House Prices/Making Houses More Appealing
- Data Wrangling -

Introduction

This report relates to my Capstone project titled “Making Houses More Appealing for buyers”. All the analysis are done over the House_Prices dataset.
Throughout the report, there will be links to the code executed to accomplish the mentioned actions. They are all listed on the Annexe.

1. Checking for errors on categorical columns

The document “data_description.txt” shows all the allowed values for each categorical data. Let’s list all unique values on each categorical column to look for mistyped data.
IMPORTANT: there are 3 categorical columns that are represented by numbers ('MSSubClass','OverallQual','OverallCond'). Please note the highlighted line where we added those 3 columns. CODE 1

Here is a part of the output 1. With a quick inspection, we notice some NaNs that are not “missing data”. The highlighted nan, for example, means “no garage”, but it could also mean “missing data” in other columns. We’ll look into that later.

Comparing the output with the data description, we noticed small errors that are very easy to miss if we inspect manually. So we created a .csv file called “AllPossibleValuesForCatCols.csv”, using the data_description.txt as the source, to list all possible values those categorical columns may have. We’ll import this file as a Dataframe and match the actual values found on the dataset with the possible values for that column. This way we can find typos easier. This is the CODE 2 to find all typos:

Analyzing the Output 2, we noticed some small variations on the text of some columns, compared to the data_description data. To match the values exactly, we changed the following, by running CODE 3:

- Column **BldgType**: “Duplex” should be “Duplx” and “2fmCon” should be “2FmCon”. “Twnhs” should be “TwnhsI”.
- Column **MSZoning:** “C (all)” should be “C”
- Column **Exterior2nd:** “Wd Shng” should be “WdShing” because wecan see “Wd Shng” on the data. “CmentBd” should be “CemntBd” and “Brk Cmn” should be “BrkComm”
- Column **Neighborhood**: “NAmes” should be “Names” (meaning North Ames)

If we run the same code again to check typos, the output is empty, meaning the code ran successfully.

Back to the unique values from all columns, a few other things that caught our attention. Below are the modifications we performed on the columns **MasVnrType** and **MasVnrArea,** performed by CODE 4.

:

- There are 8 values that are “None” values and NaN on the MasVnrType column. Replace NaN with ‘None’ on column MasVnrType and replace NaN with 0 on the MasVnrArea column
- when the MasVnrType is None, MasVnrArea should be zero. Analysing this table, generated by code 4, we find values different than zero, which is not what we expected. To deal with that, the two records with Areas = 1.0 will be replaced by Area = 0.0, and the other three values will be replaced by the most common Masonry veneer type, which is type = ‘BrkFace’.

This type of consistency will be checked for every column later. For example, if the house has no garage, all columns related to “garage” should show NA.

2. **Cleaning NaN from categorical columns**

We already know that missing values are stored as NaN. CODE 5 lists the categorical columns with their unique values. Analyzing Output 5, we notice that, for all columns, it makes sense to replace nan with “No Item”, except for the column “Electrical”, because It doesn’t make sense not to have an electrical system.

After replacing all NaN from the above columns with “No Item”, CODE 6 perform the following operations on the “Electrical” column

- Find the number of records we’re dealing with
- Checks how the price of this house stands against the other types of electrical systems. You can check the plot here
- Replace “No Item” by “SBrkr”

3. **Cleaning NaN from numerical columns**

Code 7 performs the following operations:

- Create a DataFrame with only the numeric columns;
- Inspect the NaN with this table, generated by code 7
    - Column **GarageYrBlt**: Hopefully, those NaN are for houses without garage. Check the unique values of the column GarageType when GarageYrBlt is NaN
    - Column **LotFrontage**: those are legitimate NaNs, too many to discard. Inspecting the columns, we found a **LotArea.** Maybe a scatter plot will show a relationship between those two variables
        - We can easily identify the presence of outliers on both dimensions. To better identify the relationship, let’s “zoom in” in the graphic.
        - As suspected, there is a big correlation between those two variables.

Filling those 259 NaN from LotFrontage with values that are proportional to the column LotArea makes sense. To do that, we'll find the equation that best represents the linear regression between these two columns and apply it to replace the NaNs. Excel will be used to find the equation, which is **y = 0.0046x + 27.113.**

Based on this equation, the NaN from LotFrontage will be filled according to CODE 8. The comparison between before and after code 8 can be seen here.

4. **Consistency among columns**

If a house has no garage, all columns related to "garage" should reflect this fact. We've already done this for MasVnrType column on item 1 of this report, now let's take the same approach for garage, Basement, fireplace, pool and miscellaneous features. Please check the CODE 9, which resulted in empty outputs only, meaning no action needs to be taken.

5. **Checking outliers**

As shown on the scatter plot previously presented, both LotFrontage and LotArea has outliers. In this section, we are going through a few relevant numerical columns to inspect outliers and decide whether to keep or discard them.

- LotArea and LotFrontage

When we print a scatter plot of these two variables, we notice that we introduced outliers on our attempt to fill in the NaNs. As lotFrontage almost never surpasses the 200 units, we will limit this dimension to 200 regardless of the lot Area. It's like we are using one regression equation for lotFrontage until 200 and another one for greater than 200.

Using Excel, the equation is :

y = 0.0006x + 19.657

Where x is LotArea

Here is the full CODE 10, Applying this equation for LotFrontage > 200, followed by the output.

We can see highlighted in yellow the two equations used to get rid of NaNs. Even though it's tempting to just discard the values where LotArea is greater than 50,000, the price of the properties has a strong correlation to the Lot Area, so this data is valuable. The two values inside the red circle, however, they do not bear relation to the house selling price, meaning they can undermine the precision of our future model. We'll drop those two values.

- Violin plots for the other variables

We used CODE 11 to plot violin plots to spot outliers on the other variables. Inspecting each one of the curves, the following called our attention:

- Pool diagram

Looking back into the data, this behavior is expected as we only have 7 values different than 0. Maybe this data can help predict house selling prices for specific cases, so we'll keep this column as-is.

The same argument goes to columns 3SsdPorch, BsmtFinSF2, LowQualFinSF, MiscVal, ScreenPorch and BsmtHalfBath.

- MasVnrArea Diagram

This variable has more than 500 samples different than zero, so let's see if we can find outiers in a violin distribution that ignores the zeros. Here is the code and the output.

Now this distribution has only a few outliers that can be useful for some cases and may have a combined effect with other variables, so we'll keep it as is.

6. Conclusion

Based on my experience, this dataset is rather clean, allowing us to keep almost all records. We were able to spot typos and filled some NA values in a way that will foster the predicting power or our model. This dataset is now ready for the next steps.

7. Annexe

Please find in this section all the codes and outputs ran throughout the report.

CODE 1

```
import pandas as pd
df = pd.read_csv('house_price.csv, header=0, index_col='Id')
dfnumbers = df._get_numeric_data()
catcols = set(df.columns) - set(dfnumbers.columns)
catcols = catcols.union(set(['MSSubClass','OverallQual','OverallCond']))
for col in catcols:
    print(col, ' - ',df[col].unique())
```

Output 1

```
GarageQual  -  ['TA' 'Fa' 'Gd' nan 'Ex' 'Po']
Foundation  -  ['PConc' 'CBlock' 'BrkTil' 'Wood' 'Slab' 'Stone']
CentralAir  -  ['Y' 'N']
ExterQual  -  ['Gd' 'TA' 'Ex' 'Fa']
BsmtExposure  -  ['No' 'Gd' 'Mn' 'Av' nan]
MSZoning  -  ['RL' 'RM' 'C (all)' 'FV' 'RH']
RoofStyle  -  ['Gable' 'Hip' 'Gambrel' 'Mansard' 'Flat' 'Shed']
```

CODE 2

```
df = pd.read_csv('train.csv', header=0, index_col='Id')
dfAll = pd.read_csv('AllPossibleValuesForCatCols.csv', header=0)
dfAll = dfAll.applymap(lambda x: x.strip() if isinstance(x, str) else x)
diffs = {}
for col in dfAll.columns:
    diffs[col] = set(df[col]) - set(dfAll[col])
for key, value in diffs.items():
    if(len(value)>0): print(key, ' - ',value)
```

Output 2

```
MSZoning  -  {'C (all)'}
Neighborhood  -  {'NAmes'}
BldgType  -  {'Duplex', '2fmCon', 'Twnhs'}
Exterior2nd  -  {'Wd Shng', 'CmentBd', 'Brk Cmn'}
```

CODE 3

```
df.BldgType = df.BldgType.replace('Twnhs', 'TwnhsI')
df.BldgType = df.BldgType.replace('Duplex', 'Duplx')
df.BldgType = df.BldgType.replace('2fmCon', '2FmCon')
df.MSZoning = df.MSZoning.replace('C (all)', 'C')
df.Exterior2nd = df.Exterior2nd.replace('Wd Shng', 'WdShing')
df.Exterior2nd = df.Exterior2nd.replace('CmentBd', 'CemntBd')
df.Exterior2nd = df.Exterior2nd.replace('Brk Cmn', 'BrkComm')
df.Neighborhood = df.Neighborhood.replace('NAmes', 'Names')
```

CODE 4

```
df = df.replace({'MasVnrType':{np.nan : 'None'}, 'MasVnrArea': {np.nan : 0}}, value = None)
print(df[['MasVnrType', 'MasVnrArea']][(df.MasVnrType == 'None') & (df.MasVnrArea > 0)])
df.MasVnrArea = df.MasVnrArea.replace(1.0,0.0)
print(df[['MasVnrType', 'MasVnrArea']][df.MasVnrArea > 0].MasVnrType.value_counts())
for index, row in df.iterrows():
    if((row['MasVnrType'] == 'None') & (row['MasVnrArea'] > 0)):
        df.loc[index, 'MasVnrType'] = 'BrkFace'
```

Output table Code 4

Id	MasVnrType	MasVnrArea
625	None	288.0
774	None	1.0
1231	None	1.0
1301	None	344.0
1335	None	312.0

CODE 5

```
nulls = df.isnull().sum()
catcolswithnan = set(dfAll.columns) & set(nulls[nulls>0].index)
for col in catcolswithnan:
    print(col, ' - ',df[col].unique())
```

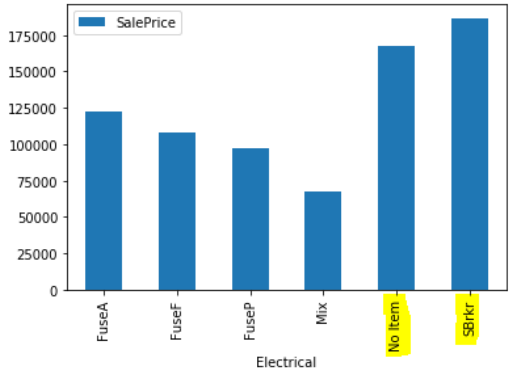
Output 5

```
PoolQC  -  [nan 'Ex' 'Fa' 'Gd']
Alley  -  [nan 'Grvl' 'Pave']
BsmtQual  -  ['Gd' 'TA' 'Ex' nan 'Fa']
BsmtCond  -  ['TA' 'Gd' nan 'Fa' 'Po']
Electrical  -  ['SBrkr' 'FuseF' 'FuseA' 'FuseP' 'Mix' nan]
BsmtExposure  -  ['No' 'Gd' 'Mn' 'Av' nan]
GarageCond  -  ['TA' 'Fa' nan 'Gd' 'Po' 'Ex']
BsmtFinType2  -  ['Unf' 'BLQ' nan 'ALQ' 'Rec' 'LwQ' 'GLQ']
GarageType  -  ['Attchd' 'Detchd' 'BuiltIn' 'CarPort' nan 'Basment' '2Types']
GarageQual  -  ['TA' 'Fa' 'Gd' nan 'Ex' 'Po']
GarageFinish  -  ['RFn' 'Unf' 'Fin' nan]
Fence  -  [nan 'MnPrv' 'GdWo' 'GdPrv' 'MnWw']
FireplaceQu  -  [nan 'TA' 'Gd' 'Fa' 'Ex' 'Po']
BsmtFinType1  -  ['GLQ' 'ALQ' 'Unf' 'Rec' 'BLQ' nan 'LwQ']
MiscFeature  -  [nan 'Shed' 'Gar2' 'Othr' 'TenC']
```

CODE 6

```
print(df[df.Electrical == 'No Item']['Electrical'].count())
#plot code 6
df[['Electrical', 'SalePrice']].groupby('Electrical').mean().plot(kind = 'bar')
plt.show()

df.Electrical = df.Electrical.replace('No Item', 'SBrkr')
```

Plot code 6



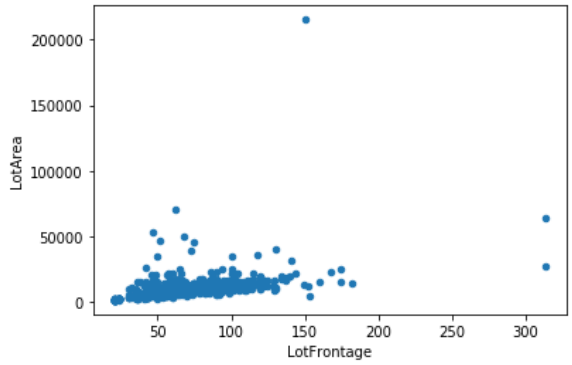
CODE 7

```
dfNumericCols = set(df.columns) - set(dfAll.columns)
dfNumeric = df[list(dfNumericCols)]
print(dfNumeric.isnull().sum()[dfNumeric.isnull().sum()>0])
print(df[['GarageType', 'GarageYrBlt']][df.GarageYrBlt.isnull()].GarageType.unique())
df[['LotFrontage', 'LotArea']].plot(x = 'LotFrontage', y = 'LotArea', kind = 'scatter')
df[['LotFrontage', 'LotArea']].plot(x = 'LotFrontage', y = 'LotArea',
                                    xlim=(0,150), ylim=(0,20000), kind = 'scatter')
```

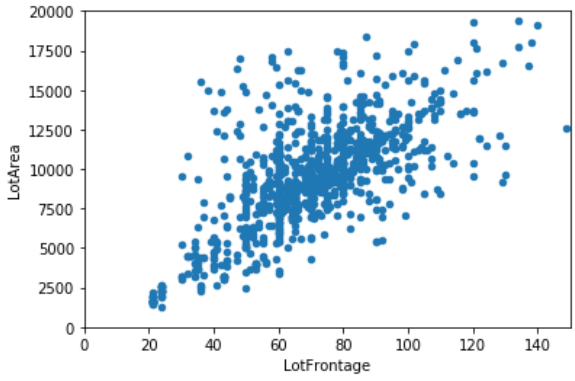
Table Code 7

```
GarageYrBlt      81
LotFrontage     259
```

Scatter Plot Code 7



Graphic Code 7

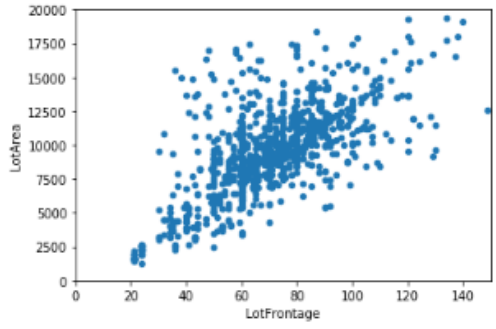


CODE 8

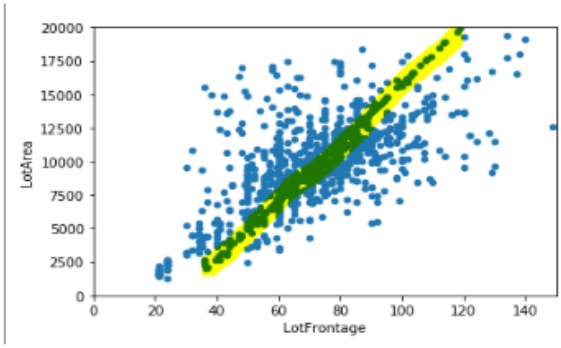
```
for index, col in df.iterrows():
    if(np.isnan(col['LotFrontage'])):
        df.loc[index, 'LotFrontage'] = 0.0046*df.loc[index, 'LotArea'] + 27.113
```

Before X After code 8

BEFORE



AFTER

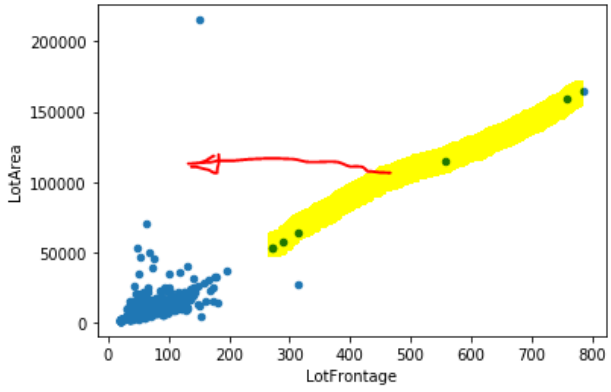


CODE 9

```
#Garage
collist = ['GarageYrBlt', 'GarageFinish', 'GarageCars', 'GarageArea', 'GarageQual', 'GarageCond']
df.GarageType== 'No Item'
myDict = {}

for col in collist:
    myDict[col] = []

for index, row in df.iterrows():
    if(df.loc[index, 'GarageType'] == 'No Item'):
        for col in collist:
            myDict[col].append(row[col])

dfGarage = pd.DataFrame(myDict)

for col in dfGarage.columns:
    print(col, '- ', dfGarage[col].unique())

#Basement
collist = ['BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinSF1', 'BsmtFinType2', 'BsmtFinSF2',
           'BsmtUnfSF', 'TotalBsmtSF', 'BsmtFullBath', 'BsmtHalfBath']
df.BsmtQual == 'No Item'
myDict = {}

for col in collist:
    myDict[col] = []

for index, row in df.iterrows():
    if(df.loc[index, 'BsmtQual'] == 'No Item'):
        for col in collist:
            myDict[col].append(row[col])

dfBsmt = pd.DataFrame(myDict)

for col in dfBsmt.columns:
    print(col, '- ', dfBsmt[col].unique())

#Fireplace
 print(df[['Fireplaces', 'FireplaceQu']][(df.FireplaceQu == 'No Item') & (df.Fireplaces != 0 )])

#Pool
print(df[['PoolArea', 'PoolQC']][(df.PoolQC == 'No Item') & (df.PoolArea != 0 )])

#miscellaneous
print(df[['MiscFeature', 'MiscVal']][(df.MiscFeature == 'No Item') & (df.MiscVal != 0 )])
```

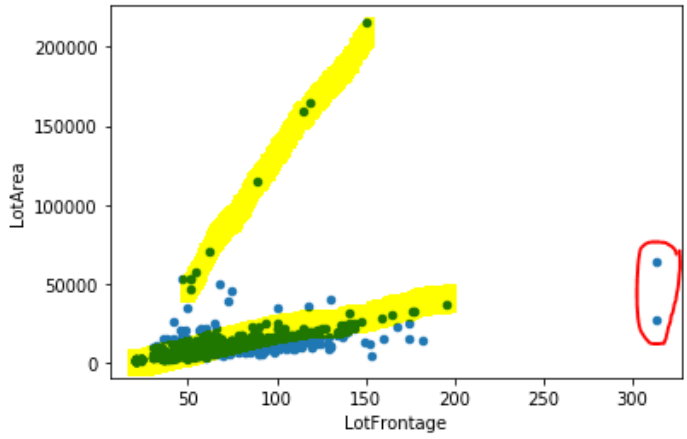
Scatter plot LotFrontage X LotArea

```
df[['LotFrontage', 'LotArea']].plot(x = 'LotFrontage', y = 'LotArea', kind = 'scatter')
```



CODE 10

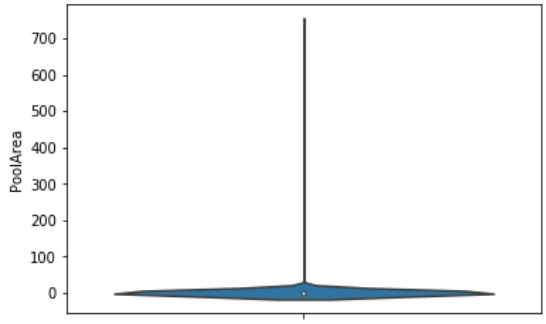
```
aux = 0
for index, col in df.iterrows():
    if(np.isnan(col['LotFrontage'])):
        aux = 0.0046*df.loc[index, 'LotArea'] + 27.113
        if(aux>200):
            df.loc[index, 'LotFrontage'] = 0.0006*df.loc[index, 'LotArea'] + 19.657
        else:
            df.loc[index, 'LotFrontage'] = aux

df[['LotFrontage', 'LotArea']].plot(x = 'LotFrontage', y = 'LotArea', kind = 'scatter')
plt.show()
```



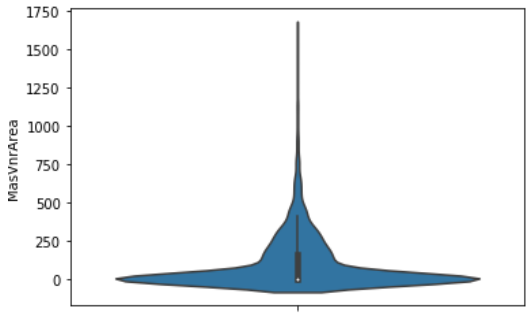
CODE 11

```
for col in dfNumericCols:
    ax = sns.violinplot(y=col, data=df)
    plt.show()
```

Pool diagram



MasVnrArea Diagram



MasVnrArea Diagram ignore zeros

```
ax = sns.violinplot(y='MasVnrArea', data = df[df.MasVnrArea > 0])
```