House hunting — the data scientist way

Atma Mani Follow
Oct 27, 2018 · 13 min read

At some point in time, each of us would have went through the process of either renting or buying a house. Whether we realize or not, a lot of factors we consider important are heavily influenced by location. In this article, we apply the data wrangling capabilities of scientific Python ecosystem and geospatial data visualization & analysis capabilities of the **ArcGIS platform** to build a model that will help shortlist good properties (houses). You may ask why do this as there are a number of real estate websites that promise something similar. I hope by the end of this article, you will be able to answer that yourself.

Note: *I recently presented this as a keynote speech at a GeoDev meetup hosted by Esri in Portland.* ***The Python notebooks used for*** *this blog can be found* *here:* https://github.com/Esri/arcgis-python-api/tree/master/talks/GeoDevPDX2018 *and the slides here:* *https://slides.com/atma_mani/deck-1*

Data collection and cleaning

Housing data for the city of Portland was collected from a popular real estate website. It came in a few CSV files of different sizes. Data was read using Pandas as `DataFrame` objects. These `DataFrame`s form the bedrock of this study upon which both spatial and attribute analysis are performed. The CSVs were merged to obtain an initial list of about `4200` properties listed for sale.

Missing value imputation

An initial and critical step in any data analysis and machine learning project is wrangling and cleaning of the data. The data collected, in this case, suffers from duplicates, illegal characters in column names, and outliers. Pandas makes it extremely easy to sanitize tabular data. Different strategies were used to impute for missing values. Centrality measures such as *mean* and *median* were used to impute for missing values in 'lot size', 'price per sqft.' and 'sq ft' columns whereas, frequency measures like *mode* were used for columns such as 'ZIP'. Rows that had missing values in critical columns such as 'beds', 'baths', 'price', 'year built', 'latitude' and 'longitude' were dropped as there was no reliable way of salvaging them. After removing such records, there were `3652` properties available for analysis.

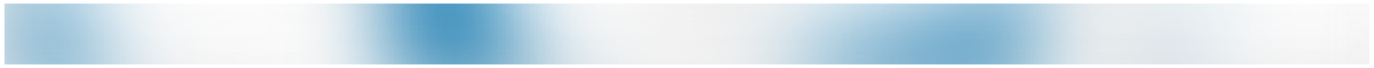
Removing outliers

Outliers in real estate data could be due to a number of reasons. For example, erroneous data formats, bad default values, typographical errors during data entry etc. can lead to outliers. Plotting the distribution of numeric columns can give a sense of outliers in the data set.

```
# explore distribution of numeric columns
ax_list = prop_df.hist(bins=25, layout=(4,4), figsize=(15,15))
```

Histograms of numeric columns show the presence of outliers

From the first two histograms, it appears as though all the houses have the same number of beds and baths, which is simply not true. This is a sign that a small number of high values (outliers) are skewing the distribution. A few different approaches exist to filter outliers. A popular technique is to use a **6 sigma** filter which removes values that are greater than `3` standard deviations from *mean*. This filter assumes that the data follows a normal distribution and uses *mean* as the measure of centrality. However, when data suffers heavily from outliers, as in this case, *mean* can get distorted. An **Inter Quartile Range (IQR)** filter which uses *median*, a more robust measure of centrality, can filter out outliers that are at a set distance from median in a more reliable fashion. After removing outliers using the IQR filter, the distribution of numeric columns looks much healthier.


Histogram of numeric columns after missing value imputation and removal of outliers

Exploratory data analysis

As seen so far, Pandas provides an efficient API to explore the statistical distribution of the numeric columns. To explore the spatial distribution of this data set, ArcGIS API for Python is used.

```
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

from arcgis.gis import GIS
from arcgis.features import GeoAccessor, GeoSeriesAccessor
```

The `GeoAccessor` and `GeoSeriesAccessor` classes add **spatial capabilities** to **Pandas DataFrame** objects. Any regular DataFrame object with location columns can be transformed into a **Spatially Enabled DataFrame** using these classes.

```
>>> prop_sdf = pd.DataFrame.spatial.from_xy(prop_df,
'LONGITUDE','LATITUDE')
```

Similar to plotting a statistical chart out of a `DataFrame` object, a spatial plot on an interactive map widget can be plotted out of a Spatially Enabled DataFrame.

MLS listings of properties for sale in Portland market

Renderers such as heat-maps can be applied to quickly visualize the density of the listings.

Plotting the Spatially Enabled DataFrame with a heatmap renderer shows the presence of hotspots near downtown and Clover Hill neighborhoods

The ArcGIS API for Python comes with an assortment of sophisticated renderers that help visualize the spatial variation in columns such as ‘property price’, ‘age’, ‘square footage’, ‘HoA’ etc.

Spatial and statistical distributions of property price, square footage and monthly HoA

Combining maps with statistical plots as shown above yield deeper insights into the real estate market of interest. For instance, maps such as these help investigate general assumptions that properties are higher priced, smaller in size, older in age around the downtown area and change progressively as you move toward the periphery.

Running an initial shortlist

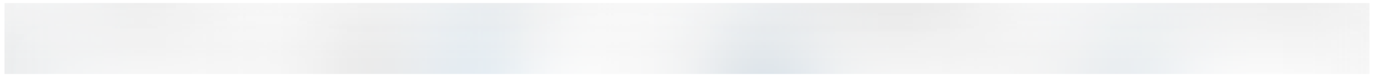
The following rules based on the intrinsic features of the houses were used to build a shortlist.

```
>>> filtered_df = prop_sdf[(prop_df['BEDS']>=2) &
                           (prop_df['BATHS']>1)&
                           (prop_df['HOA PER MONTH']<=200) &
                           (prop_df['YEAR BUILT']>=2000) &
                           (prop_df['SQUARE FEET'] > 2000) &
                           (prop_df['PRICE']<=700000)]
>>> filtered_df.shape
(331, 23)
```

This narrows down the list of eligible properties to `331`, from an original `3624`. When plotted on a map, these shortlisted properties are spread across the city.

Shortlisted properties are spread throughout the Portland market

From the histograms below, most houses in the shortlist have `4` beds and the majority of them are skewed toward the upper end of the price spectrum.



Histograms of shortlisted properties

Feature engineering — quantifying access to facilities

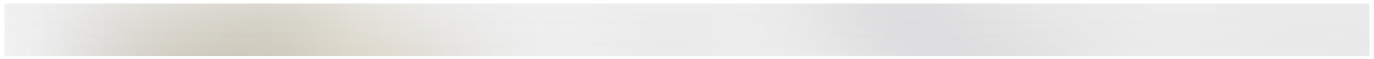
When buying a house, buyers look for proximity to facilities such as groceries, pharmacies, urgent care, parks etc. These comprise a house's location properties. The `geocoding` module of the ArcGIS API for Python can be used to search for such facilities within a specified distance around a house.

```
from arcgis.geocoding import geocode

# search for restaurants in neighborhood
restaurants = geocode('restaurant', search_extent=prop_buffer.extent,
max_locations=200)

# search for hospitals
hospitals = geocode('hospital', search_extent=prop_buffer.extent,
max_locations=50)
```

The map below displays facilities such as groceries, restaurants, hospitals, coffee shops, bars, gas stations, shops & service, travel & transport, parks and educational institutions that fall within a `5` mile `buffer` around a house.



This map shows some common facilities home buyers generally look for around a house. A house chosen at random is symbolized with a red star and the 5 mile buffer is symbolized with the black circle. The blue line represents the fastest route from the house to a designated destination (Esri Portland R&D office in this case).

Another important aspect that buyers consider is the time it takes to commute to work / school. The `network` module of the ArcGIS API for Python provides tools to compute driving directions and duration based on historic traffic information. For instance, the snippet below calculates the directions between the chosen house and Esri Portland R&D office and the time it would take on a typical Monday morning at 8 AM.

```
route_result = route_service.solve(stops, return_routes=True,
                return_stops=True, return_directions=True,
                impedance_attribute_name='TravelTime',
                start_time=644511600000)
...
print("route length: {} miles, route duration
      {}".format(round(route_length,3)))


>>>
route length: 10.273 miles, route duration: 27m, 48.39s
```

When routing, you can add multiple stops such as to daycare, gym or other places you visit as part of your commute and take those into account as well. This information can be turned into a Pandas DataFrame and visualized as a table or a bar chart. Thus, houses can be compared against one another based on access to neighborhood facilities.

Left: List of neighborhood facilities for a house chosen at random, as a table. Right: Number of such facilities under each category.

Feature engineering

The above steps were run on a batch mode against each of the `331` shortlisted properties. Different neighborhood facilities were added as new columns to the data set and the count of the number of facilities a property has access to (within a specified distance) wasadded as the column value. The idea is if there are a lot of facilities of the same kind near a property, all of them compete for the same market. This competition keeps a check on the prices and also improves the quality of service. Such houses are more attractive than the rest.

The histograms below show the distribution of facility counts around the `331` shortlisted properties.

Histograms of facilities around the shortlisted properties

Based on the histogram, there is a healthy distribution of coffee shops, gas stations, hospitals and general purpose shops around a large number of houses. Further, the Portland market appears to perform really well when it comes to commute duration & commute length (when computed to Esri downtown office), number of schools, grocery shops, parks and restaurants as most houses have a large number of options to choose from.

Thus, features such as the number of beds, baths, square footage, price, age, etc. form the **intrinsic features** of a property. Through a series of **spatial enrichment** steps as seen above, location based attributes such as the number of restaurants, grocery stores, hospitals, etc. a property has access to were computed. These become a property's **spatial features** and were added to the original data set as part of the **feature engineering** process.

Scoring properties

Evaluating houses is a deeply personal process. Different buyers look for different characteristics in a house and not all aspects are considered equal. Thus, it is possible to assign different weights for each of the features and arrive at a weighted sum (a score) for each house. The higher the score, the more desirable a house is.

Below is a scoring function that reflects the relative importance of each feature in a house. Attributes that are considered desirable are weighed positively, while those undesirable are weighed negatively.

```python
def set_scores(row):
    score = ((row['PRICE']*-1.5) + # penalize by 1.5 times
             (row['BEDS']*1)+
             (row['BATHS']*1)+
             (row['SQUARE FEET']*1)+
             (row['LOT SIZE']*1)+
             (row['YEAR BUILT']*1)+
             (row['HOA PER MONTH']*-1)+  # penalize by 1 times
             (row['grocery_count']*1)+
             (row['restaurant_count']*1)+
             (row['hospitals_count']*1.5)+  # reward by 1.5 times
             (row['coffee_count']*1)+
             (row['bars_count']*1)+
             (row['shops_count']*1)+
             (row['travel_count']*1.5)+  # reward by 1.5 times
             (row['parks_count']*1)+
             (row['edu_count']*1)+
             (row['commute_length']*-1)+  # penalize by 1 times
             (row['commute_duration']*-2)  # penalize by 2 times
            )
    return score
```

Scaling your data

While a scoring function can be extremely handy to compare feature engineered, shortlisted houses when applied directly (without any scaling), it returns a set of scores that are heavily influenced by a small number of attributes whose values are numerically very large. For instance, attributes such as property price tend to be really large numbers (hundreds of thousands) compared to, say, number of beds (<10) and when used without scaling, it tends to dominate the scores beyond its allotted weight.

Effects of scaling: Property scores before and after scaling the numerical columns

The left side of the chart above shows scores computed without scaling. They appear extremely correlated with the property price variable. While property price is an important consideration for most buyers, it cannot be the only criteria that determines a property's rank.

To rectify this, all numerical columns were scaled to a uniform range of `0-1` using the `MinMaxScaler` function from `scikit-learn` library and a new set of scores were computed. The right side of the chart above shows the results of this scaling. The scores appear normally distributed and the scatter between property price and scores shows only a weak correlation. Thus, the scaling function has performed well and would allow the scoring algorithm to effectively consider all the other attributes of a house in addition to property price.

Ranking properties

Once the properties were scored, they were sorted in descending order of their scores and assigned a rank. The house with highest score gets rank `1` and so on. From here, the home buyer could pick the top `10` or `50` houses as a refined shortlist and perform field visits.

Top 50 houses. Lighter shades (yellow) represent lower prices.

Interestingly, the top `50` houses are spread across the city of Portland without any signs of a strong spatial clustering. Property prices on the other hand appear in clusters. The histograms below show that most houses in the top `50` list have `2` baths, `4` beds (although the shortlist criteria was just a minimum of `2` beds), priced on the upper end of the spectrum, under `2500` sq ft. and built fairly recently (2015). In terms of access to neighborhood facilities, most houses appear well connected with a large number of grocery stores, restaurants, gas stations, educational institutions and hospitals to choose from. Majority of the properties are within `10` miles from the said destination (Esri downtown office) and can be reached within `25` minutes of driving.

Histograms of intrinsic and spatial features of the top 50 houses

The scaling function ensured that there is no single feature that dominates a property's score beyond its allotted weight. However, there may be some features that tend to correlate with scores. To visualize this, the `pairplot()` function of `seaborn` library is used to produce a scatter plot of each variable against one another, as shown in the matrix below. The diagonals of the matrix represent histograms of the corresponding variable.

Scatter plots of rank vs spatial features of houses

In the scatter grid above, we notice randomness and stratification in the `rank` variable. The only two places where the scatters show sings of correlation is understandably between `commute_duration` and `commute_length`. In other words, as the commute distance decreases, the commute duration also decreases, meaning traffic conditions don't affect duration as much as distance does.

The second scatter grid below is between rank and various intrinsic features of the properties. The scatter between rank and property price is quite random, meaning it is possible to buy a house with a higher rank for lower than average price. The scatter between rank and square footage shows an interesting 'U' shape, meaning as property sizes increase, their rank gets better, but after a point, they get worse.

Scatter plot of rank vs intrinsic features of houses

Building a housing recommendation engine

So far, the data set was feature engineered with intrinsic and spatial attributes. Weights for different features were explicitly defined, using which properties were scored and ranked. In reality, buyers decision making process, although logical, is a little less calculated and a bit fuzzier. Buyers are likely to be content with certain shortcomings (for instance, fewer bedrooms than preferred or fewer shopping centers in the neighborhood) if they are highly impressed with some other characteristic (such as larger square footage to compensate with fewer bedrooms). Thus, we could get buyers to simply ‘favorite’ and ‘blacklist’ a set of houses and let a machine learning model infer their preferences.

Since it is difficult to collect such training data for a large number of properties, a mock data set was synthesized using the top `50` houses marked as favorites and the remaining `281` as blacklists. This was fed to a machine learning logistic regression model.

As this model learns from the training data, it attempts to assign weights to each of the predictor variables (intrinsic and spatial features) and can predict whether or not a house will be preferred by a buyer. Thus, as newer properties hit the market, this model can predict whether or not a buyer would favorite a new property and present only such relevant results. Below is the accuracy of such a model that was run on this data set.

```
>>> classification_report(y_test, test_predictions,
                          target_names=['blacklist','favorite'])


             precision    recall   f1-score

  blacklist      0.94       0.98       0.96
```

```
    favorite       0.88       0.71       0.79
     average       0.93       0.93       0.92
```

> `Precision` *refers to the model's ability to correctly identify whether a given property is favorite or not.* `Recall` *on the other hand refers to its ability to identify all favorites in the test set. The* `f1-score` *computes the harmonic mean of* `precision` *and* `recall` *to provide a combined score of the model's accuracy.*

The training data used in this case study is small by today's standards and is imbalanced because there are fewer properties that are favorites compared to blacklists (50 vs 281). Yet, the model performs appreciably well with high `f1` scores for eliminating properties that are likely to be blacklists.

The weights assigned by the regression model (code snippet on the right in the image below) show what the model learnt to be the relative importance of each feature based on the training data. When compared against the weights that were manually assigned (see section "Scoring properties"), the logistic regression model has penalized property price, commute length and duration only mildly. It has weighed certain features such as lot size, number of grocery stores, shops, parks and educational institutions negatively and the rest positively. Features such as hospital counts, coffee shops, bars and gas stations are weighed higher than what was assigned manually.

Conclusion

The type of recommendation engine built in this study is called '**content based filtering**' as it uses just the intrinsic and spatial features engineered for prediction. For this type of recommendation to work, we need a really large training set. In reality nobody can generate such a large set manually. In practice however, another type of recommendation called '**community based filtering**' is used. This type of recommendation engine uses the features engineered for the properties, combined with favorite / blacklist data to find similarity between a large number of *buyers*. It then pools the training set from similar buyers to create a really large training set and learns on that.

In this case study, the input data set was spatially enriched with information about access to different facilities. This can be extended further by engineering **socio-economic features** such as age, income, education level, marital status, population density and a host of other parameters using the `geoenrichment` module of the ArcGIS API for Python. Another aspect that could be incorporated is to use authoritative data shared by local governments under the **open data initiative**. For instance, the city of

Portland's open data site lists a host of useful spatial layers that can be used to further enrich this data set.

This study demonstrates how data science and machine learning can be employed to one aspect of the real estate industry. Buying a home is a personal process, however a lot of decisions are heavily influenced by location. As shown in this study, Python libraries such as Pandas can be used for visualization and statistical analysis, and libraries such as the ArcGIS API for Python for spatial analysis. The methods adopted in this study can be applied to any other real estate market to build a recommendation engine of your own.

*The **Python notebooks** used for this blog can be found here:* https://github.com/Esri/arcgis-python-api/tree/master/talks/GeoDevPDX2018 *and the **slides** here: https://slides.com/atma_mani/deck-1*

Data Science Python Spatial Analysis Machine Learning Real Estate

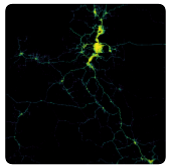
503 claps

WRITTEN BY

Atma Mani

Follow

Atma Mani is the lead product engineer for ArcGIS API for Python at Esri. He enjoys applying advanced analytics to solve spatial problems.



GeoAI

Follow

Geospatial Artificial Intelligence: thoughts about where AI and GIS intersect

See responses (11)

More From Medium

More from GeoAI

Parking Lot Vehicle Detection Using Deep Learning

David Yu in GeoAI
Aug 27, 2018 · 12 min read

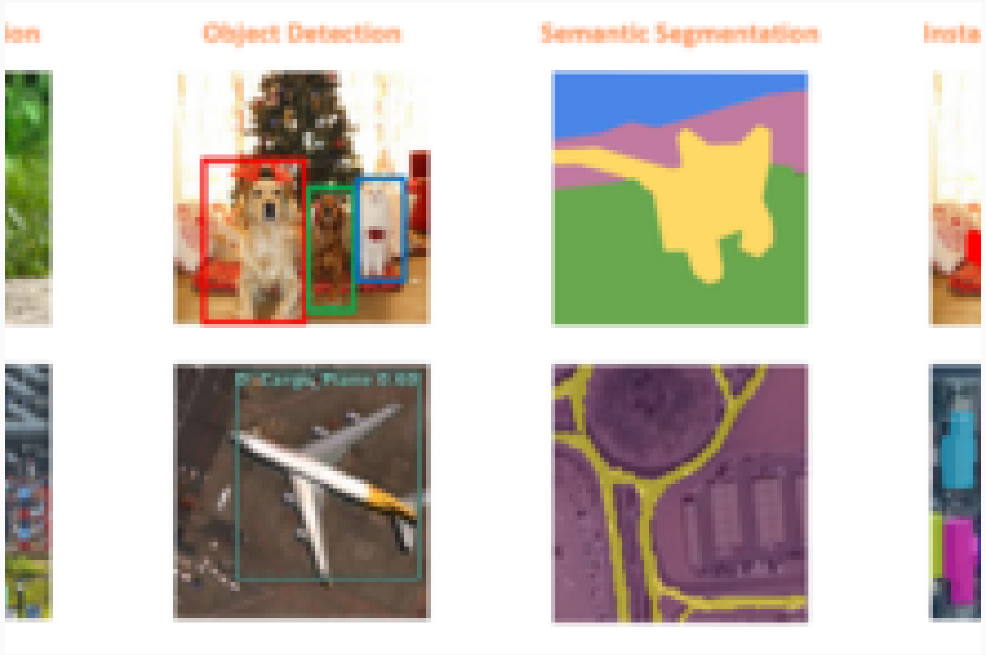
842

More from GeoAI

Integrating Deep Learning with GIS

Rohit Singh in GeoAI
Feb 23 · 8 min read

556



More from GeoAI

Using Forest-based Classification & Regression to Model and Estimate House Values

Alberto Nieto in GeoAI
Oct 27, 2018 · 7 min read

72