Python

House Price Predect

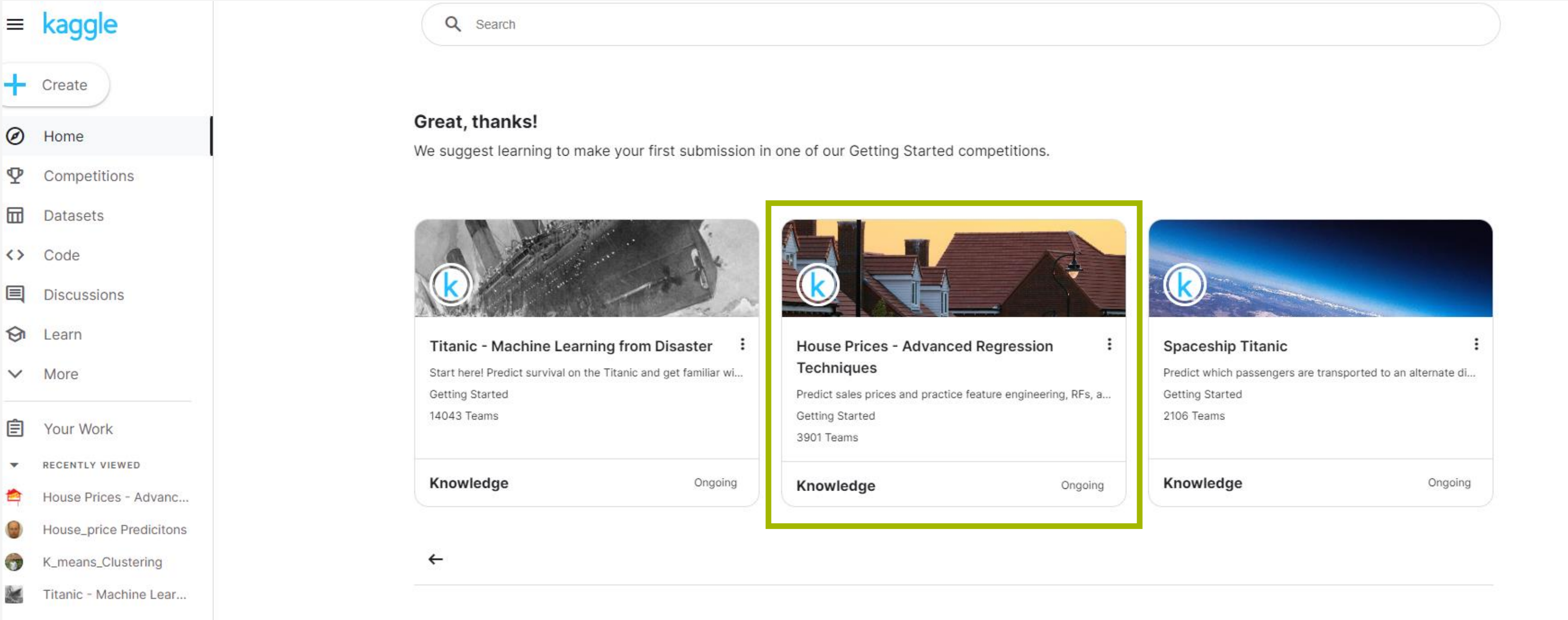
Kaggle / 김남우

Kaggle의 competition 자료이용

CONTENTS

빅데이터 예측 밑 시각화

Data 출처 및 프로그램 목적



I. Kaggle의 한 경쟁파일을 선택하여 진행

II. 기존 데이터로 다양한 범주의 data보유

III. 예측 확인으로 비교하여 다른 이들과 차이확인

Data 출처 및 프로그램 목적

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
print(pd.__version__ , sns.__version__,np.__version__)
```

```
1.4.4 0.12.0 1.23.2
```

```python
train_df=pd.read_csv("data/train.csv")
test_df=pd.read_csv("data/test.csv")
print(train_df.shape,test_df.shape)
```

```
(1460, 81) (1459, 80)
```

```python
combine=[train_df,test_df]
for dataset in combine:
    print(dataset.isna().sum())
```

I. 필요 기능 Import

II. 파일을 데이터 폴더아래 저장

III. 데이터 정보 확인

Data의 정의 및 시각화

자료 내 요약 과 'object' 속성의 요약 확인

```
train_df.describe()
```

	Id	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	...	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch
count	1460.000000	1460.000000	1201.000000	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000	1452.000000	1460.000000	...	1460.000000	1460.000000	1460.000000	1460.000000	1460.000000
mean	730.500000	56.897260	70.049958	10516.828082	6.099315	5.575342	1971.267808	1984.865753	103.685262	443.639726	...	94.244521	46.660274	21.954110	3.409589	15.060959
std	421.610009	42.300571	24.284752	9981.264932	1.382997	1.112799	30.202904	20.645407	181.066207	456.098091	...	125.338794	66.256028	61.119149	29.317331	55.757415
min	1.000000	20.000000	21.000000	1300.000000	1.000000	1.000000	1872.000000	1950.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000
25%	365.750000	20.000000	59.000000	7553.500000	5.000000	5.000000	1954.000000	1967.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000
50%	730.500000	50.000000	69.000000	9478.500000	6.000000	5.000000	1973.000000	1994.000000	0.000000	383.500000	...	0.000000	25.000000	0.000000	0.000000	0.000000
75%	1095.250000	70.000000	80.000000	11601.500000	7.000000	6.000000	2000.000000	2004.000000	166.000000	712.250000	...	168.000000	68.000000	0.000000	0.000000	0.000000
max	1460.000000	190.000000	313.000000	215245.000000	10.000000	9.000000	2010.000000	2010.000000	1600.000000	5644.000000	...	857.000000	547.000000	552.000000	508.000000	480.000000

8 rows × 38 columns

```
train_df.describe(include="object")
```

	MSZoning	Street	Alley	LotShape	LandContour	Utilities	LotConfig	LandSlope	Neighborhood	Condition1	...	GarageType	GarageFinish	GarageQual	GarageCond	PavedDrive	PoolQC	Fence	MiscFeature	Sal
count	1460	1460	91	1460	1460	1460	1460	1460	1460	1460	...	1379	1379	1379	1379	1460	7	281	54	
unique	5	2	2	4	4	2	5	3	25	9	...	6	3	5	5	3	3	4	4	
top	RL	Pave	Grvl	Reg	Lvl	AllPub	Inside	Gtl	NAmes	Norm	...	Attchd	Unf	TA	TA	Y	Gd	MnPrv	Shed	
freq	1151	1454	50	925	1311	1459	1052	1382	225	1260	...	870	605	1311	1326	1340	3	157	49	

4 rows × 43 columns

Data의 정의 및 시각화

```
all_data_na = (train_df.isnull().sum()/len(train_df))*100
all_data_na = all_data_na.drop(all_data_na[all_data_na == 0].index).sort_values(ascending=False)[:15]
missing_data = pd.DataFrame({'Missing Data' : all_data_na})
missing_data
```

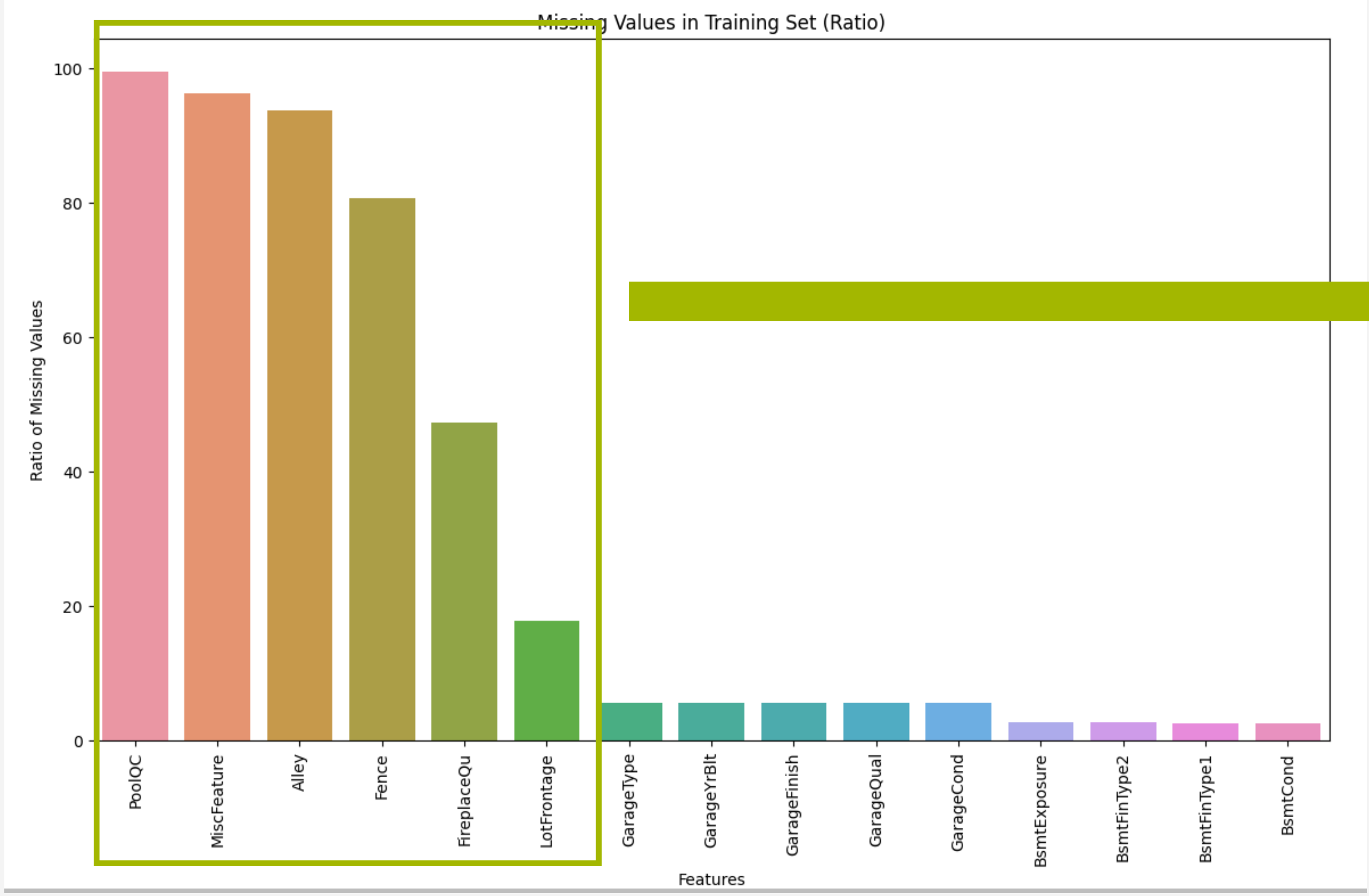
	Missing Data
PoolQC	99.520548
MiscFeature	96.301370
Alley	93.767123
Fence	80.753425
FireplaceQu	47.260274
LotFrontage	17.739726
GarageType	5.547945
GarageYrBlt	5.547945
GarageFinish	5.547945
GarageQual	5.547945
GarageCond	5.547945
BsmtExposure	2.602740
BsmtFinType2	2.602740
BsmtFinType1	2.534247
BsmtCond	2.534247

모든 Colomn의 결측치의 비율 확인

Data의 정의 및 시각화

```
fig,ax = plt.subplots(figsize=(14,8))
sns.barplot(x=all_data_na.index ,y=all_data_na)
plt.xticks(rotation = 90)
plt.xlabel('Features')
plt.ylabel('Ratio of Missing Values')
plt.title('Missing Values in Training Set (Ratio)')
plt.show()
```

결측치 비율의 시각화

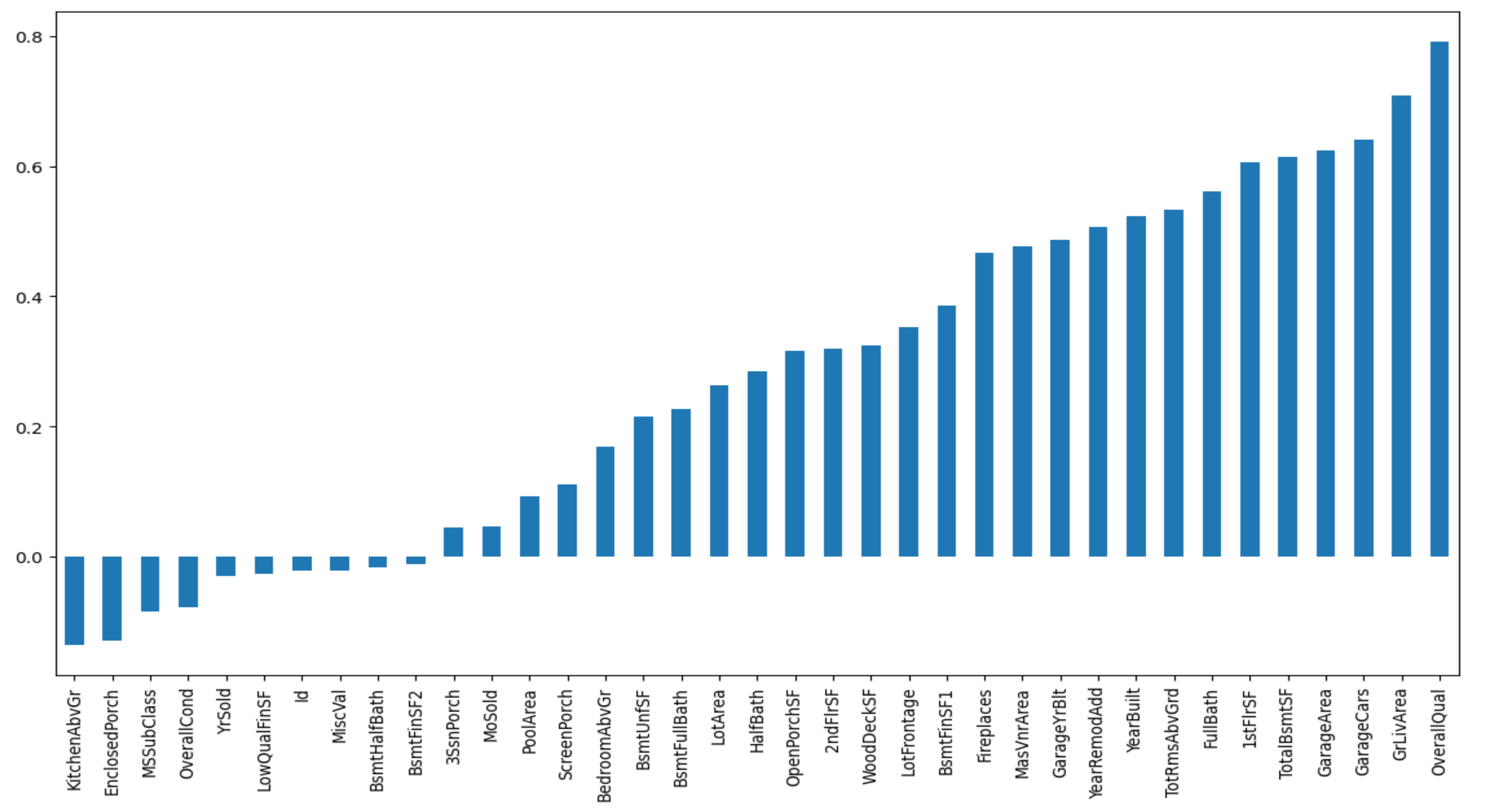


과한 결측치
내역확인

Data의 정의 및 시각화

자료 내 숫자형 자료와 SalePrice간의 상관관계 시각화

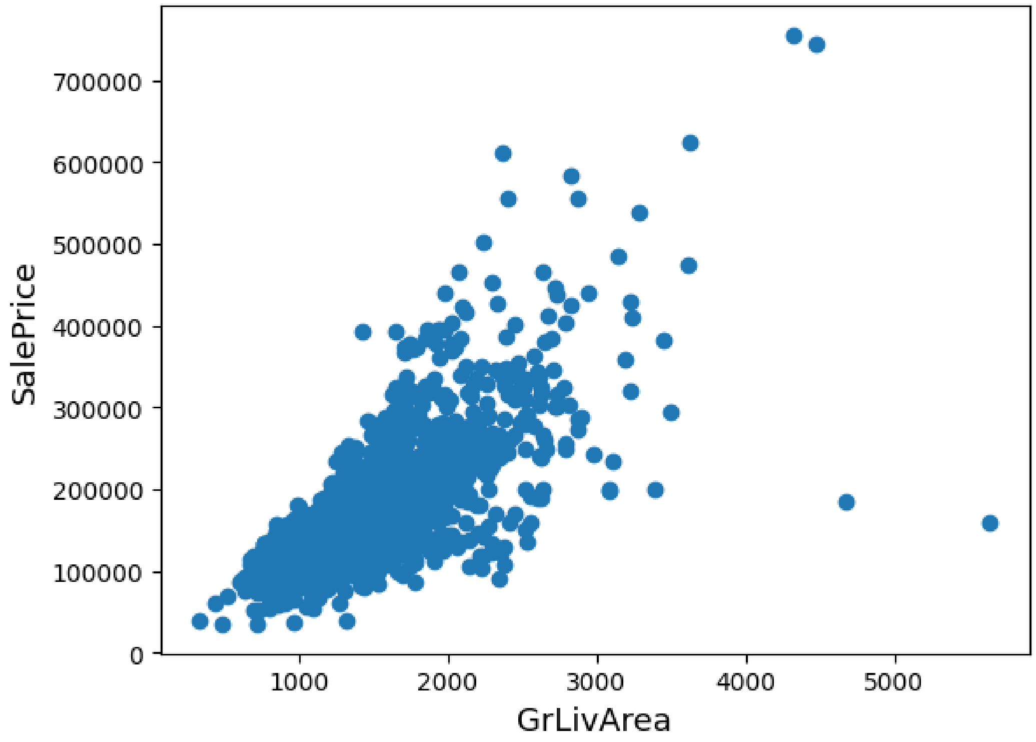
```
plt.figure(figsize=(14,8))
train_df.corr()['SalePrice'].sort_values()[:-1].plot(kind='bar')
plt.show()
```



Data의 정의 및 시각화

자료 내 숫자형 자료와 SalePrice간의 상관관계 시각화

```
fig, ax=plt.subplots()
ax.scatter(x= train_df['GrLivArea'],y= train_df["SalePrice"])
plt.ylabel("SalePrice",fontsize=13)
plt.xlabel("GrLivArea",fontsize=13)
plt.show()
```



기존 SalePrice와 가장 관계성이 높은 GrLivArea의 시각화

Data의 정의 및 시각화

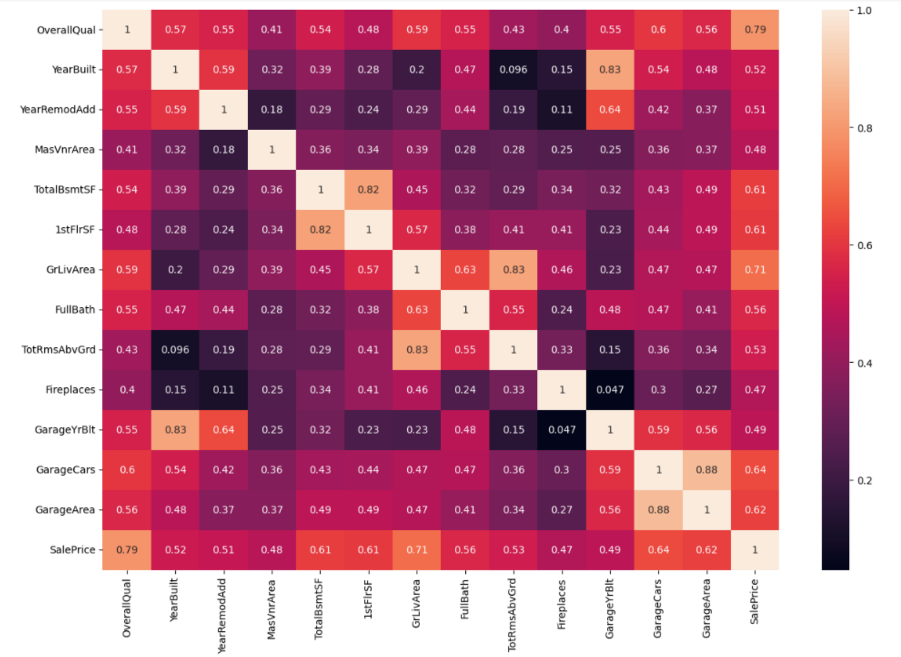
자료 내 숫자형 자료와 SalePrice간의 상관관계 시각화

```
cor= train_df.corr()
cor_fe=cor.index[abs(cor["SalePrice"]) >=0.45]
cor_fe
```

```
Index(['OverallQual', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'TotalBsmtSF',
       '1stFlrSF', 'GrLivArea', 'FullBath', 'TotRmsAbvGrd', 'Fireplaces',
       'GarageYrBlt', 'GarageCars', 'GarageArea', 'SalePrice'],
      dtype='object')
```

```
plt.figure(figsize=(15,10))
sns.heatmap(train_df[cor_fe].corr(),annot=True)
```

```
<AxesSubplot:>
```



Data의 정의 및 시각화

자료 내 숫자형 자료와 SalePrice간의 상관관계 시각화

```
plt.figure(figsize=(14,8))
sns.distplot(train_df['SalePrice'], kde=True, bins=50)
plt.show()
```

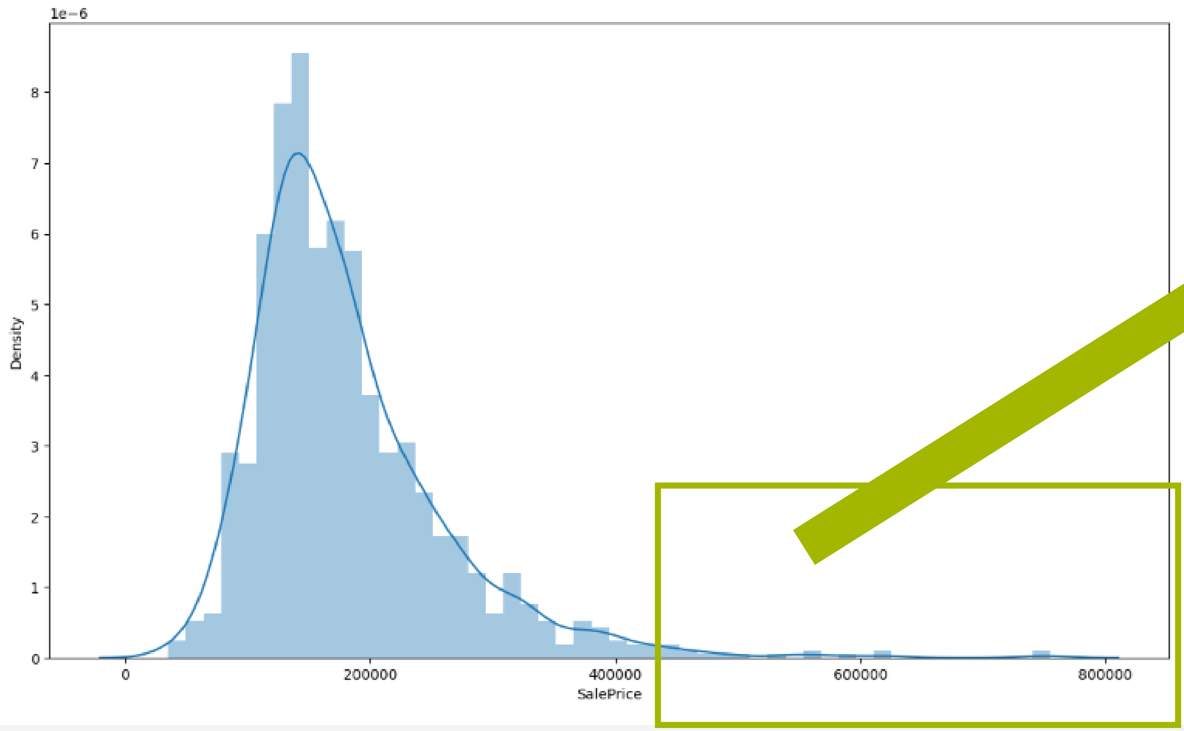
```
C:\Users\KB\AppData\Local\Temp\ipykernel_5420\328497871.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(train_df['SalePrice'], kde=True, bins=50)
```



일부 통계에 잡히지 않거나
예외적인 파일 삭제 필요 확인

자료 변형(범주형의 자료 변형)

자료 내 범주형 자료 변경

```
sim=train_df
combine=[sim,test_df]
title_mapping={"EX":5,"Gd":4,"TA":3, "Fa":2, "Po":1}
for dataset in combine:
    dataset["GarageQual"]=dataset["GarageQual"].map(title_mapping)
    dataset["GarageQual"]=dataset["GarageQual"].fillna(0)
sim[["GarageQual","SalePrice"]].groupby(["GarageQual"],as_index=Fals
```

	GarageQual	SalePrice
0	0.0	108234.523810
1	1.0	100166.666667
2	2.0	123573.354167
3	3.0	185969.791890
4	4.0	215860.714286

```
title_mapping={"EX":5,"Gd":4,"TA":3, "Fa":2, "Po":1}
for dataset in combine:
    dataset["GarageCond"]=dataset["GarageCond"].map(
    dataset["GarageCond"]=dataset["GarageCond"].fill
sim[["GarageCond","SalePrice"]].groupby(["GarageCond
```

	GarageCond	SalePrice
0	0.0	103815.662651
1	1.0	108500.000000
2	2.0	114654.028571
3	3.0	186384.136157
4	4.0	179930.000000

```
title_mapping={"BuiltIn":8,"Attchd":7,"Basment":5,"2Types" : 4,"Detchd":3, "CarPort" : 1}
for dataset in combine:
    dataset["GarageType"]=dataset["GarageType"].map(title_mapping)
    dataset["GarageType"]=dataset["GarageType"].fillna(0)
sim[["GarageType","SalePrice"]].groupby(["GarageType"],as_index=False).mean()
```

	GarageType	SalePrice
0	0.0	103317.283951
1	1.0	109962.111111
2	3.0	134091.162791
3	4.0	151283.333333
4	5.0	160570.684211
5	7.0	200669.692841
6	8.0	254751.738636

```
bsm_mapping={"Gd":6,"Av":4, "Mn":3, "No":2}
for dataset in combine:
    dataset["BsmtExposure"]=dataset["BsmtExposure"].map(bsm_mapping)
    dataset["BsmtExposure"]=dataset["BsmtExposure"].fillna(0)
for dataset in combine:
    dataset["Bsmtpoint"]=dataset["BsmtQual"]+dataset["BsmtCond"]+dataset["Bsmt
sim[["Bsmtpoint","SalePrice"]].groupby(["Bsmtpoint"],as_index=False).mean()
```

	Bsmtpoint	SalePrice
0	0.0	105652.891892
1	5.0	67000.000000
2	6.0	110625.000000
3	7.0	120981.250000
4	8.0	137007.746421
5	9.0	191110.984655
6	10.0	209800.058065
7	11.0	209409.513514
8	12.0	245149.432836
9	13.0	228451.314286
10	14.0	339400.800000
11	15.0	465000.000000

```
lotslp_mapping={"Sev":3, "Mod":2, "Gtl":1}
for dataset in combine:
    dataset["LandSlope"]=dataset["LandSlope"].map(lotslp_mapping)
    dataset["LandSlope"]=dataset["LandSlope"].fillna(0)
sim[["LandSlope","SalePrice"]].groupby(["LandSlope"],as_index=False).mean()
```

	LandSlope	SalePrice
0	1	178493.207547
1	2	196734.138462
2	3	204379.230769

```
title_mapping={"Ex":5,"Gd":4,"TA":3, "Fa":2, "Po":1}
for dataset in combine:
    dataset["BsmtQual"]=dataset["BsmtQual"].map(title_mapping)
    dataset["BsmtQual"]=dataset["BsmtQual"].fillna(0)
sim[["BsmtQual","SalePrice"]].groupby(["BsmtQual"],as_index=False).mean()
```

	BsmtQual	SalePrice
0	0.0	105652.891892
1	2.0	115692.028571
2	3.0	140759.818182
3	4.0	202688.478964
4	5.0	314831.700855

```
title_mapping={"Ex":5,"Gd":4,"TA":3, "Fa":2, "Po":1}
for dataset in combine:
    dataset["BsmtCond"]=dataset["BsmtCond"].map(title_mapping)
    dataset["BsmtCond"]=dataset["BsmtCond"].fillna(0)
sim[["BsmtCond","SalePrice"]].groupby(["BsmtCond"],as_index=False).mean()
```

```
combine = [sim,test_df]
lotc_mapping={"HLS":4, "Low":3, "Lvl":2, "Bnk":1}
for dataset in combine:
    dataset["LandContour"]=dataset["LandContour"].map(lotc_mapping)
    dataset["LandContour"]=dataset["LandContour"].fillna(0)
sim[["LandContour","SalePrice"]].groupby(["LandContour"],as_index=False).mean().sort_values(by="SalePrice",ascending=False)
```

	LandContour	SalePrice
3	4	231533.940000
2	3	203661.111111
1	2	178641.342770
0	1	143104.079365

```
title_mapping={"CulDSac":7, "FR3":6, "FR2":3, "Corner":4, "Inside":2}
for dataset in combine:
    dataset["LotConfig"]=dataset["LotConfig"].map(title_mapping)
    dataset["LotConfig"]=dataset["LotConfig"].fillna(0)
sim[["LotConfig","SalePrice"]].groupby(["LotConfig"],as_index=False).mean().sort_values(by="SalePrice",ascending=False)
```

	LotConfig	SalePrice
4	7	219541.225806
3	6	208475.000000
1	3	177934.574468
2	4	177268.049808
0	2	176524.423406

자료 변형(결측치 제거, 채우기)

결측이 된 량이 많은 몇몇 칼럼 삭제 또는 채우기고 상관관계가 낮은 칼럼 삭제

```
sim.corr()['SalePrice'].sort_values(ascending=False)[1:]
```

```
OverallQual      0.798004
Neighborhood     0.704835
ExterQual        0.693246
GrLivArea        0.691034
GarageCars       0.651630
KitchenQual      0.634925
GarageArea       0.634058
TotalBsmtSF      0.605791
1stFlrSF         0.596491
FullBath         0.558121
GarageFinish     0.549166
YearBuilt        0.539240
TotRmsAbvGrd     0.528314
YearRemodAdd     0.526217
Gagagepoint      0.514096
Bsmtpoint        0.512100
Exter            0.510827
GarageType       0.503313
Foundation       0.500378
MasVnr           0.466157
Fireplaces       0.464401
MasVnrArea       0.445564
MasVnrType       0.404258
BsmtFinSF1       0.360712
OpenPorchSF      0.329547
WoodDeckSF       0.324530
2ndFlrSF         0.293729
GarageCond       0.280386
HalfBath         0.279427
GarageQual       0.272500
GarageYrBlt      0.272159
LotArea          0.261000
BsmtFullBath     0.231081
BsmtUnfSF        0.229938
LandContour      0.170209
BedroomAbvGr     0.162933
Heating          0.123900
ScreenPorch      0.123285
LotConfig        0.119334
MoSold           0.062310
LandSlope        0.058463
3SsnPorch        0.049468
PoolArea         0.029813
BsmtFinSF2      -0.006743
MiscVal         -0.020869
LowQualFinSF    -0.024955
YrSold          -0.025256
Id              -0.034198
BsmtHalfBath    -0.035687
OverallCond     -0.077834
ExterCond       -0.083986
MSSubClass      -0.087090
EnclosedPorch   -0.129846
KitchenAbvGr    -0.140343
Name: SalePrice, dtype: float64
```

```
for dataset in combine:
    dataset["GarageCars"]=dataset["GarageCars"].fillna(0)
    dataset["GarageArea"]=dataset["GarageArea"].fillna(0)
```

```
for dataset in combine:
    dataset["TotalBsmtSF"]=dataset["TotalBsmtSF"].fillna(0)
```

```
sim=sim.drop(["BsmtQual","BsmtCond","BsmtExposure","Street","LotShape","Utilities"],axis=1)
```

```
sim=sim.drop(["MSZoning","BldgType","HouseStyle","RoofStyle","RoofMatl","Exterior1st","Exterior2nd","BsmtFinType1","BsmtFinType2","HeatingQC",\
              "CentralAir","Electrical","Functional","PavedDrive","SaleType","SaleCondition"],axis=1)
```

```
test_df=test_df.drop(["MSZoning","BldgType","HouseStyle","RoofStyle","RoofMatl","Exterior1st","Exterior2nd","BsmtFinType1","BsmtFinType2","HeatingQC",\
              "CentralAir","Electrical","Functional","PavedDrive","SaleType","SaleCondition"],axis=1)
```

```
test_df=test_df.drop(["Street","LotShape","Utilities"],axis=1)
```

```
test_df=test_df.drop(["BsmtQual","BsmtCond","BsmtExposure"],axis=1)
```

```
sim.shape,test_df.shape
```

```
((1456, 25), (1459, 24))
```

상관관계 0.45미만 삭제

○ + 자료 변형(0.45기준 낮은 상관관계 제거)

```
sim=sim.drop(["MasVnrType","BsmtFinSF1","OpenPorchSF","WoodDeckSF","2ndFlrSF","GarageCond","HalfBath","GarageQual","LotArea","BsmtFullBath","BsmtUnfSF","Condition1","LandContour",\
"BedroomAbvGr","Heating","ScreenPorch","LotConfig","Condition2","MoSold","LandSlope","3SsnPorch","PoolArea","BsmtFinSF2","MiscVal","LowQualFinSF","YrSold","BsmtHalfBath","OverallCond",\
"ExterCond","MSSubClass","EnclosedPorch","KitchenAbvGr"],axis=1)
```

```
test_df=test_df.drop(["MasVnrType","BsmtFinSF1","OpenPorchSF","WoodDeckSF","2ndFlrSF","GarageCond","HalfBath","GarageQual","LotArea","BsmtFullBath","BsmtUnfSF","Condition1","LandContour",\
"BedroomAbvGr","Heating","ScreenPorch","LotConfig","Condition2","MoSold","LandSlope","3SsnPorch","PoolArea","BsmtFinSF2","MiscVal","LowQualFinSF","YrSold","BsmtHalfBath","OverallCond",\
"ExterCond","MSSubClass","EnclosedPorch","KitchenAbvGr"],axis=1)
```

```
sim.corr()['SalePrice'].sort_values(ascending=False)[1:]
```

```
OverallQual     0.798004
Neighborhood    0.704835
ExterQual       0.693246
GrLivArea       0.691034
GarageCars      0.651630
KitchenQual     0.634925
GarageArea      0.634058
TotalBsmtSF     0.605791
1stFlrSF        0.596491
FullBath        0.558121
GarageFinish    0.549166
YearBuilt       0.539240
TotRmsAbvGrd    0.528314
YearRemodAdd    0.526217
Gagagepoint     0.514096
Bsmtpoint       0.512100
Exter           0.510827
GarageType      0.503313
Foundation      0.500378
MasVnr          0.466157
Fireplaces      0.464401
MasVnrArea      0.445564
GarageYrBlt     0.272159
Id             -0.034198
Name: SalePrice, dtype: float64
```

자료 변형(결측치 확인)

```
sim.isna().sum()
```

```
Id              0
Neighborhood    0
OverallQual     0
YearBuilt       0
YearRemodAdd    0
MasVnrArea      0
ExterQual       0
Foundation      0
TotalBsmtSF     0
1stFlrSF        0
GrLivArea       0
FullBath        0
KitchenQual     0
TotRmsAbvGrd    0
Fireplaces      0
GarageType      0
GarageYrBlt     0
GarageFinish    0
GarageCars      0
GarageArea      0
SalePrice       0
Gagagepoint     0
Bsmtpoint       0
MasVnr          0
Exter           0
dtype: int64
```

```
test_df.isna().sum()
```

```
Id              0
Neighborhood    0
OverallQual     0
YearBuilt       0
YearRemodAdd    0
MasVnrArea      0
ExterQual       0
Foundation      0
TotalBsmtSF     0
1stFlrSF        0
GrLivArea       0
FullBath        0
KitchenQual     0
TotRmsAbvGrd    0
Fireplaces      0
GarageType      0
GarageYrBlt     0
GarageFinish    0
GarageCars      0
GarageArea      0
Gagagepoint     0
Bsmtpoint       0
MasVnr          0
Exter           0
dtype: int64
```

결측이 없음을 확인

결과 예측

측정 변경 준비

```
X_train=sim.drop(["SalePrice","Id"],axis=1)
Y_train=sim["SalePrice"]
X_test=test_df.drop("Id",axis=1).copy()
X_train.shape, Y_train.shape, X_test.shape
```

```
((1456, 23), (1456,), (1459, 23))
```

결과 예측

측정 변경 준비

```
X_train=sim.drop(["SalePrice","Id"],axis=1)
Y_train=sim["SalePrice"]
X_test=test_df.drop("Id",axis=1).copy()
X_train.shape, Y_train.shape, X_test.shape
```

```
((1456, 23), (1456,), (1459, 23))
```

결과 예측

예측 실행

```
from sklearn.svm import SVC, LinearSVC
svc=SVC()
svc.fit(X_train,Y_train)
Y_pred=svc.predict(X_test)
acc_svc=round(svc.score(X_train,Y_train)*100,2)
acc_svc
```

1.92

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(X_train,Y_train)
Y_pred=knn.predict(X_test)
acc_knn=round(knn.score(X_train,Y_train)*100,2)
acc_knn
```

21.84

```
from sklearn.naive_bayes import GaussianNB
gaussian=GaussianNB()
gaussian.fit(X_train,Y_train)
Y_pred=gaussian.predict(X_test)
acc_gaussian=round(gaussian.score(X_train,Y_train)*100,2)
acc_gaussian
```

51.72

```
from sklearn.linear_model import Perceptron
perceptron=Perceptron()
perceptron.fit(X_train,Y_train)
Y_pred=perceptron.predict(X_test)
acc_perceptron=round(perceptron.score(X_train,Y_train)*100,2)
acc_perceptron
```

0.69

```
from sklearn.linear_model import SGDClassifier
sgd=SGDClassifier()
sgd.fit(X_train,Y_train)
Y_pred=sgd.predict(X_test)
acc_sgd=round(sgd.score(X_train,Y_train)*100,2)
acc_sgd
```

1.3

```
from sklearn.tree import DecisionTreeClassifier
decision_tree=DecisionTreeClassifier()
decision_tree.fit(X_train,Y_train)
Y_pred=decision_tree.predict(X_test)
acc_decision_tree=round(decision_tree.score(X_train,Y_train)*100,2)
acc_decision_tree
```

99.59

```
from sklearn.ensemble import RandomForestClassifier
random_forest=RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train,Y_train)
Y_pred=random_forest.predict(X_test)
acc_random_forest=round(random_forest.score(X_train,Y_train)*100,2)
acc_random_forest
```

99.59

```
models=pd.DataFrame({"Model":["SVM","KNN","Logistic Regression","Random Forest","Naive Bayes","Perseptron","SGD","Decision Tree"],
                     "Score":[acc_svc,acc_knn,acc_logreg,acc_random_forest,acc_gaussian,acc_perceptron,acc_sgd,acc_decision_tree]
                     })
models.sort_values(by="Score",ascending=False)
```

	Model	Score
3	Random Forest	99.59
7	Decision Tree	99.59
4	Naive Bayes	51.72
1	KNN	21.84
2	Logistic Regression	9.55
0	SVM	1.92
6	SGD	1.30
5	Perseptron	0.69

측정 결과

결과 예측

예측 실행

```
from sklearn.svm import SVC, LinearSVC
svc=SVC()
svc.fit(X_train,Y_train)
Y_pred=svc.predict(X_test)
acc_svc=round(svc.score(X_train,Y_train)*100,2)
acc_svc
```

1.92

```
from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier()
knn.fit(X_train,Y_train)
Y_pred=knn.predict(X_test)
acc_knn=round(knn.score(X_train,Y_train)*100,2)
acc_knn
```

21.84

```
from sklearn.naive_bayes import GaussianNB
gaussian=GaussianNB()
gaussian.fit(X_train,Y_train)
Y_pred=gaussian.predict(X_test)
acc_gaussian=round(gaussian.score(X_train,Y_train)*100,2)
acc_gaussian
```

51.72

```
from sklearn.linear_model import Perceptron
perceptron=Perceptron()
perceptron.fit(X_train,Y_train)
Y_pred=perceptron.predict(X_test)
acc_perceptron=round(perceptron.score(X_train,Y_train)*100,2)
acc_perceptron
```

0.69

```
from sklearn.linear_model import SGDClassifier
sgd=SGDClassifier()
sgd.fit(X_train,Y_train)
Y_pred=sgd.predict(X_test)
acc_sgd=round(sgd.score(X_train,Y_train)*100,2)
acc_sgd
```

1.3

```
from sklearn.tree import DecisionTreeClassifier
decision_tree=DecisionTreeClassifier()
decision_tree.fit(X_train,Y_train)
Y_pred=decision_tree.predict(X_test)
acc_decision_tree=round(decision_tree.score(X_train,Y_train)*100,2)
acc_decision_tree
```

99.59

```
from sklearn.ensemble import RandomForestClassifier
random_forest=RandomForestClassifier(n_estimators=100)
random_forest.fit(X_train,Y_train)
Y_pred=random_forest.predict(X_test)
acc_random_forest=round(random_forest.score(X_train,Y_train)*100,2)
acc_random_forest
```

99.59

```
models=pd.DataFrame({"Model":["SVM","KNN","Logistic Regression","Random Forest","Naive Bayes","Perseptron","SGD","Decision Tree"],
                     "Score":[acc_svc,acc_knn,acc_logreg,acc_random_forest,acc_gaussian,acc_perceptron,acc_sgd,acc_decision_tree]
                    })
models.sort_values(by="Score",ascending=False)
```

	Model	Score
3	Random Forest	99.59
7	Decision Tree	99.59
4	Naive Bayes	51.72
1	KNN	21.84
2	Logistic Regression	9.55
0	SVM	1.92
6	SGD	1.30
5	Perseptron	0.69

이를 사용(가장 높은 수치)

측정 결과

결과 예측

```
submission=pd.DataFrame({
    "Id":test_df["Id"],
    "SalePrice":Y_pred
})
submission.head()
```

	Id	SalePrice
0	1461	109500
1	1462	139000
2	1463	181000
3	1464	181000
4	1465	180000

submission2.csv 0.23633

5 days ago by N-Tress

add submission details

0.0000에서 3100번째 번 예측

THANKS
FOR WATCHING