Assignment Two

Will Peck - wrp3

9th March 2017

Contents

1 Introduction

This report discusses the methods and decision making processes used to analyse the house price dataset.

2 Data Cleaning

The data cleaning exercise applied to the dataset included three stages. Completing the stages resulted in useful data for determining correlations and projections. By recursively completing the data cleaning process; the data naturally scaled to a managable size.

2.1 Removing Inapplicable Data

The original number of rows contained within the dataset totalled 27395. The following filtering process removed a large number of rows where the data is unusable because of crucial missing data.

Redundant data was defined by rows where no sale price value was given and then where the neighbourhood was unknown. Therefore, the first cleaning operation involved the removal of any row where the price was unknown, followed by removing rows where the neighbourhood field was empty.

```
...
# Remove rows where 0 values for 'SALE_PRICE'
   rawdata.replace(0, np.nan, inplace = True)
   rawdata = rawdata[pd.notnull(rawdata['SALE_PRICE'])]

# Replace all spaces with '_' and remove rows where
# value is '_' in column 'NEIGHBORHOOD'
   rawdata.replace(r'\s+', '_', regex = True, inplace = True)
   rawdata = rawdata.loc[rawdata['NEIGHBORHOOD'] != '_']
...
```

The next operation removed all duplicate entries where identical rows were discovered when compared across all columns. At this stage, the number of rows was reduced to 18145.

Whilst the dataset length could be reduced further by removing other missing values from other columns; it was considered that initially outliers might be more accurately identified when the highest possible number of complete neighbourhood and sale price rows were used.

```
...
# Remove duplicates
   pd.DataFrame.drop_duplicates(rawdata, keep=False,           inplace=True)
...
```

Some columns contained data that when loaded, Python interpreted the values as Strings. These values needed to be parsed into numeric datatypes.

```
...
# Remove ',' and '$' from value String
   rawdata['SALE_PRICE'] = rawdata['SALE_PRICE'].str.replace(',','')
    rawdata['SALE_PRICE'] = rawdata['SALE_PRICE'].str.replace('$','')

# Change 'SALE_PRICE' datatype to numeric
   rawdata['SALE_PRICE'] = pd.to_numeric(rawdata['SALE_PRICE'])
...
```

2.2 Identifying Outliers and Data Verfication

To define the significant data within the dataset; outliers were filtered with the objective of retaining only data values reasonably close to the interquartile range.

Furthermore; to create an accurate as possible model whilst working with a managable dataset length, it was considered to be important to maintain as much of the original 'image' in the data as possible. As an analogy, when reducing the resolution of an image of a car, it is important to retain enough pixels such that the new image is still identifiable as a car. Therefore; columns with the most complete data were chosen as the basis for the initial data cleaning process. ie. Columms where the number of rows with significant values were closest to the original number of rows were selected. These columns included sale price and sale date.

To improve the accuracy further; instead of attempting to clean the outliers in one pass against the whole dataframe. It was considered that accuracy could be improved by applying the function against the properties location. This was based on an assumption that property sale prices would vary in this way. The graph below demonstrates the difference in average selling price between the different neighbourhoods.

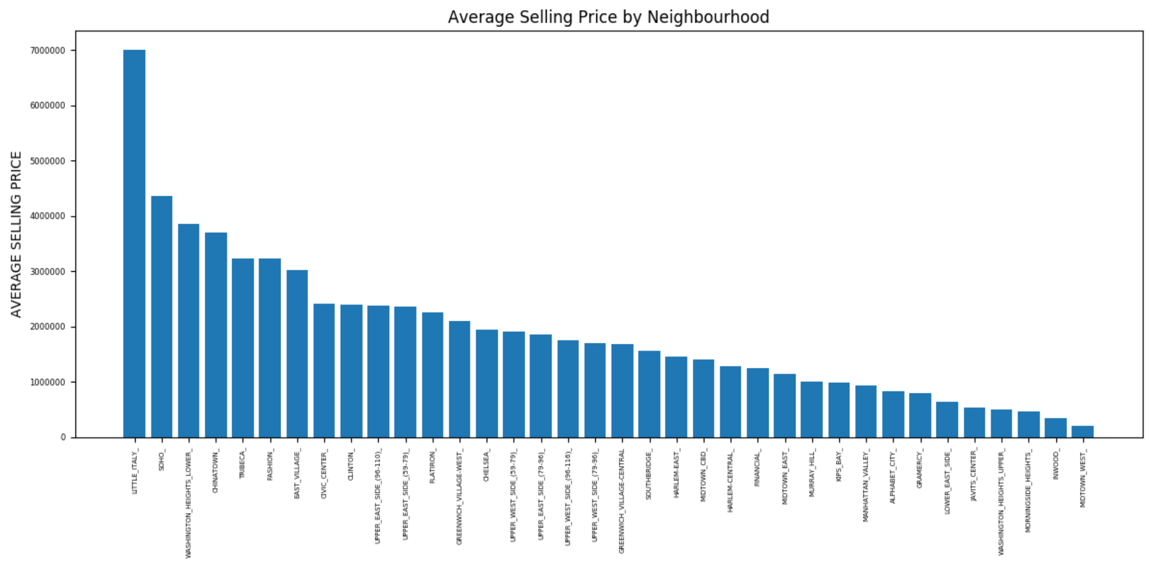

Figure 1: ASP by Neighbourhood - After filtering outliers

For each neighbourhood; the outliers were recursively identified as any value recorded either outside of 1.5 times interquartile range minus the lower quartile or 1.5 times the interquartile range plus the upper quartile range. The resulting data is illustrated in the two example scatter graphs below.

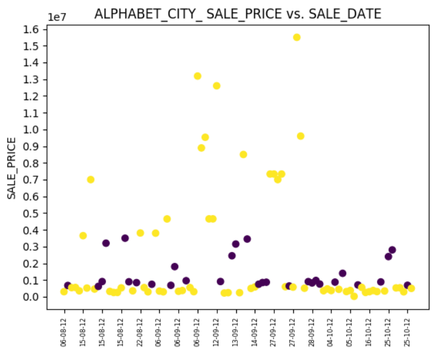

Figure 2: Sale Prices vs. Sale Date - Before filtering outliers

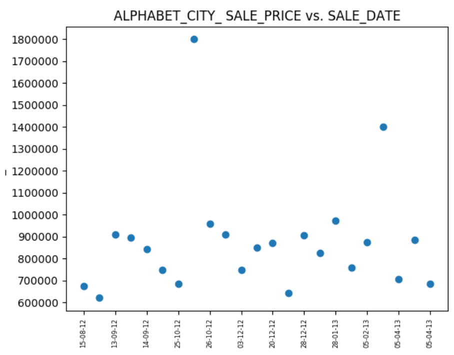

Figure 3: Sale Prices vs. Sale Date - After filtering outliers

The building class column was also passed into the function alongside saleprice and used instead of the data to further identify and remove outliers.

Once the outlier removal process completed; the total number of rows within the dataframe reduced to 4299. This left a number of columns which contained incomplete numeric data which using imputation could back-filled. For example, the missing values for gross square feet could be estimated by calculating the current mean percent penetration into the average selling price from the existing data. Then replacing null square foot values by multiplying this penetration value by the actual selling price for each row.

```
...
# Av. Gross SQFT / ASP
  spdesc = clndata['SALE_PRICE'].describe()
  sfdesc = clndata['GROSS_SQ_FT'].describe()
  asp = spdesc[1]
  asqf = sfdesc[1]
  rat = asqf / asp

  temp = clndata[clndata['GROSS_SQ_FT'] == 0]
  temp['GROSS_SQ_FT'] = temp['SALE_PRICE'].apply(lambda x : x * rat)
  dfs = [clndata, temp]
  clndata = pd.concat(dfs)
  clndata.drop(clndata[clndata['GROSS_SQ_FT'] == 0], inplace = True)
...
```

The function above was applied to gross and land square feet columns. Other potential columns were residential, commercial and total units. However, when applied the function returns values less than 0. It was decided that commercial and residential unit columns should be dropped and the total units column back-filled and rounded to the next integer. This is because it is not feasible to predict the commercial status of a property with the given dataset.

To further improve the predictability of the model; the sale date and year columns were converted into numeric types and represented as integers from 0 for the first entry (columns 'NUMERIC DATE' and 'NUM YEAR BUILT'). The sale date integers were incremented according to the number of months and for year built the integer represents number of years. Once converted in this way, date and year could be used more readily in the linear regression.

3 Analytics and Correllation

The scatter matrix below allows for comparison between the variables recorded across different columns. Some correlations are immediately obvious. For example; the density of plots within 'NUM YEAR BUILT' correlates with the peak in the 'SALE PRICE' plot. It could be argued that this would be expected because the number of properties sold of any value would likely come from the years where the most properties were built. Similarly, this is reflected in the 'NUMERIC DATE' (numeric representation of date sold) and 'NUM YEAR BUILT' correlation across highest density of points. Prior to generating the scatter plot the data was normalised to plot different scales against each other.

Furthermore, the 'GROSS SQ FT' plot is almost identical to the 'SALE PRICE'. Whilst this might be expected; the caveat here is that a significant proportion of this data was back-filled. Therefore, it could be argued that the predicted values were made to fit the model. However, the accuracy of 'TOTAL UNITS' where a similar method was applied could be supported by correllating against 'NUMERIC DATE' which is original and complete data. The pattern and shape of the plot is similar.

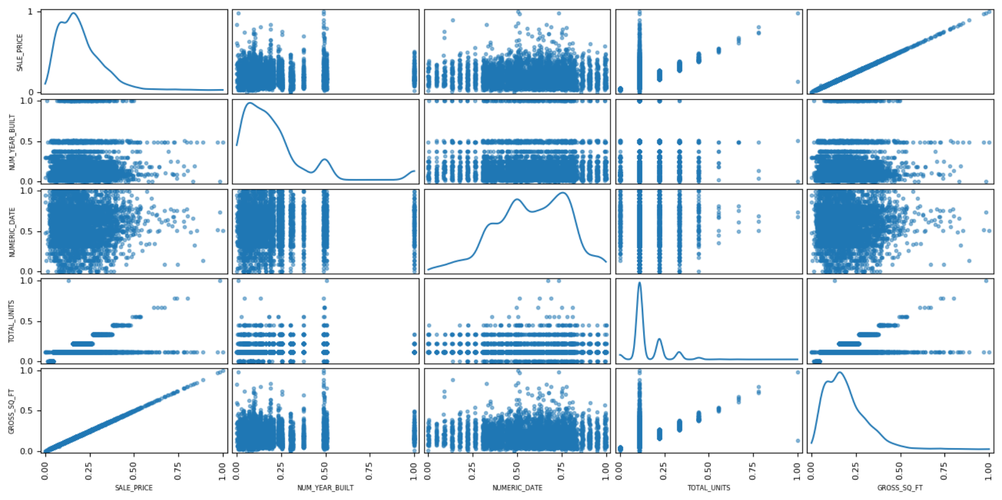

Figure 4: Scatter Matrix

4 Projection

Linear regression was used as the basis for the predictive model due to its simple implementation and ability to consider multiple variables to support the output. It has provided a method to discover which of these variables within the limited dataset impact property selling prices for Manhatten.

The formala and equation used for the linear regression is detailed below.

$y = y{-}intercept = 0.3907..$ $df = tableofdatarelatedtoManhattenproperties$ $b_0 = df['BLOCK'] = -0.006..$

$b_1 = df['LOT'] = 0.074..$

$b_2 = df['TOTAL\_UNITS'] = 0.243..$

$b_3 = df['GROSS\_SQ\_FT'] = 0.932..$

$y = f(x)$

$f(x) = b_0x_0 + b_1x_1 + b_2x_2 + b_3x_3 + b_4x_4 = 0.3907x_0 - 0.006x_1 + 0.074x_2 + 0.243x_3 + 0.932x_4$

The above model was implemented in Python using the 'pandas' library from code adapted from the lab 6 worksheet.

```
Weight coefficients:
                BLOCK: -0.0060720369211053485
                  LOT: 0.07445895000580982
          TOTAL_UNITS: 0.24330553346717051
          GROSS_SQ_FT: 0.9326038385200704
R squared for the training data 0.8687918687919031
Score against test data: 0.8471340271534613
R squared for the test data is: 0.847134027153
```

The R squared value is the measure between 0 and 1 of how well the test data fits to the regression model. In this case; the R squared value indicates that this model explains a high proportion of the variability of the data. However, in order to prove the model, the output would need to be verified by checking for residual errors.

Assuming the accuracy of the model, and when examining the graph plot of the model output; the plot appears to follow a logarithmic shape. This means that any predicted sale values have a ceiling which would have an interesting impact on investment decisions or perhaps help inform on the best time to buy or sell a property.

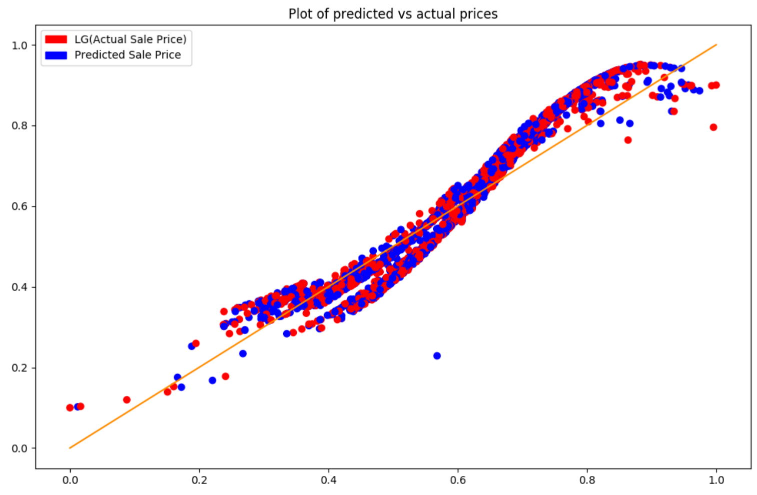

Figure 5: Linear Regression - Best Fit

5 Conclusion

Throughout this assignment; an attempt was made to preserve as much of the original data as possible whilst making sure as many missing values were either removed or replaced. The data cleaning process involved a facinating but steep learning curve before revealing any value in the data.