Data Analysis for housing model - Documentation

Created By: Raj Kumar and Aadhikya Ghimire

Abstract

This purpose of this project is to tune housing model data set and apply machine learning algorithms to make accurate predictions for housing prices. This documentation provides detail steps of tuning the data set to make if a perfect fit for prediction models and later apply those models to make predictions of previously unseen housing data.

Introduction

Data analysis on housing model will be done using Python libraries such as Pandas, Scikit-learn, matplotlib, and Seaborn. These libraries contains methods that will help in determining the attributes that contribute the most while making a sale prediction for a house. Once the model is all set, meaning we have our data all ready we will be implementing *Gradient boosting classifier* model to fit the data and make predictions. How accurate our predictions are will be determined from the accuracy mean given by our model.

Data Description

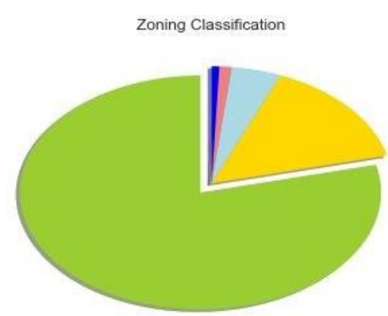
The given data set contains attributes that describes different features in a house. These features can be its location, the size of the house, the year built, and etc. Imagine you are going to buy a house and you would like to have some ideal things planned in your head that you want to see in your dream house. Those expectation can include things like a yard, garage size, how old the house is, its neighborhood, and much more. Those expected features are one of the example that our data set has.

Experiment

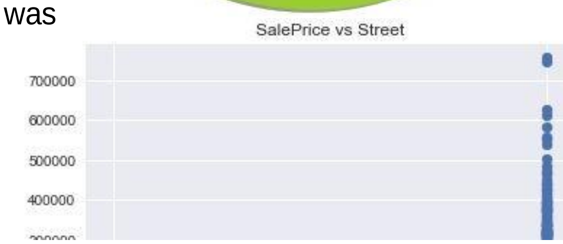
- **Pre-Processing and Analysis:**

The pre-processing of data set was done by intuition, trial and error and combinations of graphs that will make the best out of the data set. First, we wrote a function to check the percentage of missing values in the columns and dropped the columns that had more than 90 percent of missing values. We converted classification columns into numbers by mapping them into value. We filled the N/A values with the mean or the mode of the column depending on its nature.
We converted the data into integers and tried to figure out the correlation between them and SalePrice. We kept the data with high correlation and used it in our algorithm. Below are the few attributes that were preprocessed. However, we preprocessed all of the data and provided comments in the file - accordingly.

Neighborhood: Other than few extra spikes, the SalePrice is not really affected that much by Neighborhood, as it is all between 100000 - 200000. Aadhikya and I believe, the Neighborhood really doesn't matter. Hence, we would drop this column'''



MSZoning: We discretized it because a large portion of the data is of three values. As shown in the picture. ['RL - 78.8%', 'RM - 14.9%', 'C (all) - 4.5%', 'FV - 1.1%', 'RH - 0.7 %']. However, it's corr value with SalePrice was very low; hence, we decided not to keep this.



Street: The graph shows that majority of the data is made up Pave value; therefore, we will not use this column in our final evaluation.

Alley, MiscFeature, PoolQC, Utilities, Fence, FireplaceQu, LotFrontage: All of these attributes have a high percentage of NA values. Therefore, it seems sensible to drop these columns because we don't have a large portion of the data.

```
PoolQC          99.486804
Fence           79.838710
MiscFeature     96.260997
```

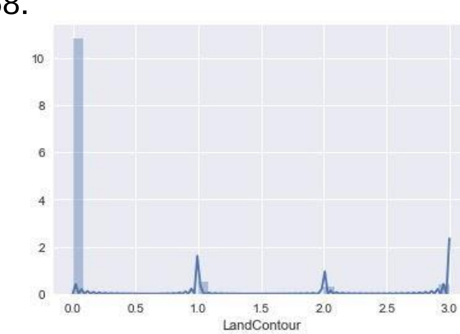
```
MSSubClass       0.000000
MSZoning         0.000000
LotFrontage     18.035191
LotArea          0.000000
Street           0.000000
Alley           95.381232

FireplaceQu     45.747801
GarageType       5.425220
GarageYrBlt      5.425220
GarageFinish     5.425220
```

LotShape: [Reg - 62.2 %', 'IR1 - 34.4 %', 'IR2 - 2.6 %', 'IR3 - 0.7 %'] - the percentage of the values show that it is all Reg, IR1 and IR2. Therefore, after discreting it we tried to figure out its correlation with the SalePrice - we got 0.289812715368.
Hence, we decided not to use this column.

LandContour, LotConfig, LandSlope: Doesn't have a high correlation with SalePrice.

```
['Inside - 71.5 %', 'FR2 - 18.3 %', 'Corner - 6.7 %',
'CulDSac - 3.2 %', 'FR3 - 0.2 %']
0.0297332320677
```



HouseStyle: 'It is a high negative correlation with SalePrice - therefore we would keep it.

RoofStyle: Data represents a lot like HouseStyle; therefore, assuming a high correlation between roofstyle and housestyle - we would drop this as well.

And few more...

After the entire preprocessing we figured that

['Id', 'LotArea', 'HouseStyle', 'OverallQual', 'OverallCond', 'YearBuilt', 'YearRemodAdd', 'MasVnrArea', 'ExterQual', 'Foundation', 'BsmtCond', 'TotalBsmtSF', '1stFlrSF', 'GrLivArea', 'BsmtFullBath', 'GarageYrBlt', 'GarageArea', 'SaleCondition', 'Duplex', 'Unf']

these were the final columns that we wanted to use in the algorithm. - According to the correlation table below.

```
Id              -0.018440
MSSubClass      -0.078692
LotArea          0.360997
LandSlope       -0.044338
HouseStyle      -0.185197
OverallQual      0.809959
OverallCond     -0.221416
YearBuilt        0.560706
YearRemodAdd     0.521565
MasVnrArea       0.504359
ExterQual       -0.140797
Foundation      -0.498121
BsmtCond        -0.150422
BsmtFinSF1       0.345986
BsmtFinSF2      -0.060187
BsmtUnfSF        0.246869
TotalBsmtSF      0.652839
HeatingQC        0.330178
1stFlrSF         0.637372
2ndFlrSF         0.278487
LowQualFinSF    -0.030120
GrLivArea        0.746729
BsmtFullBath     0.235787
BsmtHalfBath    -0.073076
FullBath         0.597104
HalfBath         0.183964
BedroomAbvGr     0.148608
KitchenAbvGr    -0.118979
TotRmsAbvGrd     0.607896
Fireplaces       0.476300
GarageYrBlt      0.520014
GarageCars       0.666636
GarageArea       0.668532
WoodDeckSF       0.264310
OpenPorchSF      0.341129
EnclosedPorch   -0.045918
3SsnPorch        0.046743
ScreenPorch      0.082497
PoolArea         0.008878
MiscVal         -0.021625
MoSold           0.083156
YrSold          -0.035464
SaleCondition    0.403459
SalePrice        1.000000
Duplex          -0.106118
Fin              0.376185
RFn              0.100855
Unf             -0.427710
Name: SalePrice, dtype: float64
```

Algorithms and Parameterization

<u>Methods Explanation:</u>

def encode(*df*, *cols*): Takes two parameters - a dataframe and a list of cols. This method implements binary encoding on particular columns in the dataframe and returns that dataframe.

def barPlot(df, var): Takes two parameters - a data frame and a column (called var). It plots a graph between "Saleprice" and the column that is passed to this function

def checkRange(df, var): Given a column of variables, checkRange will show the percentage of each element in that column.

def histogram(df, var): Plots histogram of a column in the dataframe.

def join(df1, df2): Combines two dataframes together.

def scatterPlot(df, var): Returns a scatter plot of a column with respect to the "SalePrice".

def checkMissingValues(df): Returns the percentage of NA values in a column.

def encode(df, s): Creates new columns for non numerical attribute -- (s) and does Binary Encoding on them. This will increase the dimensionality of a column.

def c2(df, colName): Return the correlation of a particular column with "SalePrice".

Conclusion:

After we have our dataset all set, we implemented Gradient Boosting Regressor model to run predictions. We ended up with a mean of ***"0.85".*** This means that our model is 85% accurate to make predictions on unseen problems.