사이의 숫자로
print(df_scaler) # 배열로 바뀜!
df_sca = pd.DataFrame(df_scaler, columns=['mpg','cylinders',
df_sca.head()
```

```
[[0.2393617  1.         0.61757106 0.5361497  0.23
809524]
 [0.15957447 1.         0.72868217 0.58973632 0.20
833333]
 [0.2393617  1.         0.64599483 0.51686986 0.17
857143]
 ...
 [0.61170213 0.2        0.17312661 0.19336547 0.21
428571]
 [0.50531915 0.2        0.13436693 0.2869294  0.63
095238]
 [0.58510638 0.2        0.13178295 0.31386447 0.67
857143]]
```

	mpg	cylinders	displacement	weight	accel
0	0.239362	1.0	0.617571	0.536150	0.238
1	0.159574	1.0	0.728682	0.589736	0.208
2	0.239362	1.0	0.645995	0.516870	0.178
3	0.186170	1.0	0.609819	0.516019	0.238
4	0.212766	1.0	0.604651	0.520556	0.148

In [ ]:

```
df_sca.describe()
```

	mpg	cylinders	displacement	weight
count	398.000000	398.000000	398.000000	398.0000
mean	0.386026	0.490955	0.324098	0.384867
std	0.207872	0.340201	0.269431	0.240103
min	0.000000	0.000000	0.000000	0.000000
25%	0.226064	0.200000	0.093669	0.173164
50%	0.372340	0.200000	0.208010	0.337539
75%	0.531915	1.000000	0.501292	0.565637
max	1.000000	1.000000	1.000000	1.000000

상관계수

- 두 변수 사이의 상관 관계의 강도와 방향성을 나타내는 지표입니다. 상관계수는 -1과 1 사이의 값을 가짐

상관관계

- 두 변수 사이의 관계를 나타내는 것입니다. 특히, 한 변수가 변할 때 다른 변수도 함께 변하는 경향이 있을 때 두 변수는 서로 상관관계가 있다고 말합니다. 상관관계는 양의 상관관계와 음의 상관관계로 나뉩니다.
- 두 변수가 함께 변하는 경향성을 의미합니다. 두 변수가 같이 증가하거나, 같이 감소하거나, 한 변수가 증가하면 다른 변수가 감소하는 관계 등을 의미합니다.
- 양의 상관관계는 한 변수가 증가할 때 다른 변수도 증가하는 경우를 의미합니다.
- 음의 상관관계는 한 변수가 증가할 때 다른 변수가 감소하는 경우를 의미합니다. = 상관관계는 두 변수 사이의 관계를 보여주지만 인과관계를 나타내지는 않습니다. 즉, 두 변수 사이에 상관관계가 있을지라도 그 관계가 인과관계를 의미하지는 않을 수 있습니다. 따라서 상관관계를 분석할 때에는 추가적인 분석과 실험을 통해 인과관계를 파악해야 합니다.

종속변수와 독립변수

- 종속변수: 회귀분석에서 예측하고자 하는 변수
- 독립변수: 종속 변수와 관련된 다른 변수들로, 종속 변수의 값에 영향을 미치는 변수

In [ ]:

```
# corr() - 상관계수를 계산하는 함수
# DataFrame 내의 column 간의 pairwise correlation(상관관계)을
# 상관관계는 -1부터 1사이의 값을 가지며, -1에 가까울수록 음의
# 0일 경우에는 서로 독립적인 관계입니다. (상관관계 적음)
# 상관관계가 낮은 것은 제외하고, 상관관계가 높을 것으로 변수

df_sca.corr()
```

	mpg	cylinders	displacement	weigl
mpg	1.000000	-0.775396	-0.804203	-0.831
cylinders	-0.775396	1.000000	0.950721	0.896(
displacement	-0.804203	0.950721	1.000000	0.932{
weight	-0.831741	0.896017	0.932824	1.000(
acceleration	0.420289	-0.505419	-0.543684	-0.417

- 편차(deviation) 편차(deviation)는 데이터가 평균에서 얼마나 떨어져 있는지를 나타내는 값으로, 각 데이터 값에서 평균값을 뺀 값을 의미합니다. 데이터의 개수가 N일 때, 각 데이터의 편차는 다음과 같이 계산됩니다.

편차(deviation) = 각 데이터 - 평균

편차는 데이터가 평균값을 중심으로 얼마나 퍼져 있는지를 나타내며, 평균과 같은 중심 경향성 지표를 계산하는 데에 사용됩니다. 예를 들어, 분산과 표준편차를 계산할 때에는 각 데이터의 편차를 사용합니다.

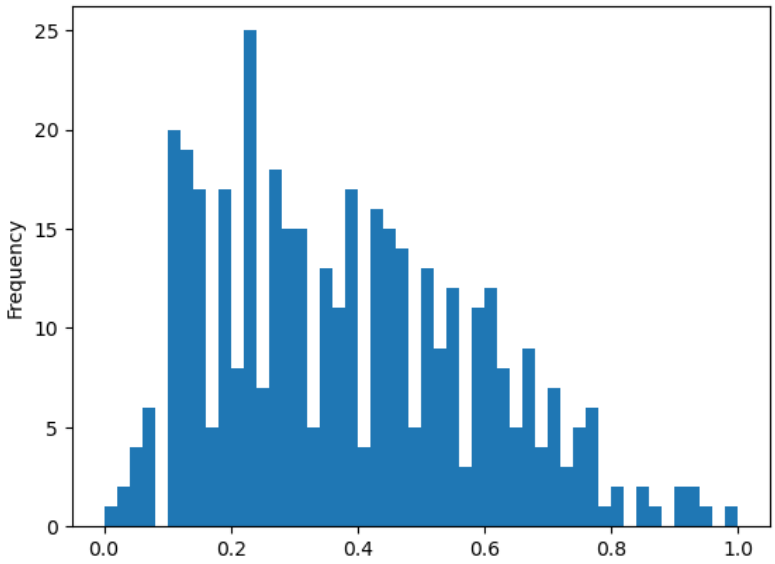
- std(standard deviation)
  - 표준편차, 데이터의 분산 정도, 평균에서 데이터가 얼마나 멀리 흩어져 있는가
  - 분산의 양의 제곱근으로 계산됨
  - 표준 편차가 작을수록 데이터가 평균 주변에 모여있고, 클수록 데이터가 넓게 퍼져있다는 것(흩어져 있음)을 의미함

즉, 각 데이터가 평균에서 얼마나 떨어져 있는지와, 데이터가 평균에서 얼마나 흩어져 있는지의 차이

In [ ]:

```
# plot() - 판다스 라이브러리의 데이터를 시각화하기 위한 메서
# 히스토그램 - 하나의 변수의 분포
# bins는 히스토그램의 구간을 나타내는 파라미터 (binning), 계
# 보통 bins의 값이 크면 데이터의 분포를 더 부드럽게 나타내지

df_sca['mpg'].plot(kind='hist', bins=50)
```

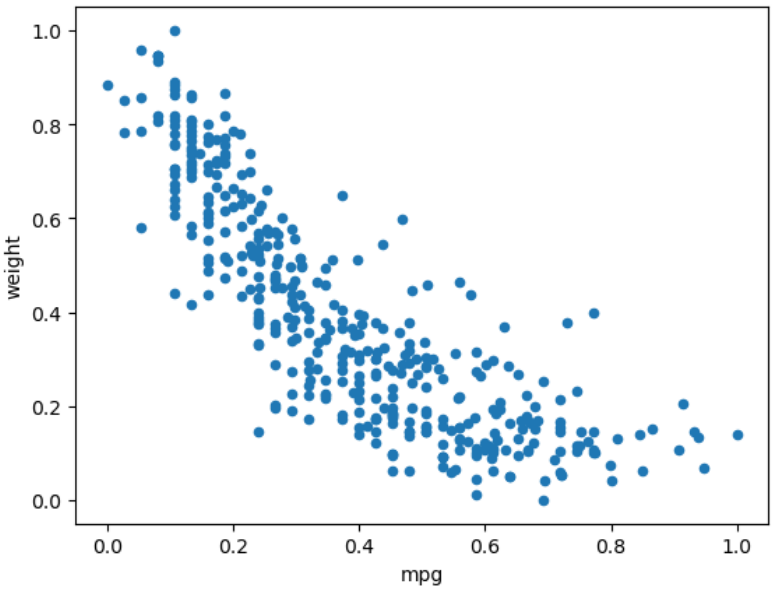
```
<AxesSubplot:ylabel='Frequency'>
```



In [ ]:

```
df_sca[['mpg', 'weight']].plot(kind='scatter', x='mpg', y='w
```

```
<AxesSubplot:xlabel='mpg', ylabel='weight'>
```



In [ ]:

```
기술 통계학, 추측 통계학

확률 - 모집단과 표본집단이 어떻게 나오냐? 확률로 나온다

분포 -> 확률분포 -> 정규분포, 표준정규분포

모집단 - it 업계에 취업하려는 모든 사람들
표본집단 - 우리가 표본, 표본을 뽑는다
분포 - 표본집단으로 모집단을 예측하려면 분포가 같아야 한다

(T분표, 지수분포, ...)

정규분포 - 평균과 표준편차에 의해 결정됨 (종 모양의 대칭 분포
(평균 - 분포의 중심)
(표준편차 - 분포의 폭, 분산의 제곱근임)

표본집단의 분포가 정규분포가 아니면, 정규분포에 가깝게 해줘야

머신러닝, 딥러닝이 들어오면서, 물론 위와 같은 것이 지켜지지만
기계학습, 학습을 통해서 찾아내기 때문에

범주형 데이터 - origin(카테고리, 숫자적 의미보다) - 이런건 O
```

정규분포(Normal Distribution)와 표준정규분포(Standard Normal Distribution)는 모두 확률분포의 종류 중 하나입니다.

정규분포는 평균(mu)과 분산(sigma^2)의 두 매개변수로 특징지어지는 연속확률분포로, 종 모양의 대칭적인 분포를 가집니다. 즉, 평균(mu) 주변에서 대부분의 값이 발생하며, 평균에서 멀어질수록 발생하는 값의 빈도수가 낮아지는 형태를 가지고 있습니다.

반면에, 표준정규분포는 평균(mu)이 0이고, 분산(sigma^2)이 1인 정규분포를 말합니다. 즉, 모든 정규분포의 평균과 분산을 표준화하여 만든 정규분포입니다. 이렇게 표준화된 분포를 이용하면, 평균이 0이고, 표준편차가 1인 표준정규분포 테이블을 이용하여 확률 계산을 용이하게 할 수 있습니다.

따라서, 정규분포와 표준정규분포의 가장 큰 차이점은 평균과 분산이 서로 다르다는 것입니다. 정규분포는 임의의 평균(mu)과 분산(sigma^2)을 가지고 있지만, 표준정규분포는 평균이 0이고, 분산이 1인 고정된 분포입니다.

[과제] auto-mpg.csv 데이터 셋을 탐색한 후 필요한 전처리를 하여 다음 사항을 수행하세요.

- mpg를 예측하는 모델을 생성하기 위한 컬럼들을 선정(3개)하고 그 근거를 기술
- 선정된 3개의 컬럼에 대한 데이터 분포도, mpg와의 상관 관계를 시각화
- 선정된 3개의 컬럼에 대한 통계적 인사이트에 대한 기술

In [ ]:

```
# 참고 - 내가 한 것

# pd.to_numeric()
    # 시리즈나 데이터프레임의 열을 숫자형으로 변환하기 위해
    # error='coerce' 파싱이나 변환이 불가능한 데이터를 NaN으
    # 기본 반환값은 float임
# Numpay 라이브러리의 nan_to_num() - NaN 값을 다른 값으로 대
# np.mean()
# np.where() - Numpy 라이브러리 함수, NaN 값을 찾아냄
# np.isnan() - Numpy 라이브러리 함수, NaN 값에 대하여 True,
    # df.isnull()이 있고, np.usnan()이 있음
# -> True면 두 번째 인자로, False면 세 번째 인자로 반환)


# 1차로 car name은 제외함
import pandas as pd
import numpy as np

df_auto = pd.read_csv('sample_data/auto-mpg.csv')
# df_auto = pd.read_csv('auto-mpg.csv')
print(df_auto)
print(df_auto['horsepower'].unique())

# 문자열 -> 숫자 변경 and 변경 불가능 문자 NaN 처리
df_auto['horsepower'] = pd.to_numeric(df_auto['horsepower'],
df_auto.horsepower.unique()

# 평균값 구하기 위해 nan값을 0으로 바꿈
df_temp = np.nan_to_num(df_auto.horsepower) # , nan=0 없어도

# 평균값 구함
df_auto_mean = np.mean(df_temp)

# nan -> 평균값으로 대체
print(df_auto.horsepower)
# df_auto.horsepower = np.where(np.isnan(df_auto.horsepower)
df_auto.horsepower = df_auto.horsepower.fillna(df_auto_mean)
print(df_auto.horsepower.unique())

# Scaling
from sklearn.preprocessing import MinMaxScaler # 라이브러리-

df_s = df_auto[['mpg','cylinders','displacement','horsepower
scaler= MinMaxScaler()
df_scaler = scaler.fit_transform(df_s)
df_sca = pd.DataFrame(df_scaler, columns=['mpg','cylinders',
df_sca.head()

# 상관계수를 계산하는 함수
print(df_sca.corr())

# 상관관계에 근거하여 mpg, displacement, weight를 선정함
df_s = df_auto[['mpg', 'cylinders', 'displacement', 'weight'

scaler= MinMaxScaler()
df_scaler = scaler.fit_transform(df_s)
df_sca = pd.DataFrame(df_scaler, columns=['mpg', 'cylinders'
```

```
58 df_sca.head()
59
60 df_sca['mpg'].plot(kind='hist', bins=50)
61
62 df_sca[['mpg', 'cylinders']].plot(kind='scatter', x='mpg', y
63
64 df_sca[['mpg', 'displacement']].plot(kind='scatter', x='mpg'
65
66 df_sca[['mpg', 'weight']].plot(kind='scatter', x='mpg', y='w
```

```
      mpg  cylinders  displacement horsepower  we
ight  acceleration  W
0    18.0          8         307.0        130
3504          12.0
1    15.0          8         350.0        165
3693          11.5
2    18.0          8         318.0        150
3436          11.0
3    16.0          8         304.0        150
3433          12.0
4    17.0          8         302.0        140
3449          10.5
..    ...        ...           ...        ...
...           ...
393  27.0          4         140.0         86
2790          15.6
394  44.0          4          97.0         52
```

Q. horsepower 열의 누락 데이터 '?'을 삭제한 후 NaN 값의 갯수를 출력하세요.

In [ ]:

```
import pandas as pd
import numpy as np

auto_df = pd.read_csv('sample_data/auto-mpg.csv')
# auto_df = pd.read_csv('../../dataset/auto-mpg.csv')
# df_auto = pd.read_csv('auto-mpg.csv')
auto_df.head()
```

	mpg	cylinders	displacement	horsepower	weig
0	18.0	8	307.0	130	3504
1	15.0	8	350.0	165	3693
2	18.0	8	318.0	150	3436
3	16.0	8	304.0	150	3433
4	17.0	8	302.0	140	3449

In [ ]:

```
# 뭔 차이?

import pandas as pd
import numpy as np

df = pd.DataFrame({'A': [1, 2, np.nan, 4], 'B': [5, np.nan,
print(df.isnull())
print(np.isnan(df))
```

```
       A      B
0  False  False
1  False   True
2   True  False
3  False  False
       A      B
0  False  False
1  False   True
2   True  False
3  False  False
```

In [ ]:

```
# inplace 파라미터를 사용 or 덮어씌우기

import numpy as np

# auto_df['horsepower'].replace('?', np.nan, inplace = True)
auto_df.horsepower = auto_df.horsepower.replace('?', np.nan)
auto_df.horsepower.unique()
```

```
array([130., 165., 150., 140., 198., 220., 215., 2
25., 190., 170., 160.,
        95.,  97.,  85.,  88.,  46.,  87.,  90., 1
13., 200., 210., 193.,
        nan, 100., 105., 175., 153., 180., 110.,
72.,  86.,  70.,  76.,
        65.,  69.,  60.,  80.,  54., 208., 155., 1
12.,  92., 145., 137.,
       158., 167.,  94., 107., 230.,  49.,  75.,
91., 122.,  67.,  83.,
        78.,  52.,  61.,  93., 148., 129.,  96.,
71.,  98., 115.,  53.,
        81.,  79., 120., 152., 102., 108.,  68.,
58., 149.,  89.,  63.,
        48.,  66., 139., 103., 125., 133., 138., 1
35., 142.,  77.,  62.,
       132.,  84.,  64.,  74., 116.,  82.])
```

In [ ]:

```
# 데이터프레임 또는 시리즈 객체의 각 원소가 NaN(결측치)인지
# isnull() 함수는 NaN인 원소가 있으면 해당 원소를 True로, Na
    # df.isnull()이 있고, np.isnan()이 있음 - DataFrame에는

auto_df.isnull().sum()
```

```
mpg             0
cylinders       0
displacement    0
horsepower      6
weight          0
acceleration    0
model year      0
origin          0
car name        0
dtype: int64
```

In [ ]:

```
auto_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        398 non-null    int64  
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(4), int64(4), object(1)
memory usage: 28.1+ KB
```

In [ ]:

```
# dropna() - 함수는 데이터프레임 또는 시리즈 객체에서
# NaN(결측치) 값을 가지고 있는 행(row) 또는 열(column)을 삭제
# dropna() 함수는 데이터프레임 또는 시리즈 객체의 복사본을 반
# subset은 dropna() 메서드에서 제거할 결측치가 있는 열(colum

auto_df.dropna(subset=['horsepower'], axis =0, inplace=True)
auto_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    int64  
 5   acceleration  392 non-null    float64
 6   model year    392 non-null    int64  
 7   origin        392 non-null    int64  
 8   car name      392 non-null    object 
dtypes: float64(4), int64(4), object(1)
memory usage: 30.6+ KB
```

In [ ]:

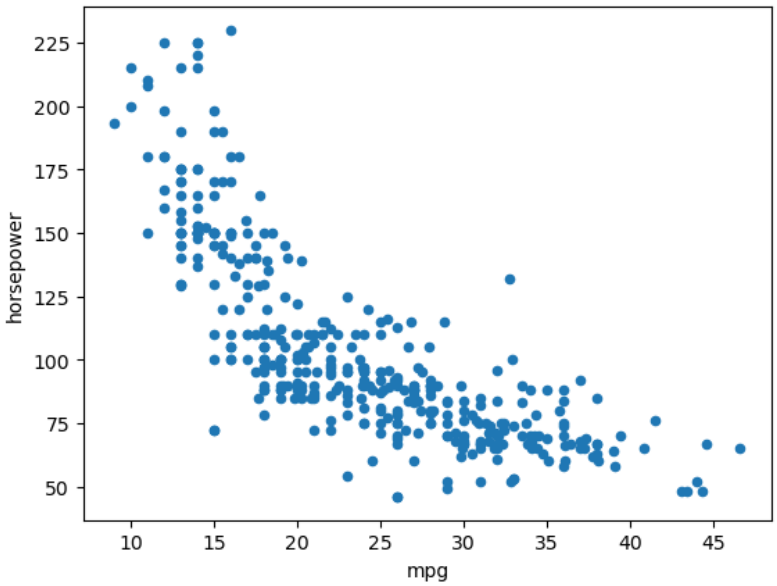
```
### Q. horsepower 문자열을 실수형으로 변환한 후 자료형을 확인

import numpy as np
import pandas as pd
# auto_df['horsepower']=auto_df['horsepower'].astype('float'
auto_df['horsepower']=auto_df.horsepower.astype('float')
auto_df['hp'] = auto_df.horsepower.astype('int')
print()
auto_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    int64  
 5   acceleration  392 non-null    float64
 6   model year    392 non-null    int64  
 7   origin        392 non-null    int64  
 8   car name      392 non-null    object 
 9   hp            392 non-null    int64  
dtypes: float64(4), int64(5), object(1)
memory usage: 33.7+ KB
```

In [ ]:

```
auto_df[['mpg', 'horsepower']].plot(kind='scatter', x='mpg',
```

```
<AxesSubplot:xlabel='mpg', ylabel='horsepower'>
```



In [ ]:

```
auto_df.head()
```

	mpg	cylinders	displacement	horsepower	wei
0	18.0	8	307.0	130.0	3504
1	15.0	8	350.0	165.0	3693
2	18.0	8	318.0	150.0	3436
3	16.0	8	304.0	150.0	3433
4	17.0	8	302.0	140.0	3449

In [ ]:

```
# value_counts() - pandas 라이브러리에서 제공하는 메서드 중
# 시리즈(Series) 객체에서 각 값(value)의 등장 횟수(count)를
# 내림차순으로 나옴

auto_df.origin.value_counts()
```

```
1    245
3     79
2     68
Name: origin, dtype: int64
```

Q. 아래 사항을 처리 하세요

- origin 열의 고유값을 출력하세요.
- 정수형 데이터를 문자형 데이터로 변환한 후 고유값을 출력하세요.

In [ ]:

```
# origin 열의 고유값 확인
# print(auto_df['origin'].unique())
print(auto_df.origin.unique())

# 정수형 데이터를 문자형 데이터로 변환
auto_df['origin'].replace({1:'USA', 2:'EU', 3:'JAPAN'}, inpl

print(auto_df['origin'].unique())
```

```
[1 3 2]
['USA' 'JAPAN' 'EU']
```

In [ ]:

```
# Q. origin 열의 자료형을 확인하고, 범주형으로 변환하여 출력

# origin의 int는 숫자로 의미가 없음, 범주형(category)으로 바

# astype() - pandas 라이브러리에서 제공하는 메서드 중 하나로
# 데이터프레임(DataFrame)이나 시리즈(Series)의 데이터 타입(d

# dtypes - pandas 라이브러리에서 제공하는 속성 중 하나로,
# 데이터프레임의 각 열(column)에 대한 데이터 타입(data type)

auto_df['origin'] = auto_df.origin.astype('category')
print(auto_df.origin.dtypes)
print()
print(auto_df['model year'].head(3))
print()
auto_df['model year'] = auto_df['model year'].astype('catego
print(auto_df['model year'].head(3))
```

```
category

0    70
1    70
2    70
Name: model year, dtype: int64

0    70
1    70
2    70
Name: model year, dtype: category
Categories (13, int64): [70, 71, 72, 73, ..., 79,
80, 81, 82]
```

In [ ]:

```
auto_df.head()
```

	mpg	cylinders	displacement	horsepower	wei
0	18.0	8	307.0	130.0	3504
1	15.0	8	350.0	165.0	3693
2	18.0	8	318.0	150.0	3436
3	16.0	8	304.0	150.0	3433
4	17.0	8	302.0	140.0	3449

to_pickle() 과 read_pickle()

df.to_pickle() 메서드는 Pandas DataFrame을 Pickle 파일로 저장하는 메서드입니다. Pickle 파일은 파이썬의 객체를 바이트 형태로 직렬화하여 저장하는 방식으로, 객체를 저장할 때 유용하게 사용

In [ ]:

```
# to_pickle()은 pandas 라이브러리에서 제공하는 메서드 중 하나
# 데이터프레임(DataFrame)이나 시리즈(Series) 객체를 pickle 파

auto_df.to_pickle('auto_df.pkl')
auto_df=pd.read_pickle('auto_df.pkl')
auto_df.head()
```

	mpg	cylinders	displacement	horsepower	weig
0	18.0	8	307.0	130.0	3504
1	15.0	8	350.0	165.0	3693
2	18.0	8	318.0	150.0	3436
3	16.0	8	304.0	150.0	3433
4	17.0	8	302.0	140.0	3449

In [ ]:

```
# Q. 'mpg'를 'kpl'로 환산하여 새로운 열을 생성하고 처음 3개
# 반올림하여 출력하세요.
# mile : km = 1 : 0.425144

mpg_to_kpl = 0.425144
auto_df['kpl'] = auto_df['mpg'] * round(mpg_to_kpl, 2)
auto_df[['mpg', 'kpl']].head(3)
```

	mpg	kpl
0	18.0	7.74
1	15.0	6.45
2	18.0	7.74

In [ ]:

```
auto_df
```

	mpg	cylinders	displacement	horsepower	we
0	18.0	8	307.0	130.0	35
1	15.0	8	350.0	165.0	36
2	18.0	8	318.0	150.0	34
3	16.0	8	304.0	150.0	34
4	17.0	8	302.0	140.0	34
...	...	...	...	...	...
393	27.0	4	140.0	86.0	27
394	44.0	4	97.0	52.0	21
395	32.0	4	135.0	84.0	22
396	28.0	4	120.0	79.0	26
397	31.0	4	119.0	82.0	27

392 rows × 11 columns

범주형(카테고리) 데이터 처리

In [ ]:

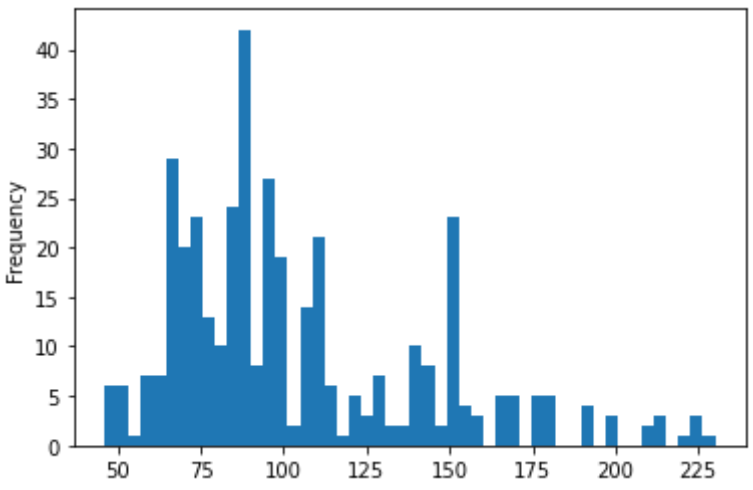
```
auto_df.horsepower.unique()
```

```
array([130., 165., 150., 140., 198., 220., 215., 2
25., 190., 170., 160.,
        95.,  97.,  85.,  88.,  46.,  87.,  90., 1
13., 200., 210., 193.,
       100., 105., 175., 153., 180., 110.,  72.,
86.,  70.,  76.,  65.,
        69.,  60.,  80.,  54., 208., 155., 112.,
92., 145., 137., 158.,
       167.,  94., 107., 230.,  49.,  75.,  91., 1
22.,  67.,  83.,  78.,
        52.,  61.,  93., 148., 129.,  96.,  71.,
98., 115.,  53.,  81.,
        79., 120., 152., 102., 108.,  68.,  58., 1
49.,  89.,  63.,  48.,
        66., 139., 103., 125., 133., 138., 135., 1
42.,  77.,  62., 132.,
        84.,  64.,  74., 116.,  82.])
```

In [ ]:

```
# horsepower를 3개의 구간으로 구분
# bins (binning - 구간화)

auto_df.horsepower.plot(kind='hist',bins=50)
```

```
<Axes: ylabel='Frequency'>
```



In [ ]:

```
# 숫자 데이터라 하더라도 범주화할 필요도 있다
# np.histogram - 데이터를 히스토그램으로 시각화하기 위해 사용
# -> 시각화는 Matplotlib, plot() 등 이용
# 값은 1차원 배열 형태로 입력(시리즈)
# count - np.histogram 함수가 계산한, 각 계급 구간(bin)에 속
# pd.cut()
    # 연속적인 수치형 데이터를 일정한 계급 구간(bin)으로 나누
    # x - 변환하고자 하는 1차원 배열 또는 시리즈
    # bins - 데이터를 나누고자 하는 구간(bin)을 나타내는 값들
    # labels - bins에 지정한 각 계급 구간(bin)에 대해 사용할
    # include_lowest - 구간(bin)의 최소값을 구간에 포함할지

# 46~107 107~168 168~230 3개 구간임

count, bin_dividers = np.histogram(auto_df.horsepower, bins=
print(count, 'Wn', bin_dividers)

bin_names = ['저출력', '보통출력', '고출력']
auto_df['hp_bin'] = pd.cut(x=auto_df.horsepower,
                           bins = bin_dividers,
                           labels = bin_names,
                           include_lowest=True) # 첫 경계값 포

auto_df[['horsepower', 'hp_bin']].head()
```

```
[257 103  32]
 [ 46.         107.33333333 168.66666667 230.
]
```

	horsepower	hp_bin
0	130.0	보통출력
1	165.0	보통출력
2	150.0	보통출력
3	150.0	보통출력
4	140.0	보통출력

In [ ]:

```
# Q. horsepower 컬럼을 5개로 범주화 하세요.
# EH, H, M, L, EL

auto_df = pd.read_pickle('auto_df.pkl')
auto_df.head(3)

count, bin_dividers = np.histogram(auto_df.horsepower, bins=
bin_names = ['EH', 'H', 'M', 'L', 'EL']
auto_df['hp_bin'] = pd.cut(x=auto_df.horsepower,
                           bins = bin_dividers,
                           labels = bin_names,
                           include_lowest = True)

auto_df[['horsepower', 'hp_bin']].head()
```

	horsepower	hp_bin
0	130.0	M
1	165.0	L
2	150.0	M
3	150.0	M
4	140.0	M

In [ ]:

```
1
```

원핫 인코딩 (One-Hot Encoding)

- 범주형 데이터를 수치형 데이터로 변환하는 방법 중 하나
- 카테고리를 나타내는 범주형 데이터를 회귀분석 등 ml 알고리즘에 바로 사용할 수 없는 경우
- 컴퓨터가 인식 가능한 입력값으로 변환 -> 숫자 0 또는 1로 표현되는 더미 변수 사용.
- 0, 1은 크고 작음과 상관없고 어떤 특성 여부만 표시(존재 1, 비존재 0) -> one-hot-encoding(one hot vector로 변환한다는 의미)

- 예시>
  - 색상 - 빨강, 파랑, 노랑이 있으면, 대부분의 머신러닝 알고리즘은 수치형 데이터만 다루기 때문에, 이 문자열 값들을 숫자로 변환해야 합니다. 원-핫 인코딩은 이러한 문자열 값을 0 또는 1로 이루어진 벡터로 변환합니다

In [ ]:

```
# pd.get_dummies - 함수는 범주형(categorical) 데이터를 원-핫
# pd.get_dummies(data, columns)
    # data - 원본 데이터 프레임
    # columns - 원-핫 인코딩을 수행할 열 이름 또는 열 이름 리

auto_df = pd.read_pickle('auto_df.pkl')
auto_df = pd.get_dummies(auto_df, columns=['origin'])
auto_df[['origin_EU', 'origin_JAPAN', 'origin_USA']].tail(5)
```

	origin_EU	origin_JAPAN	origin_USA
393	0	0	1
394	1	0	0
395	0	0	1
396	0	0	1
397	0	0	1

정규화

- 각 변수의 숫자 데이터의 상대적 크기 차이 때문에 ml 분석 결과가 달라질 수 있음. (a변수 1 ~ 1000, b변수 0 ~ 1)
- 숫자 데이터의 상대적 크기 차이를 제거할 필요가 있으며 각열(변수)에 속하는 데이터 값을 동일한 크기 기준으로 나누어 정규화 함.
- 정규화 결과 데이터의 범위는 0 ~ 1 또는 -1 ~ 1(음수값이 있는 경우).
- 각 열의 값 / 최대값 or (각 열의 값 - 최소값) / (해당 열의 최대값 - 최소값)

표준화

- 평균이 0이고 분산(or 표준편차)이 1인 가우시안 표준 정규분포를 가진 값으로 변환하는 것

정규화(Normalization)와 표준화(Standardization)는 모두 데이터의 값 범위를 변경하여 모델의 학습을 원활하게 만드는 기법입니다. 하지만 둘은 목적과 방법이 다릅니다.

정규화(Normalization) 정규화는 데이터의 값을 일정한 범위로 변환하는 기법입니다. 주로 0과 1 사이의 값으로 변환하는 Min-Max 스케일링 방법이 많이 사용됩니다. 이를 통해 데이터의 분포를 일정한 범위 내에서 비교할 수 있습니다.

예를 들어, 데이터의 최소값을 0, 최대값을 1로 설정하여 모든 데이터를 일정한 범위 내로 변환할 수 있습니다. 이때, 각 데이터가 가진 상대적인 크기 비교는 유지됩니다.

표준화(Standardization) 표준화는 데이터를 평균이 0, 표준편차가 1인 정규 분포로 변환하는 기법입니다. 이를 통해 데이터의 분포를 평균값을 중심으로, 표준편차를 기준으로 비교할 수 있습니다.

예를 들어, 데이터의 평균값을 0, 표준편차를 1로 설정하여 모든 데이터를 일정한 범위 내로 변환할 수 있습니다. 이때, 각 데이터가 가진 상대적인 위치는 달라질 수 있습니다.

따라서, 정규화는 데이터 분포의 범위를 일정하게 조정하여 비교하는 기법이며, 표준화는 데이터 분포의 평균값과 표준편차를 이용하여 정규분포로 변환하여 비교하는 기법입니다.

In [ ]:

```
# 정규화 - 수작업

auto_df = pd.read_pickle('auto_df.pkl')
print(auto_df.horsepower.describe(), '₩n')

auto_df.horsepower = auto_df.horsepower / auto_df.horsepower
# auto_df.horsepower = (auto_df.horsepower - auto_df.horsepo
print(auto_df.horsepower.describe())
```

```
count    392.000000
mean     104.469388
std       38.491160
min       46.000000
25%       75.000000
50%       93.500000
75%      126.000000
max      230.000000
Name: horsepower, dtype: float64

count    392.000000
mean       0.454215
std        0.167353
min        0.200000
25%        0.326087
50%        0.406522
75%        0.547826
max        1.000000
Name: horsepower, dtype: float64
```

전처리, 모델 생성 및 평가

1. 전처리
   - 데이터셋을 로드하고, 컬럼명을 지정합니다.
   - horsepower 열의 결측값인 ?을 NaN으로 변경하고, 이를 포함하는 행을 삭제합니다.
   - car name 열을 삭제합니다.
   - 범주형 데이터인 origin을 one-hot 인코딩합니다. (보통 문자로 돼있음 -> 이걸 숫자로 바꾸는 것, 그래야 알로리즘에 넣어줄 수 있음)
   - 데이터를 입력 변수 X와 출력 변수 y로 분할합니다. (y는 mpg)

- 입력 변수 X를 정규화합니다. (입력 변수 = 독립 변수)

2. 모델 학습

- 학습 데이터셋과 테스트 데이터셋으로 데이터를 분할합니다.
- 선형 회귀 모델을 생성하고, 학습 데이터셋으로 모델을 학습합니다.

3. 모델 성능 평가

- 테스트 데이터셋으로 모델을 예측하고, R^2 점수를 계산하여 모델의 성능을 평가합니다.
- mpg를 예측하는 선형 회귀 모델의 성능 평가
  - R^
  - MSE (Mean Squared Error)와 RMSE (Root Mean Squared Error)
    - MSE는 예측값과 실제값의 차이를 제곱한 값들의 평균이고, RMSE는 MSE의 제곱근입니다.

※ R^2는 결정 계수(Coefficient of Determination)를 의미합니다. 결정 계수는 회귀 분석에서 독립 변수(X)가 종속 변수(Y)에 대해 얼마나 잘 설명하는지를 나타내는 지표로, 0에서 1사이의 값을 가집니다. (100점 만점)

R^2의 값이 1에 가까울수록 회귀 모델이 데이터를 더 잘 설명한다는 의미입니다. 반대로 R^2의 값이 0에 가까울수록 모델이 데이터를 설명하는 능력이 낮다는 것을 의미합니다. R^2의 값이 음수일 수도 있는데, 이는 모델이 데이터보다 더 나쁘게 예측한 경우입니다.

R^2는 다른 성능 평가 지표와 함께 모델의 성능을 종합적으로 평가할 수 있습니다. 그러나 R^2는 독립 변수와 종속 변수 간의 관계를 단순화하므로, 모델의 성능 평가만으로는 충분하지 않을 수 있습니다. 따라서 다른 성능 평가 지표와 함께 사용하여 모델의 성능을 정확히 평가하는 것이 중요합니다.

In [ ]:

```
# car name 컬럼 삭제 - 참고(나)

# new_columns = ['mpg', 'cylinders', 'displacement', 'horsep
#            'origin']

# df = df.reindex(new_columns, axis = 1)

df = df.drop('car name', axis=1)

df
```

In [1]:

```
# names
    # CSV 파일에서 컬럼명으로 사용할 값들을 리스트 형태로 입
    # 이 인자를 생략하면 CSV 파일의 첫번째 행이 컬럼명으로 사
# delim_whitespace
    # 이 인자가 True인 경우, 파일을 공백(띄어쓰기, 탭 등)으로
    # 이 인자를 생략하거나 False로 지정하면 파일을 쉼표(,)로
    # delim은 delimiter의 축약어로, 구분자라는 의미를 가지고
    # CSV 파일에서 각 열(column)들은 쉼표(,) 또는 탭(tab) 등
    # 이러한 문자들을 delimiter 또는 separator라고 합니다. (
    # delim_whitespace=True는 CSV 파일의 구분자를 공백으로 사
    # 이렇게 공백을 구분자로 사용하는 CSV 파일을 TSV(Tab Sep

import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# 데이터셋 url
url = "https://archive.ics.uci.edu/ml/machine-learning-datab

# 컬럼명 지정 - df.info()로 컬럼 개수 알 수 있음
columns = ['mpg', 'cylinders', 'displacement', 'horsepower',
           'origin', 'car name']

# 데이터 가져오기
df = pd.read_csv(url, names=columns, delim_whitespace=True)

# horsepower 열의 결측값 '?'을 NaN으로 변경하고, 이를 포함하
df = df.replace('?', np.nan)
df = df.dropna()

print(df.horsepower.unique())

# 'car name' 열 삭제
df = df.drop('car name', axis=1)

# 범주형 데이터인 'origin'을 one-hot 인코딩
df = pd.get_dummies(df, columns=['origin'])

# 데이터 분할
X = df.drop('mpg', axis=1)
y = df['mpg']

# 정규화
scaler = StandardScaler()   # 객체 만들기
X = scaler.fit_transform(X)     # X를 표준화해 줌

# 결과 확인
print(X[:5])
print(y[:5])
```

```
['130.0' '165.0' '150.0' '140.0' '198.0' '220.0'
'215.0' '225.0' '190.0'
 '170.0' '160.0' '95.00' '97.00' '85.00' '88.00'
'46.00' '87.00' '90.00'
 '113.0' '200.0' '210.0' '193.0' '100.0' '105.0'
'175.0' '153.0' '180.0'
 '110.0' '72.00' '86.00' '70.00' '76.00' '65.00'
'69.00' '60.00' '80.00'
 '54.00' '208.0' '155.0' '112.0' '92.00' '145.0'
'137.0' '158.0' '167.0'
 '94.00' '107.0' '230.0' '49.00' '75.00' '91.00'
'122.0' '67.00' '83.00'
 '78.00' '52.00' '61.00' '93.00' '148.0' '129.0'
'96.00' '71.00' '98.00'
 '115.0' '53.00' '81.00' '79.00' '120.0' '152.0'
'102.0' '108.0' '68.00'
 '58.00' '149.0' '89.00' '63.00' '48.00' '66.00'
'139.0' '103.0' '125.0'
 '133.0' '138.0' '135.0' '142.0' '77.00' '62.00'
'132.0' '84.00' '64.00'
 '74.00' '116.0' '82.00']
[[ 1.48394702  1.07728956  0.66413273  0.62054034
-1.285258   -1.62531533
   0.77459667 -0.45812285 -0.50239045]
 [ 1.48394702  1.48873169  1.57459447  0.84333403
-1.46672362 -1.62531533
   0.77459667 -0.45812285 -0.50239045]
 [ 1.48394702  1.1825422   1.18439658  0.54038176
-1.64818924 -1.62531533
   0.77459667 -0.45812285 -0.50239045]
 [ 1.48394702  1.04858429  1.18439658  0.53684535
-1.285258   -1.62531533
   0.77459667 -0.45812285 -0.50239045]
 [ 1.48394702  1.02944745  0.92426466  0.5557062
-1.82965485 -1.62531533
   0.77459667 -0.45812285 -0.50239045]]
0    18.0
1    15.0
2    18.0
3    16.0
4    17.0
Name: mpg, dtype: float64
```

[과제] 사용자 함수를 이용하여 재작성 하세요.

In [6]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler

# 데이터셋 url
url = "https://archive.ics.uci.edu/ml/machine-learning-datab

# 컬럼명 지정 - df.info()로 컬럼 개수 알 수 있음
columns = ['mpg', 'cylinders', 'displacement', 'horsepower',
           'origin', 'car name']

# 데이터 가져오기
df = pd.read_csv(url, names=columns, delim_whitespace=True)

# horsepower 열의 결측값 '?'을 NaN으로 변경하고, 이를 포함하
df = df.replace('?', np.nan)
df = df.dropna()
df['horsepower']=df.horsepower.astype('float')

# 'car name' 열 삭제
df = df.drop('car name', axis=1)

# 범주형 데이터인 'origin'을 one-hot 인코딩
df = pd.get_dummies(df, columns=['origin'])

# 데이터 분할
X = df.drop('mpg', axis=1)
y = df['mpg']

# 정규화
scaler = StandardScaler()   # 객체 만들기
X = scaler.fit_transform(X)     # X를 표준화해 줌

# 결과 확인
print(X[:5])
print(y[:5])
```

```
[130. 165. 150. 140. 198. 220. 215. 225. 190. 170.
160.  95.  97.  85.
  88.  46.  87.  90. 113. 200. 210. 193. 100. 105.
175. 153. 180. 110.
  72.  86.  70.  76.  65.  69.  60.  80.  54. 208.
155. 112.  92. 145.
 137. 158. 167.  94. 107. 230.  49.  75.  91. 122.
67.  83.  78.  52.
  61.  93. 148. 129.  96.  71.  98. 115.  53.  81.
79. 120. 152. 102.
 108.  68.  58. 149.  89.  63.  48.  66. 139. 103.
125. 133. 138. 135.
 142.  77.  62. 132.  84.  64.  74. 116.  82.]
[[ 1.48394702  1.07728956  0.66413273  0.62054034
-1.285258   -1.62531533
   0.77459667 -0.45812285 -0.50239045]
 [ 1.48394702  1.48873169  1.57459447  0.84333403
-1.46672362 -1.62531533
   0.77459667 -0.45812285 -0.50239045]
 [ 1.48394702  1.1825422   1.18439658  0.54038176
-1.64818924 -1.62531533
   0.77459667 -0.45812285 -0.50239045]
 [ 1.48394702  1.04858429  1.18439658  0.53684535
-1.285258   -1.62531533
   0.77459667 -0.45812285 -0.50239045]
 [ 1.48394702  1.02944745  0.92426466  0.5557062
-1.82965485 -1.62531533
   0.77459667 -0.45812285 -0.50239045]]
0    18.0
1    15.0
2    18.0
3    16.0
4    17.0
Name: mpg, dtype: float64
```

In [7]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64  
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    float64
 5   acceleration  392 non-null    float64
 6   model year    392 non-null    int64  
 7   origin_1      392 non-null    uint8  
 8   origin_2      392 non-null    uint8  
 9   origin_3      392 non-null    uint8  
dtypes: float64(5), int64(2), uint8(3)
memory usage: 25.6 KB
```

In [1]:

```
import pandas as pd
import numpy as np
# from sklearn.preprocessing import StandardScaler

# 함수 정의
def preprocess_data(url):
    # 데이터 가져오기
    df = pd.read_csv(url, delim_whitespace=True, header = Nc

    # 컬럼명 지정 - df.info()로 컬럼 개수 알 수 있음
    columns = ['mpg', 'cylinders', 'displacement', 'horsepow
               'origin', 'car name']
    df.columns = columns

    # horsepower 열의 결측값 '?'을 NaN으로 변경하고, 이를 포
    df = df.replace('?', np.nan)
    df = df.dropna()
    df['horsepower']=df.horsepower.astype('float')

    # 'car name' 열 삭제
    df = df.drop('car name', axis=1)

    # 범주형 데이터인 'origin'을 one-hot 인코딩
    df = pd.get_dummies(df, columns=['origin'])

    return df

# 함수 호출
url = "https://archive.ics.uci.edu/ml/machine-learning-datab
df = preprocess_data(url)

# 데이터 확인
print(df.head())


# # 데이터 분할
# X = df.drop('mpg', axis=1)
# y = df['mpg']

# # 정규화
# scaler = StandardScaler()   # 객체 만들기
# X = scaler.fit_transform(X)     # X를 표준화해 줌

# # 결과 확인
# print(X[:5])
# print(y[:5])
```

```
    mpg  cylinders  displacement  horsepower  weig
ht  acceleration  W
0  18.0          8         307.0       130.0  350
4.0          12.0
1  15.0          8         350.0       165.0  369
3.0          11.5
2  18.0          8         318.0       150.0  343
6.0          11.0
3  16.0          8         304.0       150.0  343
3.0          12.0
4  17.0          8         302.0       140.0  344
9.0          10.5

   model year  origin_1  origin_2  origin_3
0          70         1         0         0
1          70         1         0         0
2          70         1         0         0
3          70         1         0         0
4          70         1         0         0
```

In [13]:

```
1
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 392 entries, 0 to 397
Data columns (total 10 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   mpg           392 non-null    float64
 1   cylinders     392 non-null    int64
 2   displacement  392 non-null    float64
 3   horsepower    392 non-null    float64
 4   weight        392 non-null    float64
 5   acceleration  392 non-null    float64
 6   model year    392 non-null    int64
 7   origin_1      392 non-null    uint8
 8   origin_2      392 non-null    uint8
 9   origin_3      392 non-null    uint8
dtypes: float64(5), int64(2), uint8(3)
memory usage: 25.6 KB
```

In [ ]:

```
# 평균, 표준편차

print(X.mean())     # 평균 0
print(X.std())      # 표준편차 1
```

```
-1.409807015397024e-16
1.0
```

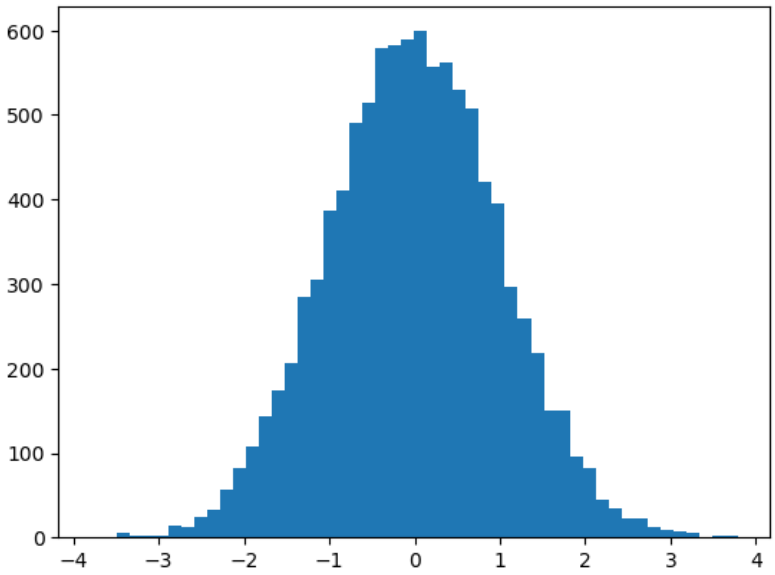
In [ ]:

```
# 모델 학습 및 평가

import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error

# 데이터 분할 (학습 데이터셋과 테스트 데이터셋) - test sampl
X_train, X_test, y_train, y_test = train_test_split(X, y, te

# 다중 선형 회귀 모델 생성
lr_model = LinearRegression()

# 모델 학습
lr_model.fit(X_train, y_train)

# 모델 예측
y_pred = lr_model.predict(X_test)

# R^2 평가 (r2 score - 얼마나 잘 맞는지 설명)
r2 = r2_score(y_test, y_pred)

# MSE 평가 (실제값과 예측값 간의 오차)
mse = mean_squared_error(y_test, y_pred)

# RMSE 평가 (정확한 오차는 이걸로 알게 됨)
rmse = np.sqrt(mse)

# 결과 확인
print("R^2 Score: ", r2) # 79프로
print("MSE: ", mse)
print("RMSE: ", rmse)
```

```
R^2 Score:  0.792277471402258
MSE:  10.602279011688372
RMSE:  3.256114096847402
```

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

# 평균이 0이고, 표준편차가 1인 표준정규분포를 따르는 데이터
data = np.random.normal(0, 1, 10000)

# 히스토그램 그리기
plt.hist(data, bins=50)
plt.show()

```



In [ ]:

```
카테고리로 바꿔 놨는지
그 다음 horsepower 범주화하는지
origin도 체크

```

In [7]:

```
# 연습

# 모듈 불러오기
import pandas as pd
import numpy as np
# from sklearn.preprocessing import StandardScaler

# 함수 정의
def preprocess_data(url):

    # 데이터 불러오기
    df = pd.read_csv(url, delim_whitespace = True, header =

    # 컬럼 정하기
    df.columns = ['mpg', 'cylinders', 'displacement', 'horse
#     columns =
#     df.columns = columns

    # 전처리 위해 데이터 살피기
    df
    df.info()
    df.horsepower.unique()

    # '?' -> NaN으로 바꿔주기
    df.replace('?', np.nan, inplace = True)

    # NaN이 속한 행 삭제하기
    df.dropna()

    # str -> int형으로 바꿔주기
    df['horsepower'] = df.horsepower.astype('float') # inpla
    df.horsepower = df.horsepower.astype('float')

    # 불필요한 data, 'car name' 삭제하기
    df.drop('car name', axis = 1, inplace = True) # drop이 ㅎ

    # 좀 더 의미있게 horsepower -> category, 범주화 하기
    count, bin_dividers = np.histogram(df.horsepower, bins =
    bin_names = ['저출력', '보통출력', '고출력']
    df['hp_bin'] = pd.cut(x = df.horsepower,
                          bins = bin_dividers,
                          labels = bin_names,
                          include_lowest = True)

    # origin을 알고리즘에 적용할 수 있도록 one-hot encoding
    pd.get_dummies(df, columns=['origin'], inplace = True) #
#     df = pd.get_dummies(df, columns=['origin'])

    return df

# 데이터 url
url = "https://archive.ics.uci.edu/ml/machine-learning-datab

# 함수 호출
df = preprocess_data(url)

# 데이터 확인
```

```
print(df.head())

# 데이터 분할
X = df.drop('mpg', axis=1)
y = df['mpg']

# 정규화
scaler = StandardScaler()   # 객체 만들기
X = scaler.fit_transform(X)     # X를 표준화해 줌

# 결과 확인
print(X[:5])
print(y[:5])
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   mpg           398 non-null    float64
 1   cylinders     398 non-null    int64  
 2   displacement  398 non-null    float64
 3   horsepower    398 non-null    object 
 4   weight        398 non-null    float64
 5   acceleration  398 non-null    float64
 6   model year    398 non-null    int64  
 7   origin        398 non-null    int64  
 8   car name      398 non-null    object 
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

```
---------------------------------------------
-----------------------------
ValueError                                Traceba
ck (most recent call last)
~WAppDataWLocalWTempWipykernel_5264W363949133
6.py in <module>
     53
     54 # 함수 호출
---> 55 df = preprocess_data(url)
     56
     57 # 데이터 확인

~WAppDataWLocalWTempWipykernel_5264W363949133
6.py in preprocess_data(url)
     36
     37     # 좀 더 의미있게 horsepower -> cate
gory, 범주화 하기
---> 38     count, bin_dividers = np.histogram(d
f.horsepower, bins = 3)
     39     bin_names = ['저출력', '보통출력',
'고출력']
     40     df['hp_bin'] = pd.cut(x = df.horsepowe
r,

<__array_function__ internals> in histogram(*a
rgs, **kwargs)

C:WappWlibWsite-packagesWnumpyWlibWhistogram
s.py in histogram(a, bins, range, normed, weig
hts, density)
    791     a, weights = _ravel_and_check_weights
(a, weights)
    792
--> 793     bin_edges, uniform_bins = _get_bin_ed
ges(a, bins, range, weights)
    794
    795     # Histogram is an integer or a flo
at array depending on the weights.

C:WappWlibWsite-packagesWnumpyWlibWhistogram
s.py in _get_bin_edges(a, bins, range, weights)
    424             raise ValueError('`bins` mus
t be positive, when an integer')
    425
```

```
--> 426         first_edge, last_edge = _get_out
er_edges(a, range)
    427
    428     elif np.ndim(bins) == 1:
```

In [ ]:

```
1
```

```
~\AppData\Local\Programs\Python\Python39\lib\site-packages\numpy\lib\histogram
s.py in _get_outer_edges(a, range)
    321         first_edge, last_edge = a.min(),
a.max()
    322         if not (np.isfinite(first_edge) a
nd np.isfinite(last_edge)):
--> 323             raise ValueError(
```