In [7]:

```
## 데이터마이닝 7주차 과제
```

1. 도요타 자동차(ToyotaCorolla.csv)를 이용하여 아래 두 문항에 대해 답하시오.

https://github.com/reisanar/datasets/blob/master/ToyotaCorolla.csv
(https://github.com/reisanar/datasets/blob/master/ToyotaCorolla.csv)

① 선형회귀모델을 구축하시오.

i. 1000개의 인스턴스를 이용하시오. ii. ‘age_08_04’, ‘km’, ‘fuel_type’, ‘hp’, ‘met_color’, ‘automatic’, ‘cc’, doors’,‘quarterly_tax’, ‘weight’ 변수를 이용하여 ‘price’를 예측하시오. iii. Training dataset과 test dataset을 7:3으로 분할하고, 이 때 random_state는 42로 설정하시오. iv. 각 변수에 대한 coefficient를 도출하시오. v. Test dataset에 대한 모델의 MSE 및 MAE 값을 도출하시오.

> 경로 /Users/shimyuna/Desktop/✳/Python-Data-Mining/datamining/ToyotaCorolla.csv

In [8]:

```
import pandas as pd
import numpy as np
import sklearn as sk

from sklearn.model_selection import train_test_split  # 훈련/평가 분할
from sklearn.metrics import r2_score # 결정계수 함수 (이 통계 모델로 대상을 얼마나 잘 설명할 수 
from sklearn.linear_model import LinearRegression #선형 회귀 분석
from sklearn.metrics import mean_squared_error, mean_absolute_error

#데이터로부터 데이터 셋을 생성하고, 불러오는 코드
Toyota_df = pd.read_csv("/Users/shimyuna/Desktop/✳/Python-Data-Mining/datamining/Toy
```

In [9]:

```
#1) 100개의 인스턴스만 사용
#2) 'age_08_04','km','fuel_type','hp','met_color','automatic','cc','doors','quarteri
#3) Training dataset과 test dataset을 7:3으로 분할하고, 이때 random_state는 42로 설정하시오
#4) 각 변수에 대한 coefficient를 도출
# 5) Test dataset에 대한 모델의 MSE 및 MAE 값을 도출

Toyota_df = Toyota_df.iloc[:1000] # 처음부터 1000번째 행까지 추출

# 회귀분석을 위한 열 선택
predictors = ['age_08_04','km','fuel_type','hp','met_color','automatic','cc','doors'
outcome = 'price'

# 데이터 분할
X = pd.get_dummies(Toyota_df[predictors], drop_first=True)
y = Toyota_df[outcome]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_stat

# 선형 회귀 모델 학습
model = LinearRegression()
model.fit(X_train, y_train)

# coefficients 출력
print('intercept', model.intercept_)
print(pd.DataFrame({'predictor' : X.columns, 'coefficient': model.coef_}))


# 테스트 데이터에 대한 예측
y_pred = model.predict(X_test)

# MSE와 MAE 계산
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)

print('\n')
print("MSE: ", mse)
print("MAE: ", mae)
```

```
intercept -6.439222488552332e-10
           predictor   coefficient
0          age_08_04  2.809117e-14
1                 km  1.024181e-14
2                 hp  3.944764e-14
3          met_color -5.222718e-14
4          automatic  1.637817e-13
5                 cc -1.098026e-16
6              doors  2.615235e-14
7      quarterly_tax  1.359769e-14
8             weight -3.790100e-16
9              price  1.000000e+00
10  fuel_type_Diesel -1.126410e-13
11  fuel_type_Petrol -4.672758e-14


MSE:  1.185797708517499e-19
MAE:  2.6208605656089884e-10
```

② Ridge와 Lasso의 차이에 대해 간략히 서술하시오.

> Ridge 회귀와 Lasso 회귀는 둘 다 선형 회귀 모델의 일종으로 이 두 모델의 차이는 회귀 계수를 제한하는 방법이다. Ridge 회귀는 회귀 계수를 축소하지만 0(0으로 근접시킴)으로 만들지는 않으며, Lasso 회귀는 회귀 계수를 축소하고 0으로 만들 수 있다.

Ridge 회귀는 L2 규제를 사용하여 회귀 계수를 제한한다. L2 규제는 모든 회귀 계수에 대해 큰 값을 가지는 페널티를 부여하여, 회귀 계수의 크기를 작게 만든다. Ridge 회귀는 이를 통해 모델의 복잡도를 감소시키고 일반화 성능을 향상시킨다.

Lasso 회귀는 L1 규제를 사용하여 회귀 계수를 제한한다. L1 규제는 일부 회귀 계수만을 큰 값을 가지는 페널티를 부여하여, 회귀 계수의 크기를 0으로 만든다. 따라서 Lasso 회귀는 회귀 계수를 0으로 만들어 feature selection의 효과를 가질 수 있다.

2. 잔디깎이 기계 데이터(RidingMowers.csv)를 이용하여 아래 두 문항에 대해 답하시오.

> https://github.com/reisanar/datasets/blob/master/RidingMowers.csv (https://github.com/reisanar/datasets/blob/master/RidingMowers.csv)

① K-NN 분류기에서 이웃 결정 시 가장 보편적으로 사용되는 거리 측도는 유클리드 거리다. 해당 측도가 적합하지 않은 상황과 그 이유에 대해 간략히 서술하시오.

> 유클리드 거리 측도가 적합하지 않은 상황은 큰 차원의 데이터인 상황이 있습니다.

- 유클리드 거리는 각 변수(또는 차원) 사이의 차이를 제곱하여 더한 후 제곱근을 취한 값으로, 데이터 간 거리를 계산하는 가장 보편적인 방법 중 하나입니다. 하지만 차원이 증가할수록 변수들 간의 거리가 급격히 커지면서, 데이터의 분포가 희소해지는 문제가 발생하기때문에 해당 이렇게 데이터의 차원이 큰 경우 해당 측도가 적합하지 않습니다. => 이러한 경우를 <차원의 저주>라고 부름
- 차원의 저주는 데이터 간 거리 측정에 대한 문제를 초래하며, 이로 인해 예측 모델의 성능이 저하시킬 수 있기 때문에 이렇게 데이터의 차원이 큰 경우에는 해당 측도가 적합하지 않습니다. 이런 경우, 차원 축소 기법 등을 사용하여 데이터의 차원을 줄여주는 방식을 통해 성능을 개선할 수 있습니다.

---

② K-NN 분류기는 K 값에 민감한 경향을 보인다. K ∈ {1, 5, 10, 15} 에 대한 검증 정확도(accuracy)를 측정한 후, 가장 적합한 K 값을 선택하고 이에 대한 이유를 간략히 서술하시오.

> 위의 K-NN 분류기를 통해 나온 코드 결과 값을 통해 가장 적합한 K 값은 5라고 생각했습니다.

가장 높은 정확도를 가지고 있는 K = 10 과 k = 5 인 두가지 경우 중, k가 5인 경우가 적합하다고 생각한 이유는 다음과 같습니다.

- 첫째, k 값이 10일 경우 가장 높은 정확도를 보이지만 짝수이기 때문에 분류를 할 때 5:5로 분류 기준이 동일할 경우 분류가 어려워질 수 있다고 판단하여,홀수 인 값인 1,5,15 중 하나를 선택하는 것이 적합하다고 생각했습니다.
- 둘째, k 가 10인 경우는 k가 5인 경우에 비해 판단을 위한 값들이 너무 많이 퍼져있어, 데이터의 지역적 구조를 파악할 수 있는 능력을 놓칠 수 있다고 생각하였습니다.(underfitting의 가능성이 높아짐)
- 결과적으로, 하이퍼 파라미터 값을 변동 시켜서 나온 성능 판단에서 정확도가 높은 k 값 5,10 중 홀수이면서, 데이터가 너무 퍼져있어 오분류의 위험이 적은 값을 선택한 결과 <k = 5> 인 경우가 가장 적합한 값이라고 생각했습니다.

In [63]:

```
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.neighbors import NearestNeighbors,KNeighborsClassifier
import matplotlib.pylab as plt
import dmba
```

In [64]:

```
# 데이터 준비
mower_df = dmba.load_data('RidingMowers.csv')
mower_df['Number'] = mower_df.index + 1 # 인덱스 처리
mower_df.head(5)
```

Out[64]:

	Income	Lot_Size	Ownership	Number
0	60.0	18.4	Owner	1
1	85.5	16.8	Owner	2
2	64.8	21.6	Owner	3
3	61.5	20.8	Owner	4
4	87.0	23.6	Owner	5

In [65]:

```
# 누락데이터 및 유효데이터 확인을 통해 전처리할 내용 확인
print(mower_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 24 entries, 0 to 23
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Income     24 non-null     float64
 1   Lot_Size   24 non-null     float64
 2   Ownership  24 non-null     object 
 3   Number     24 non-null     int64  
dtypes: float64(2), int64(1), object(1)
memory usage: 896.0+ bytes
None
```

In [66]:

```
# i. Training dataset과 validation dataset을 7:3으로 분할하고, 이 때 random_state는 42로 설
trainData, validData = train_test_split(mower_df, test_size= 0.3, random_state= 42)
print(trainData.shape, validData.shape)

newHousehold = pd.DataFrame([{'Income' : 60, 'Lot_Size' : 20}])
newHousehold
```

```
(16, 4) (8, 4)
```

Out[66]:

	Income	Lot_Size
0	60	20

In [67]:

```
# ii. 'Income'과 'Lot_Size'에 대해 StandardScaler를 이용하여 정규화하고, 이를 변수로 사용하시오.
# 정규화된 학습, 검증 및 전체 데이터 프레임 초기화
scaler = preprocessing.StandardScaler()
scaler.fit(trainData[['Income', 'Lot_Size']])

# 전체 데이터 변환
mowerNorm = pd.concat([pd.DataFrame(scaler.transform(mower_df[['Income','Lot_Size']]
                                    columns=['zIncome','zLot_Size']), mower_df[['Owr
trainNorm = mowerNorm.iloc[trainData.index]
validNorm = mowerNorm.iloc[validData.index]
newHouseholdNorm = pd.DataFrame(scaler.transform(newHousehold),columns=['zIncome','z
```

In [68]:

```
# iii. 'Ownership'을 Y값으로 설정하시오.

# 훈련, 검증 데이터 분할 후 k의 값에 따른 민감도를 측정해 가장 정확한 숫자 판단
train_X = trainNorm[['zIncome','zLot_Size']]
train_y = trainNorm['Ownership']
valid_X = validNorm[['zIncome','zLot_Size']]
valid_y = validNorm['Ownership']

# k에 1,5,10,15를 넣어보고 검증 정확도 측정하기
results = []
for k in [1, 5, 10, 15]:
    knn = KNeighborsClassifier(n_neighbors=k).fit(train_X, train_y)
    results.append({
        '검증한 k 값': k,
        '정확도 accuracy': accuracy_score(valid_y, knn.predict(valid_X))
    })

# 결과값을 데이터 프레임으로 변환
results = pd.DataFrame(results)
print(results)
```

```
   검증한 k 값  정확도 accuracy
0         1          0.375
1         5          0.500
2        10          0.625
3        15          0.375
```