머신러닝

20135174 정태이

1. 데이터 준비

import os

import tarfile

import urllib.request

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import StratifiedShuffleSplit

import matplotlib.pyplot as plt

from pandas.plotting import scatter\_matrix

from sklearn.preprocessing import Imputer

from sklearn.preprocessing import OneHotEncoder

from sklearn.base import BaseEstimator, TransformerMixin

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.pipeline import FeatureUnion

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score, mean\_squared\_error

from sklearn.tree import DecisionTreeRegressor

from sklearn.model\_selection import cross\_val\_score

from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import GridSearchCV

from sklearn.model\_selection import RandomizedSearchCV

from scipy.stats import randint

from scipy import stats

from scipy.stats import norm, skew

import seaborn as sns

import warnings

warnings.filterwarnings('ignore')

DATA\_PATH = os.path.join("all")

def load\_data(data\_path=DATA\_PATH):

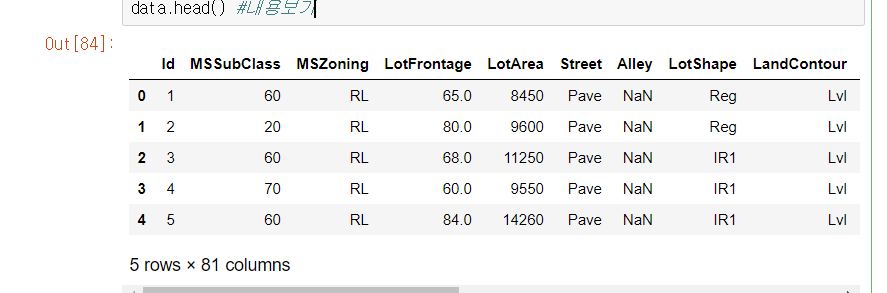
csv\_path = os.path.join(data\_path, "train.csv")

return pd.read\_csv(csv\_path)

data = load\_data()

data\_test = pd.read\_csv(os.path.join(DATA\_PATH,"test.csv")) #test세트도 불러옴

data.head() #내용보기



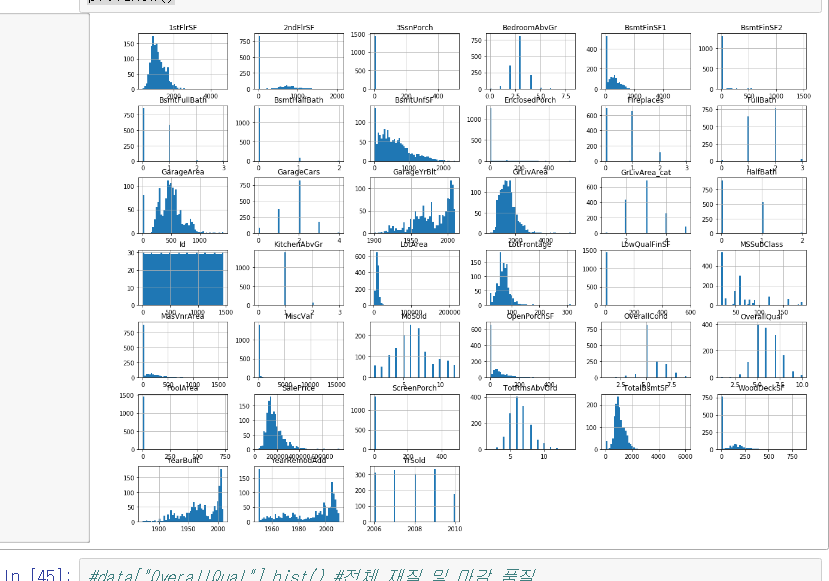
2. 데이터 설명

data.info() #간략한 설명, 특성의 데이터 타입과 널이 아닌 값의 개수 확인 유용

data.describe() #숫자형 특성 요약, null값이 제외된다.

data.hist(bins=50, figsize=(20,15)) #bins는 hist의 한 구간, figsize는 (row,col)

plt.show() #데이터 시각화



#상관계수

corr\_matrix = data.corr()

corr\_matrix["SalePrice"].sort\_values(ascending=False)

#1에 가까울수록 양의 상관관계를 나타내는데, OverallQual이 가장 1에 가까운 0.79이다.

#즉, 가격은 전체 재질이나 마감 품질이 올라갈 때 증가하는 경향이 있다.



#상관관계 시각화, 바로 위 결과에서 위에서 4개를 속성으로 짜서 시각화.

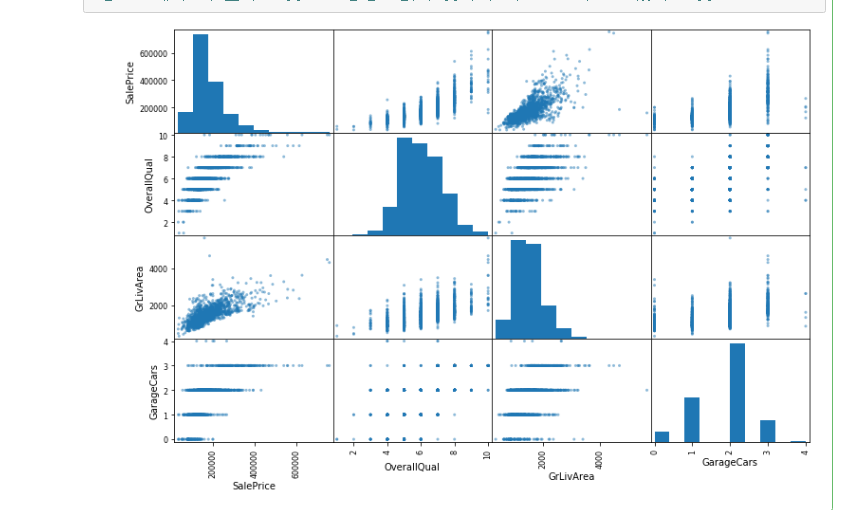
attributes = ["SalePrice","OverallQual","GrLivArea","GarageCars"]

scatter\_matrix(data[attributes],figsize=(12,8))

plt.show()

#오버롤퀄이 가장 상관관계가 높다는걸 알아냈다고 하고 진행한다

#상관관계가 더 높다는 것은 음양상관없이 숫자가 더 크면 더 관련있다는 것.

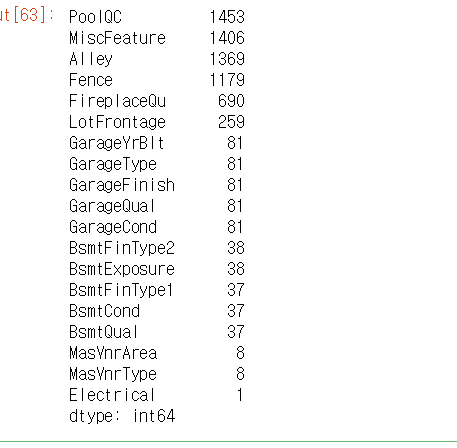


3. 데이터 전처리(데이터 정제)

check\_train = data.isnull().sum() #train세트에서 널값이 있는걸 더해준다

check\_train = check\_train[check\_train>0] #널이 있는 것들만 다시 저장

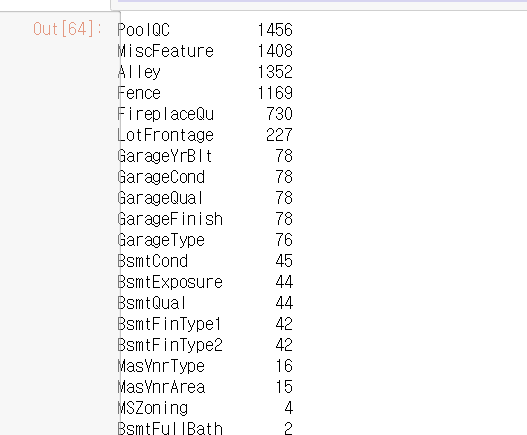
check\_train.sort\_values(ascending=False) #내림차순으로 정렬해준다.



check\_test = data\_test.isnull().sum() #test세트에서도 널값이 있는걸 더해준다

check\_test = check\_test[check\_test>0] #널이 있는 것들만 다시 저장

check\_test.sort\_values(ascending=False) #내림차순으로 정렬해준다.



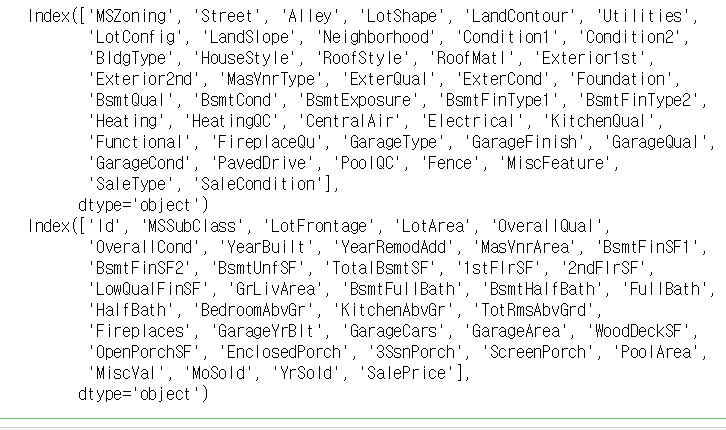
#숫자형과 카테고리를 나눠주기

object\_data = data.select\_dtypes(include=['object']).columns

print(object\_data)# object타입만 담아서 전달받음

num\_data = data.select\_dtypes(exclude = ['object']).columns

print(num\_data) # object타입은 제외한 나머지 숫자형만 저장



train\_num=data[num\_data] # 분리 후 그 분리된 부분만 나오도록 새 변수들에 각각 저장

train\_obj=data[object\_data]

#빈 값 대체

imputer = Imputer(strategy="median")

imputer.fit(train\_num)

#비교

print(imputer.statistics\_)

print(train\_num.median().values)

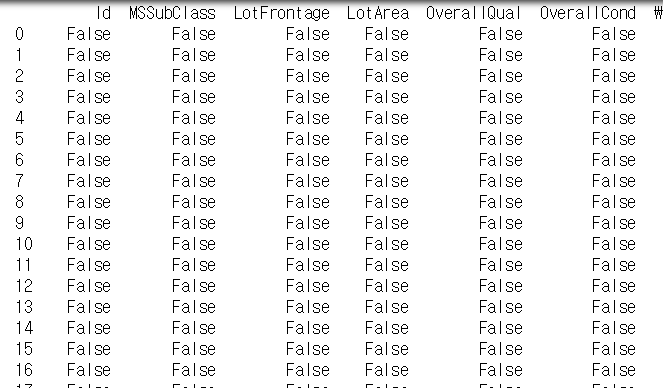
#학습한 중간값으로 바꿈

X = imputer.transform(train\_num)

num\_tr = pd.DataFrame(X, columns=train\_num.columns,

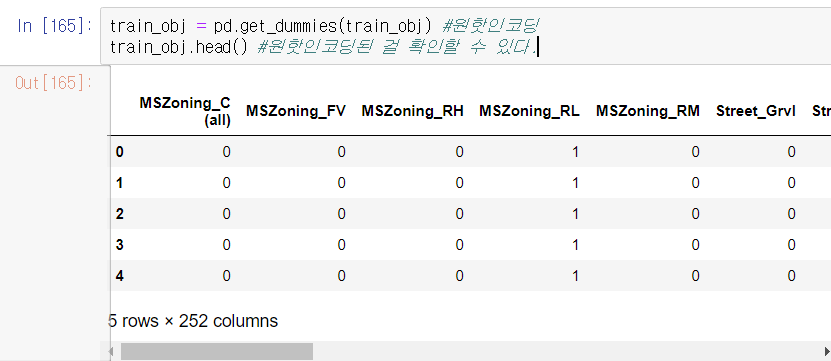
index = list(data.index.values))

print(num\_tr.isnull()) # 널값이 남아있는지 확인 -> False임으로 없음



train\_obj = pd.get\_dummies(train\_obj) #원핫인코딩

train\_obj.head() #원핫인코딩된 걸 확인할 수 있다.

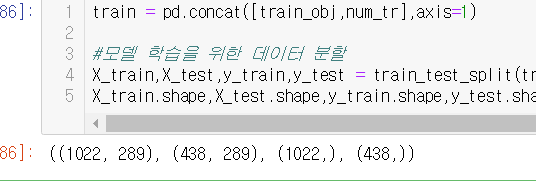


train = pd.concat([train\_obj,num\_tr],axis=1)

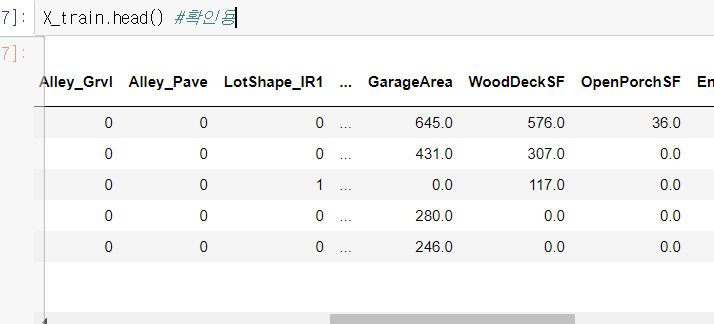
#모델 학습을 위한 데이터 분할

X\_train,X\_test,y\_train,y\_test = train\_test\_split(train,data.SalePrice,test\_size = 0.3,random\_state= 0)

X\_train.shape,X\_test.shape,y\_train.shape,y\_test.shape #확인용



X\_train.head() #확인용



4.모델선택, 학습 및 평가

#test와 train 각각의 함수를 만들어준다

n\_folds = 5

from sklearn.metrics import make\_scorer

from sklearn.model\_selection import KFold

scorer = make\_scorer(mean\_squared\_error,greater\_is\_better = False)

def rmse\_CV\_train(model):

kf = KFold(n\_folds,shuffle=True,random\_state=42).get\_n\_splits(train.values)

rmse = np.sqrt(-cross\_val\_score(model,X\_train,y\_train,scoring ="neg\_mean\_squared\_error",cv=kf))

return (rmse)

def rmse\_CV\_test(model):

kf = KFold(n\_folds,shuffle=True,random\_state=42).get\_n\_splits(train.values)

rmse = np.sqrt(-cross\_val\_score(model,X\_test,y\_test,scoring ="neg\_mean\_squared\_error",cv=kf))

return (rmse)

lr = LinearRegression() #선형회귀

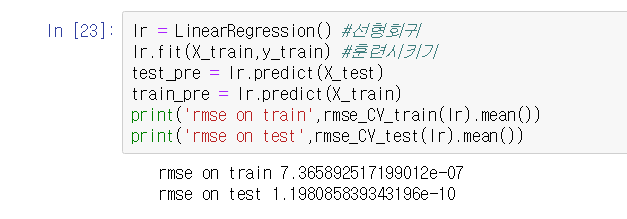
lr.fit(X\_train,y\_train) #훈련시키기

test\_pre = lr.predict(X\_test)

train\_pre = lr.predict(X\_train)

print('rmse on train',rmse\_CV\_train(lr).mean())

print('rmse on test',rmse\_CV\_test(lr).mean())



#예측한 값과의 차이 보기

plt.scatter(train\_pre, train\_pre - y\_train, c = "red", label = "Training data")

plt.scatter(test\_pre,test\_pre - y\_test, c = "blue", label = "Validation data")

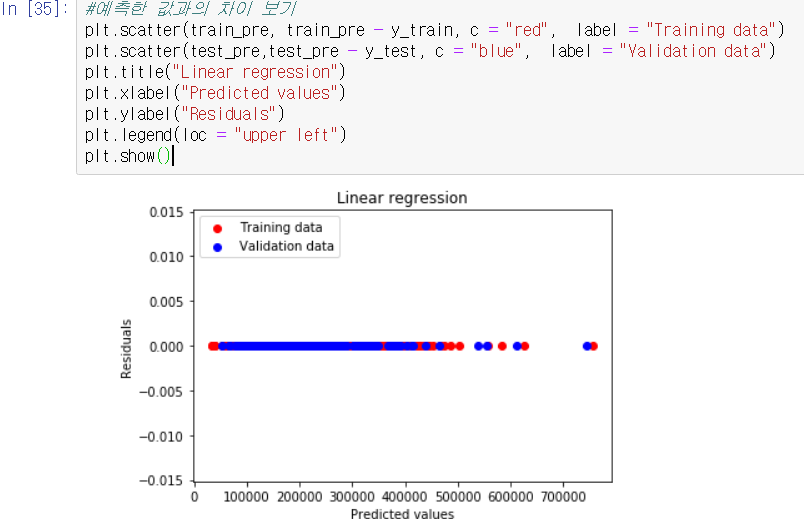
plt.title("Linear regression")

plt.xlabel("Predicted values")

plt.ylabel("Residuals")

plt.legend(loc = "upper left")

plt.show()



# 예측 - 실제 값

plt.scatter(train\_pre, y\_train, c = "red", label = "Training data")

plt.scatter(test\_pre, y\_test, c = "blue", label = "Validation data")

plt.title("Linear regression")

plt.xlabel("Predicted values")

plt.ylabel("Real values")

plt.legend(loc = "upper left")

plt.show()

