HOMEWORK - 3
ZILLOW CHALLENGE

KAGGLE GROUP NAME – DRACARYSS

Deepak Gupta (111485745)

Jatin Garg (111462753)

Vipul Gandhi (111483224)

Question 1:

Build a scoring function to rank houses by desirability

Cleaning Phase

- We removed NaN values form our dataset first. Latitude and Longitude NaN values do not serve any purpose and are substituted by 0.
- As per Zillow dictionary, the feature ‘roomcnt’ stands for the total number of rooms in a house. For values in the dataset, where ‘roomcnt’ value was not available or was 0, we substituted ‘roomcnt’ value by the total of bedroom count and bathroom count. And for other households, where ‘roomcnt’ value was less than sum of bed and bath, we changed those room count values too by the sum of bedroom count and bathroom count.
- As finished area of a house is an important feature, NaN values will effect model scoring. For this, we calculated average area per room for all households (other than NaN) by dividing ‘calculatedfinishedsquarefeet' with 'roomcnt'. And for houses where ‘calculatedfinishedsquarefeet' value was not available, we substituted it by (average area per room * room count)
- Columns with majority NaN values are dropped and other columns with NaN values are substituted by 0 or mean or other simple mathematical techniques.
- Categorical columns of our interest, namely 'buildingqualitytypeid' , 'heatingorsystemtypeid' and 'propertylandusetypeid' are distributed into individual columns with binary values. For a category where there are a lot more different subtypes (like 'propertylandusetypeid' has 24 subtypes) we have only used the top 4 subtypes because other subtypes have very less count for houses and moreover adding many additional columns to our dataset will add on to unnecessary extension in dataset width. Like column ‘propertylandusetypeid’ has been categorized to 4 different columns 'propertylandusetypeid_261', ‘propertylandusetypeid_266', 'propertylandusetypeid_246' and 'propertylandusetypeid_269’

External Data Merging

We have used feature ‘regionIdCity’ for a house to fetch city name from Zillow API. Then the city names are ranked by their desirability score, which is scrapped from Niche (https://niche.com). We have talked about external data merging in more detail in Question 4 (Integration of external Data).

Selecting Features

Selected Features for scoring function should positively co-relate with the value of a property. Based on this, the list of features selected for the scoring function is:

1. 'poolcnt'
2. 'buildingqualitytypeid_7','buildingqualitytypeid_4','buildingqualitytypeid_1', buildingqualitytypeid_10'

3. 'heatingorsystemtypeid_2','heatingorsystemtypeid_7','heatingorsystemtypeid_6', 'heatingorsystemtypeid_24'
4. 'roomcnt_zscore'
5. 'calculatedfinishedsquarefeet_zscore'
6. 'garagetotalsqft_zscore'
7. 'taxvaluedollarcnt_zscore'
8. 'CityRank_zscore'

Deciding weights

Zscores for all columns are calculated first because a house with high finished area value (in 100 thousands) will diminish the effect of weights contributed by our other features. So, it is important to normalize the selected features first.

We gave 35% weight to the city rank we calculated by the external dataset because we feel location is of utmost importance when deciding the value or desirability of a property (reasons for which we have provided in Section 4).

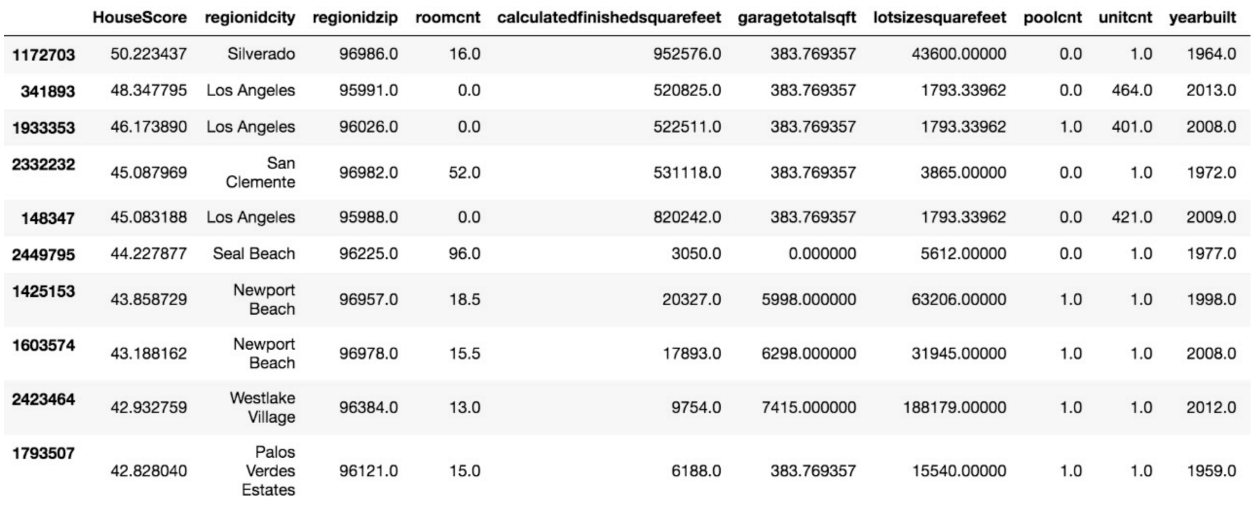
The finished area and garage area hold 20% and 10% weight respectively as both these features contribute significantly to value of a house. The weight matrix we mutually decided on is as below:

1. 'poolcnt' **– 2.5%**
2. 'buildingqualitytypeid_7','buildingqualitytypeid_4','buildingqualitytypeid_1', buildingqualitytypeid_10' **– (1.5%, 3%, 4.5%, 1%)**
3. 'heatingorsystemtypeid_2','heatingorsystemtypeid_7','heatingorsystemtypeid_6', 'heatingorsystemtypeid_24' **– (1%, 0.5%, 0.8%, 0.2%)**
4. 'roomcnt_zscore' **– 10%**
5. 'calculatedfinishedsquarefeet_zscore' **-20%**
6. 'garagetotalsqft_zscore' **-10%**
7. 'taxvaluedollarcnt_zscore' **-10%**
8. 'CityRank_zscore' **– 35%**

Overall scoring function

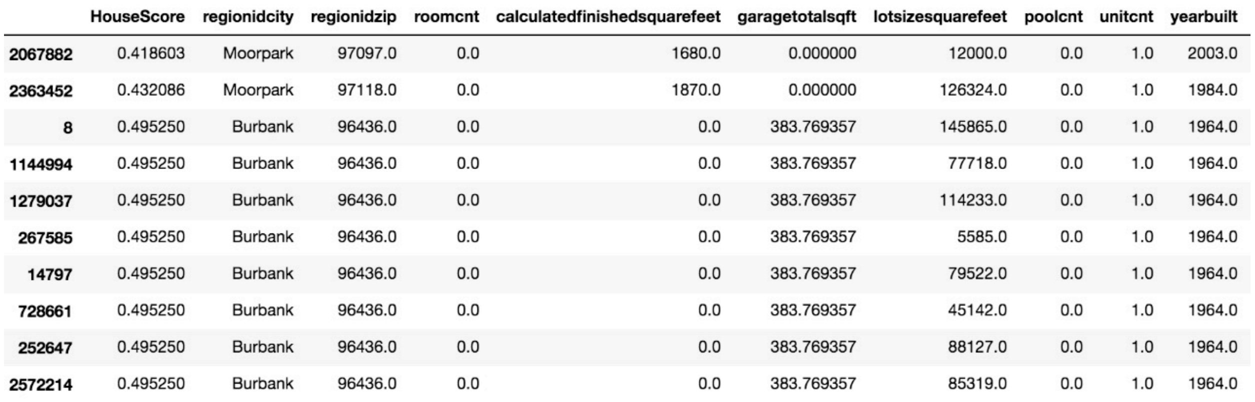
We built a new column named 'House Score' which gives us the cumulative score of all the features of a house multiplied by their column weight. Row-wise computation for House Score calculation on each row (house) took a very long time. So, we applied the column-wise multiplication and then assigned sum of the individual column scores to the new column. Pandas function applied is: dfFilter['HouseScore'] = dfScore[retainList].sum(axis=1)

Top desirable houses (Rank 1 to 10)



	HouseScore	regionidcity	regionidzip	roomcnt	calculatedfinishedsquarefeet	garagetotalsqft	lotsizesquarefeet	poolcnt	unitcnt	yearbuilt
1172703	50.223437	Silverado	96986.0	16.0	952576.0	383.769357	43600.00000	0.0	1.0	1964.0
341893	48.347795	Los Angeles	95991.0	0.0	520825.0	383.769357	1793.33962	0.0	464.0	2013.0
1933353	46.173890	Los Angeles	96026.0	0.0	522511.0	383.769357	1793.33962	1.0	401.0	2008.0
2332232	45.087969	San Clemente	96982.0	52.0	531118.0	383.769357	3865.00000	0.0	1.0	1972.0
148347	45.083188	Los Angeles	95988.0	0.0	820242.0	383.769357	1793.33962	0.0	421.0	2009.0
2449795	44.227877	Seal Beach	96225.0	96.0	3050.0	0.000000	5612.00000	0.0	1.0	1977.0
1425153	43.858729	Newport Beach	96957.0	18.5	20327.0	5998.000000	63206.00000	1.0	1.0	1998.0
1603574	43.188162	Newport Beach	96978.0	15.5	17893.0	6298.000000	31945.00000	1.0	1.0	2008.0
2423464	42.932759	Westlake Village	96384.0	13.0	9754.0	7415.000000	188179.00000	1.0	1.0	2012.0
1793507	42.828040	Palos Verdes Estates	96121.0	15.0	6188.0	383.769357	15540.00000	1.0	1.0	1959.0

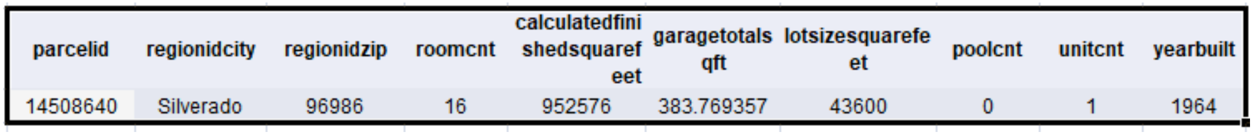
Least desirable houses (Rank -1 to -10)



	HouseScore	regionidcity	regionidzip	roomcnt	calculatedfinishedsquarefeet	garagetotalsqft	lotsizesquarefeet	poolcnt	unitcnt	yearbuilt
2067882	0.418603	Moorpark	97097.0	0.0	1680.0	0.000000	12000.0	0.0	1.0	2003.0
2363452	0.432086	Moorpark	97118.0	0.0	1870.0	0.000000	126324.0	0.0	1.0	1984.0
8	0.495250	Burbank	96436.0	0.0	0.0	383.769357	145865.0	0.0	1.0	1964.0
1144994	0.495250	Burbank	96436.0	0.0	0.0	383.769357	77718.0	0.0	1.0	1964.0
1279037	0.495250	Burbank	96436.0	0.0	0.0	383.769357	114233.0	0.0	1.0	1964.0
267585	0.495250	Burbank	96436.0	0.0	0.0	383.769357	5585.0	0.0	1.0	1964.0
14797	0.495250	Burbank	96436.0	0.0	0.0	383.769357	79522.0	0.0	1.0	1964.0
728661	0.495250	Burbank	96436.0	0.0	0.0	383.769357	45142.0	0.0	1.0	1964.0
252647	0.495250	Burbank	96436.0	0.0	0.0	383.769357	88127.0	0.0	1.0	1964.0
2572214	0.495250	Burbank	96436.0	0.0	0.0	383.769357	85319.0	0.0	1.0	1964.0

Evaluation of how well our scoring function worked

As per the results given by our scoring function, the top ranked property in the desirable list is below:



parcelid	regionidcity	regionidzip	roomcnt	calculatedfinishedsquarefeet	garagetotalsqft	lotsizesquarefeet	poolcnt	unitcnt	yearbuilt
14508640	Silverado	96986	16	952576	383.769357	43600	0	1	1964

We assigned 10% weight to the finished area in our scoring function for house desirability. Above is the house with best score as per our scoring function. A high finished area value gives a high component for finished area score. Thus, it increases the overall cost/desirability for this house.

Different scoring functions will output different properties at top ranks. Our scoring function gives most priority to the city in which the property is located, which we feel is an important parameter to base our scoring function on. The parameter cityRank covers almost all aspects of a desirable location like nearby schools, crime rates, job opportunities in the area. However, properties with big finished sizes in moderately ranked cities (as per our cityRank list) can turn out to be more costly and desirable for commercial works (like, offices or warehouses) than comparatively smaller area properties in costly cities (high ranked cities). We have considered these conditions and we feel our scoring function predicts house desirability as would prices/desirability for houses work in real estate market.

Question 2:

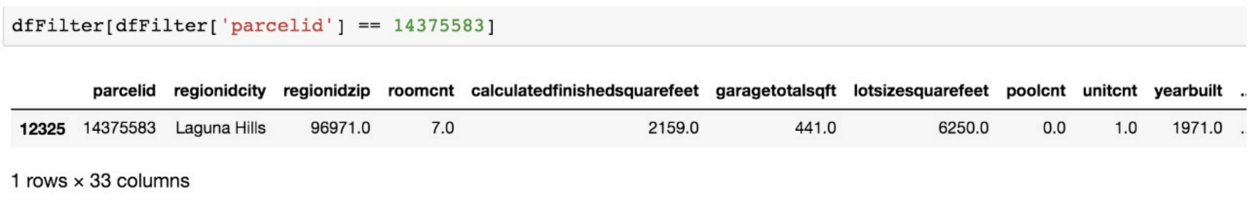
<u>Define a house “pairwise distance function”, which measures the similarity of two properties.</u>

- We have used manhattan distance for the pairwise distance function in between the houses. For two houses, Manhattan distance is calculated by taking the absolute difference between the z-score values for every corresponding feature. We have filtered out parameters from the original dataframe to a definite set of features that contribute towards distance calculation. Features like ‘assessmentyear’, ‘storytypeid’, ‘fireplaceflag’, ‘fullbathcnt’ etc are dropped because it is not logical to measure distances based on these parameters. For distance calculation, we have taken into consideration both geographical variables like ‘CityRank’ and property specific variables like ‘roomcnt’, 'calculatedfinishedsquarefeet’, 'garagetotalsqft’ etc.
- We started by computing distances between 2 houses based on the real values of features and not on the zscore values of the features. This observation didn’t pass our sniff test. This gave irregular distances as a high difference in finished squarefeet area often diminished the effect of all other parameters, which though were very similar and thus would have resulted in a very low distance, had we reduced the effect of finished squarefeet area. In order to normalize this effect, we substituted real feature values by the z_score values which give better overall results.

Evaluation of how well pairwise distance function worked

Houses with low distance (Very Similar houses):

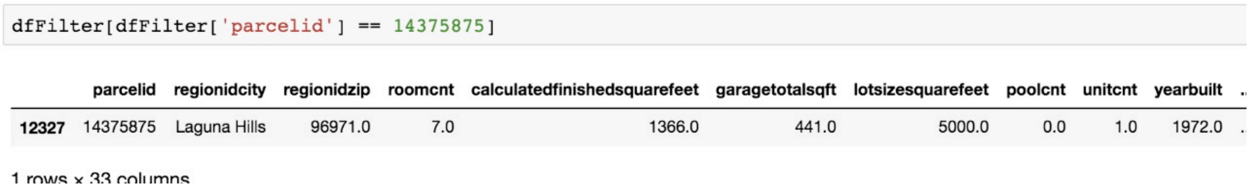
Below pair of two houses is amongst some of the top pairs of houses with very less distance. All the features of both the houses are very similar. They lie in the same city, i.e. Laguna Hills. They have same number of rooms. Both houses were built around the same time with a difference of almost an year between them. The distance between two houses is 0.082.



```
dfFilter[dfFilter['parcelid'] == 14375583]
```

	parcelid	regionidcity	regionidzip	roomcnt	calculatedfinishedsquarefeet	garagetotalsqft	lotsizesquarefeet	poolcnt	unitcnt	yearbuilt	..
12325	14375583	Laguna Hills	96971.0	7.0	2159.0	441.0	6250.0	0.0	1.0	1971.0	.

1 rows × 33 columns



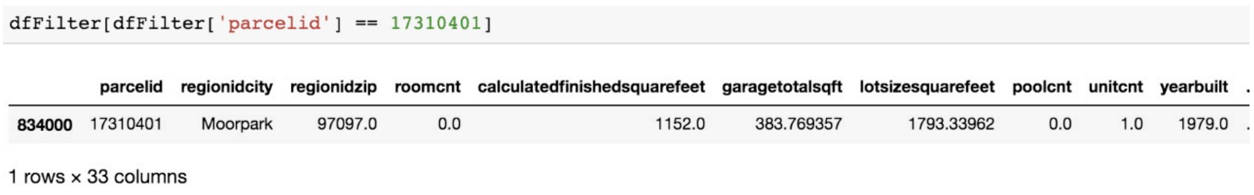
```
dfFilter[dfFilter['parcelid'] == 14375875]
```

	parcelid	regionidcity	regionidzip	roomcnt	calculatedfinishedsquarefeet	garagetotalsqft	lotsizesquarefeet	poolcnt	unitcnt	yearbuilt	..
12327	14375875	Laguna Hills	96971.0	7.0	1366.0	441.0	5000.0	0.0	1.0	1972.0	.

1 rows × 33 columns

Houses with high distance (Dissimilar houses): Reason

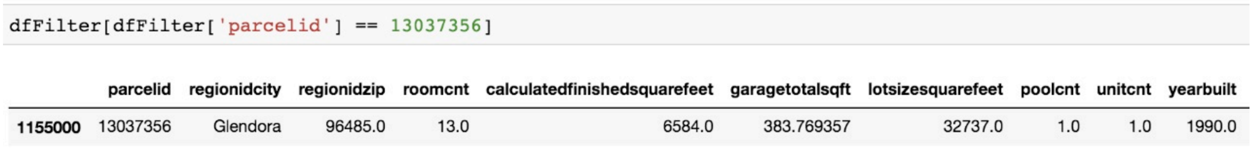
The two houses shown below have distance of 7.18. Almost all features are different. So, the absolute of the difference of zscore of all features of these 2 houses is very high.



```
dfFilter[dfFilter['parcelid'] == 17310401]
```

	parcelid	regionidcity	regionidzip	roomcnt	calculatedfinishedsquarefeet	garagetotalsqft	lotsizesquarefeet	poolcnt	unitcnt	yearbuilt
834000	17310401	Moorpark	97097.0	0.0	1152.0	383.769357	1793.33962	0.0	1.0	1979.0

1 rows × 33 columns



```
dfFilter[dfFilter['parcelid'] == 13037356]
```

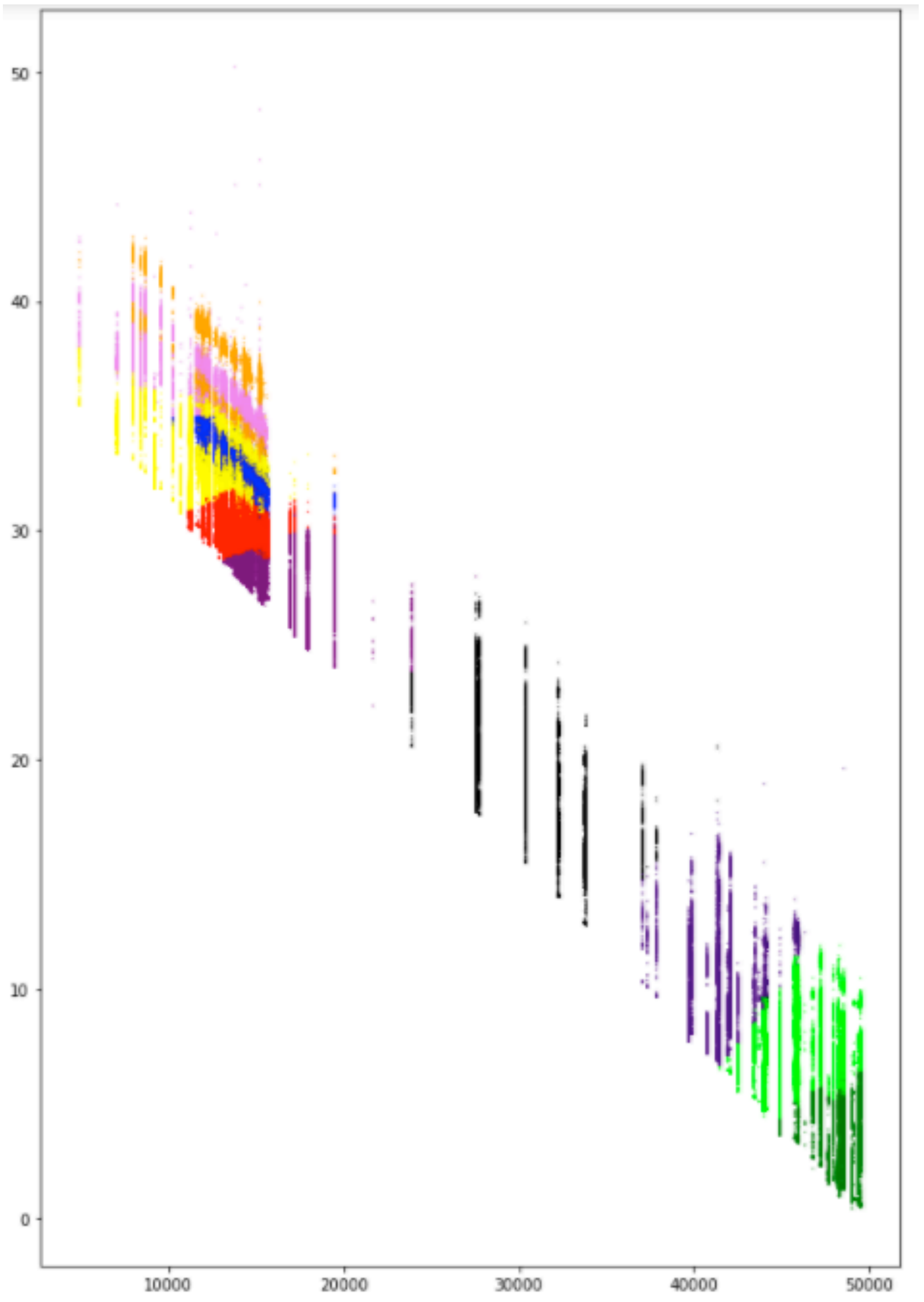
	parcelid	regionidcity	regionidzip	roomcnt	calculatedfinishedsquarefeet	garagetotalsqft	lotsizesquarefeet	poolcnt	unitcnt	yearbuilt
1155000	13037356	Glendora	96485.0	13.0	6584.0	383.769357	32737.0	1.0	1.0	1990.0

We think our pairwise distance function did a pretty decent job as we handled the effect of high outliers for every feature by normalizing them first and then calculating distance function on it. We evaluated the distance function on around 100 random pair of houses in the dataset provided. The houses are more similar as distance is near to 0 while distance increases as houses become more dissimilar which is expected property of how a distance function should perform.

Question 3:

Analysis of what your clusters seem to be capturing

We have applied k-means clustering on the filtered dataset we used for our distance function. We have clustered our data in 10 clusters. **The graph is plotted against the score value on y-axis (0-50.22) given by our scoring function and city rank on x-axis (1-50,000)**. The scoring function has a 30% weight to the city rank. City rank in itself is from external dataset and is derived on the basis of nearby schools, crime rates, job opportunities in the area.



The plot shows whole of the 3 million houses in the dataset. For lower city ranks (i.e.: better city), the value given by the scoring function is high. So, as the city rank improves, the house score is also better. So, the houses with better value of HouseScore are clustered together. So, one can find top ranked houses in the first cluster while bottom ranked houses in the last cluster.

How well they work?

The clusters are a bit uneven and wide, which we expected from our dataset because houses clustered by k-means clustering algorithm are dependent on many features (like roomcnt, unitcnt, totaldollartaxvaluecnt, etc) in addition to the house score. So, it is highly possible that some houses with the similar score value but different city ranks lie in the same cluster. This is the reason some clusters extend horizontally.

Question 4:

Integration of external data

We calculated City name from the Zillow API by passing regionidcity from our dataset. This gave us a list of all the unique cities in our dataset. Zillow limits 1000 queries per day so, it was not wise to query Zillow API for every unique zipcode id. We have talked more about this in the 'Interesting experiences' section in Question 5.

Now, we had to rank the list of cities received by zillow api. We ranked cities by forming a city rank list by building a scraper and scraping data from https://www.niche.com/. Niche provides an assessment of best places to live in America on the basis of different factors. Niche takes into account the quality of local schools, crime rates, housing trends, employment statistics, access to various amenities and many other parameters in an attempt to measure the overall quality of an area. Niche gathers data from public sources like US Census, Department of Education, FBI, ACS, NCDC and also from personal experiences, surveys and reviews from real people.[1]

Though we had built lists for individual data like crime rate (from website of FBI) and number of district schools in an area (from website of LAUSD). But after browsing on the internet and google for some time, we landed on niche.com. Our dataset was also fine but Niche has a team of data scientists that evaluates countless data sets to produce rankings, report cards, and in-depth profiles. [1] We compared our dataset for crime rate and number of schools with the rankings available on Niche. The rankings often matched and we were satisfied with the niche rankings. So, trusting niche we decided to proceed with its dataset as it saved us a lot of time and provided a comprehensive dataset which is a collection of 7-8 different features which would have taken up a lot of our time to add to our dataset manually. Adding niche rankings surely makes our dataset rich.

Analysis of how it helps with prediction

City rank is a major feature in the scoring function we have built. The scores generated by the scoring function are used for the final model prediction. The chief idea behind adding city ranks is that geographical aspects of a house predominantly decide the value or the desirability of the house. The external data is added to the current dataset in an attempt to train the model on more real features. Real features mean that they contribute to our model the way real world factors like job and crime rate affect a house's desirability.

Question 5:

How well prediction model works

- After cleaning process and finalizing the feature list, we trained our model on selected features.
- The chosen features were: roomcnt, calculatedfinishedsquarefeet_zscore, garagetotalsqft_zscore, lotsizesquarefeet_zscore, unitcnt_zscore, yearbuilt_zscore, taxvaluedollarcnt_zscore, CityRank_zscore.
- The R-squared value is 0.008 without intercept, which means that only 0.8% of the log errors can be explained by our predictor variables.
- As the dependent variable logerror is a very small value for each result, coefficient values for each predictor variable (provided by statsmodels ols api) are also very small. This means that even a major increase in the value of our predictor variable will only cause a small change to logerror.
- Though for every predictor variable the standard error is very low. But to confidently say this, metric evaluation of the variables needs to be performed.
- Even after taking the z_score of the predictor variables, the results remain the same. So, normalization does not affect our model.
- The r-squared value is low because even small improvements in the zillow model are hard to come by as it is already a very good model
- After training and testing out several models, we decided to go with OLS linear regression model as it was producing better results as compared to other models. For example, decision tree regressor was producing MSE of 0.06645 while OLS linear model regressor was at 0.0284292.
- I believe this model performs pretty well as this model is taking several property specific features as well as geographical information into account.
- The score that we got from Kaggle is 0.0650.

Interesting experiences

Failed Efforts while external data merging:

In an effort to get location for every house, we tried many things out which didn't work out the way we expected them to. Getting location by passing lat and long to geocode api would not have worked anyway because of the max limit of the queries(which is very less) you can hit geocode api with in a time limit.

Then, we tried to fetch location by using zipcode in the geocode api because as we figured out that there were only 405 unique zipcodes in the dataset. At this time, we didn't know that zip codes in our dataset are not real zipcodes but are zillow substituted zipcodes. So, passing zipcodes from our dataset to the geocode api returned malicious locations with cities even in Brazil and Germany.

Ultimately we had to settle with city names (and not zipcode) by calling Zillow api with regionidcity parameter. So, we had city names for a particular household.

Question 6:

P-test on our prediction model

After cleaning process and finalizing the feature list, we trained our model on selected features.

After training and testing out several models, we decided to go with OLS linear regression model as it was performing better than others.

Now we want to evaluate the significance of our model by performing permutation test.

For this purpose we decided to use NumPy np.random.permutation method to permute the log-errors of test data. For the permuted column of log-error, each time we predicted log error and calculated mean square error.

We did 500 permutations and the average mean squared error was : 0.047

Mean squared error on original log-error: 0.023

Since the mean squared error was significantly higher than the original model MSE, we believe that our original model was significant and was doing real prediction.