Data Mining

판다스 (Pandas)

학습 목표

- 데이터를 처리하고 분석하기 위한 라이브러리인 Pandas의 기능에 대해 살펴본다.

주요 내용

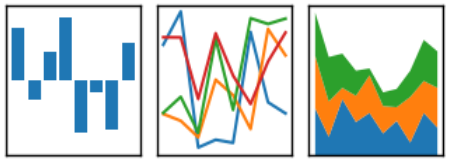
- 판다스 (Pandas)
- 시리즈 (Series)
- 데이터프레임 (DataFrame)
- 데이터 탐색
- 데이터 전처리

1. 판다스 (Pandas)

판다스 (Pandas)

pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$



데이터 구조를 분석하기 위한 라이브러리

- **테이블 구조의 데이터**를 **인덱스**를 통해 다루는 방식
- 다양한 파일 I/O : **CSV**, **텍스트 파일**, **Excel**, **SQL DB**, **HDF5** 등
- 데이터 객체 연산 시 **인덱스 결합 방식**으로 처리
- 유연한 데이터 구조 변환 및 **통계 요약**
- 대량 데이터의 **슬라이싱**, **인덱싱**, **부분집합** 처리
- **용이한 컬럼 추가** 방식
- **Group by** 엔진으로 데이터의 요약 및 변환
- 고성능으로 데이터셋을 **결합 (Merge and Joining)**
- **계층적 인덱싱 (Hierarchical axis indexing)**
- **시계열** 기능 : 날짜 범위 생성 및 빈도 변환, 이동 통계량, 날짜 이동 및 지연, 다른 시간 간격을 갖는 시계열 결합
- 주요 코드 부분이 Cython 또는 C로 작성되어 **성능이 최적화** 됨

테이블 형태의 자료 구조

데이터 프레임(DataFrame)**은 테이블 형태의 데이터를 표현하며 '시리즈**(Series)**의 시리즈 ' 구조이다.**

열 시리즈

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
393	27.0	4	140.0	86.0	2790.0	15.6	82	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	2
395	32.0	4	135.0	84.0	2295.0	11.6	82	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	1

열 인덱스

행 시리즈 →

행 인덱스

dataframe = {'MPG' : {...}, 'Cylinders' : {....}, ..., 'Weight' " {...}, ...}

- **시리즈** (Series) : (인덱스, 값) 구조로 되어 있는 데이터 구조 (딕셔너리와 유사)
- **데이터 프레임** (DataFrame) : 열의 시리즈로 구성된 데이터 구조, 열마다 데이터 타입이 다를 수 있음

2. 시리즈 (Series)

시리즈 생성

시리즈는 어떤 넘파이 타입도 저장이 가능

리스트로 생성

```
obj = pd.Series([3,5,-2,1])
obj
```

인덱스 데이터 값

```
0    3
1    5
2   -2
3    1
dtype: int64
```

- Obj는 (Index, Value)로 구성

딕셔너리로 생성

```
data = {'a': 30, 'b': 70, 'c': 160, 'd': 5}
obj = pd.Series(data)
obj
```

```
a     30
b     70
c    160
d      5
dtype: int64
```

- 딕셔너리의 key가 인덱스가 됨

값과 인덱스 조회

값 조회

```
obj.values
```

```
array([ 30,  70, 160,   5])
```

인덱스 조회

```
obj.index
```

```
Index(['a', 'b', 'c', 'd'], dtype='object')
```

인덱스 지정

시리즈를 생성할 때 인덱스를 리스트로 지정할 수 있다.

```
data = {'a': 30, 'b': 70, 'c': 160, 'd': 5}
index = ['a','b','c','d','g']
obj = pd.Series(data, index=index)
obj
```

```
a     30.0
b     70.0
c    160.0
d      5.0
g      NaN
dtype: float64
```

인덱스는 있는데 데이터가 없는 경우

- 인덱스에 해당하는 데이터가 없으면 NaN (Not a Number)로 표시됨

시리즈 연산

산술 연산

```
obj *2
```

```
a     60
b    140
c    320
d     10
dtype: int64
```

쿼리 연산

```
obj[obj>2]
```

```
a     30
b     70
c    160
d      5
dtype: int64
```

Series 연산은 NumPy 연산과 동일하다.

시리즈 연산

시리즈 간에 연산을 할 때는 같은 인덱스를 갖는 항목을 찾아서 연산한다.

```
student1 = pd.Series({'국어':100, '영어':80,'수학':90})
student2 = pd.Series({'수학':80, '국어':90, '영어':80})
print(student1)
print('\n')
print(student2)
```

```
[Output]
국어   100
영어    80
수학    90
dtype: int64

수학    80
국어    90
영어    80
dtype: int64
```

```
add_data = student1 + student2 # 덧셈
sub_data = student1 - student2 # 뺄셈
mul_data = student1 * student2 # 곱셈
div_data = student1 / student2 # 나눗셉
# 결과를 데이터 프레임으로 합치기 (시리즈 -> 데이터프레임)
result = pd.DataFrame([add_data,sub_data,mul_data,div_data],
                      index = ['덧셈','뺄셈','곱셈','나눗셈'])
print(result)
```

```
[Output]
          국어        수학     영어
덧셈   190.000000  170.000  160.0
뺄셈    10.000000   10.000    0.0
곱셈  9000.000000 7200.000 6400.0
나눗셈    1.111111    1.125    1.0
```

Null 데이터 확인

시리즈 연산 후에 동일한 인덱스가 없다면 Null 데이터가 생길 수 있다.

데이터가 Null인지 확인

```
pd.isnull(obj)
```

```
a    False
b    False
c    False
d    False
g     True
dtype: bool
```

데이터가 Null이 아닌지 확인

```
pd.notnull(obj)
```

```
a     True
b     True
c     True
d     True
g    False
dtype: bool
```

3. 데이터프레임 (DataFrame)

Auto MPG 데이터셋

1970년대 후반과 1980년대 초반의 자동차의 정보와 연비 데이터셋

Auto MPG 데이터셋

- 이 기간에 출시된 자동차 정보 제공
- 398개 인스턴스, 8개 속성

https://archive.ics.uci.edu/ml/datasets/auto+mpg

속성	타입	설명
MPG	continuous	갤런 당 마일수
Cylinders	multi-valued discrete	실린더 수
Displacement	continuous	배기량
Horsepower	continuous	마력
Weight	continuous	공차 중량
Acceleration	continuous	가속
Model Year	multi-valued discrete	모델 연식
Origin	multi-valued discrete	제조국
Car name	string	차량 이름

데이터셋 다운로드

```
import requests

URL = "http://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
response = requests.get(URL)

dataset_path = "auto-mpg.data"
with open(dataset_path,"w") as f:
  f.write(response.text)
```

데이터 읽기

```
column_names = ['MPG','Cylinders','Displacement','Horsepower','Weight',
                'Acceleration', 'Model Year', 'Origin']
raw_dataset = pd.read_csv(dataset_path, names=column_names,
                          na_values = "?", comment='\t',
                          sep=" ", skipinitialspace=True)

dataset = raw_dataset.copy()
dataset.tail()
```

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
393	27.0	4	140.0	86.0	2790.0	15.6	82	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	2
395	32.0	4	135.0	84.0	2295.0	11.6	82	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	1

4. 데이터 탐색

데이터 탐색

변수/특징 구성

- 분석의 목적에 맞는 변수 구성
- 개별 변수의 이름이나 설명, 데이터 타입

개별 데이터 관찰

- 무작위로 표본을 추출해서 관찰
- 전체적인 추세와 특이 사항
- 데이터 중복 및 결측치/이상치 확인
- 데이터 시각화

요약 통계량/분포

- 데이터의 개수
- 범위 (최소, 최대)
- 중심 경향성 (평균, 중앙값)
- 퍼짐 (표준 편차, IQR)
- 4분위 (25%, 50%, 75%)
- 히스토그램

변수 간의 관계

- 변수 간의 상관 관계

변수/특징 구성 컬럼 이름/타입 확인

```
dataset.columns
```

```
Index(['MPG', 'Cylinders', 'Displacement', 'Horsepower', 'Weight',
       'Acceleration', 'Model Year', 'Origin'],
      dtype='object')
```

컬럼 이름의 데이터 타입

```
dataset.dtypes
```

```
MPG             float64
Cylinders         int64
Displacement    float64
Horsepower      float64
Weight          float64
Acceleration    float64
Model Year        int64
Origin            int64
dtype: object
```

개별 데이터 관찰 컬럼 이름으로 조회

컬럼 이름으로 조회

```
dataset['MPG']
```

```
0      18.0
1      15.0
2      18.0
3      16.0
4      17.0
      ...
393    27.0
394    44.0
395    32.0
396    28.0
397    31.0

Name: MPG, Length: 398, dtype: float64
```

여러 컬럼 조회

칼럼 이름을 리스트로 명시

```
dataset[['MPG', 'Weight']]
```

	MPG	Weight
1	18.0	3504.0
2	15.0	3693.0
3	18.0	3436.0
4	16.0	3433.0
...	...	...
393	27.0	2790.0
394	44.0	2130.0
395	32.0	2295.0
396	28.0	2625.0
397	31.0	2720.0

398 rows × 2 columns

개별 데이터 관찰 컬럼 조건 검색

여러 조건으로 행 선택

```
dataset[(dataset['Origin']==2) & (dataset['Horsepower']>70)].head(4)
```

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
20	25.0	4	110.0	87.0	2672.0	17.5	70	2
21	24.0	4	107.0	90.0	2430.0	14.5	70	2
22	25.0	4	104.0	95.0	2375.0	17.5	70	2
23	26.0	4	121.0	113.0	2234.0	12.5	70	2

- 차량 제조국이 Europe이고 마력이 70이상인 차량 선택
- head()는 default로 처음 5개 항목을 반환, tail()은 default로 마지막 5개 항목을 반환

개별 데이터 관찰 표준 슬라이싱

```
dataset[1:3]
```

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
1	15.0	8	350.0	165.0	3693.0	11.5	70	1
2	18.0	8	318.0	150.0	3436.0	11.0	70	1

개별 데이터 관찰 행 인덱스로 조회

인덱스 위치를 지정해서 행 선택 (iloc 함수)

인덱스 위치

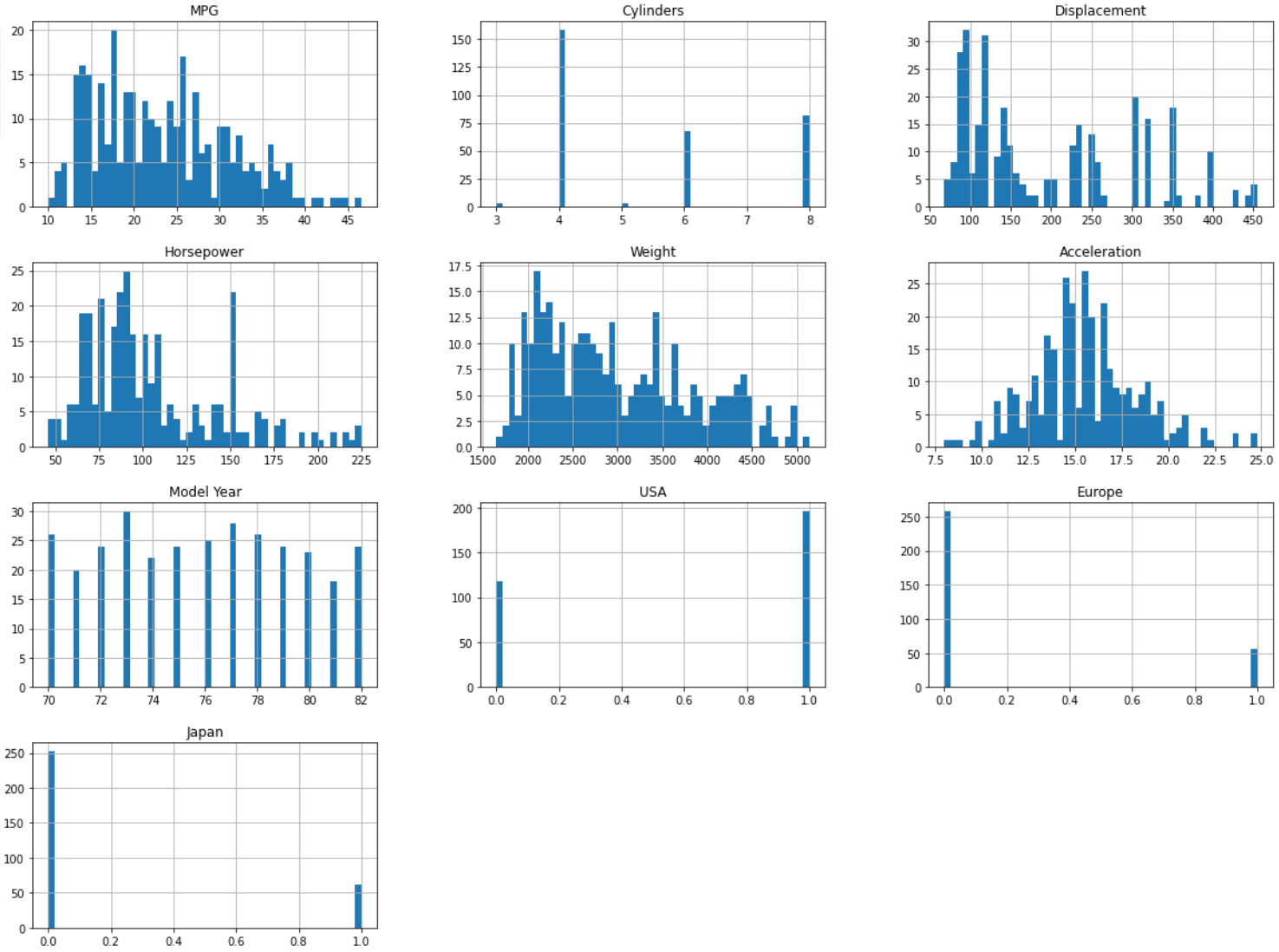
```
dataset.iloc[:5]
```

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
0	18.0	8	307.0	130.0	3504.0	12.0	70	1
1	15.0	8	350.0	165.0	3693.0	11.5	70	1
2	18.0	8	318.0	150.0	3436.0	11.0	70	1
3	16.0	8	304.0	150.0	3433.0	12.0	70	1
4	17.0	8	302.0	140.0	3449.0	10.5	70	1

- iloc는 인덱스 위치에 대해 실행
- 그외 ix(), loc() : 인덱스 레이블을 지정할 수도 있음

요약 통계량/분포 데이터 분포

```
dataset.hist(bins=50, figsize=(20,15))
```



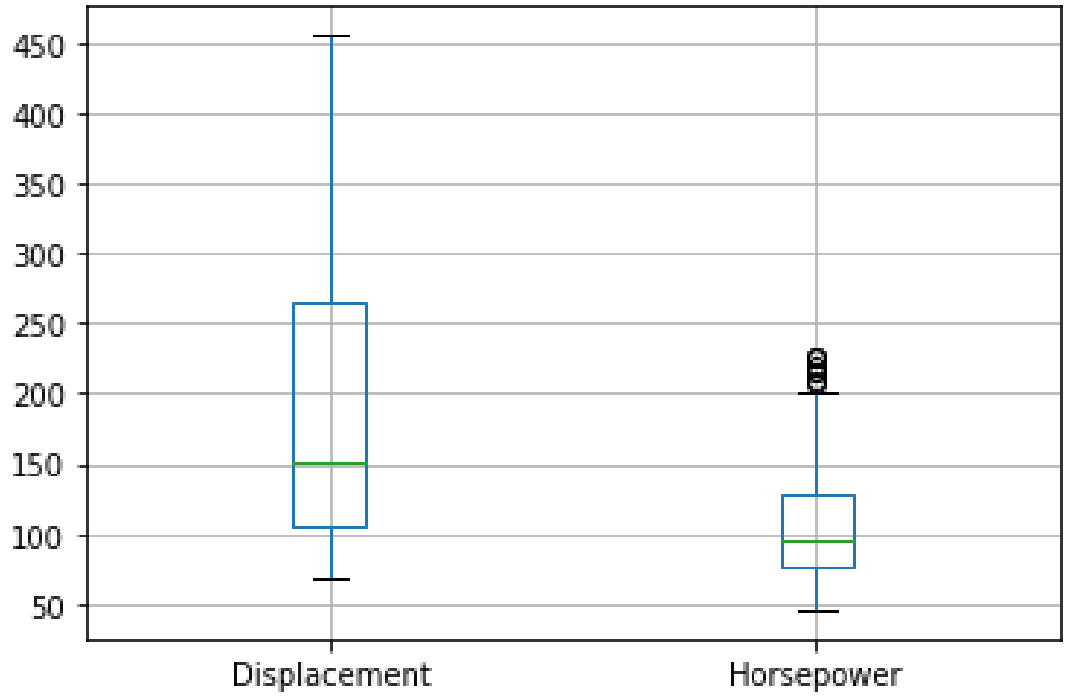
요약 통계량/분포 요약통계

```
stats = dataset.describe()
stats.pop("MPG")
stats = stats.transpose()
stats
```

	count	mean	std	min	25%	50%	75%	max
Cylinders	314.0	5.477707	1.699788	3.0	4.00	4.0	8.00	8.0
Displacement	314.0	195.318471	104.331589	68.0	105.50	151.0	265.75	455.0
Horsepower	314.0	104.869427	38.096214	46.0	76.25	94.5	128.00	225.0
Weight	314.0	2990.251592	843.898596	1649.0	2256.50	2822.5	3608.00	5140.0
Acceleration	314.0	15.559236	2.789230	8.0	13.80	15.5	17.20	24.8
Model Year	314.0	75.898089	3.675642	70.0	73.00	76.0	79.00	82.0
USA	314.0	0.624204	0.485101	0.0	0.00	1.0	1.00	1.0
Europe	314.0	0.178344	0.383413	0.0	0.00	0.0	0.00	1.0
Japan	314.0	0.197452	0.398712	0.0	0.00	0.0	0.00	1.0

요약 통계량/분포 요약통계

```
train_dataset.boxplot(column=["Displacement","Horsepower"])
```

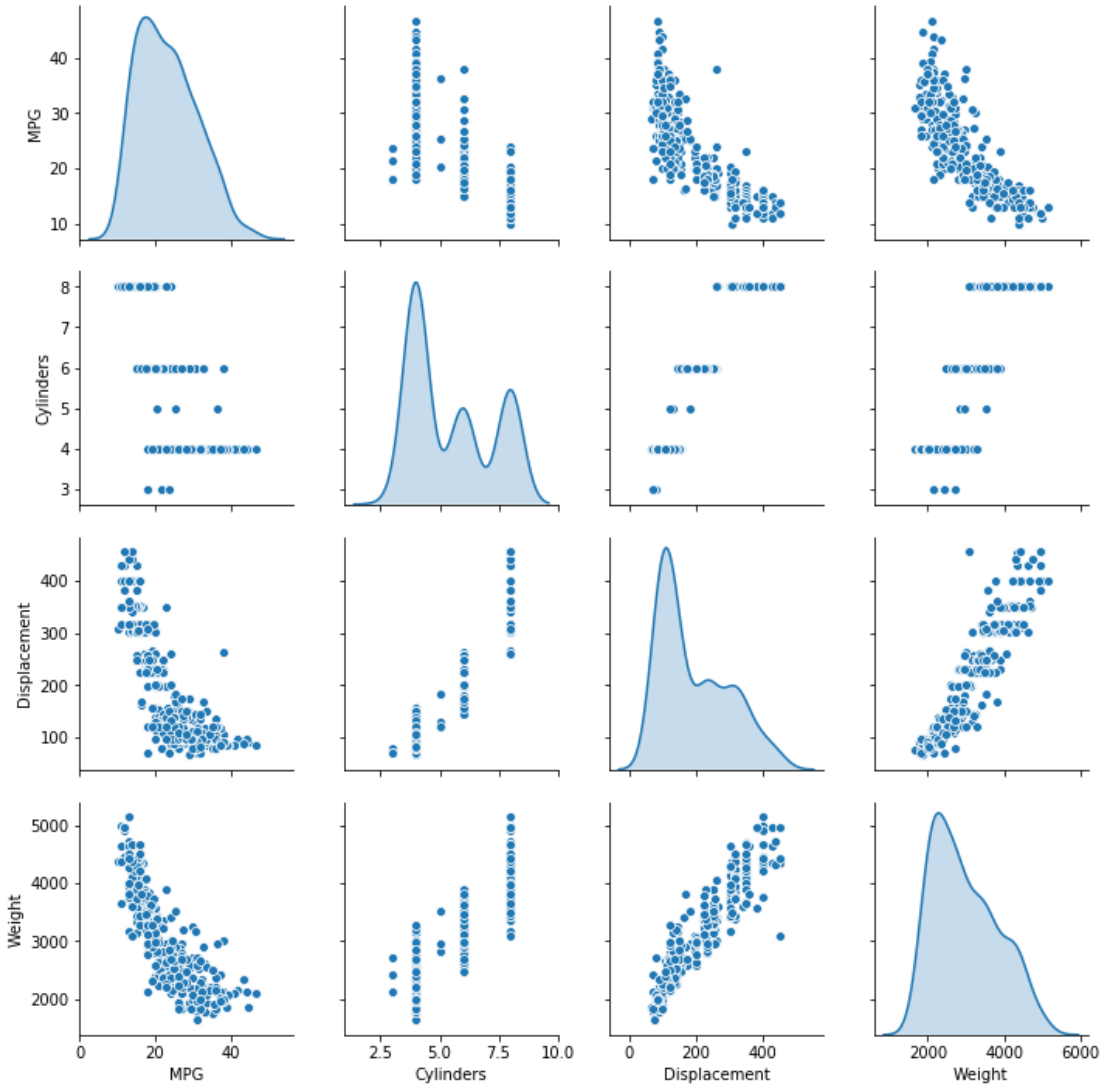


변수 간의 관계 산점도 행렬

산점도 행렬로 데이터 분포 및 상관성 조사

```
sns.pairplot(train_dataset[["MPG",
"Cylinders", "Displacement", "Weight"]],
diag_kind="kde")
```

kde : Use kernel density estimates for univariate plots



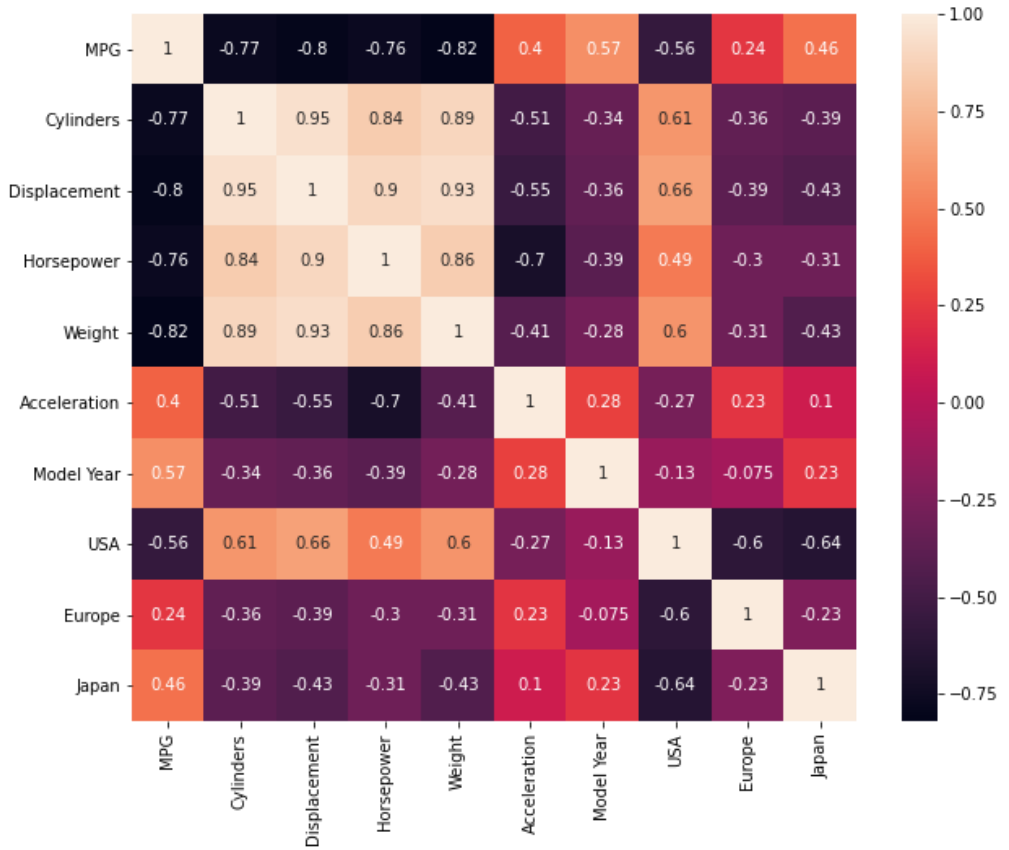
변수 간의 관계 상관관계 표

```
dataset.corr()
```

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	USA	Europe	Japan
MPG	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329	0.580541	-0.565161	0.244313	0.451454
Cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.345647	0.610494	-0.352324	-0.404209
Displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.369855	0.655936	-0.371633	-0.440825
Horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361	0.489625	-0.284948	-0.321936
Weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.309120	0.600978	-0.293841	-0.447929
Acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.290316	-0.258224	0.208298	0.115020
Model Year	0.580541	-0.345647	-0.369855	-0.416361	-0.309120	0.290316	1.000000	-0.136065	-0.037745	0.199841
USA	-0.565161	0.610494	0.655936	0.489625	0.600978	-0.258224	-0.136065	1.000000	-0.591434	-0.648583
Europe	0.244313	-0.352324	-0.371633	-0.284948	-0.293841	0.208298	-0.037745	-0.591434	1.000000	-0.230157
Japan	0.451454	-0.404209	-0.440825	-0.321936	-0.447929	0.115020	0.199841	-0.648583	-0.230157	1.000000

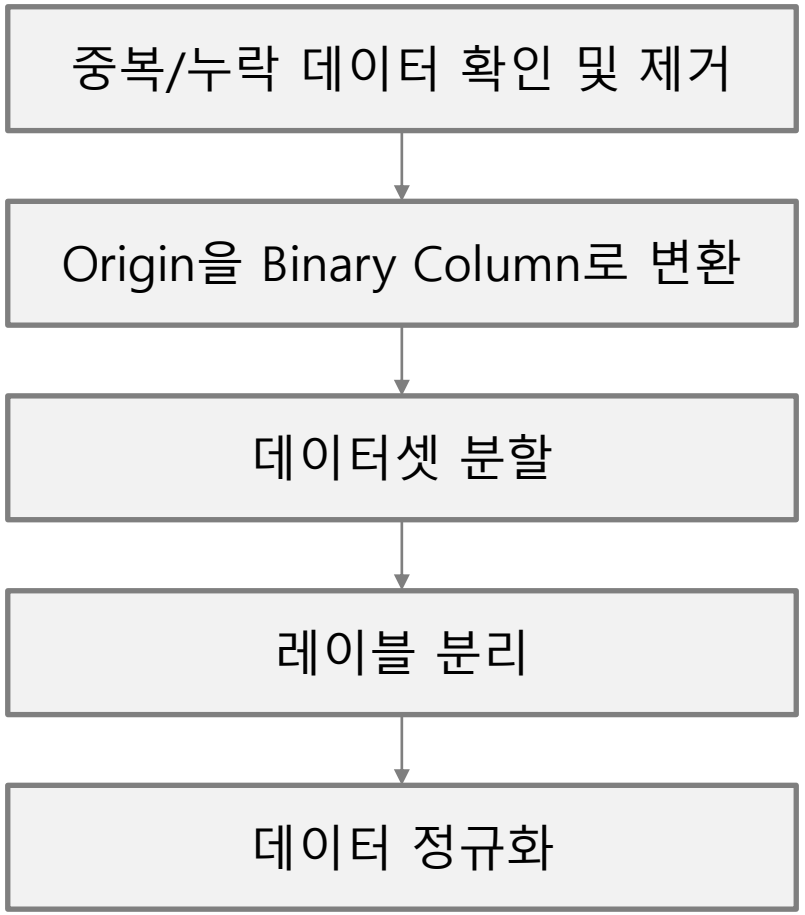
변수 간의 관계 상관관계 히트맵

```
plt.figure(figsize=(10,8))
sns.heatmap(dataset.corr(), annot=True)
```



5. 데이터 전처리

데이터 전처리



중복 데이터 : dataset.duplicated(), dataset.drop_duplicates(),
누락 데이터 : dataset.isna(), dataset.dropna()

Origin 1은 USA, 2는 Europe, 3은 Japan

dataset.sample(), dataset.drop() 사용

dataset.pop() 사용

표준정규분포 정규화 $Z = \frac{X - \mu}{\sigma} \sim \mathcal{N}(0,1)$

중복/누락 데이터 처리

중복 행 조회

```
dataset.duplicated()
```

중복 행 제거

```
dataset.drop_duplicates()
```

누락 데이터 조회

```
dataset.isna()
```

누락 데이터 개수 합산

```
dataset.isna().sum()
```

누락 데이터 제거/채우기

```
dataset = dataset.dropna()
dataset = dataset.fillna(0)
```

행 & 열 추가/삭제

새로운 이름으로 열 추가

```
dataset['USA’] = 1
```

drop 함수로 행 삭제

```
dataset = dataset.drop(‘Origin’, axis=1)
```

axis=0은 row, 1은 column

drop 함수로 열 삭제

```
dataset = dataset.drop(Row_Index, axis=0)
```

axis=0은 row, 1은 column

데이터 샘플링

샘플 개수로 지정

```
dataset.sample(n=5000, random_state=0)
```

샘플 비율로 지정

```
dataset.sample(frac=0.8, random_state=0)
```

Seed for the random number generator (if int), or numpy RandomState object.

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.sample.html

중복 행 체크

```
dataset.duplicated()
```

```
0      False
1      False
2      False
3      False
4      False
       ...
393    False
394    False
395    False
396    False
397    False
Length: 398, dtype: bool
```

중복 행 제거

```
dataset.drop_duplicates()
```

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
0	18.0	8	307.0	130.0	3504.0	12.0	70	1
1	15.0	8	350.0	165.0	3693.0	11.5	70	1
2	18.0	8	318.0	150.0	3436.0	11.0	70	1
3	16.0	8	304.0	150.0	3433.0	12.0	70	1
4	17.0	8	302.0	140.0	3449.0	10.5	70	1
...	...	...	...	...	...	...	...	...
393	27.0	4	140.0	86.0	2790.0	15.6	82	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	2
395	32.0	4	135.0	84.0	2295.0	11.6	82	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	1

398 rows × 8 columns

누락 데이터 확인 및 제거

누락된 데이터 확인

```
dataset.isna().sum()
```

```
MPG             0
Cylinders       0
Displacement    0
Horsepower      6
Weight          0
Acceleration    0
Model Year      0
Origin          0
dtype: int64
```

누락된 행을 삭제

```
dataset = dataset.dropna()
```

Origin을 Binary Column들로 변환

누락된 데이터 열은 수치형이 아니고 범주형이므로 Binary Column으로 변환

```
origin = dataset.pop('Origin')
dataset['USA'] = (origin == 1)*1.0
dataset['Europe'] = (origin == 2)*1.0
dataset['Japan'] = (origin == 3)*1.0
dataset.tail()
```

USA 1 (1 0 0)
Eur 2 (0 1 0)
Jp 3 (0 0 1)
one-hot encoding

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	USA	Europe	Japan
393	27.0	4	140.0	86.0	2790.0	15.6	82	1.0	0.0	0.0
394	44.0	4	97.0	52.0	2130.0	24.6	82	0.0	1.0	0.0
395	32.0	4	135.0	84.0	2295.0	11.6	82	1.0	0.0	0.0
396	28.0	4	120.0	79.0	2625.0	18.6	82	1.0	0.0	0.0
397	31.0	4	119.0	82.0	2720.0	19.4	82	1.0	0.0	0.0

데이터셋 분할

데이터셋을 훈련 세트와 테스트 세트로 분할

```
train_dataset = dataset.sample(frac=0.8, random_state=0)
test_dataset = dataset.drop(train_dataset.index)
```

random_state : seed for the random number generator

레이블 분리

레이블 분리

```
# dataset.pop() 사용
train_labels = train_dataset.pop('MPG')
test_labels = test_dataset.pop('MPG')
```

데이터 정규화

```
train_stats = train_dataset.describe()
train_stats = train_stats.transpose()
train_stats
```

	count	mean	std	min	25%	50%	75%	max
Cylinders	314.0	5.477707	1.699788	3.0	4.00	4.0	8.00	8.0
Displacement	314.0	195.318471	104.331589	68.0	105.50	151.0	265.75	455.0
Horsepower	314.0	104.869427	38.096214	46.0	76.25	94.5	128.00	225.0
Weight	314.0	2990.251592	843.898596	1649.0	2256.50	2822.5	3608.00	5140.0
Acceleration	314.0	15.559236	2.789230	8.0	13.80	15.5	17.20	24.8
Model Year	314.0	75.898089	3.675642	70.0	73.00	76.0	79.00	82.0
USA	314.0	0.624204	0.485101	0.0	0.00	1.0	1.00	1.0
Europe	314.0	0.178344	0.383413	0.0	0.00	0.0	0.00	1.0
Japan	314.0	0.197452	0.398712	0.0	0.00	0.0	0.00	1.0

데이터 정규화

표준정규분포 데이터 정규화

```
def norm(x):
  return (x - train_stats['mean']) / train_stats['std']

normed_train_data = norm(train_dataset)
normed_test_data = norm(test_dataset)
```

정규화할때 train_stats 모든 해당됨.

DataFrame 연산

DataFrame과 Series는 같은 인덱스를 가진 항목끼리 연산

train_dataset = {'MPG' : {...}, 'Cylinders' : {....}, ..., 'Weight' " {...}, ...} DataFrame은 Column들의 Series를 갖는 Series이다!

	MPG	Cylinders	Displacement	Horsepower	Weight	Acceleration	Model Year	Origin
393	27.0	4	140.0	86.0	2790.0	15.6	82	1
394	44.0	4	97.0	52.0	2130.0	24.6	82	2
395	32.0	4	135.0	84.0	2295.0	11.6	82	1
396	28.0	4	120.0	79.0	2625.0	18.6	82	1
397	31.0	4	119.0	82.0	2720.0	19.4	82	1

$$Z = \frac{X - \mu}{\sigma}$$

$$= \frac{\text{train\_dataset} - \text{train\_stats['mean']}}{\text{train\_stats['std']}}$$

train_stats = {'count' : {...}, mean' : {....}, ...}

	count	mean	std	min	25%	50%	75%	max
Cylinders	314.0	5.477707	1.699788	3.0	4.00	4.0	8.00	8.0
Displacement	314.0	195.318471	104.331589	68.0	105.50	151.0	265.75	455.0
Horsepower	314.0	104.869427	38.096214	46.0	76.25	94.5	128.00	225.0
Weight	314.0	2990.251592	843.898596	1649.0	2256.50	2822.5	3608.00	5140.0
Acceleration	314.0	15.559236	2.789230	8.0	13.80	15.5	17.20	24.8
Model Year	314.0	75.898089	3.675642	70.0	73.00	76.0	79.00	82.0
USA	314.0	0.624204	0.485101	0.0	0.00	1.0	1.00	1.0
Europe	314.0	0.178344	0.383413	0.0	0.00	0.0	0.00	1.0
Japan	314.0	0.197452	0.398712	0.0	0.00	0.0	0.00	1.0

train_stats['mean'] = {''Cylinders' : {...}, 'Displacement' : {....}, ..., 'Weight' " {...}, ...}

train_stats['std'] = {''Cylinders' : {...}, 'Displacement' : {....}, ..., 'Weight' " {...}, ...}

Thank you!