Capstone 1 - Data Wrangling Steps

Introduction

This dataset is based on almost every aspect of residential homes in Ames, Iowa. With over 79 explanatory variables, it becomes very important to take care of improperly recorded data. It becomes clear with this dataset (by reading the Data Description file) that there are many null entries whose real values can be inferred. The recording of 'Na' in many instances here has somewhat masked the true value of the data. When we override that with a more meaningful value, the visualizations we can produce become much more telling.

Data Inspection

Let's begin by taking some basic steps to read in the data and inspect its basic structure:

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

Let's check the columns of this dataset:

```
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')
```

Missing Values

Now lets check if the data contains any null values:

```
pd.isnull(df).any().sort_values(ascending=False)
```

```
FireplaceQu      True
GarageCond       True
BsmtFinType1     True
BsmtExposure     True
BsmtCond         True
BsmtQual         True
Electrical       True
MasVnrArea       True
MasVnrType       True
GarageType       True
GarageYrBlt      True
GarageFinish     True
GarageQual       True
BsmtFinType2     True
LotFrontage      True
Alley            True
MiscFeature      True
Fence            True
PoolQC           True
```

So clearly there is some work here to determine if we can replace some of these missing values with a more logical value.

To determine how much missing data is present for each column, lets create a new dataframe which contains a percent of missing values with respect to total length of the dataset for each feature:

```
#Calculate a the missing data (per attribute) as a percentage of the total (result is a pandas series)
missing_data = (df.isnull().sum() / len(df)) * 100
missing_data = missing_data.drop(missing_data[missing_data == 0].index).sort_values(ascending=False)
```

```
#Prints out the top 20 fields with missing data
print(missing_data.head(20))
```

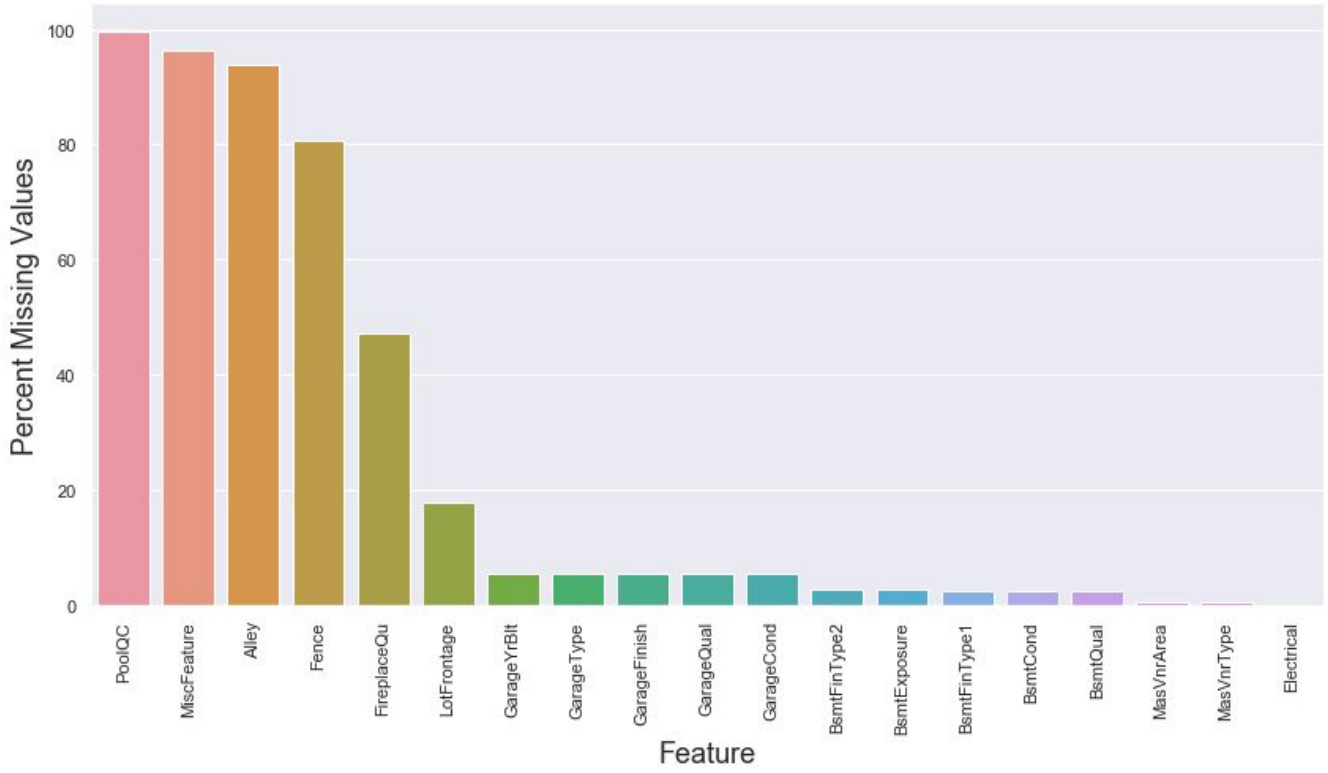
```
poolQC          99.520548
MiscFeature     96.301370
Alley           93.767123
Fence           80.753425
FireplaceQu     47.260274
LotFrontage     17.739726
GarageYrBlt      5.547945
GarageType       5.547945
GarageFinish     5.547945
GarageQual       5.547945
GarageCond       5.547945
BsmtFinType2     2.602740
BsmtExposure     2.602740
BsmtFinType1     2.534247
BsmtCond         2.534247
BsmtQual         2.534247
MasVnrArea       0.547945
MasVnrType       0.547945
Electrical       0.068493
```

This lines up with the inspection of null values we performed earlier, so thats a good sanity check. We see that a few fields are almost completely populated with null values. Lets take a look at this data in a visual form:

Now let's take a look at this missing data in a visualization:

In [72]:

```
fig, ax = plt.subplots(figsize=(14,7))
plt.xticks(rotation='90')
sns.barplot(missing_data.index, missing_data, ax=ax)
plt.ylabel('Percent Missing Values', fontsize=18)
_ = plt.xlabel('Feature', fontsize=18)
```



Lets fill some of the fields which have many missing values:
The fields PoolQC, MiscFeature, Alley, Fence and FireplaceQu all have Na values which correspond to absence of that feature. For example Na for PoolQC means the property has no pool. To simplify the data processing here we would like to convert these values to a representative string "None". See below:

```
df["PoolQC"] = df["PoolQC"].fillna("None")
```

```
df['MiscFeature'] = df['MiscFeature'].fillna("None")
```

```
df["Alley"] = df["Alley"].fillna("None")
```

```
df["Fence"] = df["Fence"].fillna("None")
```

```
df["FireplaceQu"] = df["FireplaceQu"].fillna("None")
```

From the data dictionary, LotFrontage represents the amount of square footage on the street connected to the house. Its highly unlikely that this is 0, so in this case it's most likely just unrecorded data. To treat these values, we will fill in the missing LotFrontage values with the medianLotFrontage.

```
df["LotFrontage"] = df.groupby("Neighborhood")["LotFrontage"].transform(
```

```
  lambda x: x.fillna(x.median()))
```

For GarageType, GarageQual and GarageCond we will replace missing data with 'No Garage' as indicated by the data dictionary:

```
for col in ('GarageType', 'GarageFinish', 'GarageQual', 'GarageCond'):
  df[col] = df[col].fillna('No Garage')
```

For GarageYrBlt, GarageArea and GarageCars we will replace missing data with 0:

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

For BsmtQual, BsmtCond, BsmtExposure, BsmtFinType1 and BsmtFinType2 we can replace the null values with the string "No Basement":

```
for col in ('BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2'):
  df[col] = df[col].fillna('No Basement')
```

For BsmtFinSF1, BsmtFinSF2, BsmtUnfSF, TotalBsmtSF, BsmtFullBath and BsmtHalfBath we will replace missing values with 0 since null values indicate no basement here:

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

For MasVnrArea and MasVnrType here null values indicate no veneer applied to the home. So lets fill in 0/None for those as applicable:

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

For MSZoning, since this is categorical data, lets fill in the missing values with the most common value or the mode of the entries:

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

For Functional, or 'Home Functionality' we are told to assume 'typical':

```
df["Functional"] = df["Functional"].fillna("Typical")
```

For Electrical we can replace with mode since it is categorical data (and only 1 missing value is present):

```
mode = df['Electrical'].mode()[0]
df['Electrical'] = df['Electrical'].fillna(mode)
```

For Exterior1st and Exterior2nd (categorical data) we can replace missing values with the mode as well:

```
mode1 = df['Exterior1st'].mode()[0]
df['Exterior1st'] = df['Exterior1st'].fillna(mode1)

mode2 = df['Exterior2nd'].mode()[0]
df['Exterior2nd'] = df['Exterior2nd'].fillna(mode2)
```

Finally for MSSubClass we can replace null values with "No Building Class" as suggested by the data dictionary:

```
df['MSSubClass'] = df['MSSubClass'].fillna("No Building Class")
```

Now lets check to make sure we have eliminated all of the missing values:

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

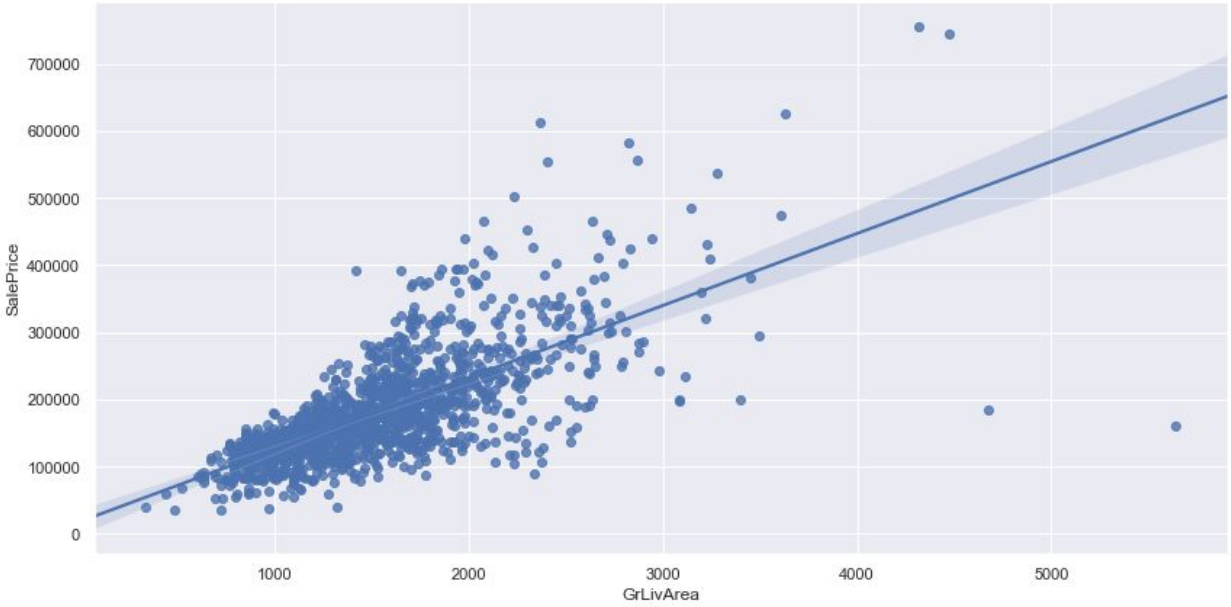
	Missing Ratio

We have successfully eliminated all missing values from the dataset!

Dealing with Outliers

The dataset author mentions that there are 4 outliers in the dataset. Specifically, anything greater than 4000 square feet living area should be removed from the dataset. Before eliminating these, let's take a look at these data points in a visualization showing GrLivArea (the total living area of the house) versus the sales price of that home (SalePrice):

```
fig, ax = plt.subplots(figsize=(14,7))
_ = sns.regplot(df.GrLivArea, df.SalePrice, ax=ax)
```



The four data points in question can be identified fairly clearly in this visualization. Lets isolate these datapoints:

becomes obvious by looking at this plot where the outliers in the dataset can be found. Lets isolate these datapoints:

In [13]:

```
df[df['GrLivArea'] > 4000][['SalePrice', 'GrLivArea']]
```

	SalePrice	GrLivArea
523	184750	4676
691	755000	4316
1182	745000	4476
1298	160000	5642

Index 692 and 1182 might actually be realistic data points since they were sold at a very high price point. For now we will leave these outliers in the dataset. For modeling purposes we do not want to assume a perfectly linear relationship so we will re evaluate whether or not to keep these in our dataset based on modeling results. Simply based on this graph it appears the two data points below 200,000 sale price