Jacob Dineen

IST 718 – Advanced Info Analytics

Lab 2

2/14/18

The following is a high-level representation of my best efforts to conduct analysis and forecast housing prices with data provided to us from Zillow, and with the inclusion of IRS and Census data fetched through external sources. I will attempt to walk through each step as it relates to questions in the assignment, making note that anything that is not touched upon is likely covered in my Jupyter notebook, zipped up alongside this word document.

Reading in the initial dataset from Zillow using the pd.read.csv function on the URL string, which is read in as a pd.dataframe, we first look at the shape of the data – 15338 rows and 268 columns. Initially, this looks daunting, but we remember from working in R that there are melt functions that transform column dimensionality. We clean some of the column names just for the ease of reference, storing the new names inplace of the old names.

The next step is dealing with NA/s or missing values. I had thought to take column means or medians, but thought this might disrupt the temporal essence of the data, thinking that taking replacing a null value sometime during the recession and replacing it with a mean that wasn’t indicative of the external dependent factors wouldn’t be appropriate, so I decided to remove all na’s- This encompassed 4700 observations. We then convert all necessary variables to factors, as they’ll likely be used in a stepwise OLS regression later.

**Develop time series plots for the following Arkansas metro areas:**

Developing plots is a relatively simple technique to shows distributions and trends of data over time. This task was a little more difficult because we had to slice the initial data to only include features = Metro and the desired date range, which was 1997 to 2013. I broke this out into two steps and merged the two dataframes together, but looking back realized that I should have melted my initial data frame before this step to be able to filter off observations rather than columns. Anyway, after melting the data and grouping by the mean of id\_var = ‘Metro’, the following code was used to join two functions together, producing individual dataframes for each group of data (I thought this may be easier in case I needed to reference back to this particular data).

#separate dfs and out in 'tab

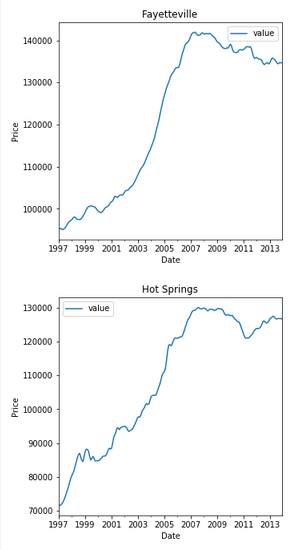
fay = pd.melt((df\_ark.filter(like = 'Fayette', axis=0)))

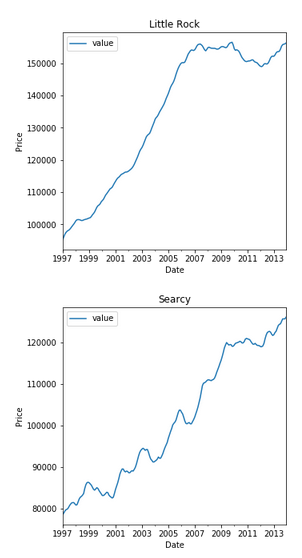
hotspr = pd.melt((df\_ark.filter(like = 'Hot', axis=0)))

littlerock = pd.melt((df\_ark.filter(like = 'Little Rock', axis=0)))

searcy = pd.melt((df\_ark.filter(like = 'Searcy', axis=0)))

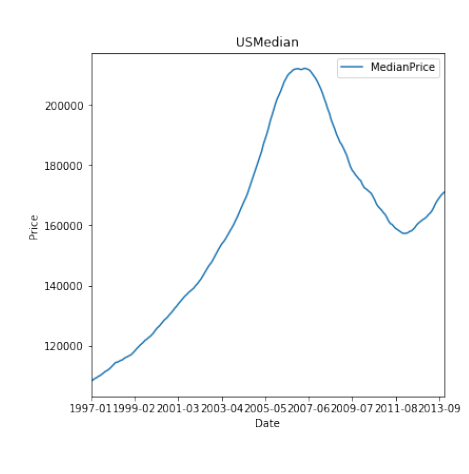
**The resulting plots look like this:**





These plots show us two things:

1. There was a very steep linear trend from 1997 through 2008 before flattening out and dropping, which is attributable to the Great Recession experienced during that year.
2. The trends seem to counteract, and even show resistance to the demise of the housing market during 2008, particularly when looking at them against the US median during the same time frame:

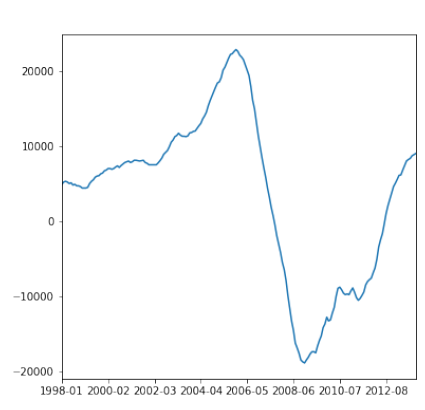


We could turn this into a project of its own, but for now we’ll digress and acknowledge that these four markets did not move in sync with the median US housing prices during the crisis of 2008 and the continued fall through 2011.

The next steps involve statistical analysis on the noise present within the data, and any correlation or clustering of movements we could directly visualize from charts before beginning to build a model.

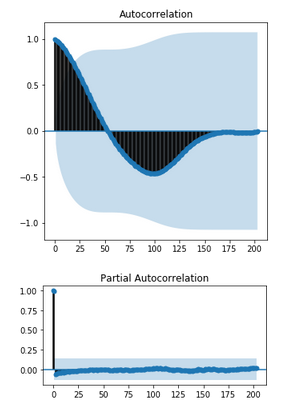
All code is available in my Jupyter notebook, to keep this compressed, and I do want to acknowledge that most of the code used to develop the ARIMA model was borrowed from the Professor’s asynch 4 notes concerning the topic.

I split my data into train and test sets, with the test set being an average of the three time periods referenced in the assignment handout. I then ran the training set through an Augmented Dickey Fuller test to look at the rendered chart, showing me that I was not dealing with stationary data, and would likely have an order of differencing prevalent in my model. I also looked at the autocorrelation and partial autocorrelation charts.



I then found the best ARIMA model for my dataset, finding the ideal order for each of the pdq values.

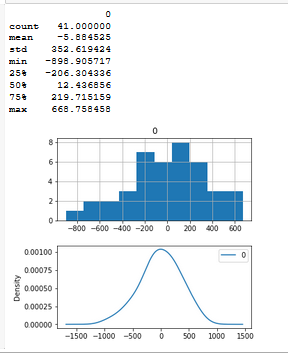
*Best ARIMA (5,2,2) RMSE = 413.571*



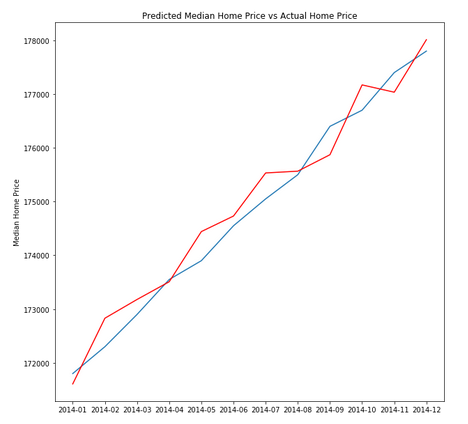
*\*Increased size to 10,10 in Jupyter notebook.*

It is difficult to absorb the AC and PAC plots, but we can visualize and understand how the best ARIMA fit was produced in terms of our p & q order of magnitude. The AC plot last breaks confidence somewhere around 25, but we have a limit of 6 lags to prevent excessive compute. The partial autocorrelation chart shows a break above the upper limit at lag 2, and no further breaks, so we can surmise that order 2 is indeed the best fit there.

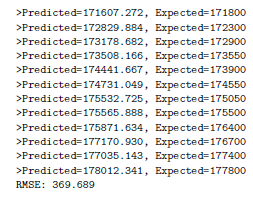
The summary statistics and visuals of the residuals generated from running the model on the training set show high density of clustering towards 0:



After outputting predictions of my training model, evaluating the residuals of that prediction and saving my model for future use, I ran my test set through the model and was met with relatively solid results:



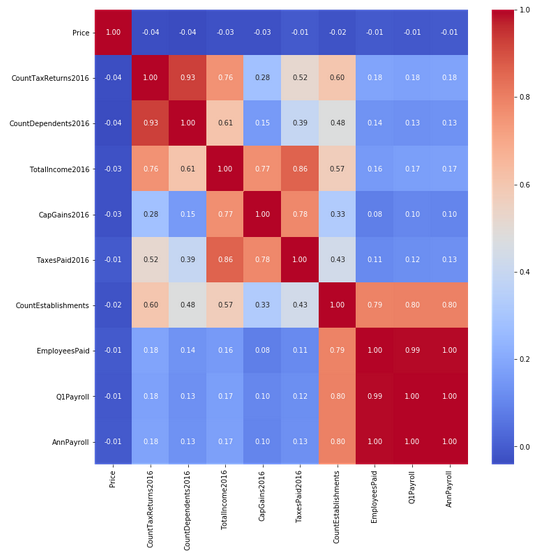
The red line denotes the prediction, while the blue line is the actual values from the test set. The root mean squared error of the predictions against the actual values was 369.689. Results of the forecast in a numerical format were also output:



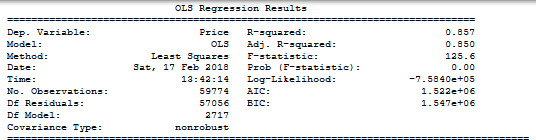
**We note here that our root mean squared error on our test set is lower than that of our training test, implying that we didn’t overfit to our training data.**

The next few steps, again, to keep this brief, involve copious amounts of data wrangling in the form of merges, concatenates, grouping by and slicing. Our output was a dataset containing various economic and labor statistics alongside our original Zillow data. We split a training and testing dataset, as per the above, with our test set being a consolidated average of the various specified date ranges, grouped at the yearly level by state and zip.

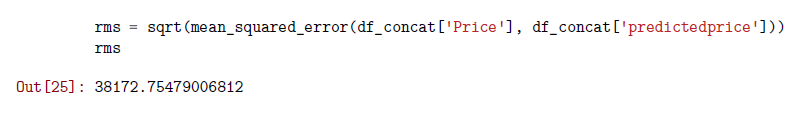
A correlation matrix of our numeric variables show that we have no relationships present between some of our dependent features and price. There are heavy correlations seen between our dependent variables, however. This doesn’t necessarily mean that there is no causal relationship present, so we will avoid column removal until determining significance.



After performing a stepwise regression (OLS) on our data, it was shown that converting zipcode and year into dummy variables, with price as our dependent feature, produced our best model in terms of R^2, which is the proportion of variance in median price that is accounted for by our dependent variables. When trying to construct a model without these dummy variables, I was left with a pretty useless model. Most of the data that was layered in had negative weights, but were show to have very small pvalues, so I decided to leave them in- Although, running a stepwise regression rendered the same R2 and rmse even if I excluded them from them model.



Now, for our prediction I applied this stored model to our test set, and computed the RMSE (Our cost function for multiple linear regression), which is relatively similar to the RMSE of our training model, indicating that we have a generalized linear model that isn’t overfitting on our training data. The computed value of 38172 tells us that the residuals are spread out considerably across our fitted lines – That tells us that the average deviation of prediction values from actual values is around 38k, which, in a dataset with a median of 150k, doesn’t lend much confidence to the predictive capabilities of our model.



Our last step is to determine the best investment opportunities for the Syracuse Real Estate Investment Trust. I went several ways on this, but decided that a viable method of conducting this analysis would be to create a separate test set with the ‘year’ column truncated and replace with the value ‘2020’. The result was a dateframe that contained the actual values of our 2014 test set against the predicted values of our 2020 test set. Some simple formulations landed us with the metrics ‘potential increase in price’ and potential percentage increase in price.

df\_concat\_2020 = df\_concat

df\_concat\_2020["year"] = 2020

df\_concat\_2020['predictedprice2020'] = train\_model\_fit1.predict(df\_concat\_2020)

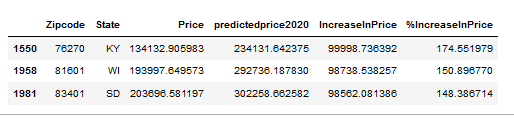
df\_concat\_2020['IncreaseInPrice'] = (df\_concat\_2020['predictedprice2020']- df\_concat\_2020['Price'])

df\_concat\_2020['%IncreaseInPrice'] = (df\_concat\_2020['predictedprice2020']/ df\_concat\_2020['Price']) \* 100

winners = df\_concat\_2020.sort\_values(by=[ 'IncreaseInPrice'],ascending=False).head(3)

winners

The results of the above show the following zip codes as having the largest potential increase in raw value from 2014-2020:



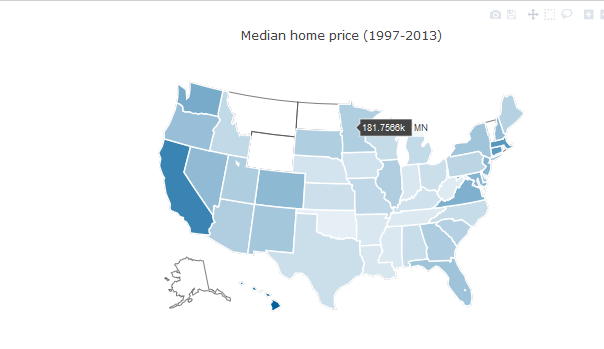
These three zipcodes are expected, per our predictive capacity, to have the largest raw increase in price over the years between 2014 and 2020. It would be interesting to validate this in a few years to calculate the error. Logically, the greatest investment opportunities would arise in highly dense, highly populated areas with tourist attractions, which none of these states, to the best of my knowledge possess in vast quantities. The homes could have very well been undervalued in their respective markets, leading way to greater potential rise in their raw values.

Using RMSE as our ‘cost’ metric, it certainly seems that the ARIMA model is a better used tool for conducting time series analysis and forecasting. The ARIMA model generated in this assignment was forecasting the US median home values, but some slight tweaks would allow us to generate this at a more granular level, such as zipcode, which could thus be used by the SREIT to analyze investment opportunities. I recently attended a conference where there was a case made for using recurrent neural networks in time series analysis due to their ability to retain the temporal aspects of data using Long Short-Term Memory (Looping after hidden layers). A demo was conducted and the resulting RMSE of the LSTM model halved the RMSE of the ARIMA model.

To kind of add to the null value that our OLS regression model added, I filtered the train and test datasets by California, rebuilt the model and tested it. We were met with an RMSE of 90348, which is likely worse than if a person were to take a shot in the dark at an estimate of a future forecast. I think the error was larger in this case because the values of our dependent variable were much higher, on average.

I found that Ridge regression imposes a penalty on the sizes of the coefficients, so I decided to build a ridge model out on my training data. Unfortunately, these only take numerical (floats) values, so I was left with my less than accurate depiction of tax and workforce features. The result wasn’t surprising, but was useful in regards to finding a new technique- This registered an R2 of .001.

Finally, we build an interactive plot using the basemap library which shows median housing prices of the United States, by State (Code from Github, and assistance from Amara). Density is tied with the median values, so we can see that California, Hawaii and New York are responsible for the highest median housing prices when looking at 1997-2013 as an aggregate:



\*Plot seemed to stop rendering in Jupyter notebook after exiting kernel. Pasted data munging operations and plotly code in notebook “Lab2\_MapVisualization\_StateLevel”.

**Sources:**

<Stackoverflow.com>

<https://pandas.pydata.org>

<https://factfinder.census.gov/faces/tableservices/jsf/pages/productview.xhtml?pid=BP_2015_00CZ1&prodType=table>

<https://www.irs.gov/statistics>

<http://scikit-learn.org/stable/modules/linear_model.html>

<http://ipython.readthedocs.io/en/stable/config/options/terminal.html>

<https://www.kaggle.com/abigaillarion/police-fatalities-in-unasited-states>

<https://www.kaggle.com/yassineghouzam/titanic-top-4-with-ensemble-modeling>