HW3 REPORT

Data Cleaning:

1) Column deletion :

- The properties data set available at kaggle has 58 columns. On observing the values in the data set we found out that for a large number of columns most of the rows are empty. Therefore, We decided a threshhold (i.e. 0.3) to remove columns that are sparse in data and removing them from the dataset. This left us with 25 columns.

- '**fips'** - Federal Information Processing Standard code is the same for most of the data.

- '**bathroomcnt'** is the same as '**calculatedbathnbr'** and **'fullbathcnt',** Hence we retain **'bathroomcnt'** and delete the other 2.

- **'calculatedfinishedsquarefeet'** and **'finishedsquarefeet12**' are similar, and calcuatedfinishedsqarefeet might be more accurate and one of them could be removed so removed **'finishedsquarefeet12'**.

- Removed redundant data **'censustractandblock'** (similar to '**rawcensustractandblock**'). Census data contains information about location i.e. block and district which can be obtained from 'regionidzip'.

- Removed '**propertycountylandusecode'** as the categorical attribute propertylandusetypeid provides same information

- Removed **'assessmentyear'** because for around 93.1% of the rows the value of this column is same.

- Removed '**roomcnt**' as a lot of fields under this column has 0 as value which is not a possible value. Therefore, we removed this column for better results.

  This left us with 16 columns.

3) filling the NaN values wisely:

- **'landtaxvaluedollarcnt' / 'structuretaxvaluedollarcnt' / 'totaltaxvaluedollarcnt'**: we observed that the relation (***landtax + structuretax = totaltax***) holds for these three columns. So we tried to fill as many NaN values as possible in the columns 'landtaxvaluedollarcnt' and 'structuretaxvaluedollarcnt' using following:

If 'landtaxvaluedollarcnt' is null:

Fill 'landtaxvaluedollarcnt'= 'totaltaxvaluedollarcnt'- 'structuretaxvaluedollarcnt' if 'totaltaxvaluedollarcnt'is not null and 'structuretaxvaluedollarcnt' is not null

If 'structuretaxvaluedollarcnt' is null:

Fill 'structuretaxvaluedollarcnt'= 'totaltaxvaluedollarcnt'- 'landtaxvaluedollarcnt' if 'totaltaxvaluedollarcnt'is not null and 'landtaxvaluedollarcnt' is not null

- **'longitude' and 'latitude'** : We had 11437 NAN values for these columns. We observed that longitude and latitude have a high relationship with 'regionidzip', 'regionidcity' and 'regionidcounty' but we found that for all empty values of latitude and longitude the corresponding 'regionidzip', 'regionidcity' and 'regionidcounty' are also empty. Therefore, we substituted the NAN values in 'longitude' and 'latitude' with the mean values.

- **'calculatedfinishedsquarefeet' and 'lotsizesquarefeet'** : We created a new dataframe with 'lotsizesquarefeet' and 'calculatedfinishedsquarefeet' and caculated the difference of the row values for these two columns. We then dropped the rows with negative difference (because for a single storey building , this diffrence will be postive. If it is negative that means bulding is other than single storey , which are very less in number in kaggle dataset). Next, we calculated the difference fraction for the remaining rows by dividing the difference computed with 'lotsizesquarefeet'. The mean of the difference fraction gave us the noise that we used to fill the empty 'lotsizesquarefeet' and 'calculatedfinishedsquarefeet' using the below listed relationships :-

  if 'lotsizesquarefeet' is null and 'calculatedfinishedsquarefeet' is not null
  lotsizesquarefeet = calculatedfinishedsquarefeet/(1-noise)

  if calculatedfinishedsquarefeet is null and lotsizesquarefeet is not null
  calculatedfinishedsquarefeet= (1-noise)*lotsizesquarefeet

For the remaining NAN values we substituted the NAN values with the mean value for 'calculatedfinishedsquarefeet' and with the median value for 'lotsizesquarefeet'.

- '**bathroomcnt' , 'bedroomcnt' :** 3-4 values were being used repeatedly in these columns. Therefore, the idea of replacing the NAN values with median value looked promising.

- **'yearbuilt' :** For this column most of the values were similar. Therefore we substituted the NAN fields with the median value.

4) Addition of external datasets (Population, Per capita income, actual cost of the house):

- We found that the 'regionidzip' values given in the data set are not real postal codes, therefore we made buckets of all the unique 'regionidzip' values and put all the latitudes and longitudes corresponding to a particular 'regionidzip' into it's bucket. We picked up the first latitude and longitude from each bucket, passed it into the google API (**pygeocoder**) and found out the correct details for county name, city name and the postal code ('regionidzip') corresponding to each unique incorrect 'regionidzip' of our data set.By this, we got mapping between regionidzip with city, county and postal code.

Addition of Population :

- Using the postal code values that we retrieved above, we calculated the population value for each postal code using the csv file downloaded from the below link (https://blog.splitwise.com/2013/09/18/the-2010-us-census-population-by-zip-code-totally-free/). This csv gave us mapping between postal codes and population.

Addition of per capita income:

- Using the postal code data, we retrieved the average per capita income for each postal code .As the data present on webpages was not available directly, webpage crawling helped us to parse the website (http://zipatlas.com/) and get the required mapping between postal code and per capita income . We observed that postal codes for our data set vary from 90001 to 95399 (indicating the areas in California region). This observation helped us to narrow down on our webpage crawling.

Addition of actual cost of the house:

- By observing the dataset, We found that dataset has only 6 unique counties. We gathered property tax rate in these counties from webpage () (as the number is small). After that, we used this property tax rate to find the actual cost of the by using tax amount .

5) Further removal of various columns after addition of external dataset

- As now , we have acutal zipcode, regionidcity , regionidcounty and regionidzip is of least use.Hence, These columns are being dropped.
- We have calculated NAN values in structuretaxvaluedollarcnt and landtaxvaluedollarcnt from taxvaluedollarcnt. Hence, taxvaluedollarcnt can be dropped.

- We have used taxamount to get the actual price of the house by using tax rate in particular county the house is located. Hence, removing the column after external dataset price got added.
- We have used actual zipcode to integrate the external dataset with zillow's original property dataset. Zipcode serves its purpose after the integration. Hence, this column can be dropped .

Now, our m*n dataset is ready with zero NAN values and integrated external data

Final columns in the dataset are { bathroomcnt , bedroomcnt, calculatedfinishedsquarefeet , latitude, longitude, lotsizesquarefeet, yearbuilt , structuretaxvaluedollarcnt, landtaxvaluedollarcnt ,price, population, income }

QUESTION1

Scoring function to measure the desirability of a house

Scoring Function = -5 (price of the house) + 4 (per capita income of the city) + 3 (lot size square feet) -2 (land tax) + (population of the region for a given postal code)

We gave ranks to the shortlisted variables and assigned the weightage to the variables according those ranks(Better rank implies more weightage)

Rank	Variable
1	price of the house (additional data)
2	per capita income of the city (additional data)
3	lot size square feet (existing data)
4	land tax (existing data)
5	population of the region for a given postal code (additional data)

Price of the house : It has a negative coefficient (-5) because if we keep the other variables constant, a higher price of the house would make the house less desirable for the buyer. It has rank 1 because according to us, this is the most important factor for a buyer and will be considered before any other factor.

Per capita income of the city : It has a positive coefficient (+4) because the more is the per capita income of the city, the more desirable the city is for a buyer to reside and therefore the houses in that city are more desirable.

Lot size square feet : It has a positive coefficient (+3) because the larger the size of the house, the more is its desirability keeping the other factors constant.

Land tax : It has a negative coefficient (-2) because the more is the land tax that the buyer has to pay, the less desirable the house is.

Population of the region : It has a positive coefficient because if the population of a place is more, it means that the place is more desirable than a place where lesser number of people chose to live.

QUESTION2

We are using Euclidean Distance metric as our paiswise distance function. The formula used to calculate it is as follows:

$$dist((x, y), (a, b)) = \sqrt{(x - a)^2 + (y - b)^2}$$

The Euclidean Distance metric is ideal for this dataset as there were no categorical data and the remaining ones had been normalized using min-max normalization. Factors or weights for each distance attribute computation was not used (like in Q1) as the negative values cannot be present while computing the square root. This gives us a reasonable estimate of the similarity index between any two houses.

Here, We are calculating the distance between every two attributes across every pair of parcel id out of our slice (10 random values for better visualization) and draw a heatmap of all the distances measured. Heat map helps to visualise the results.

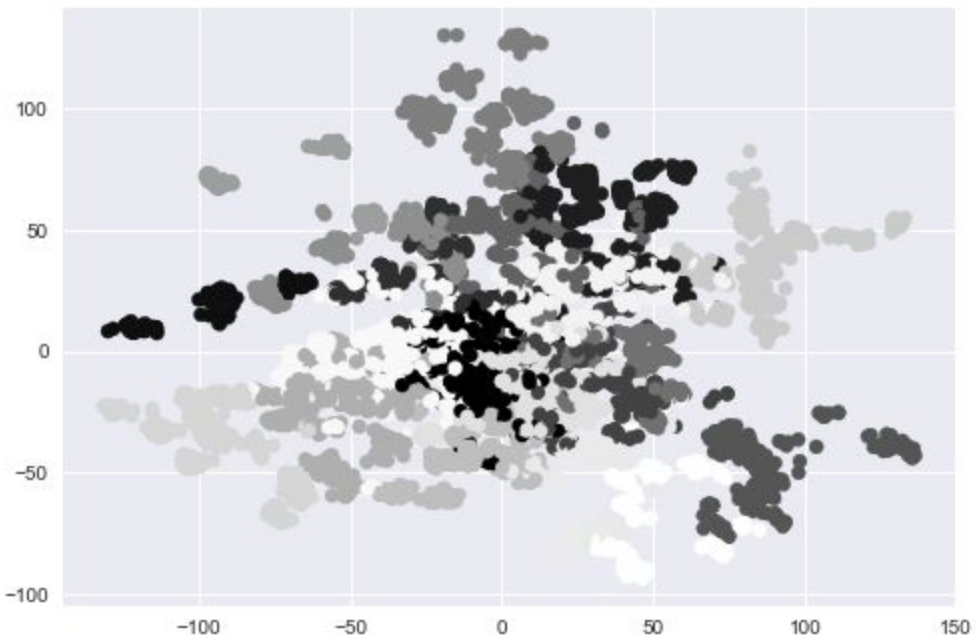
Over an average of 4-5 runs, we observed that the most of the distance values were in the range of 30 to 70 (the lighter spots) and few over 100 (the darker spots). This indicates the similarity of houses.

QUESTION3:

Clustering algorithm used - K Means
Distance metric used - Euclidean Distance

K-means is an ideal clustering algorithm in this case as dataset has been cleansed of categorical data and the remaining attributes are normalized using min-max normalization to the range of -100 to +100. The distance between two points is measured using Euclidean distance, that gives the scalar value difference for the two points/vectors.

The K-means was run on 20000 random sampled data of the 2.9M row dataset.



We can observe 7 out of 10 clusters are evenly distributed across the data while the remaining 3 seems to be at the mix of everything. This indicates that the number of clusters are slightly more than necessary. A reduction in the number of clusters will result in a single dense cluster at the center.

We picked random points from the sampled data belonging to the same cluster and checked its values in the original dataset and observed that they were similar in city id, bedroomcnt, bathroomcnt, price and location (latitude and longitude).

Repeating the above experiment for random points across different clusters had different ranges of values for the attributes mentioned above, though it had similar lotsizesquarefeet, population.

Another approach that wasn't experimented but analysed was using the desirability metric as another attribute for the clustering. Clustering houses based on desirability would be an interesting idea to help target customers.

QUESTION4:

It has been already been covered as part of data cleaning. Please refer data cleaning section above.

QUESTION5:

The model that gave us the best results is XGBRegressor

The basic idea behind this model is :

1. Fit a model to the data:
   $F_1(x) = y$

2. Fit a model to the residuals:
   $h_1(x) = y - F_1(x)$

3. Create a new model:
   $F_2(x) = F_1(x) + h_1(x)$

The working of this model is based on the idea of inserting more models that correct the errors of the previous model.

Specifically,

$$F(x) = F_1(x) \mapsto F_2(x) = F_1(x) + h_1(x) \ldots \mapsto F_M(x) = F_{M-1}(x) + h_{M-1}(x)$$

where $F_1(x)$ is an initial model fit to y

Since we initialize the procedure by fitting $F_1(x)$ where we can interpret h_m as a regression tree.

The performance of XGBRegressor:

We developed two models based on the data supplied to us-

1) Simple Linear Regressor

   Kaggle score : 0.0652527

2) XGB Regressor

Kaggle score : 0.0648128

Based on the results above we can see that XGB Regressor produced better results for us.

We came across many interesting observations while cleaning our data, when we have described above in the data cleaning section of the report.

QUESTION6:

The mean absolute value for our actual data set is 0.07132, which falls on the rank 102 in the dataframe sorted in the order of mean absolute errors for 1 to 500 times shuffled data sets. rank = 102 p value = 102/500 (0.204)

The number of permutations that produced equal or better results than the real data set = 101 fraction = 101/500