Machine Learning Engineer Nanodegree

Capstone Project

Santosh Narayan
Jun 03rd, 2018

Report

I. Definition

Overview/Domain Background

This project is aimed at predicting house prices. The Ames Housing dataset was compiled by Dean De Cock.

https://www.kaggle.com/c/house-prices-advanced-regression-techniques

The Ames Housing data set has about 80 variables that directly relate to property sales. This dataset focuses on both quantitative variables and qualitative variables of the project. The variables are the perfect representation of the questions any typical buyer would ask before buying a house/property. Example include: When was the house built? Does it have a basement? How many bathrooms does the house have? The dataset has information that can answer a typical buyers question. The dataset represents the sale of individual residential property in Ames, Iowa from 2006 to 2010. 10. The data set contains 2930 observations and many explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values.

Property/Real Estate is a very widely used measure of a person's wealth. A home is the probably the biggest investment anyone makes. It is a field with relatively low barriers to entry, anyone can participate at a level of the purchasing power they have. Flipping houses has become an occupation for quite a few. By knowing the true market value, one can help arbitrage and earn significant profits.

Hence, I have selected the Ames, Iowa dataset of housing sales prices to analyze and develop a model to predict housing prices. The project is taken from Kaggle competition. In the future, using this model as a base, I would like to be able to predict at my local area.

I would also like to cite the following research paper:
https://smartech.gatech.edu/bitstream/handle/.../Corsini_Kenneth_R_200912_mast.pdf which

discusses the housing prices in relation to other external factors such as employment rate, interest rates, New construction, Consumer Price Index etc.

Problem Statement

https://en.wikipedia.org/wiki/Kaggle

I plan to use regression techniques such as random forest and gradient boosting.

Inputs are the 79 variables and out put is to predict the price of the house.

The opportunity involves in using the data set provided to predict the housing price. The goal is to predict the sale price variable of each house. There are so many different variables in the dataset that come into play when predicting a house price. Every solution and analysis can be unique depending upon the user. I look forward to using my skills and knowledge in this domain to come up with a prediction.

https://ww2.amstat.org/publications/jse/v19n3/decock.pdf

Datasets and Inputs

The dataset is provided on Kaggle competition website. They are available to download by everyone.

https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

Data set contains information from the Ames Assessor's Office used in computing assessed values for individual residential properties sold in Ames, IA from 2006 to 2010

The data set given contains 1460 observation in the train file and 1459 on the test file and a large number of explanatory variables (23 nominal, 23 ordinal, 14 discrete, and 20 continuous) involved in assessing home values. The 14 discrete variables typically quantify the number of items occurring within the house. Most are specifically focused on the number of kitchens, bedrooms, and bathrooms (full and half) located in the basement and above grade (ground) living areas of the home. Additionally, the garage capacity and construction/remodeling dates are also recorded. There are a large number of categorical variables (23 nominal, 23 ordinal) associated with this data set. They range from 2 to 28 classes with the smallest being STREET (gravel or paved) and the largest being NEIGHBORHOOD (areas within the Ames city limits). The nominal variables typically identify various types of dwellings, garages, materials, and environmental conditions while the ordinal variables typically rate various items within the property.

data_description.txt provides the detailed description of the data.

Here's a brief version of what you'll find in the data description file. The below variables are the list provided on Kaggle by the data provider.

- SalePrice - the property's sale price in dollars. This is the target variable that you're trying to predict.
- MSSubClass: The building class
- MSZoning: The general zoning classification
- LotFrontage: Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access
- Alley: Type of alley access
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to main road or railroad
- Condition2: Proximity to main road or railroad (if a second is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Overall material and finish quality
- OverallCond: Overall condition rating
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date
- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet
- ExterQual: Exterior material quality
- ExterCond: Present condition of the material on the exterior
- Foundation: Type of foundation
- BsmtQual: Height of the basement
- BsmtCond: General condition of the basement
- BsmtExposure: Walkout or garden level basement walls
- BsmtFinType1: Quality of basement finished area
- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Quality of second finished area (if present)
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second floor square feet

- LowQualFinSF: Low quality finished square feet (all floors)
- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms
- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- Bedroom: Number of bedrooms above basement level
- Kitchen: Number of kitchens
- KitchenQual: Kitchen quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality rating
- Fireplaces: Number of fireplaces
- FireplaceQu: Fireplace quality
- GarageType: Garage location
- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet
- 3SsnPorch: Three season porch area in square feet
- ScreenPorch: Screen porch area in square feet
- PoolArea: Pool area in square feet
- PoolQC: Pool quality
- Fence: Fence quality
- MiscFeature: Miscellaneous feature not covered in other categories
- MiscVal: $Value of miscellaneous feature
- MoSold: Month Sold
- YrSold: Year Sold
- SaleType: Type of sale
- SaleCondition: Condition of sale

Overview of Approach

This is a regression problem where the Target variable (SalePrice) has to be predicted for a given set of features. A training dataset has been provided which has the given features and the actual Sales price. This data can be fit to a model. I plan to use the RandomForest and XGBoost models.

Metrics

The following are the key metrics that I plan to use to evaluate model performance.

Accuracy Score

I plan to use the accuracy 'score' function which is the mean accuracy on the given test data and labels. This gives how accurately the model can predict the target values as compared to the actual values. Hence after training, I would obtain the 'score' with the Train dataset. The higher the score and closer to 1 would mean a better trained model. Then I would obtain the score with the test dataset for the same model.

RMSE

The RMSE is the square root of the variance of the residuals. It indicates the absolute fit of the model to the data–how close the observed data points are to the model's predicted values. Whereas R-squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit. RMSE is a good measure of how accurately the model predicts the response, and it is the most important criterion for fit if the main purpose of the model is prediction.

Ref:

https://www.theanalysisfactor.com/assessing-the-fit-of-regression-models/

https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

II. Analysis

Data Exploration

The data analysis on the housing data showed the following:

```
In [6]: data.columns
```

```
Out[6]: 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')
```

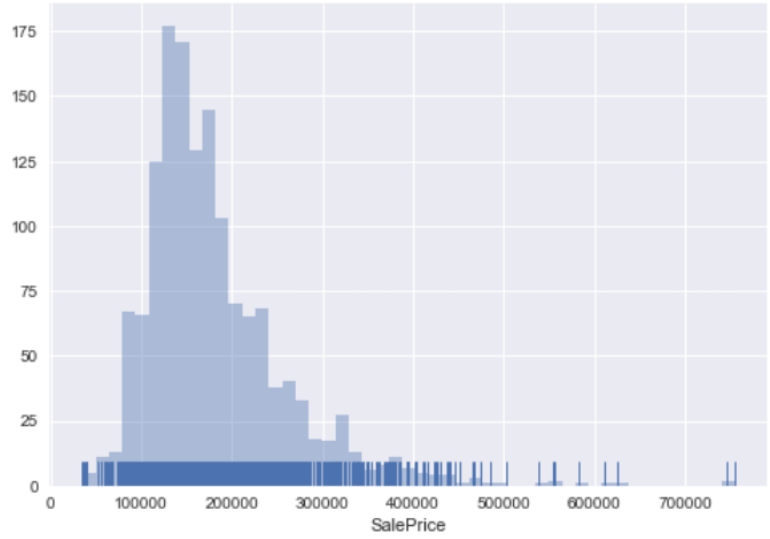
```
In [7]: data.shape
```

```
Out[7]: (1460, 81)
```

There are 1460 housing data points provided in the Training dataset.

A plot of the target variable, SalePrice shows the distribution. Most of the houses are priced between $150,000 and $200,000.

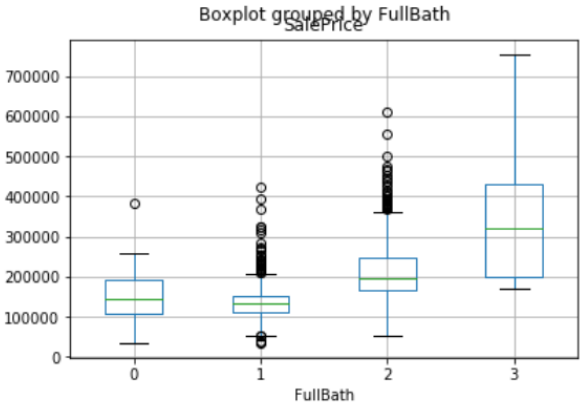
In [20]:
```
import seaborn as sb

sb.set(color_codes=True)
sb.distplot(data['SalePrice'],kde=False, rug=True);
```



On a hunch, I wanted to look at the box plot which interested me, the distribution of SalePrice with respect to number of FullBath, LivingArea and TotalBasement sqarefeet.

In [13]:
```
data.boxplot(column = 'SalePrice', by = 'FullBath' )
```

Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1754342c5c0>
```

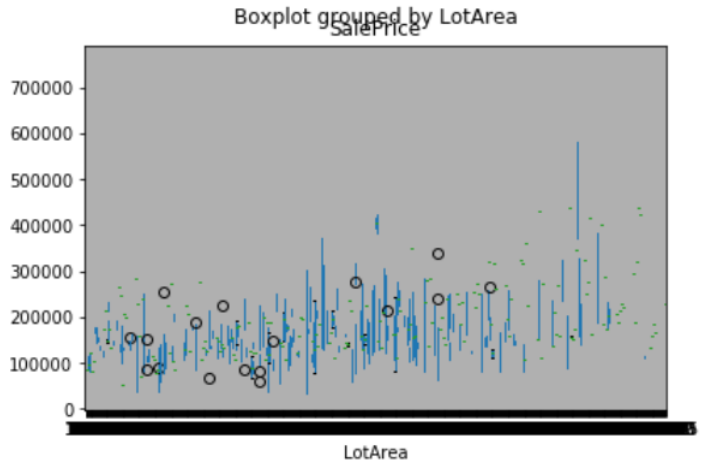


It can be seen that most of the houses have 1 or 2 Fulbaths and the SalePrice are generally proportional to the number of FullBaths. There are a low number of outliers with 0 (some lowcost houses which have only half baths and probably the shower is separate) and 3 full baths(probably indipendent houses outside the communities).

In [11]:

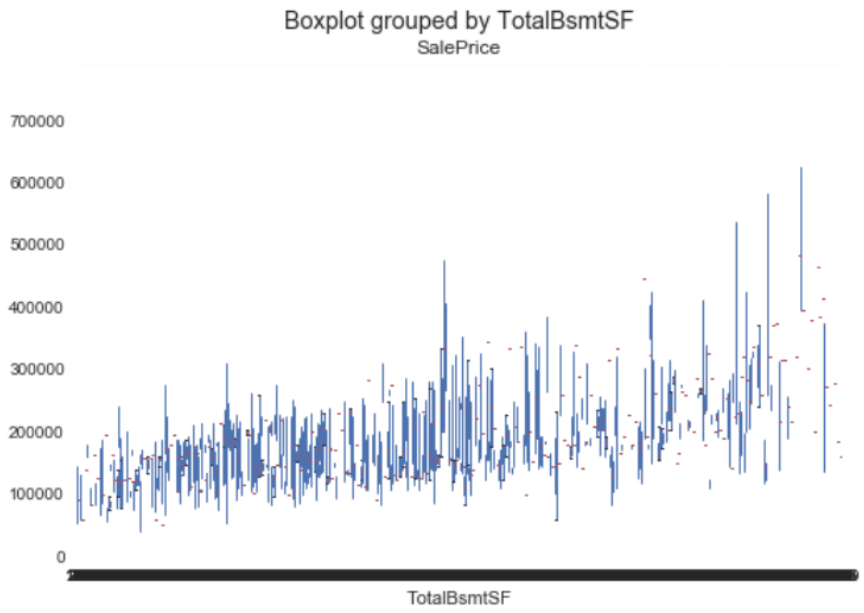
```
data.boxplot(column = 'SalePrice', by = 'LotArea' )
#print('test')
```

```
test
```



The lot area is spread all the way from 1300 sq ft to 215,000 sq ft. This is a wide spread with no concrete clusters. Also, we note that the price does not dramatically increase depending on the lot size. This might be due to even though the lot size is huge, they might be located on the outskirts of the city and may be thinly populated.

```
In [22]: data.boxplot(column = 'SalePrice', by = 'TotalBsmtSF' )
Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x17553e88780>
```



The reason I was interested to the relation to basement area is because, in many areas basement is not available, and the presence of basement would have caused and immediate jump in prices. But I don't see that by looking at the above boxplot. The Basement area is well spread out, this probably indicates that most of the houses have some basement area. In fact, looking at the file there are only 37 houses with no basement.

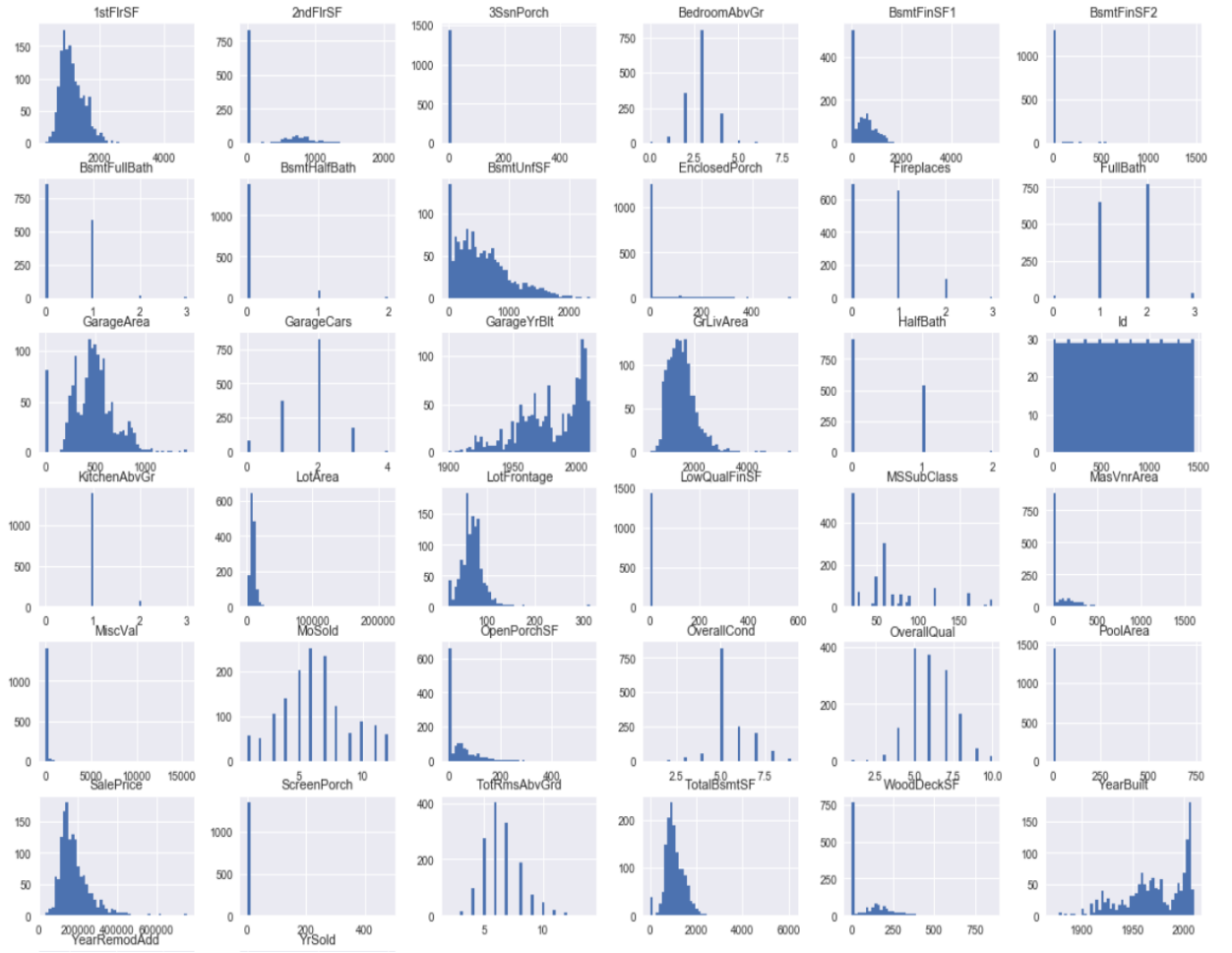
Exploratory Visualization

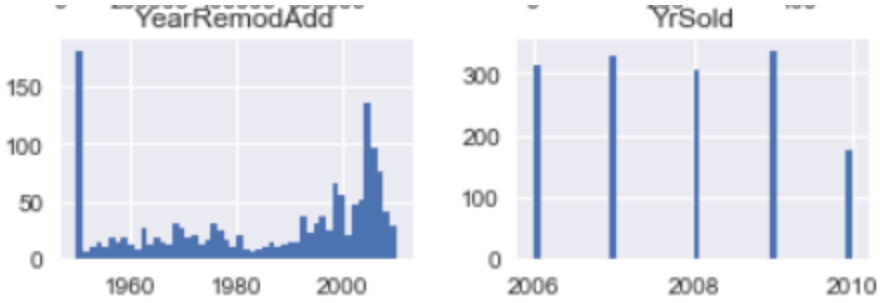
Next, the below graphs are analysis of numerical values that provide good insights.

The 1stFloor sqftt follows a bell shape as expected.
Most of the houses do not have 2nd Floor as the sq Ft is zero.
There is no house with 3 season Porch.
About half of the houses have 3 bedrooms.
Over a third of the houses don't have finished basement.
While a third have a FullBath in the basement.
Most of the houses have some area of unfinished basement.
The older homes may not have had the garage, so most of the garages were built in 1990's and 2000's.
None of them houses have LowQualFin area.
About 40% don't have an Open Porch.
OverallCond and OverallQuality are pretty high, showing a good enforcement of buuilding codes and neighborhoods built by reputable builders.
Most of the houses have two car garages.
About equal number of houses has a no or single fireplace, with a few having 2 or more.
The summer month May, Jun, Jul are the months with maximum homes sold.

None of the houses have a screen Porch
Most of the houses have 5-7 rooms above ground.
Over half the houses don't have a wood deck.
The number of houses sold year over year is fairly consistent above 300, but in 2010, there was a dip by over 30%. This is possibly due to the recession.
There is a surge in remodeling activity after the year 2000.

In [23]:
```
%matplotlib inline
import matplotlib.pyplot as plt
data.hist(bins=50, figsize=(20,15))
plt.savefig("attribute_histogram_plots")
plt.show()
```





Next, with such a large number of available features, it would be good to see correlation of each related to the SalePrice, so that we can identify which ones to focus on.

In [24]:
```
#Analyze the Correlation between the features and SalePrice
corr_matrix = data.corr()
corr_matrix["SalePrice"].sort_values(ascending=False)
```

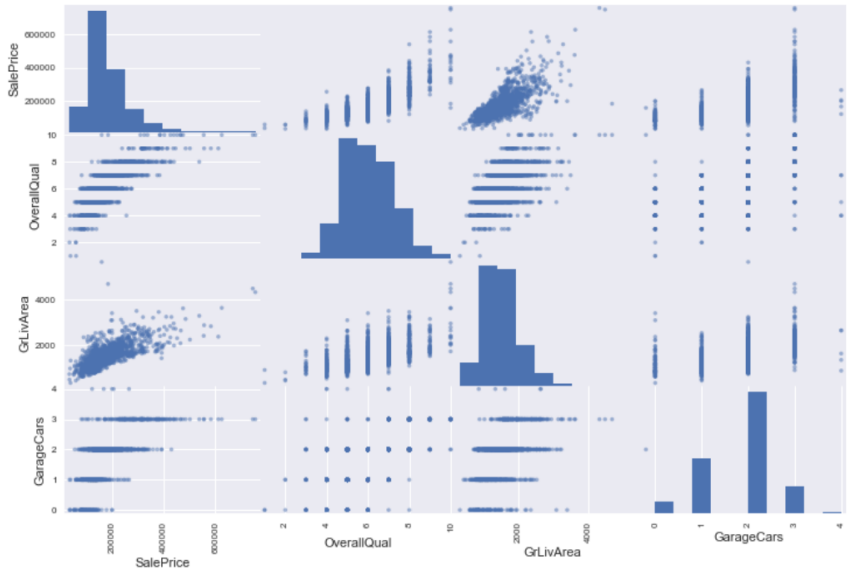
Out[24]:
```
SalePrice        1.000000
OverallQual      0.790982
GrLivArea        0.708624
GarageCars       0.640409
GarageArea       0.623431
TotalBsmtSF      0.613581
1stFlrSF         0.605852
FullBath         0.560664
TotRmsAbvGrd     0.533723
YearBuilt        0.522897
YearRemodAdd     0.507101
GarageYrBlt      0.486362
MasVnrArea       0.477493
Fireplaces       0.466929
BsmtFinSF1       0.386420
LotFrontage      0.351799
WoodDeckSF       0.324413
```

```
2ndFlrSF         0.319334
OpenPorchSF      0.315856
HalfBath         0.284108
LotArea          0.263843
BsmtFullBath     0.227122
BsmtUnfSF        0.214479
BedroomAbvGr     0.168213
ScreenPorch      0.111447
PoolArea         0.092404
MoSold           0.046432
3SsnPorch        0.044584
BsmtFinSF2      -0.011378
BsmtHalfBath    -0.016844
MiscVal         -0.021190
Id              -0.021917
LowQualFinSF    -0.025606
YrSold          -0.028923
OverallCond     -0.077856
MSSubClass      -0.084284
EnclosedPorch   -0.128578
KitchenAbvGr    -0.135907
Name: SalePrice, dtype: float64
```

Next, looked at the plots of the top 9 corelated features vs the SalePrice as these all have correlation >0.5.

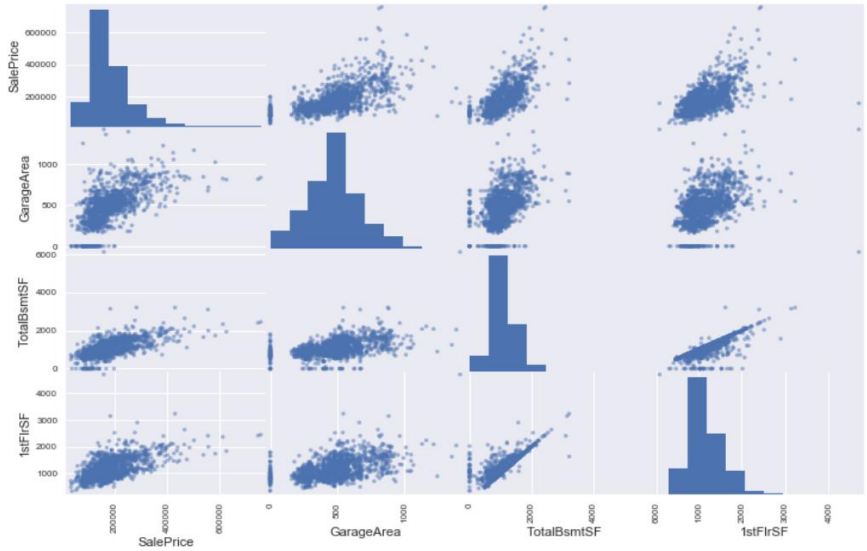
In [32]:
```
#Focus on the top 9 features which have a correlation greater than 0.5
from pandas.tools.plotting import scatter_matrix
#attributes = ["OverallQual", "GrLivArea"]
attributes = ["SalePrice", "OverallQual", "GrLivArea", "GarageCars"]
#attributes = ["SalePrice", "OverallQual", "GrLivArea", "GarageCars", "GarageArea", "TotalBsmtSF","1stFlrSF","FullBath",
#              "TotRmsAbvGrd","YearBuilt"]
scatter_matrix(data[attributes], figsize=(12, 8))
plt.savefig('matrix.png')
```

```
C:\Users\Santosh\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: FutureWarning: 'pandas.tools.plotting.scatter_matrix' is
deprecated, import 'pandas.plotting.scatter_matrix' instead.
```



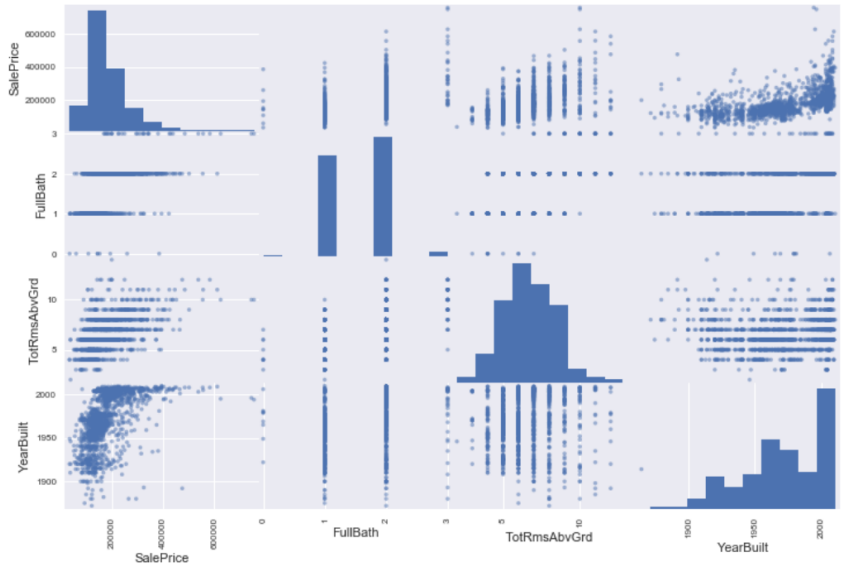
In [33]:
```
from pandas.tools.plotting import scatter_matrix

#attributes = ["SalePrice", "OverallQual", "GrLivArea", "GarageCars"]
attributes = ["SalePrice", "GarageArea", "TotalBsmtSF","1stFlrSF"]
#attributes = ["SalePrice","FullBath", "TotRmsAbvGrd","YearBuilt"]
scatter_matrix(data[attributes], figsize=(12, 8))
plt.savefig('matrix.png')
```

```
C:\Users\Santosh\Anaconda3\lib\site-packages\ipykernel_launcher.py:6: FutureWarning: 'pandas.tools.plotting.scatter_matrix' is
deprecated, import 'pandas.plotting.scatter_matrix' instead.
```



In [34]:
```
from pandas.tools.plotting import scatter_matrix

attributes = ["SalePrice","FullBath", "TotRmsAbvGrd","YearBuilt"]
scatter_matrix(data[attributes], figsize=(12, 8))
plt.savefig('matrix.png')
```

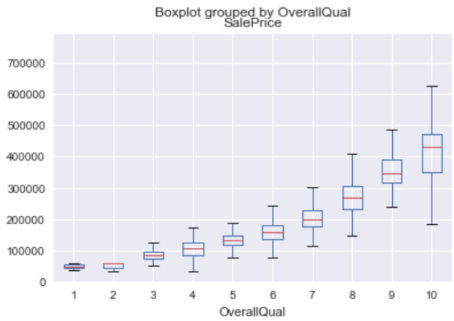
```
C:\Users\Santosh\Anaconda3\lib\site-packages\ipykernel_launcher.py:4: FutureWarning: 'pandas.tools.plotting.scatter_matrix' is
deprecated, import 'pandas.plotting.scatter_matrix' instead.
  after removing the cwd from sys.path.
```



Boxplot review of the top 9 features.

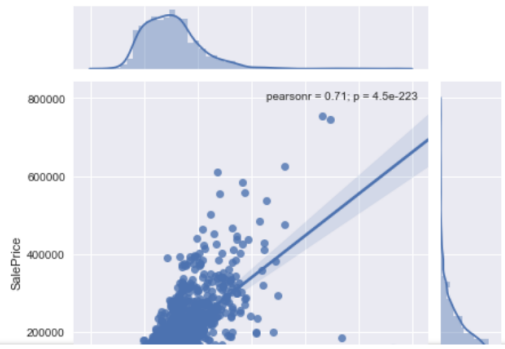
In [36]:
```
data.boxplot(column = 'SalePrice', by = 'OverallQual' )
data.boxplot(column = 'SalePrice', by = 'GrLivArea' )
data.boxplot(column = 'SalePrice', by = 'GarageCars' )
data.boxplot(column = 'SalePrice', by = 'GarageArea' )
data.boxplot(column = 'SalePrice', by = 'TotalBsmtSF' )
data.boxplot(column = 'SalePrice', by = '1stFlrSF' )
data.boxplot(column = 'SalePrice', by = 'FullBath' )
data.boxplot(column = 'SalePrice', by = 'TotRmsAbvGrd' )
data.boxplot(column = 'SalePrice', by = 'YearBuilt' )
#attributes = ["SalePrice", "OverallQual", "GrLivArea", "GarageCars", "GarageArea", "TotalBsmtSF","1stFlrSF","FullBath",
#              "TotRmsAbvGrd","YearBuilt"]
```

Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x175640ecdd8>



In [41]:
```
sb.jointplot(x=data['GrLivArea'], y=data['SalePrice'], kind='reg')
sb.jointplot(x=data['GarageCars'], y=data['SalePrice'], kind='reg')
sb.jointplot(x=data['GarageArea'], y=data['SalePrice'], kind='reg')
sb.jointplot(x=data['TotalBsmtSF'], y=data['SalePrice'], kind='reg')
sb.jointplot(x=data['1stFlrSF'], y=data['SalePrice'], kind='reg')
sb.jointplot(x=data['FullBath'], y=data['SalePrice'], kind='reg')
sb.jointplot(x=data['TotRmsAbvGrd'], y=data['SalePrice'], kind='reg')
sb.jointplot(x=data['YearBuilt'], y=data['SalePrice'], kind='reg')
```

Out[41]: <seaborn.axisgrid.JointGrid at 0x1757257e0b8>



Algorithms and Techniques

I plan to use Random Forest and XGBoost algorithms to develop model. These are ensemble methods, which means they use multiple learning models to gain better predictive results

Random Forest:

Random forests, also known as random decision forests, are a popular ensemble method that can be used to build predictive models for both classification and regression problems. Random forest model creates an entire forest of random uncorrelated decision trees to arrive at the best possible answer. This constructs a multitude of decision trees. Decision trees are simple but intuitive models that utilize a top-down approach in which the root node creates binary splits until a certain criterion is met. This binary splitting of nodes provides a predicted value based on the interior nodes leading to the terminal (final) nodes.

Ref: https://www.datascience.com/resources/notebooks/random-forest-intro

XGBoost:

XGBoost stands for eXtreme Gradient Boosting. XGBoost is very popular because of Execution speed and Model Performance. It dominates structured or tabular datasets on classification and regression predictive modeling problems. Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. Gradient boosting is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.

Ref: https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/

Benchmark Model

To Benchmark the results, the model was run for three features. The model was then trained used random forest. The Random forest and XGB provided the below accuracy score. SVC was also run (for experiment purposes) on the data set, it took a lot of time and provided very poor results and hence is not considered for further modeling.

```
In [17]: data_train_cols = data_train[['GrLivArea','FullBath','OverallQual','SalePrice']]
         score_train,score_test = rand_f(data_train_cols)

         score_train,score_test = xgbr(data_train_cols)

         #score_train,score_test = SVCM(data_train_cols)


         RANDOM F Accuracy on training set: 0.940
         RANDOM F Accuracy on test set: 0.662
         XGB Accuracy on training set: 0.935
         XGB Accuracy on test set: 0.604
```

```
SVC Accuracy on training set: 0.084
SVC Accuracy on test set: 0.017
```

III. Methodology

Data Preprocessing Step

The preprocessing that needed to take place before the data could be trained were:

- Locate and clean the Null values
- The POOLQC column had Nulls, these were filled with None.
- MiscFeature, Alley, Fence, FireplaceQu were al filled with None
- LotFrontage was filled with the median of the frontage of the Neighborhood
- 'GarageType', 'GarageFinish', 'GarageQual', 'GarageCond' were filled with None
- GarageYrBlt', 'GarageArea', 'GarageCars' were filled with 0
- 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF','TotalBsmtSF', 'BsmtFullBath', 'BsmtHalfBath' were filled with 0
- 'BsmtQual', 'BsmtCond', 'BsmtExposure', 'BsmtFinType1', 'BsmtFinType2' were filled with None
- MasVnrType was filled with none
- MasVnrArea was filled with 0.
- MSZoning was filled with the Mode of the values
- Functional was filled with Typ
- Electrical, KitchenQual, Exterior1st, Exterior2nd, SaleType were filled with mode.
- MSSubClass was filed with None

Implementation

- The data was then preprocessed by treating for basic null values.
- Due to many columns and the potential redundant features. The data set was then trained on a Random forest model and a benchmark score was obtained using top 3 correlated features

- Then increased the number of features to top nine correlated columns to the target variable. All of them had a correlation coefficient > 0.5. This data was fed into Random Forest and XGBoost. This improved the RMSE score for Random Forest from 44847 to 33005, almost a 30% improvement. The XGBoost RMSE score interestingly stayed the same. Also looking at the Accuracy of XGBoost on the training dataset is 1. Which means XGBoost model is probably overfit, with no scope for improvement.

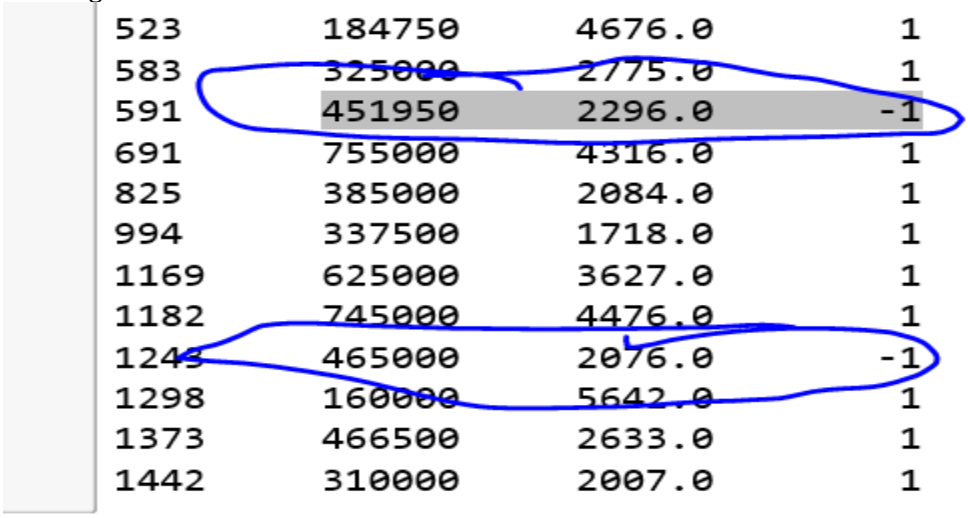
```
In [29]: attributes = ["SalePrice", "OverallQual", "GrLivArea", "GarageCars", "GarageArea", "TotalBsmtSF","1stFlrSF","FullBath",
                      "TotRmsAbvGrd","YearBuilt"]

         data_train_cols = data_train[attributes]
         score_train,score_test = rand_f(data_train_cols)

         score_train,score_test = xgbr(data_train_cols)

         #score_train,score_test = SVCM(data_train_cols)

RANDOM F Accuracy on training set: 0.966
RANDOM F Accuracy on test set: 0.852
('RMSE', 33005.33802341278)
XGB Accuracy on training set: 1.000
XGB Accuracy on test set: 0.806
('RMSE', 34716.76969790397)
```

- The data was then further cleaned now the RMSE score increased slightly to 36520, however the XGBOOST RMSE stayed the same.

```
In [31]: data_train_clean = data_train

         print clean_missing_data(data_train_clean)
         data_train_cols = data_train_clean[attributes]
         score_train,score_test = rand_f(data_train_cols)
         score_train,score_test = xgbr(data_train_cols)

('Initial Missing Data', PoolQC          1453
MiscFeature     1406
Alley           1369
Fence           1179
FireplaceQu      690
LotFrontage      259
GarageCond        81
GarageType        81
GarageYrBlt       81
GarageFinish      81
GarageQual        81
BsmtExposure      38
BsmtFinType2      38
BsmtFinType1      37
BsmtCond          37
BsmtQual          37
MasVnrArea         8
MasVnrType         8
Electrical         1
Utilities          0
dtype: int64)
SalePrice        0
SaleCondition    0
RoofMatl         0
Exterior1st      0
Exterior2nd      0
MasVnrType       0
```

```
GarageCars       0
PoolQC           0
Fence            0
MiscFeature      0
MiscVal          0
MoSold           0
YrSold           0
SaleType         0
GarageArea       0
GarageFinish     0
Electrical       0
HalfBath         0
1stFlrSF         0
2ndFlrSF         0
LowQualFinSF     0
GrLivArea        0
BsmtFullBath     0
BsmtHalfBath     0
FullBath         0
BedroomAbvGr     0
GarageYrBlt      0
KitchenAbvGr     0
KitchenQual      0
TotRmsAbvGrd     0
Functional       0
Fireplaces       0
FireplaceQu      0
GarageType       0
Id               0
Length: 80, dtype: int64
RANDOM F Accuracy on training set: 0.966
RANDOM F Accuracy on test set: 0.818
('RMSE', 36520.08495500829)
XGB Accuracy on training set: 1.000
XGB Accuracy on test set: 0.806
('RMSE', 34716.76969790397)
```

- It was noticed that there are a few outliers in the features. E.g there are a couple of houses whose sale price is high, but the Living area is actually quite modest. These are probably skewing the model.



523	184750	4676.0	1
583	325000	2775.0	1
591	451950	2296.0	-1
691	755000	4316.0	1
825	385000	2084.0	1
994	337500	1718.0	1
1169	625000	3627.0	1
1182	745000	4476.0	1
1243	465000	2076.0	-1
1298	160000	5642.0	1
1373	466500	2633.0	1
1442	310000	2007.0	1

- Hence, removed these outliers. And ran the model again. This time the RMSE score for Random Forest dropped to 31477, however the RMSE score for XGBoost more than doubled to 74696. This would indicate that the earlier XGBoost model was heavily overfitted. And the accuracy score on the training set is consistently is 1.

In [34]:
```
# remove outliers from primary training data sets
outliers = outlier_detect.loc[outlier_detect['outlier']==-1].index.values
data_train_cols.drop(outliers, inplace=True)
score_train,score_test = rand_f(data_train_cols)
score_train,score_test = xgbr(data_train_cols)
```

```
C:\Users\Santosh\Anaconda3\envs\python27\lib\site-packages\ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
  after removing the cwd from sys.path.
RANDOM F Accuracy on training set: 0.957
RANDOM F Accuracy on test set: 0.831
('RMSE', 31477.976669106163)
XGB Accuracy on training set: 1.000
XGB Accuracy on test set: 0.843
('RMSE', 74696.08270551689)
```

In [37]:
```
print(data_train_cols.shape)
```

```
(1458, 10)
```

- Next, in an attempt to improve the score tried changing some of the hyperparameters. E.g. n_estimators and gamma. There was no change by changing the gamma value.
- Increasing the n_estimators from 10 to 1000 for Random Forest improved the score to 29674. For XGBoost the n_estimators was increased from 30 to 3000. However even this did not change the RMSE values. Hence the best RMSE value obtained is for Random F.

In [46]:
```
# After changing to 100 for Random F and 3000 for xgboost
score_train,score_test = rand_f(data_train_cols)
score_train,score_test = xgbr(data_train_cols)
```

```
RANDOM F Accuracy on training set: 0.975
RANDOM F Accuracy on test set: 0.850
('RMSE', 29659.274939860414)
XGB Accuracy on training set: 1.000
XGB Accuracy on test set: 0.843
('RMSE', 74696.08270551689)
```

Results

It can be seen from the benchmark and the models that the scoring is higher in both cases than the benchmark model.

ITERATION	RandomForest			XGBoost		
	RMSE	Accuracy on Training Set	Accuracy on Testing Set	RMSE	Accuracy on Training Set	Accuracy on Testing Set
on 3 Features **BENCHMARK**	44847.53	0.938	0.726	34716.77	0.935	0.604
Increased # of features to 9	33005.34	0.966	0.852	34716.77	1	0.806
Data cleaning	36520.08	0.966	0.818	34716.77	1	0.806
Removed Outlier from GrLivingArea	31477.98	0.957	0.831	74696.08	1	0.843
Adjusted hyperparameters n_estimators and Gamma **BEST**	29674.18	0.975	0.85	74696.08	1	0.843

Conclusion

The model has been trained using 1460 rows of housing prices data. The model has the max of 85% accuracy score with RMSE Score of 29674 using Random Forest and n_estimators of 1000.

It is interesting to note that XGBoost failed in this model as the RMSE score is not changing and the accuracy score on the Training dataset is always 1.

Pre-processing steps have been applied as necessary to maximize the use of the available data for training. The model was run for different values of the hyperparameters. There are some refinements that could be done to improve the model accuracy. I considered the top nine corelated features for prediction, and it received reasonable accuracy., with some data pre-processing. We can further analyze each feature and remove outliers from other features and incorporate other features into the model.

References:

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

https://www.kaggle.com/mchatham/ames-housing-regression/code

https://www.kaggle.com/surya635/house-price-prediction/code

https://www.fabienplisson.com/random-forest-and-grid-search/

https://www.kaggle.com/gchshin/house-price-prediction-challenge/code

https://www.kaggle.com/jasonduncanwilson/predicting-iowa-house-prices

https://www.analyticsvidhya.com/blog/2016/03/complete-guide-parameter-tuning-xgboost-with-codes-python/

https://nycdatascience.com/blog/student-works/kaggles-advanced-regression-competition-predicting-housing-prices-in-ames-iowa/

https://www.analyticsvidhya.com/blog/2016/02/7-important-model-evaluation-error-metrics/

https://www.kaggle.com/neviadomski/how-to-get-to-top-25-with-simple-model-sklearn