Comprehensive data exploration with Python

Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

1. Package Import

```
#invite people for the Kaggle party
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from scipy.stats import norm
from sklearn.preprocessing import StandardScaler
from scipy import stats
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

Despite the strange names I gave to the chapters, what we are doing in this kernel is something like:

1. **Understand the problem**. We'll look at each variable and do a philosophical analysis about their meaning and importance for this problem.
2. **Univariable study**. We'll just focus on the dependent variable ('SalePrice') and try to know a little bit more about it.
3. **Multivariate study**. We'll try to understand how the dependent variable and independent variables relate.
4. **Basic cleaning**. We'll clean the dataset and handle the missing data, outliers and categorical variables.
5. **Test assumptions**. We'll check if our data meets the assumptions required by most multivariate techniques.

Comprehensive data exploration with Python

Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

2. 데이터 확인 및 분포 확인(총 : 118260개)

```
#bring in the six packs
df_train = pd.read_csv('../input/train.csv')
```

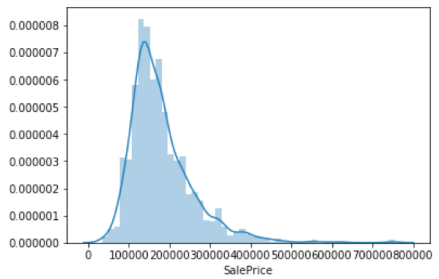
```
#check the decoration
df_train.columns
```

```
Index(['Id', 'MSSubClass', 'MSZoning', 'LotFrontage', 'LotArea', 'Street',
       'Alley', 'LotShape', 'LandContour', 'Utilities', 'LotConfig',
       'LandSlope', 'Neighborhood', 'Condition1', 'Condition2', 'BldgType',
       'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd',
       'RoofStyle', 'RoofMatl', 'Exterior1st', 'Exterior2nd', 'MasVnrType',
       'MasVnrArea', 'ExterQual', 'ExterCond', 'Foundation', 'BsmtQual',
       'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1',
       'BsmtFinType2', 'BsmtFinSF2', 'BsmtUnfSF', 'TotalBsmtSF', 'Heating',
       'HeatingQC', 'CentralAir', 'Electrical', '1stFlrSF', '2ndFlrSF',
       'LowQualFinSF', 'GrLivArea', 'BsmtFullBath', 'BsmtHalfBath', 'FullBath',
       'HalfBath', 'BedroomAbvGr', 'KitchenAbvGr', 'KitchenQual',
       'TotRmsAbvGrd', 'Functional', 'Fireplaces', 'FireplaceQu', 'GarageType',
       'GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual',
       'GarageCond', 'PavedDrive', 'WoodDeckSF', 'OpenPorchSF',
       'EnclosedPorch', '3SsnPorch', 'ScreenPorch', 'PoolArea', 'PoolQC',
       'Fence', 'MiscFeature', 'MiscVal', 'MoSold', 'YrSold', 'SaleType',
       'SaleCondition', 'SalePrice'],
      dtype='object')
```

```
#descriptive statistics summary
df_train['SalePrice'].describe()
```

```
count      1460.000000
mean     180921.195890
std       79442.502883
min       34900.000000
25%      129975.000000
50%      163000.000000
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

```
#histogram
sns.distplot(df_train['SalePrice']);
```



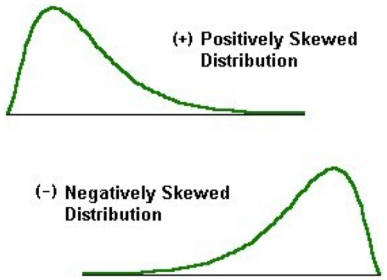
Pandas Skewness : https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.skew.html
Pandas Kurtosis : https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.kurt.html

```
#skewness and kurtosis
print("Skewness: %f" % df_train['SalePrice'].skew())
print("Kurtosis: %f" % df_train['SalePrice'].kurt())
```

```
Skewness: 1.882876
Kurtosis: 6.536282
```

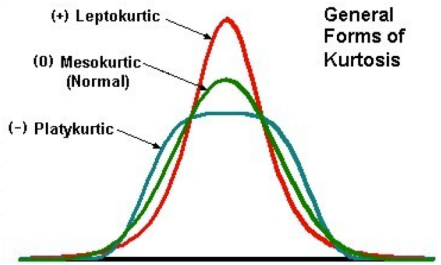
1. 왜도(skewness)

자료의 분포모양이 평균을 중심으로부터 한 쪽으로 치우쳐져 있는 경향을 나타내는 척도.
확률분포곡선에서 비대칭의 정도를 나타내는 측도.
: a=0 정규분포
: a>0 좌측으로 치우침.
: a<0 우측으로 치우침.



2. 첨도(kurtosis)

자료의 분포모양이 정규분포보다 더 중앙에 집중하는가를 나타내는 척도.
: a=3 정규분포
: a>3 뾰쪽함
: a<3 평평함



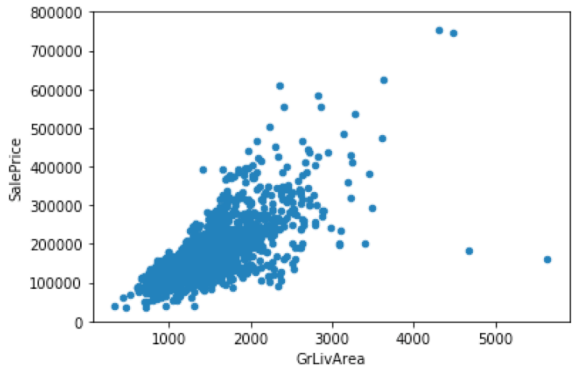
Comprehensive data exploration with Python

Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

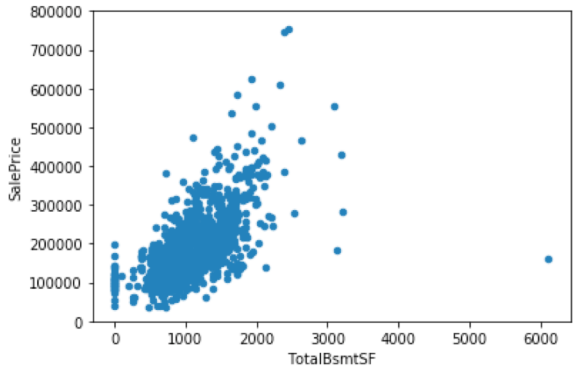
https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

3. 데이터 상관관계 확인

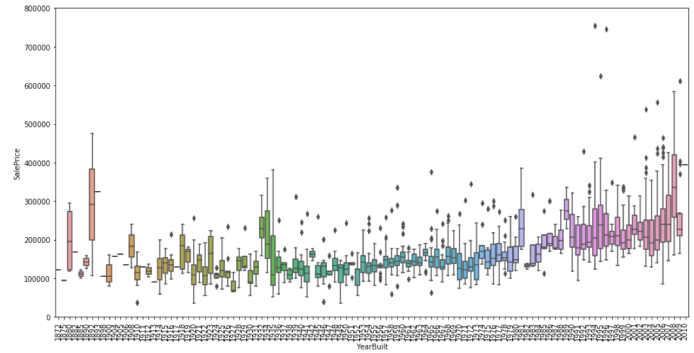
```
#scatter plot grlivarea/saleprice
var = 'GrLivArea'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```



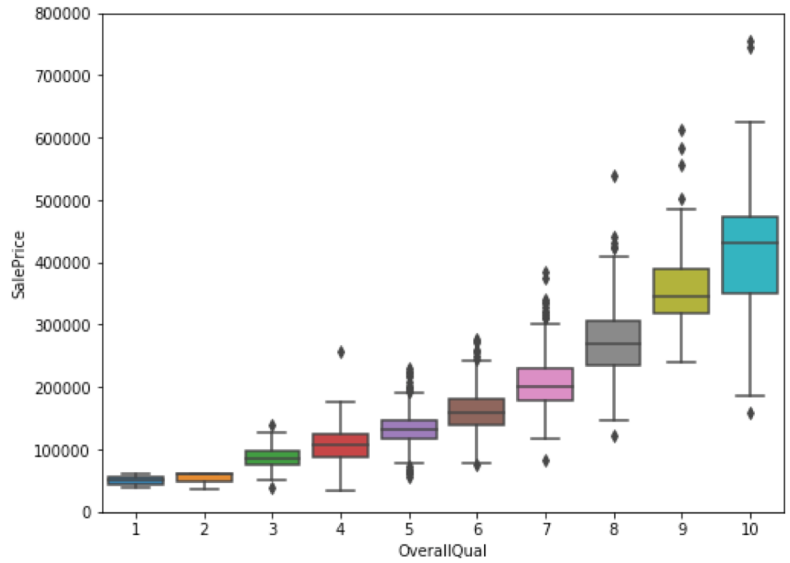

```
#scatter plot totalbsmtsf/saleprice
var = 'TotalBsmtSF'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```




```
var = 'YearBuilt'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
f, ax = plt.subplots(figsize=(16, 8))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
plt.xticks(rotation=90);
```



```
#box plot overallqual/saleprice
var = 'OverallQual'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
f, ax = plt.subplots(figsize=(8, 6))
fig = sns.boxplot(x=var, y="SalePrice", data=data)
fig.axis(ymin=0, ymax=800000);
```



Comprehensive data exploration with Python

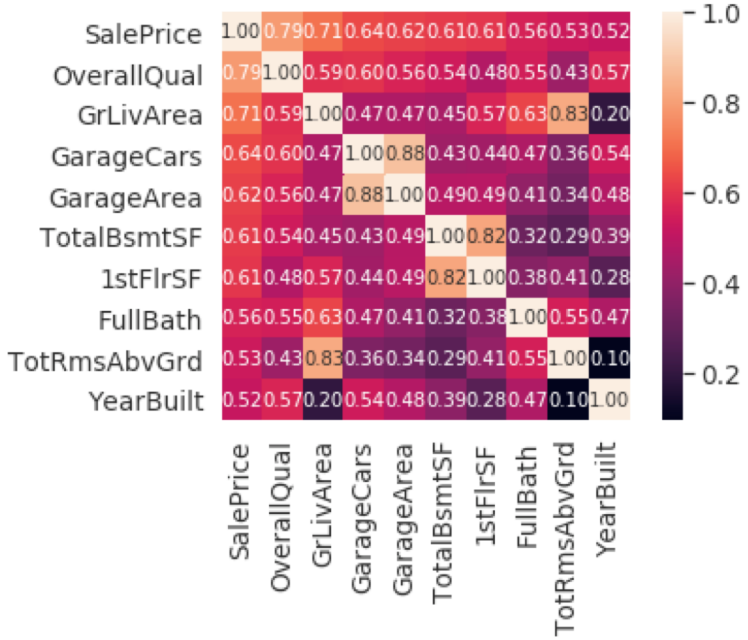
Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

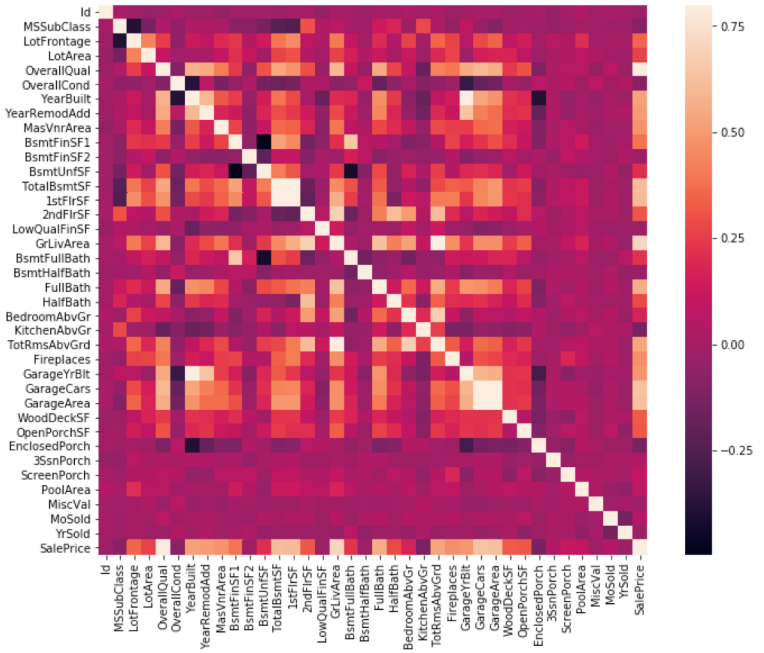
3. 데이터 상관관계 확인

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

```
#saleprice correlation matrix
k = 10 #number of variables for heatmap
cols = corrmat.nlargest(k, 'SalePrice')['SalePrice'].index
cm = np.corrcoef(df_train[cols].values.T)
sns.set(font_scale=1.25)
hm = sns.heatmap(cm, cbar=True, annot=True, square=True, fmt='.2f', annot_kws={'size': 10}, yticklabels=cols.values, xticklabels=cols.values)
plt.show()
```




```
#correlation matrix
corrmat = df_train.corr()
f, ax = plt.subplots(figsize=(12, 9))
sns.heatmap(corrmat, vmax=.8, square=True);
```



Comprehensive data exploration with Python

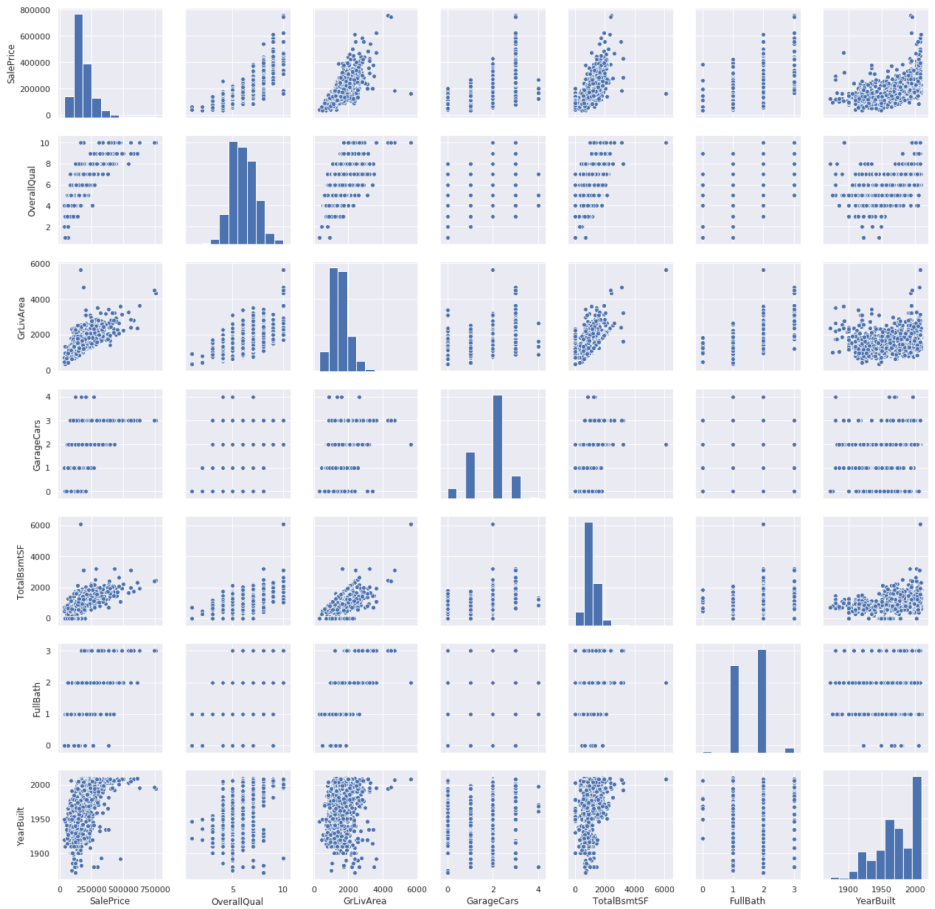
Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

3. 데이터 상관관계 확인

Sns pairplot : https://seaborn.pydata.org/generated/seaborn.pairplot.html
Sns pairplot : https://pinkwink.kr/986

```
#scatterplot
sns.set()
cols = ['SalePrice', 'OverallQual', 'GrLivArea', 'GarageCars', 'TotalBsmtSF', 'FullBath', 'YearBuilt']
sns.pairplot(df_train[cols], size = 2.5)
plt.show();
```



Although we already know some of the main figures, this mega scatter plot gives us a reasonable idea about variables relationships.

One of the figures we may find interesting is the one between 'TotalBsmtSF' and 'GrLiveArea'. In this figure we can see the dots drawing a linear line, which almost acts like a border. It totally makes sense that the majority of the dots stay below that line. Basement areas can be equal to the above ground living area, but it is not expected a basement area bigger than the above ground living area (unless you're trying to buy a bunker).

The plot concerning 'SalePrice' and 'YearBuilt' can also make us think. In the bottom of the 'dots cloud', we see what almost appears to be a shy exponential function (be creative). We can also see this same tendency in the upper limit of the 'dots cloud' (be even more creative). Also, notice how the set of dots regarding the last years tend to stay above this limit (I just wanted to say that prices are increasing faster now).

Comprehensive data exploration with Python

Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

4. 데이터 공백(Null) 확인

```
#missing data
total = df_train.isnull().sum().sort_values(ascending=False)
percent = (df_train.isnull().sum()/df_train.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

	Total	Percent
PoolQC	1453	0.995205
MiscFeature	1406	0.963014
Alley	1369	0.937671
Fence	1179	0.807534
FireplaceQu	690	0.472603
LotFrontage	259	0.177397
GarageCond	81	0.055479
GarageType	81	0.055479
GarageYrBlt	81	0.055479
GarageFinish	81	0.055479
GarageQual	81	0.055479
BsmtExposure	38	0.026027
BsmtFinType2	38	0.026027
BsmtFinType1	37	0.025342
BsmtCond	37	0.025342
BsmtQual	37	0.025342
MasVnrArea	8	0.005479
MasVnrType	8	0.005479
Electrical	1	0.000685
Utilities	0	0.000000

Let's analyse this to understand how to handle the missing data.

We'll consider that when more than 15% of the data is missing, we should delete the corresponding variable and pretend it never existed. This means that we will not try any trick to fill the missing data in these cases. According to this, there is a set of variables (e.g. 'PoolQC', 'MiscFeature', 'Alley', etc.) that we should delete. The point is: will we miss this data? I don't think so. None of these variables seem to be very important, since most of them are not aspects in which we think about when buying a house (maybe that's the reason why data is missing?). Moreover, looking closer at the variables, we could say that variables like 'PoolQC', 'MiscFeature' and 'FireplaceQu' are strong candidates for outliers, so we'll be happy to delete them.

In what concerns the remaining cases, we can see that 'Garage*X*' variables have the same number of missing data. I bet missing data refers to the same set of observations (although I will not check it; it's just 5% and we should not spend $20 in 5$ problems). Since the most important information regarding garages is expressed by 'GarageCars' and considering that we are just talking about 5% of missing data, I'll delete the mentioned 'Garage*X*' variables. The same logic applies to 'Bsmt*X*' variables.

Regarding 'MasVnrArea' and 'MasVnrType', we can consider that these variables are not essential. Furthermore, they have a strong correlation with 'YearBuilt' and 'OverallQual' which are already considered. Thus, we will not lose information if we delete 'MasVnrArea' and 'MasVnrType'.

Finally, we have one missing observation in 'Electrical'. Since it is just one observation, we'll delete this observation and keep the variable.

In summary, to handle missing data, we'll delete all the variables with missing data, except the variable 'Electrical'. In 'Electrical' we'll just delete the observation with missing data.

Comprehensive data exploration with Python

Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

4. 데이터 공백(Null) 삭제

```
#missing data
total = df_train.isnull().sum().sort_values(ascending=False)
percent = (df_train.isnull().sum()/df_train.isnull().count()).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
missing_data.head(20)
```

	Total	Percent
PoolQC	1453	0.995205
MiscFeature	1406	0.963014
Alley	1369	0.937671
Fence	1179	0.807534
FireplaceQu	690	0.472603
LotFrontage	259	0.177397
GarageCond	81	0.055479
GarageType	81	0.055479
GarageYrBlt	81	0.055479
GarageFinish	81	0.055479
GarageQual	81	0.055479
BsmtExposure	38	0.026027
BsmtFinType2	38	0.026027
BsmtFinType1	37	0.025342
BsmtCond	37	0.025342
BsmtQual	37	0.025342
MasVnrArea	8	0.005479
MasVnrType	8	0.005479
Electrical	1	0.000685
Utilities	0	0.000000

```
#dealing with missing data
df_train = df_train.drop((missing_data[missing_data['Total'] > 1]).index,1)
df_train = df_train.drop(df_train.loc[df_train['Electrical'].isnull()].index)
df_train.isnull().sum().max() #just checking that there's no missing data missing...
```

```
0
```

Comprehensive data exploration with Python

Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

5. Outlier 데이터 확인

Python standardScaler : https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
Python standardScaler : https://rfriend.tistory.com/tag/StandardScaler%28data%29.fit_transform%28data%29

```
#standardizing data
saleprice_scaled = StandardScaler().fit_transform(df_train['SalePrice'][:,np.newaxis]);
low_range = saleprice_scaled[saleprice_scaled[:,0].argsort()][:10]
high_range= saleprice_scaled[saleprice_scaled[:,0].argsort()][-10:]
print('outer range (low) of the distribution:')
print(low_range)
print('\nouter range (high) of the distribution:')
print(high_range)
```

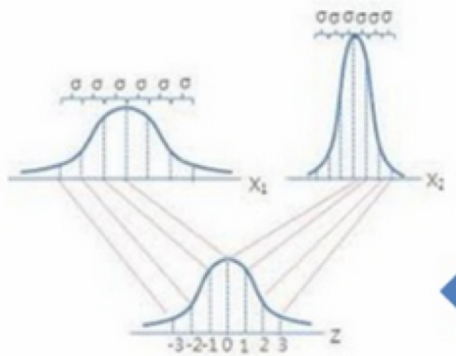
```
outer range (low) of the distribution:
[[-1.83820775]
 [-1.83303414]
 [-1.80044422]
 [-1.78282123]
 [-1.77400974]
 [-1.62295562]
 [-1.6166617 ]
 [-1.58519209]
 [-1.58519209]
 [-1.57269236]]

outer range (high) of the distribution:
[[3.82758058]
 [4.0395221 ]
 [4.49473628]
 [4.70872962]
 [4.728631  ]
 [5.06034585]
 [5.42191907]
 [5.58987866]
 [7.10041987]
 [7.22629831]]
```

python **표준정규분포 표준화**

(Standardization, mean removal and variance scaling)

- (1) numpy : z = (x - mean())/std()
- (2) scipy.stats : zscore()
- (3) sklearn.preprocessing : StandardScaler().fit_transform()



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

Comprehensive data exploration with Python

Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

- **Bivariate**는 변수가 2개
- **Univariate**는 독립변수가 여러 개라는 뜻
- **Multivariate**는 종속변수가 여러 개라는 뜻

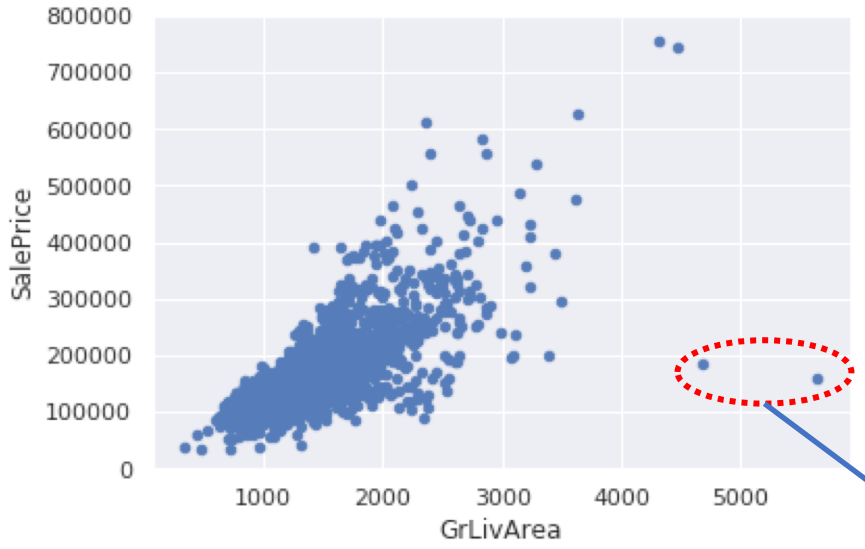
6. Bivariate analysis

Bivariate analysis : http://www.statedu.com/QnA/79573
독립변수, 종속변수 : https://drhongdatanote.tistory.com/14

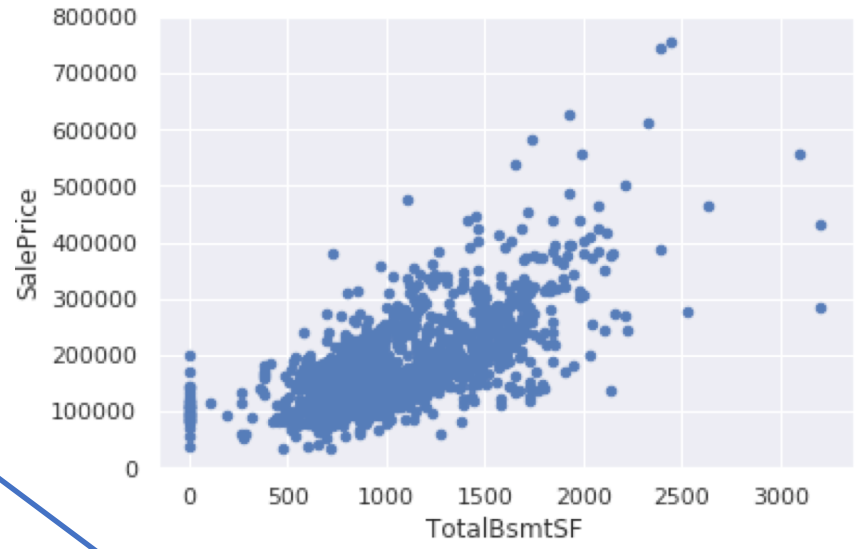
- **독립변수**란 실험에서 실험자가 직접 변경하는 변수
- **종속변수**란 독립변수의 값이 변함에 따라 달라지는 변수

https://ko.khanacademy.org/math/cc-sixth-grade-math/cc-6th-equations-and-inequalities/cc-6th-dependent-independent/a/dependent-and-independent-variables-review

```
#bivariate analysis saleprice/grlivarea
var = 'GrLivArea'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```




```
#bivariate analysis saleprice/grlivarea
var = 'TotalBsmtSF'
data = pd.concat([df_train['SalePrice'], df_train[var]], axis=1)
data.plot.scatter(x=var, y='SalePrice', ylim=(0,800000));
```



Outlier 삭제

```
#deleting points
df_train.sort_values(by = 'GrLivArea', ascending = False)[:2]
df_train = df_train.drop(df_train[df_train['Id'] == 1299].index)
df_train = df_train.drop(df_train[df_train['Id'] == 524].index)
```

Comprehensive data exploration with Python

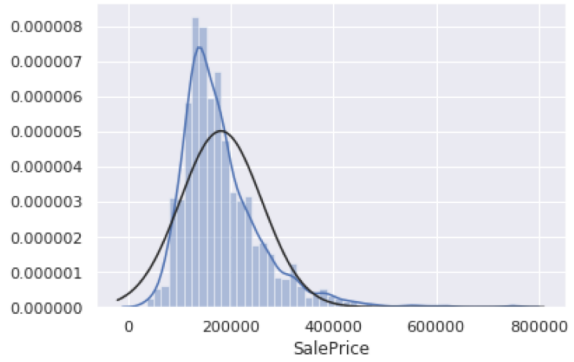
Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

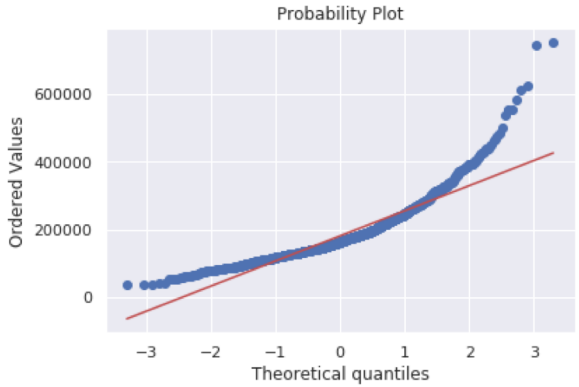
https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

6. Bivariate analysis

seaborn.distplot : https://seaborn.pydata.org/generated/seaborn.distplot.html
stats.probplot : https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.probplot.html
https://datascienceschool.net/view-notebook/76acc92d28354e86940001f9fe85c50f/
np.log : https://docs.scipy.org/doc/numpy/reference/generated/numpy.log.html

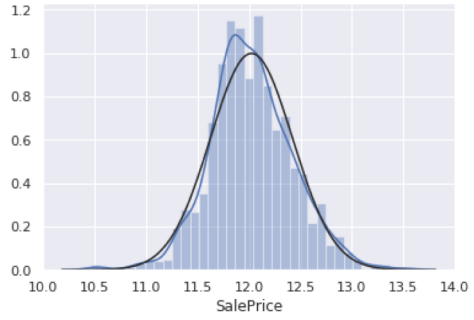
```
#histogram and normal probability plot
sns.distplot(df_train['SalePrice'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['SalePrice'], plot=plt)
```

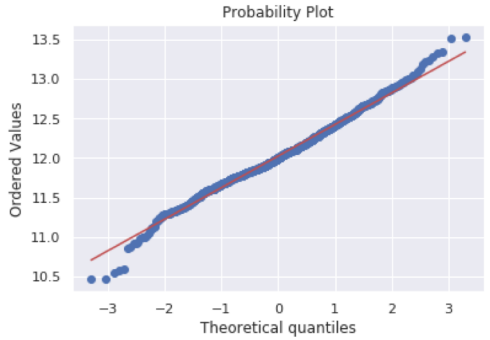





```
#applying log transformation
df_train['SalePrice'] = np.log(df_train['SalePrice'])
```

```
#transformed histogram and normal probability plot
sns.distplot(df_train['SalePrice'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['SalePrice'], plot=plt)
```





로그를 취하는 이유

- https://leebaro.tistory.com/entry/데이터-분석-시-로그를-취하는-이유
- https://rfriend.tistory.com/295
- https://datascienceschool.net/view-notebook/afb99de8cc0d407ba32079590b25180d/

Comprehensive data exploration with Python

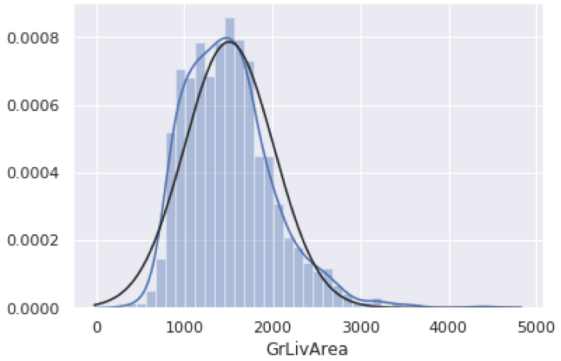
Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

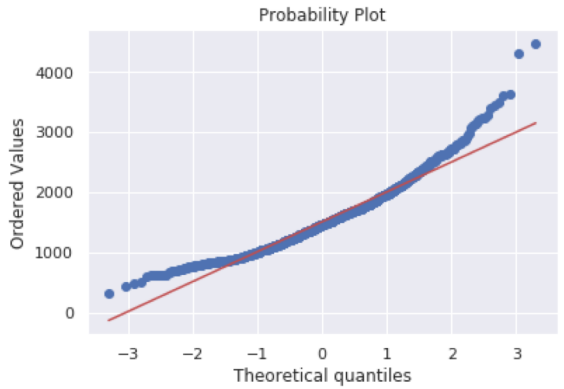
https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

6. Bivariate analysis

seaborn.distplot : https://seaborn.pydata.org/generated/seaborn.distplot.html
stats.probplot : https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.probplot.html
https://datascienceschool.net/view-notebook/76acc92d28354e86940001f9fe85c50f/
np.log : https://docs.scipy.org/doc/numpy/reference/generated/numpy.log.html

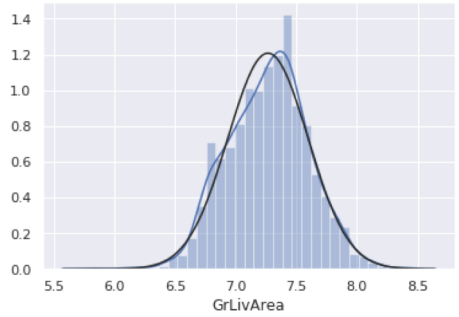
```
#histogram and normal probability plot
sns.distplot(df_train['GrLivArea'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['GrLivArea'], plot=plt)
```

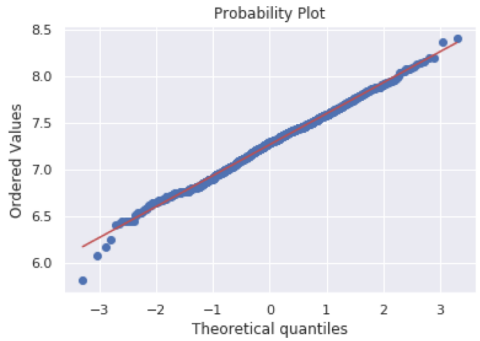





```
#data transformation
df_train['GrLivArea'] = np.log(df_train['GrLivArea'])
```

```
#transformed histogram and normal probability plot
sns.distplot(df_train['GrLivArea'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['GrLivArea'], plot=plt)
```





로그를 취하는 이유

- https://leebaro.tistory.com/entry/데이터-분석-시-로그를-취하는-이유
- https://rfriend.tistory.com/295
- https://datascienceschool.net/view-notebook/afb99de8cc0d407ba32079590b25180d/

Comprehensive data exploration with Python

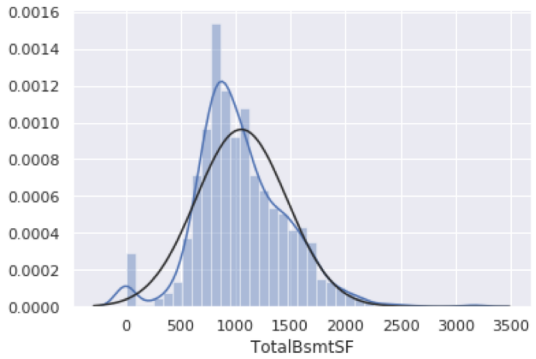
Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

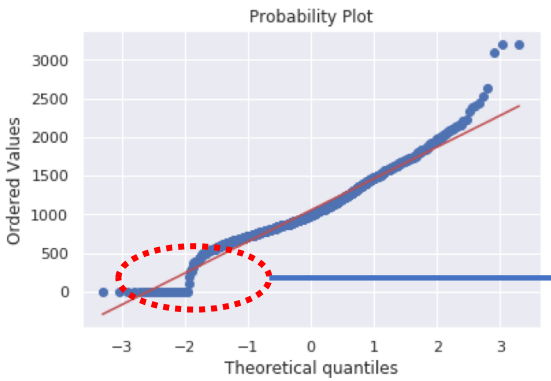
https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

seaborn.distplot : https://seaborn.pydata.org/generated/seaborn.distplot.html
stats.probplot : https://docs.scipy.org/doc/scipy-0.14.0/reference/generated/scipy.stats.probplot.html
https://datascienceschool.net/view-notebook/76acc92d28354e86940001f9fe85c50f/
np.log : https://docs.scipy.org/doc/numpy/reference/generated/numpy.log.html

6. Bivariate analysis

```
#histogram and normal probability plot
sns.distplot(df_train['TotalBsmtSF'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train['TotalBsmtSF'], plot=plt)
```





Ok, now we are dealing with the big boss. What do we have here?

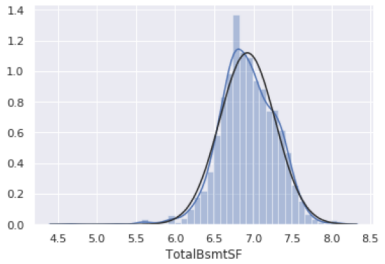
- Something that, in general, presents skewness.
- A significant number of observations with value zero (houses without basement).
- A big problem because the value zero doesn't allow us to do log transformations.

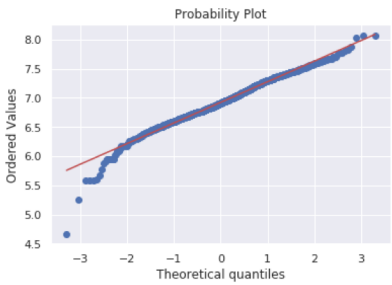
To apply a log transformation here, we'll create a variable that can get the effect of having or not having basement (binary variable). Then, we'll do a log transformation to all the non-zero observations, ignoring those with value zero. This way we can transform data, without losing the effect of having or not basement.

```
#create column for new variable (one is enough because it's a binary categorical feature)
#if area>0 it gets 1, for area==0 it gets 0
df_train['HasBsmt'] = pd.Series(len(df_train['TotalBsmtSF']), index=df_train.index)
df_train['HasBsmt'] = 0
df_train.loc[df_train['TotalBsmtSF']>0,'HasBsmt'] = 1
```

```
#transform data
df_train.loc[df_train['HasBsmt']==1,'TotalBsmtSF'] = np.log(df_train['TotalBsmtSF'])
```

```
#histogram and normal probability plot
sns.distplot(df_train[df_train['TotalBsmtSF']>0]['TotalBsmtSF'], fit=norm);
fig = plt.figure()
res = stats.probplot(df_train[df_train['TotalBsmtSF']>0]['TotalBsmtSF'], plot=plt)
```





Comprehensive data exploration with Python

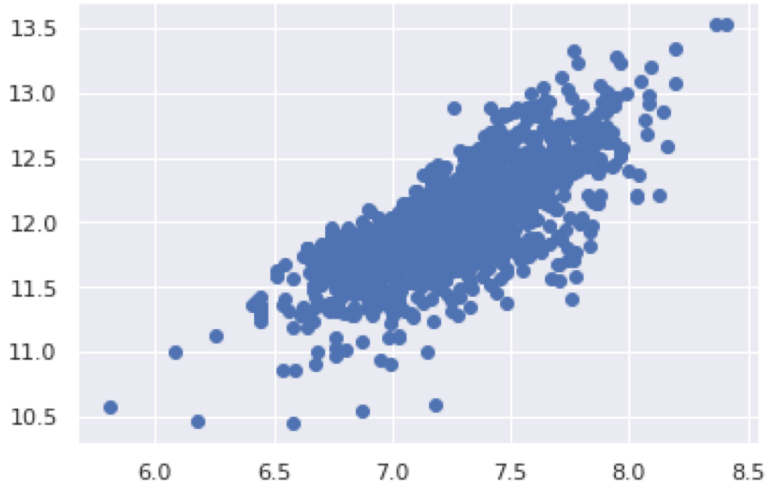
Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

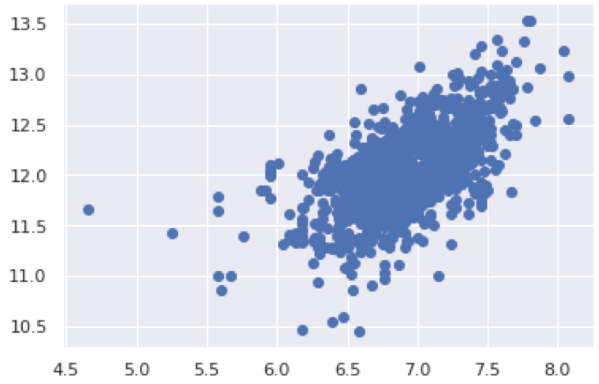
plt.scatter : https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html

6. Bivariate analysis

```
#scatter plot
plt.scatter(df_train['GrLivArea'], df_train['SalePrice']);
```




```
#scatter plot
plt.scatter(df_train[df_train['TotalBsmtSF']>0]['TotalBsmtSF'], df_train[df_train['TotalBsmtSF'
]>0]['SalePrice']);
```



Comprehensive data exploration with Python

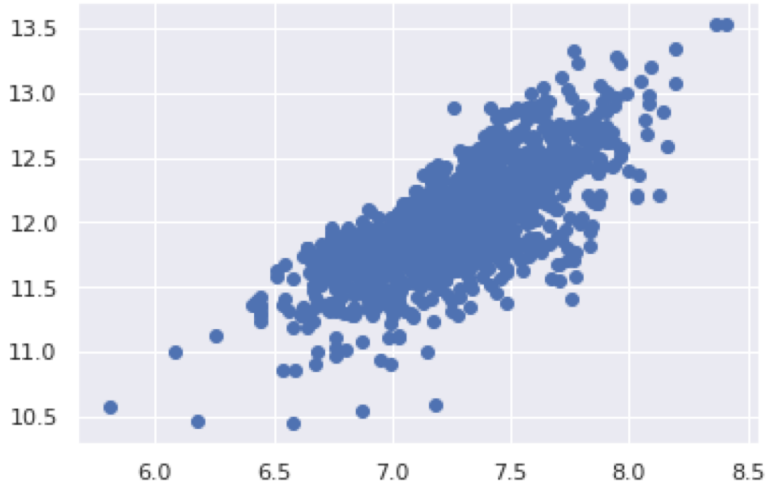
Python notebook using data from House Prices: Advanced Regression Techniques · 1,810,366 views · beginner, eda, data cleaning

https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

plt.scatter : https://matplotlib.org/api/_as_gen/matplotlib.pyplot.scatter.html

6. Bivariate analysis

```
#scatter plot
plt.scatter(df_train['GrLivArea'], df_train['SalePrice']);
```




```
#scatter plot
plt.scatter(df_train[df_train['TotalBsmtSF']>0]['TotalBsmtSF'], df_train[df_train['TotalBsmtSF'
]>0]['SalePrice']);
```