Time Series Mod 4 Project Blog Post

This project consisted of being given a time series dataset with a relatively simple question: what are the top 5 zipcodes to invest in?

At first glance, the dataset is messy. It is situated so the time is listed as columns, and there seems to me more information than necessary.

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996-05	1996-06	...
0	84654	60657	Chicago	IL	Chicago	Cook	1	334200.0	335400.0	336500.0	...
1	90668	75070	McKinney	TX	Dallas-Fort Worth	Collin	2	235700.0	236900.0	236700.0	...
2	91982	77494	Katy	TX	Houston	Harris	3	210400.0	212200.0	212200.0	...
3	84616	60614	Chicago	IL	Chicago	Cook	4	498100.0	500900.0	503100.0	...
4	93144	79936	El Paso	TX	El Paso	El Paso	5	77300.0	77300.0	77300.0	...
5	91733	77084	Houston	TX	Houston	Harris	6	95000.0	95200.0	95400.0	...
6	61807	10467	New York	NY	New York	Bronx	7	152900.0	152700.0	152600.0	...
7	84640	60640	Chicago	IL	Chicago	Cook	8	216500.0	216700.0	216900.0	...

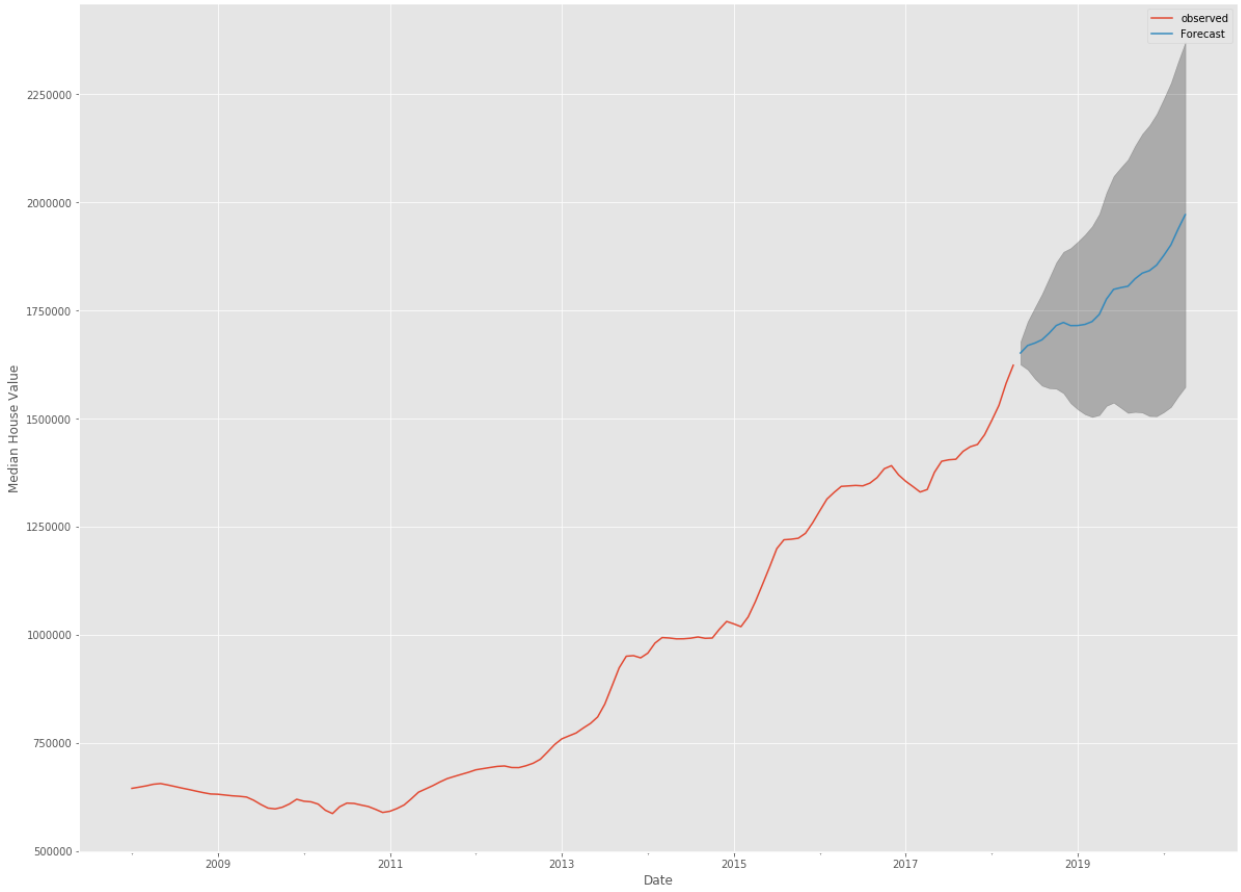
I imported the data, but immediately decided to slim it down to just be New York (bias!).
Next, I wanted to reshape it from Wide to Long format. For this, I used the melt function, and then afterwards changed the time columns (now rows) to datetime. I set them as the index, and slimmed the dataset further for anything post 2008 to account for some data variability from the housing crisis.

I figured, since I would be looking zip-code by zip-code, that writing functions for all of the operations I was going to be performing would be cleaner and simpler in the long run. I decided to write the following functions:

- **sampling:** this samples from the dataset by a provided zipcode. This only returns the median $ values and keeps the time index. We also resample at the Month to avoid errors, (even though the data is already monthly)
- **pdqz:** this is a PDQ parameter function. It uses the AICAIC in order to determine the best p, d, and q values. It takes in a dataframe of time and values. It returns p, which has the attributes, pdq, pdqd, and aic. The aic value is the minimum of all aic values tested. Passing the right PDQs accounts for stationarity in the model in the future.
- **Arimamodel:** this performs an ARIMA model on the dataset. This function takes in the dataframe, and the results of the AIC pdqs, and applies them to an arima model. It returns the output of the model.
- **MSEr:** Mean Squared Error. This function takes in the dataframe, the results of the ARIMA, and the date at which you want the ARIMA to start. It returns the mean squared error of the ARIMA predictions, vs the actual data.

- **Future_value:** This function takes in the dataframe, the results of the ARIMA, and the number of steps into the future. It returns the mean value at the most recent step, the upper bound, and the lower bound of the confidence interval
- **Plotting_forecasts:** This function takes in the dataframe, the results of the ARIMA, the amount of time in the future . The start date of your predictions, the x-axis label, and the y-axis label. It returns a graph with the observed and the forecasted results

Once all functions were defined, I wanted to test it on my home zip-code in Williamsburg, Brooklyn. I tried it out and here are the results:



Clearly we can see that the neighborhood has gone up in value since the late 2000's, with the model predicting even further valuation in the neighborhood for next year (not so great for my rent).