CARDIFF SCHOOL OF TECHNOLOGIES

4D Digidol | Data | Dylunio | Dyfodol

Cardiff Metropolitan University	
Cardiff School of Technologies	
Academic Year: 2022/2023	
Term: 1	
Module Name: Programming for Data Analysis	
Module Code: CIS7031	
Module Leader: Dr. Sandeep Singh Sengar	
MSc Programme: Data Science	
Assignment Title: Critical reflection report	
Student Name: Md Emdad Hussain Imu	**Student ID: ST 20247717**
Data Submitted: 20/12/2022	

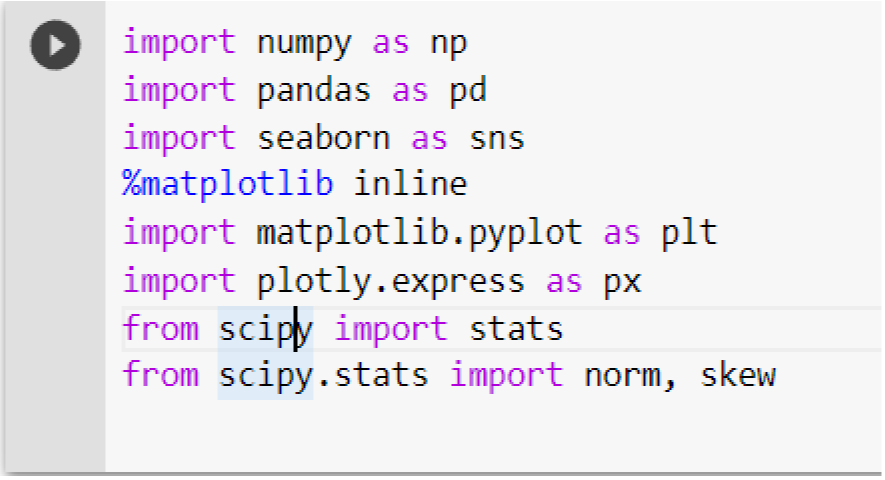
Contents

Introduction

In this write-up, we tackle the problem of predicting the sale price of houses located in Ames, Iowa, using 79 explanatory variables that explain almost every aspect of the house. It covers EDA on Iowa Housing Prices data from the Kaggle competition — House Prices: Advanced Regression Techniques. The goal of this project was to use EDA, visualization, data cleaning, preprocessing, and linear models to predict home prices given the features of the home, and interpret regression models to find out what features add value to a home.

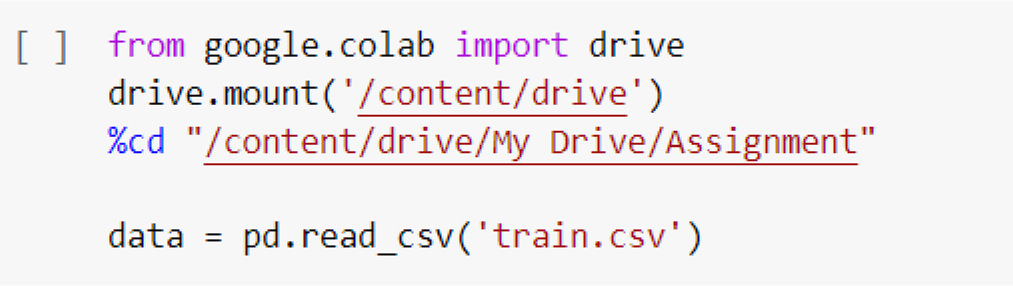
Variables Study

First of all, import necessary libraries

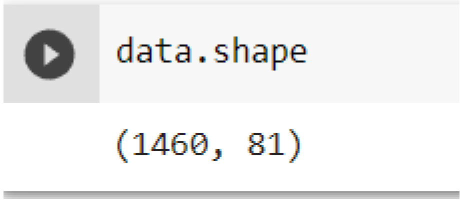


```
import numpy as np
import pandas as pd
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import plotly.express as px
from scipy import stats
from scipy.stats import norm, skew
```

Import data set from drive. There are two parts in this data set one is for training and another one for testing. But in this analysis training data only use.

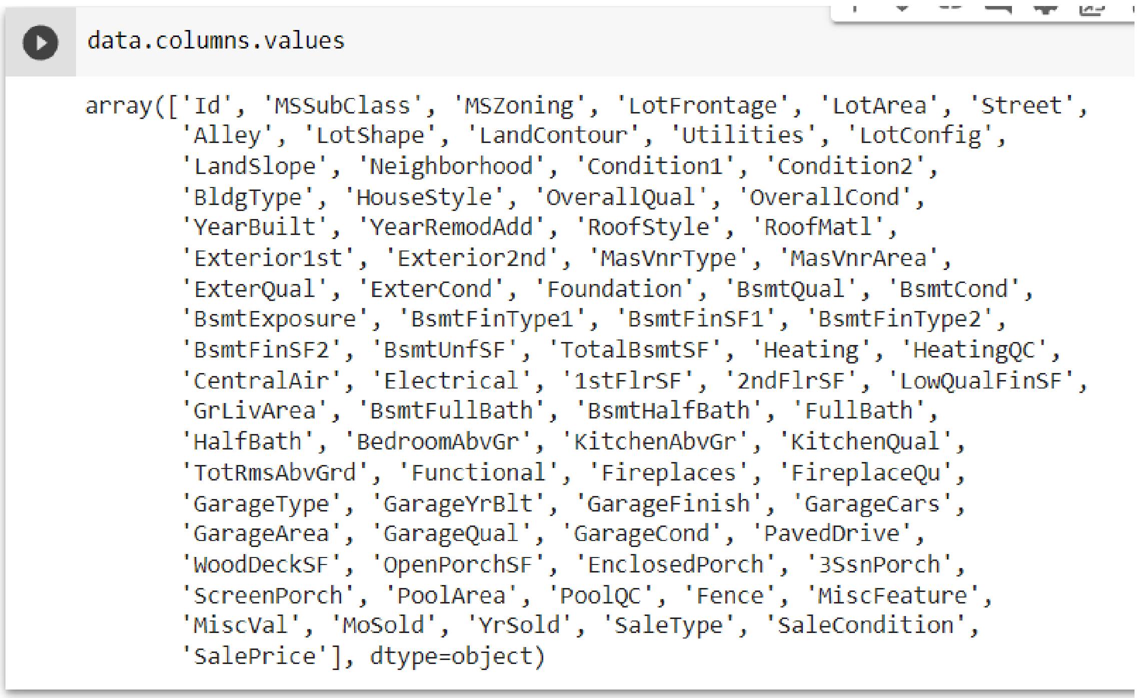


```
[ ]  from google.colab import drive
     drive.mount('/content/drive')
     %cd "/content/drive/My Drive/Assignment"

     data = pd.read_csv('train.csv')
```



```
data.shape

(1460, 81)
```

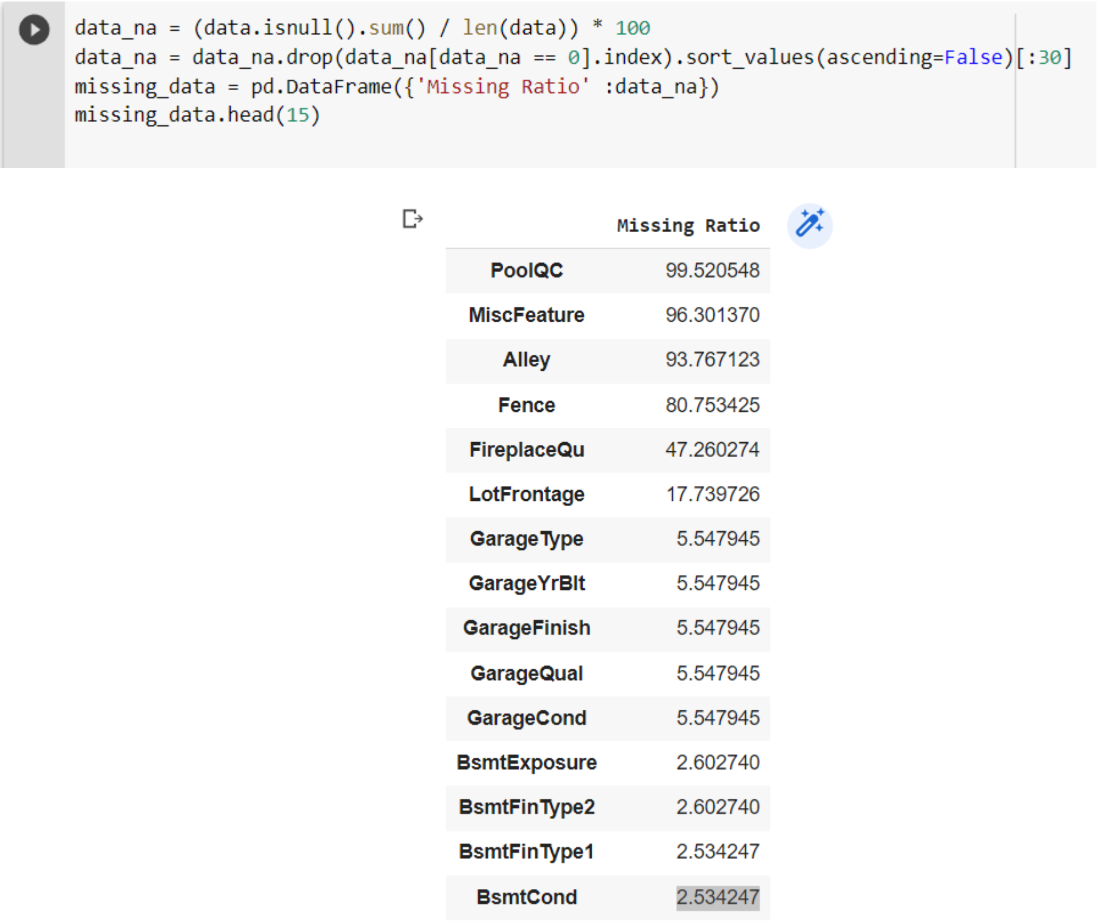
In this data set has 81 columns and 1460 items.



```
data.columns.values
```

```
array(['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)
```

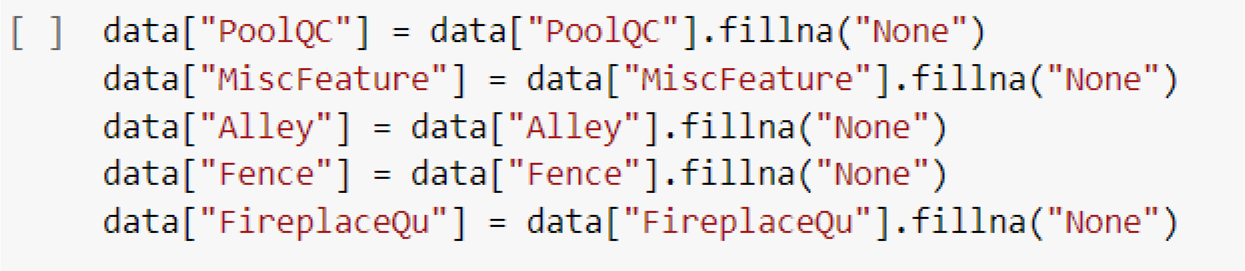
Null and missing value study



```
data_na = (data.isnull().sum() / len(data)) * 100
data_na = data_na.drop(data_na[data_na == 0].index).sort_values(ascending=False)[:30]
missing_data = pd.DataFrame({'Missing Ratio' :data_na})
missing_data.head(15)
```

	Missing Ratio
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

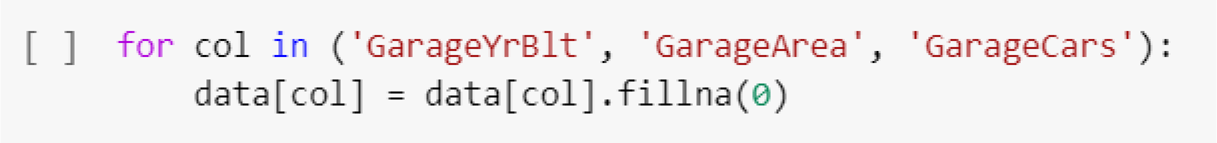
Let’s look at the top five rows of the data which has more missing values. The PoolQC, MiscFeature, Alley, Fence and FireplaceQu. All these data seem unwated in predicting the house prices. Let’s just fill the missing values with 0 or None and proceed with modeling.



```
[ ]  data["PoolQC"] = data["PoolQC"].fillna("None")
     data["MiscFeature"] = data["MiscFeature"].fillna("None")
     data["Alley"] = data["Alley"].fillna("None")
     data["Fence"] = data["Fence"].fillna("None")
     data["FireplaceQu"] = data["FireplaceQu"].fillna("None")
```

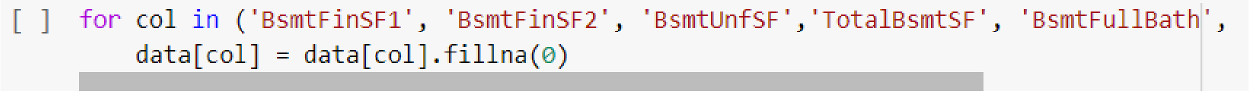
Next is LotFrontage. The dtype of LotFrontage is float. Hence it cannot fill None for missing values. In general, the LotFrontage of a house is mostly equal to the neighborhood houses. Hence, we group the data by "Neighborhood" and fill the missing values with median value of the neighborhood houses.

The dtype of the 'GarageYrBlt', 'GarageArea' and 'GarageCars' are integer. Hence filling the missing values with 0.



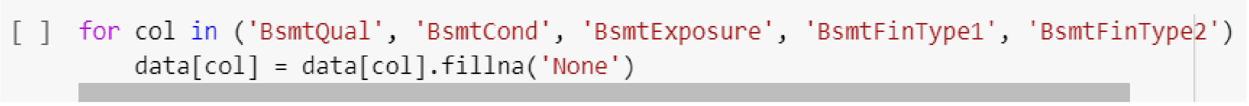
```
[ ]  for col in ('GarageYrBlt', 'GarageArea', 'GarageCars'):
         data[col] = data[col].fillna(0)
```

Now let’s see the variables related to Basement. The dtype of 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF','TotalBsmtSF', 'BsmtFullBath' and 'BsmtHalfBath' are all float. Hence, we fill the missing values with 0.



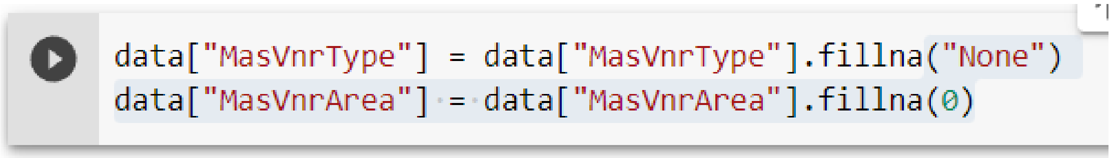
```
[ ]  for col in ('BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF','TotalBsmtSF', 'BsmtFullBath',
         data[col] = data[col].fillna(0)
```

The other variables related to Basement are 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1' and 'BsmtFinType2' which are categorical. Lets fill the missing values with "None"



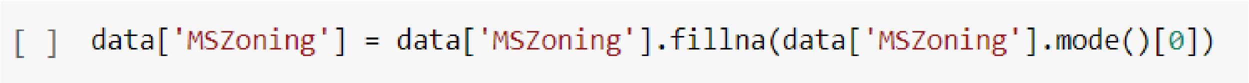
```
[ ]  for col in ('BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2')
         data[col] = data[col].fillna('None')
```

Next we take the Type and Area masonry veneer. Let’s fill 0 for missing value in area and "None" for missing value in type



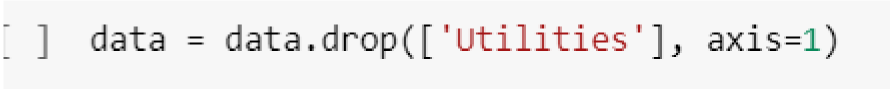
```
data["MasVnrType"] = data["MasVnrType"].fillna("None")
data["MasVnrArea"] = data["MasVnrArea"].fillna(0)
```

Now lets fill the missing values of the MSZoning with the most common value.



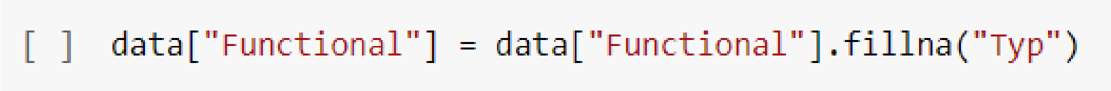
```
data['MSZoning'] = data['MSZoning'].fillna(data['MSZoning'].mode()[0])
```

If we look at the 'Utilities' column, almost all the column has the same value "AllPub", which means this variable is not going to help in prediction. Hence we drop this column.



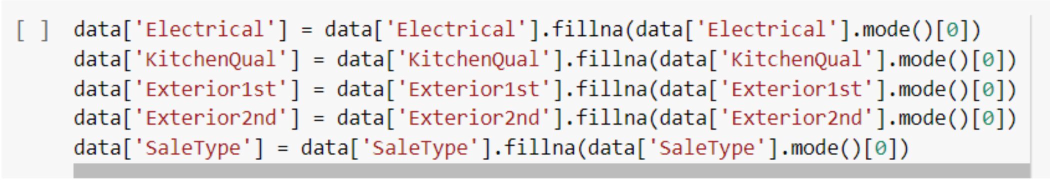
```
data = data.drop(['Utilities'], axis=1)
```

Lets fill the NA values of "Functional" column with Typical.

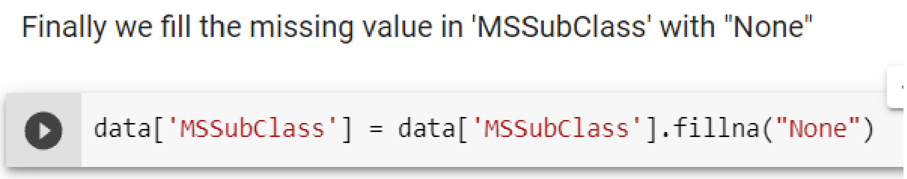


```
data["Functional"] = data["Functional"].fillna("Typ")
```

Now we fill the missing values in 'Electrical', 'KitchenQual', 'Exterior1st', 'Exterior2nd' and 'SaleType' with the most frequent value.



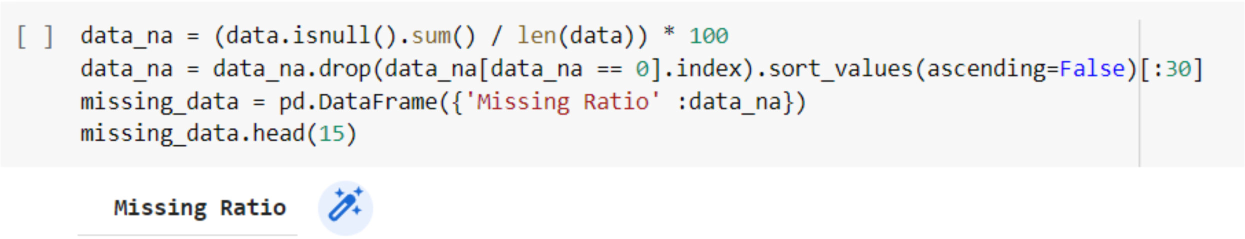
```
data['Electrical'] = data['Electrical'].fillna(data['Electrical'].mode()[0])
data['KitchenQual'] = data['KitchenQual'].fillna(data['KitchenQual'].mode()[0])
data['Exterior1st'] = data['Exterior1st'].fillna(data['Exterior1st'].mode()[0])
data['Exterior2nd'] = data['Exterior2nd'].fillna(data['Exterior2nd'].mode()[0])
data['SaleType'] = data['SaleType'].fillna(data['SaleType'].mode()[0])
```



Finally we fill the missing value in 'MSSubClass' with "None"

```
data['MSSubClass'] = data['MSSubClass'].fillna("None")
```

Now lets check whether there are any more missing value present in the dataset.



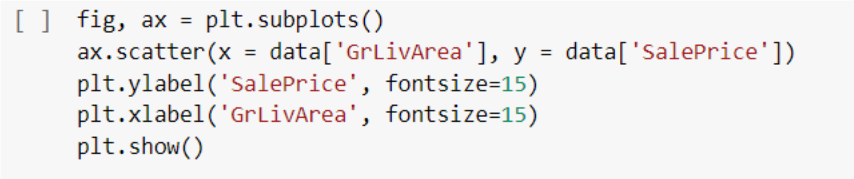
```
data_na = (data.isnull().sum() / len(data)) * 100
data_na = data_na.drop(data_na[data_na == 0].index).sort_values(ascending=False)[:30]
missing_data = pd.DataFrame({'Missing Ratio' :data_na})
missing_data.head(15)
```

```
Missing Ratio
```

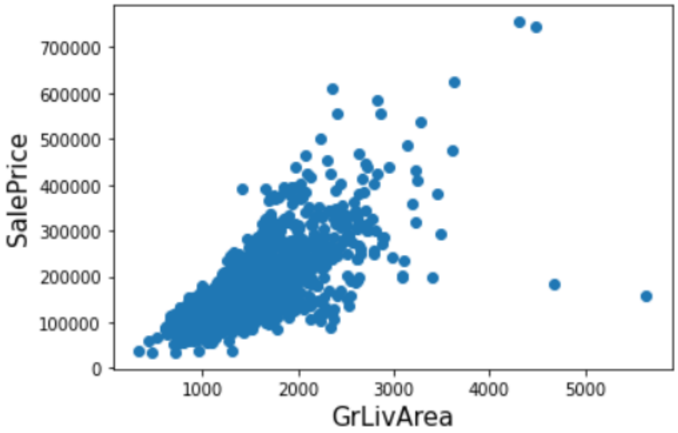
Done! There are no missing values present in the data set

EDA Analysis

The first thing we see when we are buying a house is the Living Area. So, lets analysis the Living area with our target variable SalePrice first.



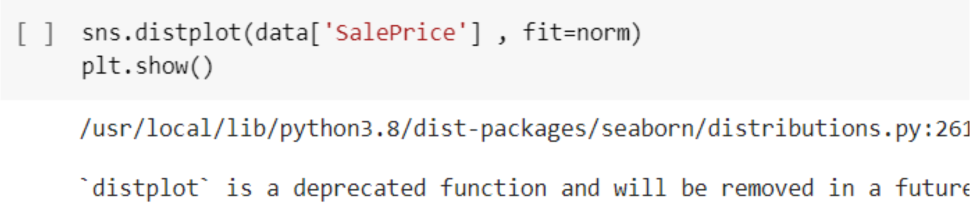
```
[ ]  fig, ax = plt.subplots()
     ax.scatter(x = data['GrLivArea'], y = data['SalePrice'])
     plt.ylabel('SalePrice', fontsize=15)
     plt.xlabel('GrLivArea', fontsize=15)
     plt.show()
```



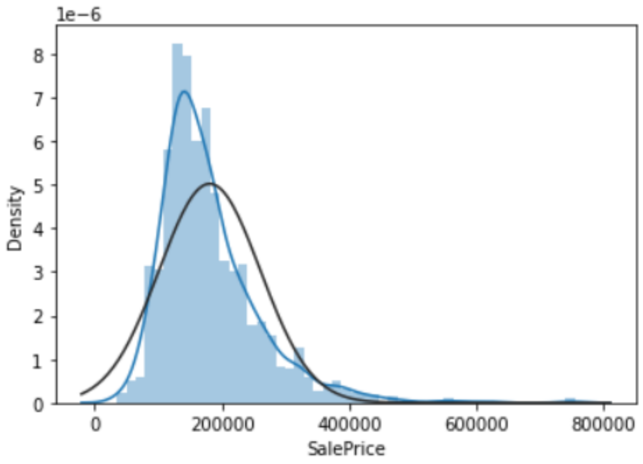
From the above plot, the Living area and Sale Price has a linear relationship.

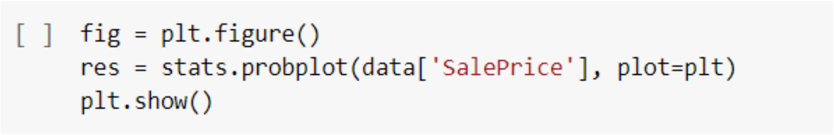
The two dots on the top shows the sudden increase in price. It may be because of the house located in the hot spot of the city.

Also note the two dots on the right side where the GrLivingArea is more and the SalePrice is very less. It could be the agricultural land nearby the city.

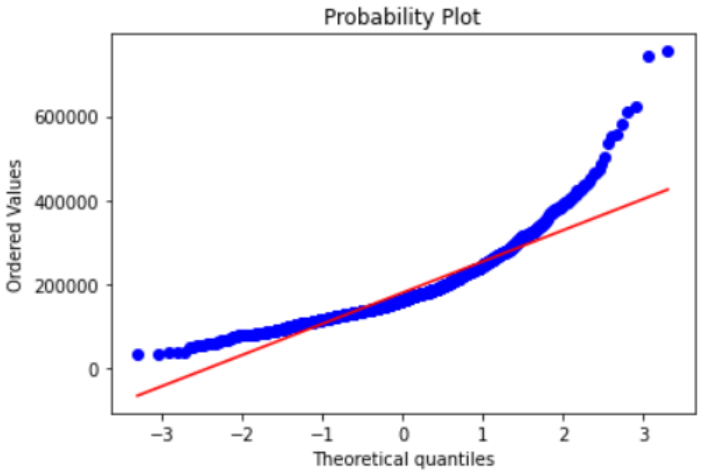


```
[ ]  sns.distplot(data['SalePrice'] , fit=norm)
     plt.show()

     /usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:261

     `distplot` is a deprecated function and will be removed in a future
```

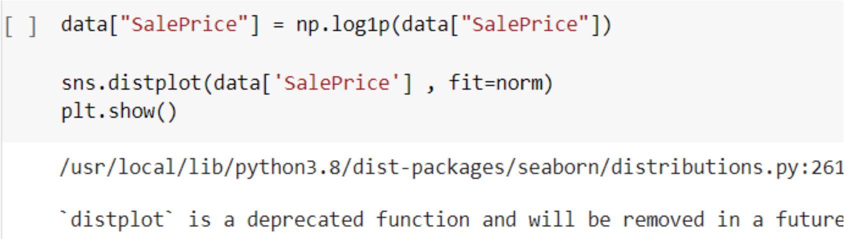




```
[ ]  fig = plt.figure()
     res = stats.probplot(data['SalePrice'], plot=plt)
     plt.show()
```

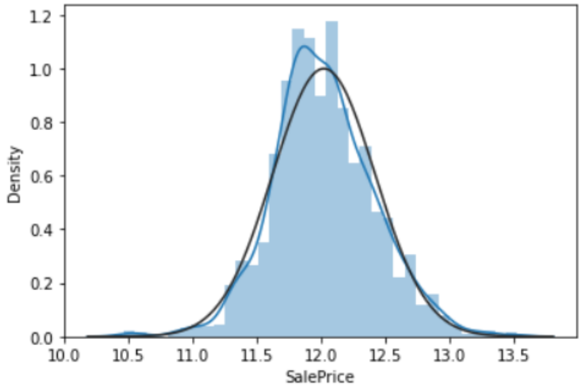


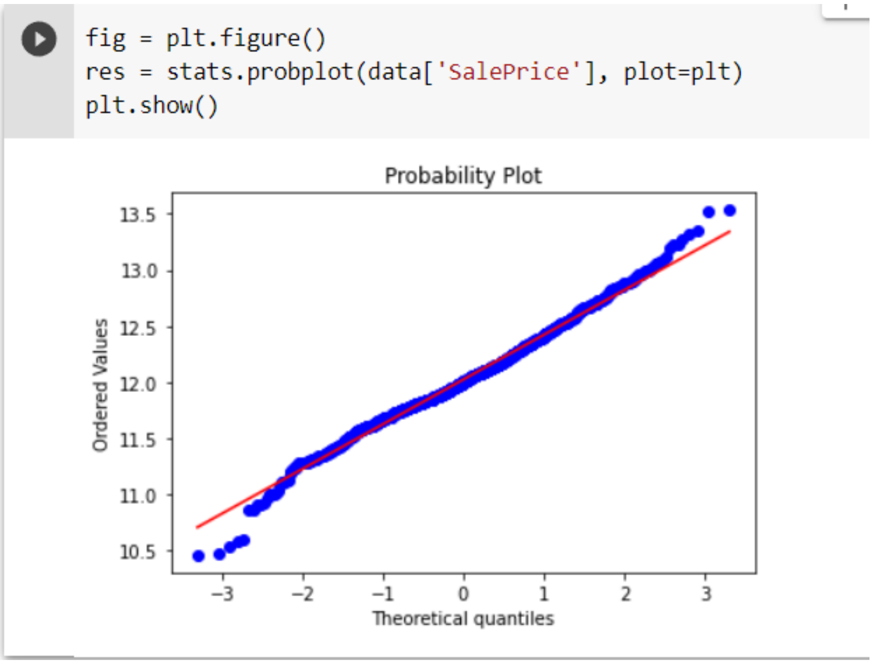
As we can see in the graph, the data is not normally distributed. The linear regression model we use needs the data to be normally distributed. Hence we need to transform the variable to normal distribution.



```
[ ]  data["SalePrice"] = np.log1p(data["SalePrice"])

     sns.distplot(data['SalePrice'] , fit=norm)
     plt.show()
```

```
/usr/local/lib/python3.8/dist-packages/seaborn/distributions.py:261

`distplot` is a deprecated function and will be removed in a future
```



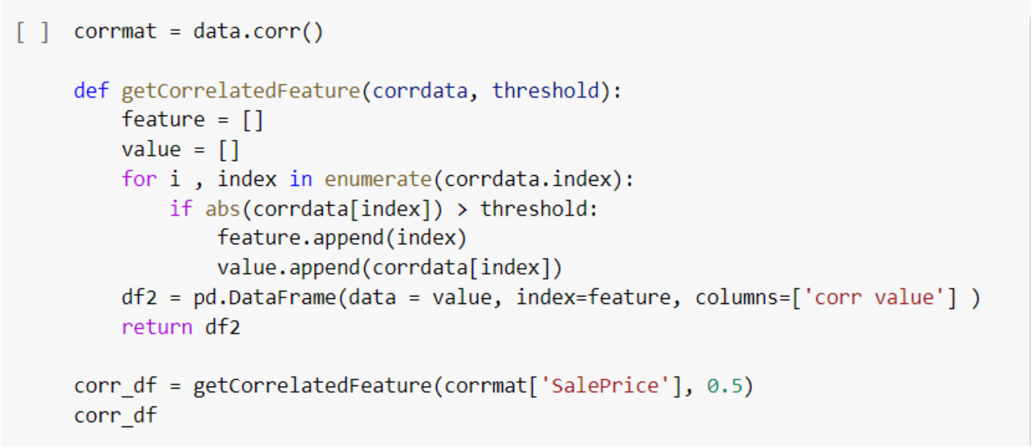


```
fig = plt.figure()
res = stats.probplot(data['SalePrice'], plot=plt)
plt.show()
```

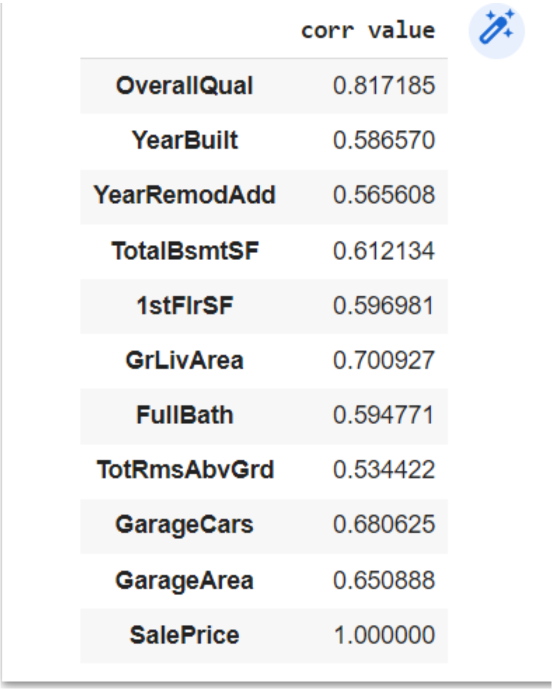
Just done log transformation to the 'SalePrice' variable.

Correlation Study

We have 80 columns in our data set. Visualizing all the data takes a lot of time. We just try to analyse the variables which are highly correlated (both positive and negative) with our target variable "SalePrice".

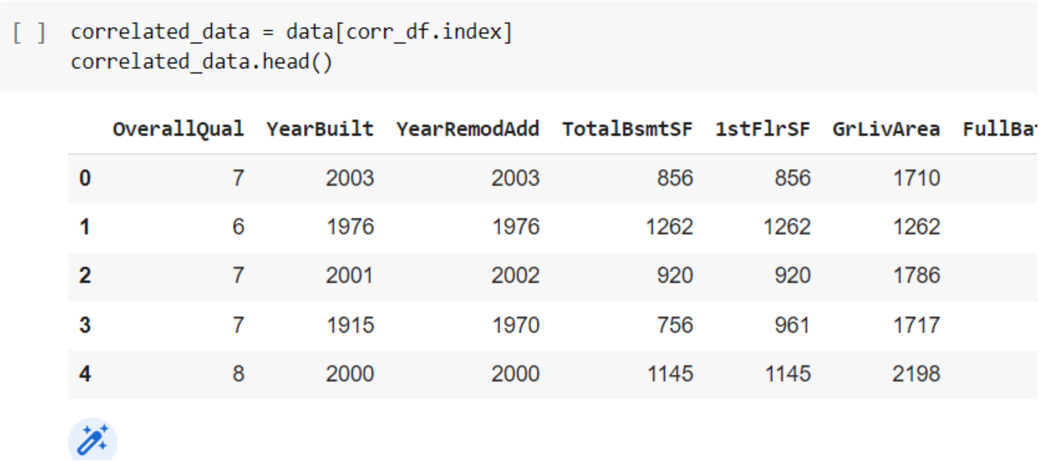


```
[ ]  corrmat = data.corr()

     def getCorrelatedFeature(corrdata, threshold):
         feature = []
         value = []
         for i , index in enumerate(corrdata.index):
             if abs(corrdata[index]) > threshold:
                 feature.append(index)
                 value.append(corrdata[index])
         df2 = pd.DataFrame(data = value, index=feature, columns=['corr value'] )
         return df2

     corr_df = getCorrelatedFeature(corrmat['SalePrice'], 0.5)
     corr_df
```



	corr value
OverallQual	0.817185
YearBuilt	0.586570
YearRemodAdd	0.565608
TotalBsmtSF	0.612134
1stFlrSF	0.596981
GrLivArea	0.700927
FullBath	0.594771
TotRmsAbvGrd	0.534422
GarageCars	0.680625
GarageArea	0.650888
SalePrice	1.000000

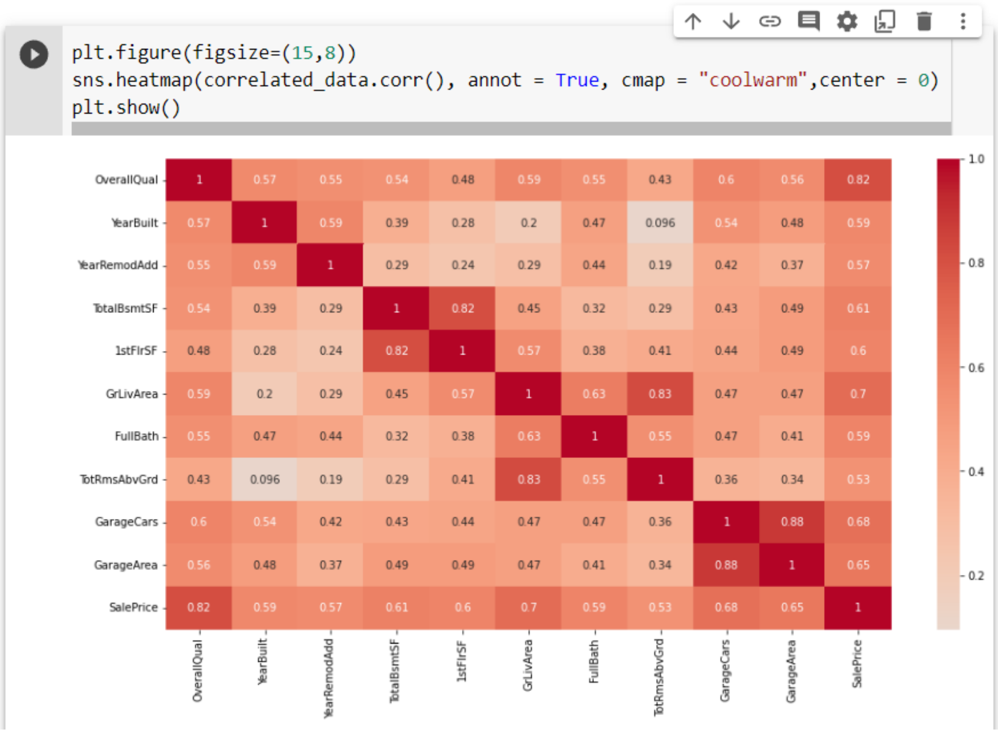
We just got all the variables which are highly correlated with our target variable. Now lets look at the values of the correlated data.



```
[ ]  correlated_data = data[corr_df.index]
     correlated_data.head()
```

	OverallQual	YearBuilt	YearRemodAdd	TotalBsmtSF	1stFlrSF	GrLivArea	FullBa
0	7	2003	2003	856	856	1710	
1	6	1976	1976	1262	1262	1262	
2	7	2001	2002	920	920	1786	
3	7	1915	1970	756	961	1717	
4	8	2000	2000	1145	1145	2198	

Lets look at the heatmap of the correlated data for a better picture.

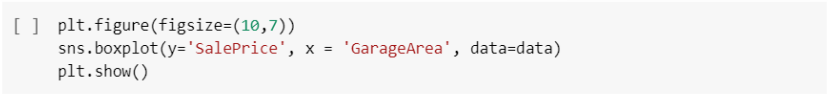


```
plt.figure(figsize=(15,8))
sns.heatmap(correlated_data.corr(), annot = True, cmap = "coolwarm",center = 0)
plt.show()
```

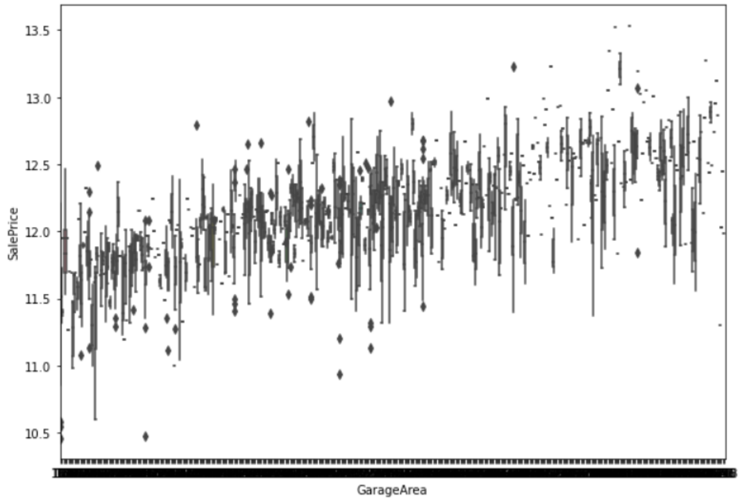
	OverallQual	YearBuilt	YearRemodAdd	TotalBsmtSF	1stFlrSF	GrLivArea	FullBath	TotRmsAbvGrd	GarageCars	GarageArea	SalePrice
OverallQual	1	0.57	0.55	0.54	0.48	0.59	0.55	0.43	0.6	0.56	0.82
YearBuilt	0.57	1	0.59	0.39	0.28	0.2	0.47	0.096	0.54	0.48	0.59
YearRemodAdd	0.55	0.59	1	0.29	0.24	0.29	0.44	0.19	0.42	0.37	0.57
TotalBsmtSF	0.54	0.39	0.29	1	0.82	0.45	0.32	0.29	0.43	0.49	0.61
1stFlrSF	0.48	0.28	0.24	0.82	1	0.57	0.38	0.41	0.44	0.49	0.6
GrLivArea	0.59	0.2	0.29	0.45	0.57	1	0.63	0.83	0.47	0.47	0.7
FullBath	0.55	0.47	0.44	0.32	0.38	0.63	1	0.55	0.47	0.41	0.59
TotRmsAbvGrd	0.43	0.096	0.19	0.29	0.41	0.83	0.55	1	0.36	0.34	0.53
GarageCars	0.6	0.54	0.42	0.43	0.44	0.47	0.47	0.36	1	0.88	0.68
GarageArea	0.56	0.48	0.37	0.49	0.49	0.47	0.41	0.34	0.88	1	0.65
SalePrice	0.82	0.59	0.57	0.61	0.6	0.7	0.59	0.53	0.68	0.65	1

From the above heatmap, GrageArea seems highly correlated with our target variable.

Since GarageArea is highly correated with traget variable, lets analyse it first.

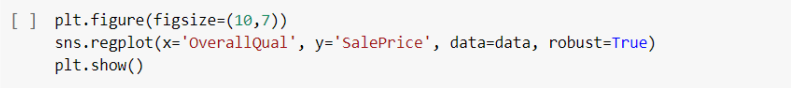


```
[ ] plt.figure(figsize=(10,7))
    sns.boxplot(y='SalePrice', x = 'GarageArea', data=data)
    plt.show()
```

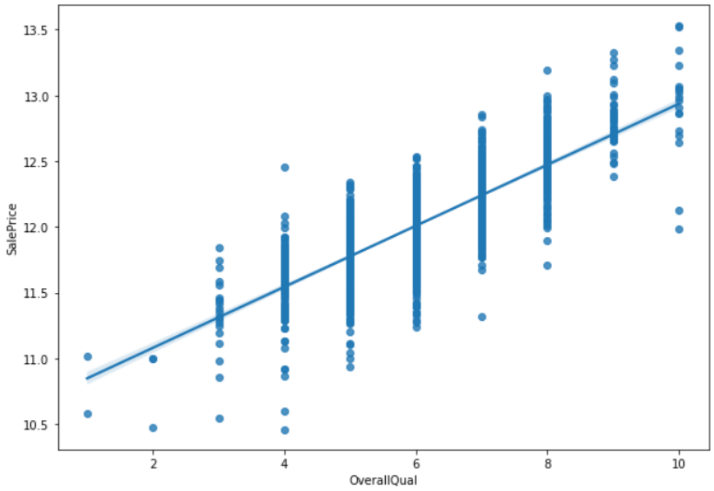


We can see a trend in data. The above plot clearly shows a linear relationship betweeb SalePrice and GarageArea. The SalePrice increases when the GarageArea increase.

Now lets analyse the next variable OverallQual since it is highly correlated with our target variable.

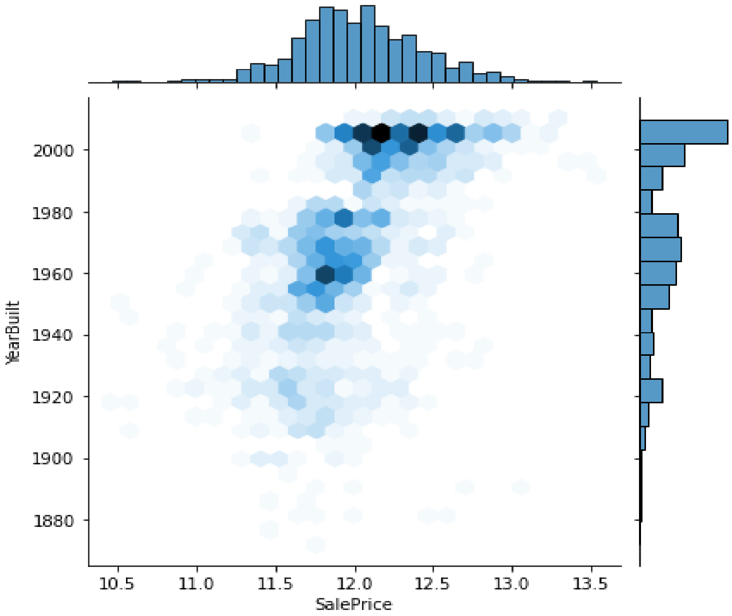


```
[ ]  plt.figure(figsize=(10,7))
     sns.regplot(x='OverallQual', y='SalePrice', data=data, robust=True)
     plt.show()
```



Obviously, there is a linear relationship between OverallQual and SalePrice.

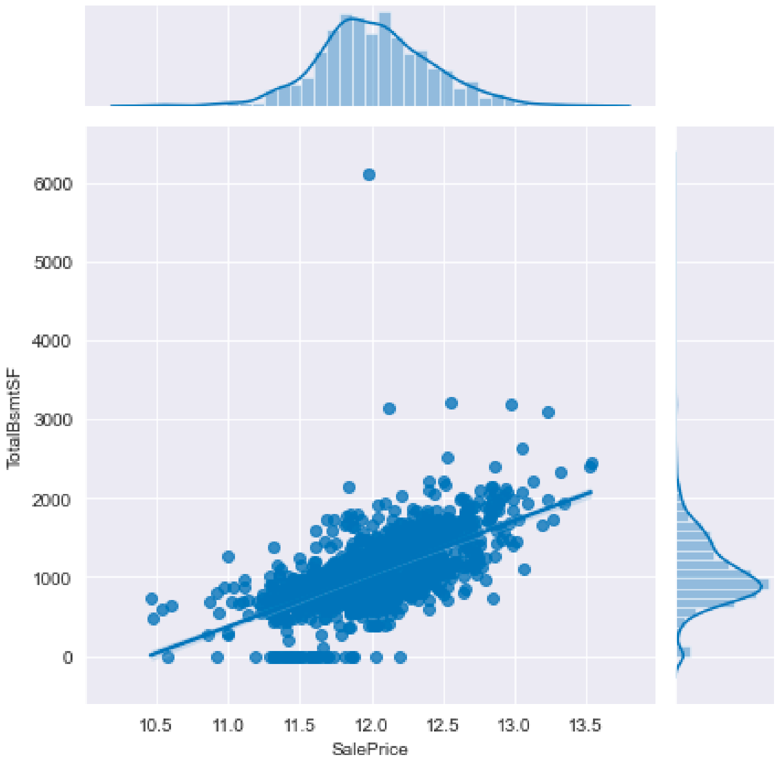
Now we consider the relationship between YearBuilt and SalePrice.



The Joint Grid plot shows us the univariate and bivariate plots of variables. The univariate plot of ‘YearBuilt’ shows that the distribution is skewed towards the year 2000 and has a long tail which extends till 1900. The linear relationship between the variables is clearer in cases of recently built houses.

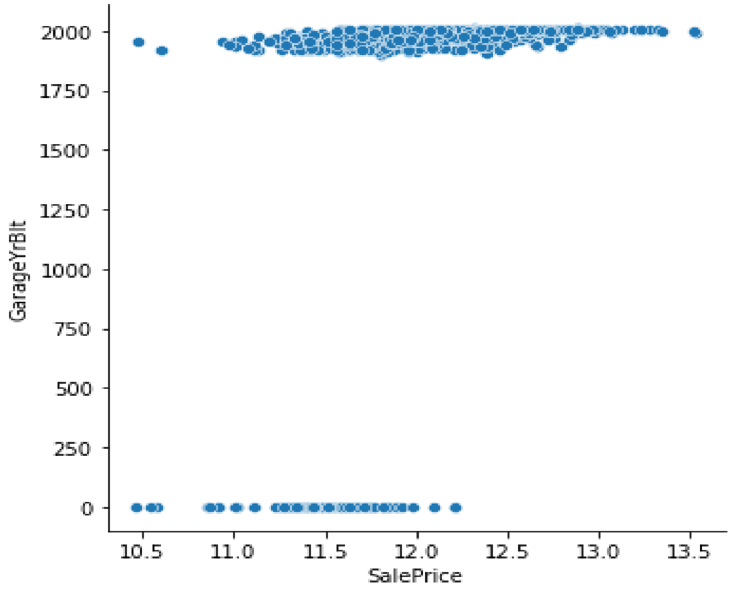
Next, analyze TotalBsmtSF.

```
<seaborn.axisgrid.JointGrid at 0x11fc70ee0>
```



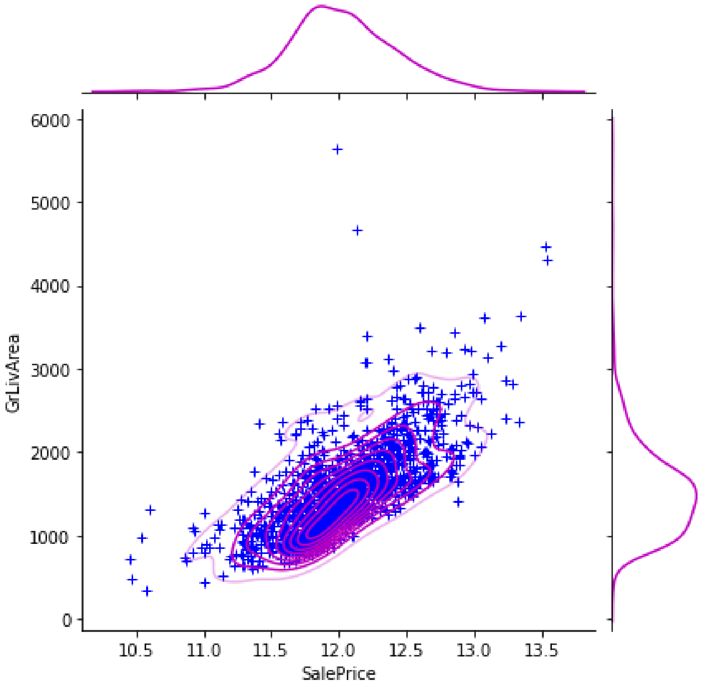
TotalBsmtSF is very highly correlated with our target variable SalePrice and also it follows a strong linear trend.

Next, we analyze GarageYrBlt.



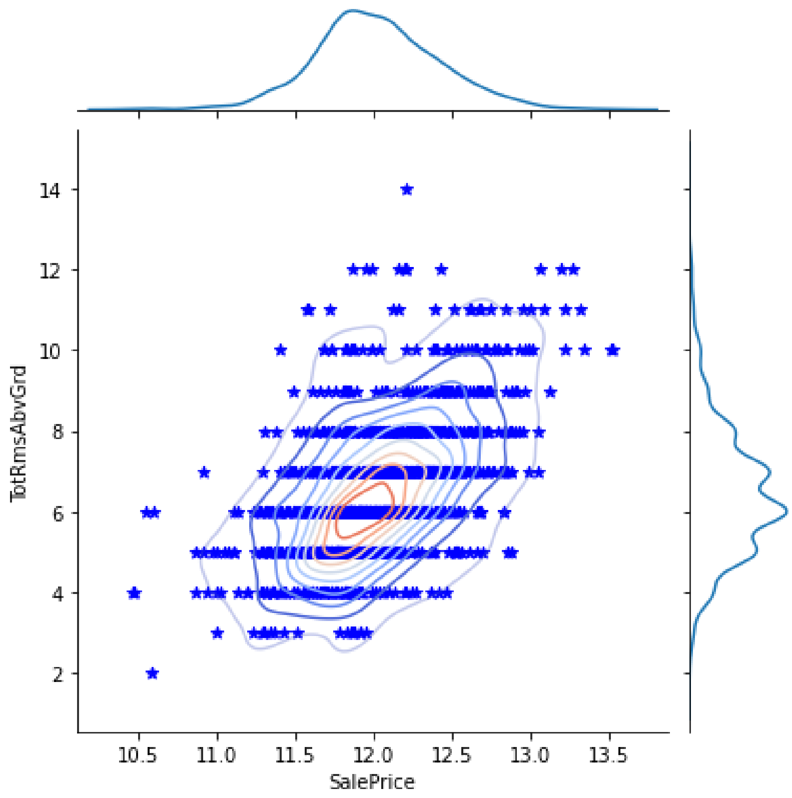
As you can see in the above plot GarageYrBlt is also highly negatively correlated with the target variable and hence we couldn't see any trend in the data.

Next, we analyze GrLivArea with SalePrice.



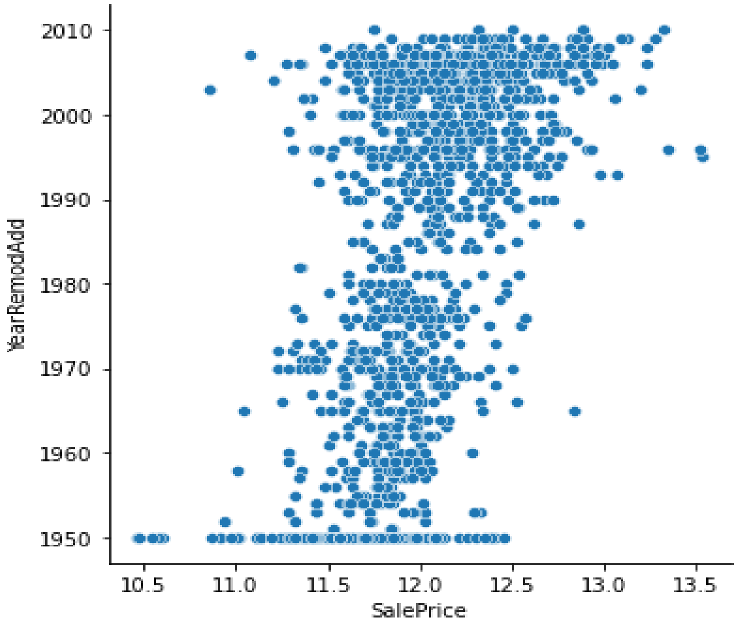
Here also we can see the linear trend in data. When the GrLivArea increases, the SalePrice increases.

Now let’s analysis TotRmsAbvGrd with SalePrice.



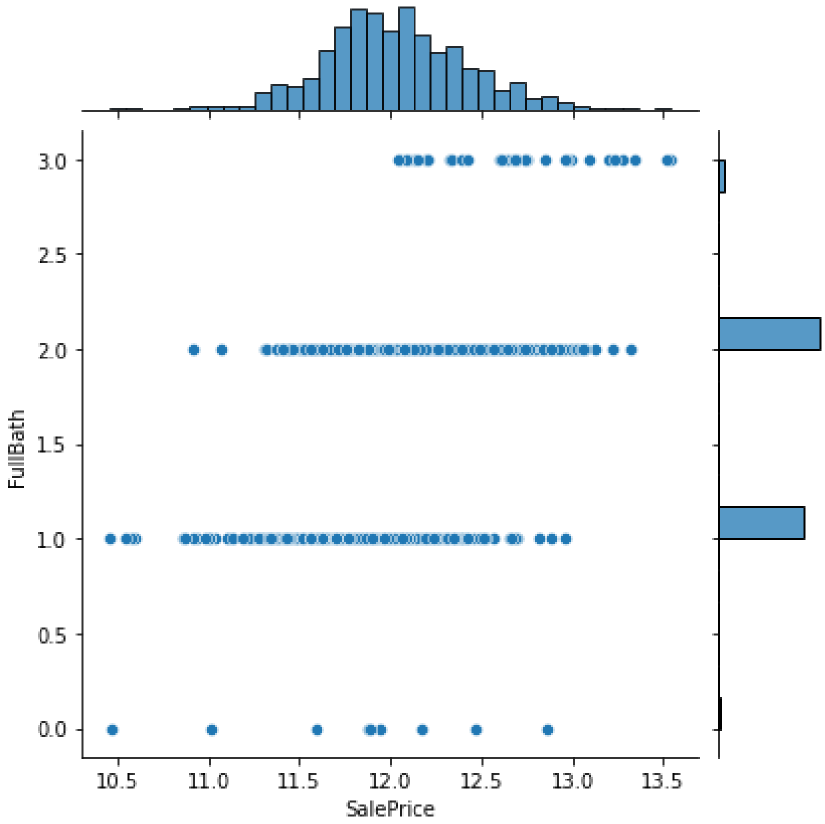
From the above plot, we could see a slight linear trend.

Next we analyse our next variable 'YearRemodAdd'.



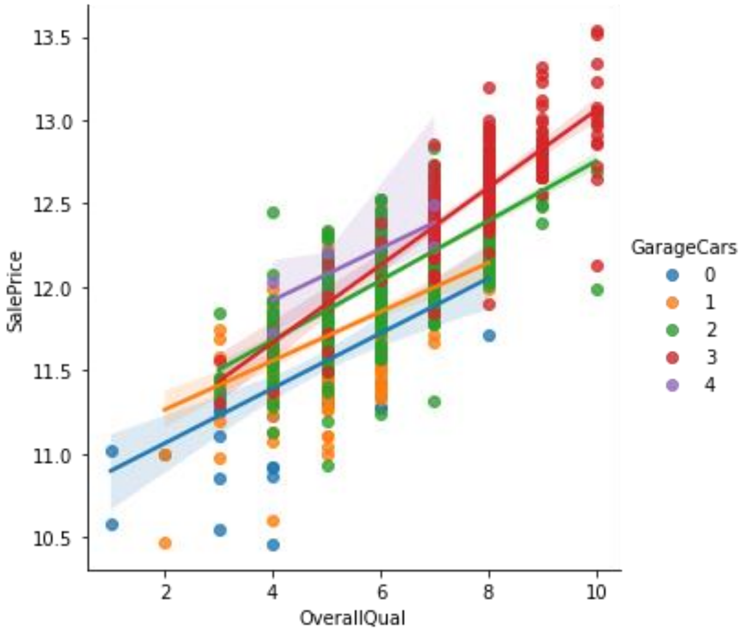
If you see the above plot, the YearRemodAdd also has a linear relationship with SalePrice.

Now we analyse the next variable 'FullBath'

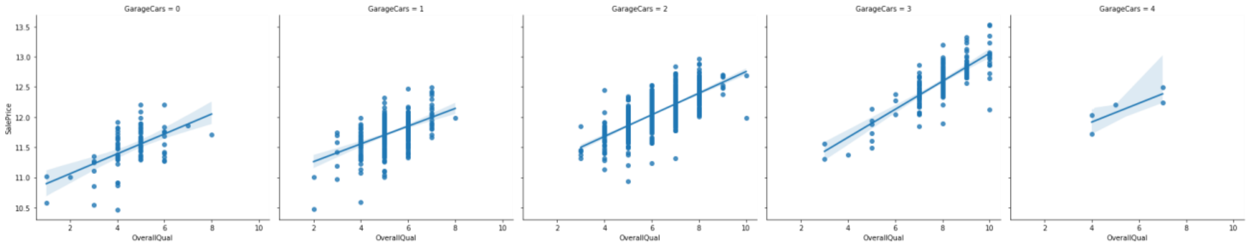


If you see here, there is no trend in data. It is highly negatively correlated with our target variable. But this helps the model to predict better.

We have already analysed OverallQual with SalePrice. Now lets include the third variable GarageCars and see if there is any change in trend in the data.



The plot above presents linear regression models fit on the data grouped based on different values of ‘GarageCars’ represented by different colors. I’ll admit that the plot is quite a bit obscure. Let’s separate the groups into a grid of plots.



The plot above presents linear regression models fit on the data grouped based on different values of ‘GarageCars’ represented by different colors. I’ll admit that the plot is quite a bit obscure. Let’s separate the groups into a grid of plots.

Conclusion

There have been some mixed conclusions throughout the analysis which may be down to the quality of the data. The early stages through EDA presented some patterns within the high prices such as overall quality and built year.

Although there is evidence that not all expensive listings will share the same features, the analysis has shown that particular features are most certainly common amongst the highest priced.