February 18, 2023

Jie wang

jwang326@SYR.EDU

ist-718

IST-718: Lab #2

**Introduction**

Property investment is often seen as an attractive option for investors, as it can offer numerous benefits such as steady cash flow, long-term appreciation, and the ability to leverage investment through borrowing. This could be achieved through rental income, capital appreciation, or a combination of both.

However, choosing the right property is one of the crucial of the investment. And the location is one of the most important factors when it comes to property investment. A property that is located in a desirable area with access to amenities such as public transport, schools, and shopping centers is likely to generate more demand and higher rental income. Additionally, a property in a location with strong economic growth and job opportunities is likely to appreciate in value over time.

This often brings to question: how does one analyze or predict investment property? To start, several Arkansas metro areas will be analyzed. Then, a national generalization will be made to gain a higher-level understanding, before addressing whether specific zip codes can be identified as better investment opportunities. Using forecasting techniques, an ARIMA model will be used to predict mean and median estimates for successive months in 2017, through 2018.

**Analysis**

**Data Sources:**

Multiple datasets (rawdata) were implemented.

* Base data available from Zillow: Zip\_Zhvi\_SingleFamilyResidence.csv
* Unemployment data (1996-2017): unemployment.csv
* Unemployment data\_ by zip code in Syracuse city: syrcode.xlsx
* 500 Most populated zip code: population zip code

Specifically, a main was collected from Zillow, then used to estimate housing worth per city and state. This dataset covers a time series between 1996-01 through 2017-09.

The unemployment rate data from 1996 to 2017 was required from <https://www.bls.gov/lau/>. This additional unemployment ratio feature during years could help improve the time series model. By incorporating historical unemployment data into the model, it may be able to provide a more comprehensive understanding of the economic conditions in the time being analyzed.

The unemployment rate data by zip code in Syracuse city was required from <https://www.bls.gov/lau/>. Filtering the unemployment rate data (<5% unemployment rate were used) for different zip codes in Syracuse can assist the Syracuse Real Estate Investment Trust in making informed investment decisions. Any zip code area with unemployment rate greater than 5% was omitted from the data aggregation.

The 500 most populated zip code data from <https://worldpopulationreview.com/zips> was used to filter the top three zip code with highest investment potential for Syracuse Real Estate Investment Trust (SREIT).

**Method:**

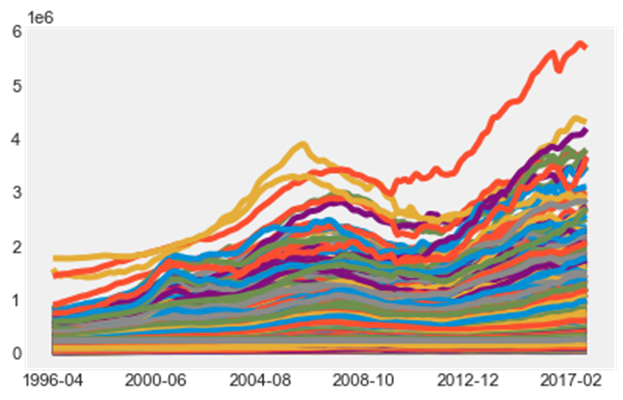
**Time series model:** Multivariant Time series Forecasting that incorporates unemployment rate using Prophet.

**Data splitting:** The threshold date is chosen to split train test data. The data before 01/2017 is considered as training dataset, and the date after 01/2017 was considered as testing data. Based on the threshold date (train\_end\_date) we set before, there are 494 data points in the training dataset and 11 data points in the testing dataset.

**House price rate**: House potential investment opportunities were evaluated by measuring house price increase rate from year 2018 to year 2017.

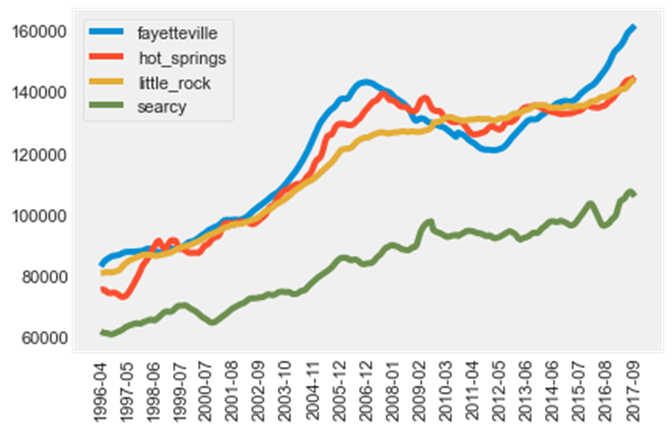
**Results**

**1. An aggregated plot of the full original dataset house price trend**

****The aggregated plot of individual zip code shows that the house price from 1996-01 through 2017-09 exhibit a similar trend (**Figure 1**). The code used to generate the plot above can be reviewed in Appendix A below.

**Figure 1**. Time series plots of house price by zip code.

**2. Time series plot for metro areas in Arkansas**

****It appears that Fayetteville, Arkansas has the greatest increase of property value since 2012, followed by a visually difficult decision between Little Rock Arkansas, Hot Springs Arkansas, and Searcy Arkansas (**Figure 2**). The code used to generate the plot above can be reviewed in Appendix B below.

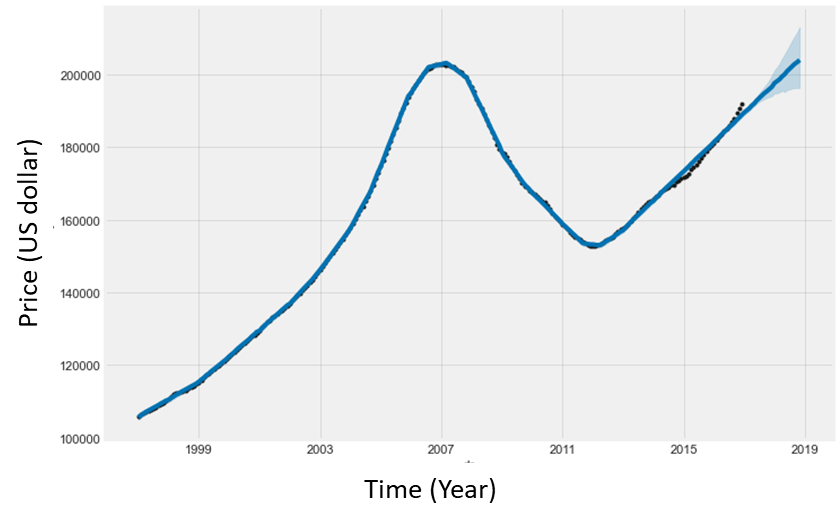
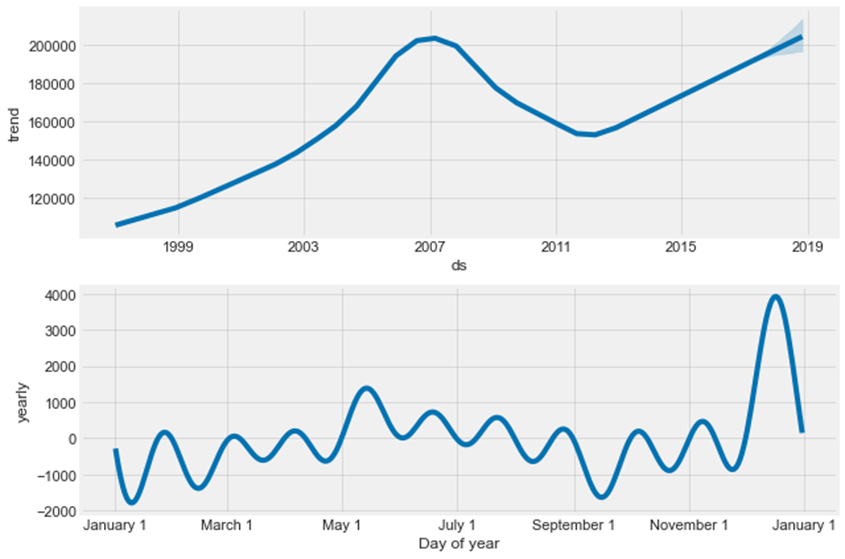
**Figure 2**. Time series plots for metro areas in Arkansas.

**3. Develop model for forecasting average median housing value for 2018 by Prophet.**

Now we build a time series model using Facebook Prophet Python using median house price from indicated time, a time series model was generated using a train dataset to determine whether a time series model could generalize housing data. This was done from 01/1997 to 01/2017, and the year 2018 median house price were also predicted (**Figure 3-4**).

Next, the forecast dataframe with the test dataframe were merged to compare the actual values with the prediction values. The mean absolute error (MAE) for the baseline model is $ 4702.24, meaning that on average, the forecast is off by $4702.24. Given the average house price in 2017 is $ 240623, the prediction is not bad.

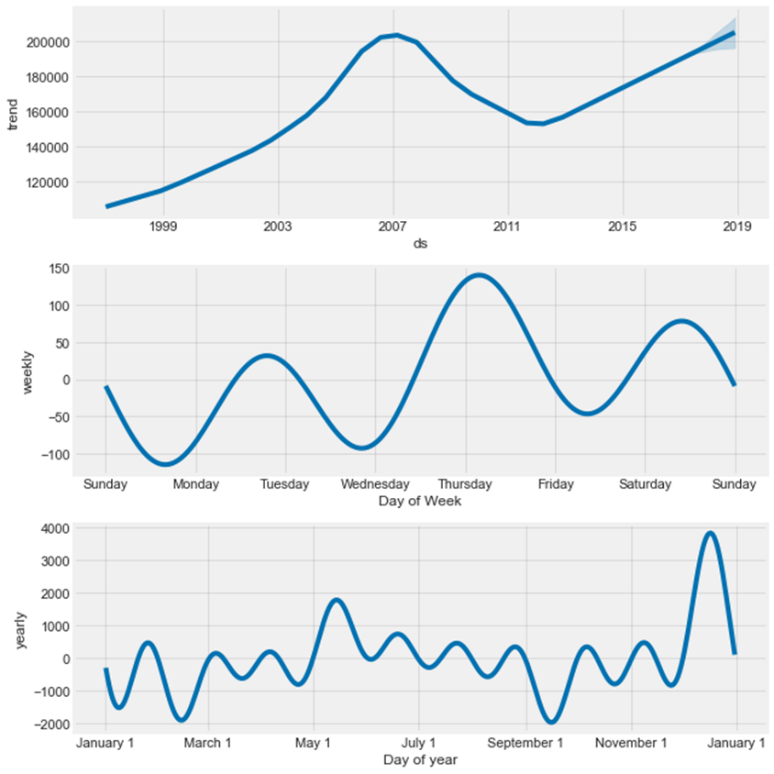
The mean absolute percent error (MAPE) for the baseline model is 2.39%, meaning that on average, the forecast is off by 2.39% of the house price. The code used to generate the plot above can be reviewed in Appendix C below.

 **Figure 3**. Baseline Prophet time series model with 2018-year prediction.

**Figure 4**. Prophet time series components using default hyperparameters.

**4. Develop model for forecasting average median housing value for 2018 by adding seasonality to the baseline model.**

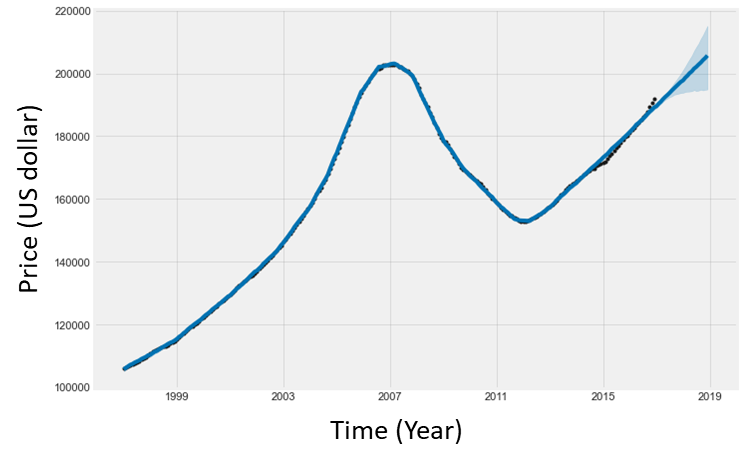
The above baseline model gives a good estimation. Now I tune the model to make the estimation and force the model to consider the yearly seasonality (**Figure 5**). This model performance does not perform better than the baseline model. The MEA for the season model is $ 4856.94; The MAPE for the season model is 2.46%. The code used to generate the plot above can be reviewed in Appendix D below.



**Figure 5.** Prophet time series model with seasonality. The black dots are the actual values. The blue line is the prediction. The blue shades are the prediction for 2018.

**5. Develop model for forecasting average median housing value for 2018 by adding unemployment rate**

To further tune the model, unemployment as an additional predictor using add\_regressor function (**Figure 6-7**). The multivariate model performance is much better than the baseline model. The MAE for the multivariate model is $ 4386.48; The MAPE for the multivariate model is 2.23%. MAE decreased to $ 4386.48 from the baseline model’s $4702.24. MAPE decreased to 2.23% from the baseline model’s 2.39%. The code used to generate the plot above can be reviewed in Appendix E below.

Chart, line chart

Description automatically generated**Figure 6**. Prophet multivariate model forecast with 2018-year prediction. The black dots are the actual values. The blue line is the prediction. The blue shades are the prediction for 2018.

**Figure 7**. Prophet multivariate model components, this components plot has one additional chart for the additional regressor.

**6. Develop model for forecasting average median housing value for 2018 by zip code**

**Table

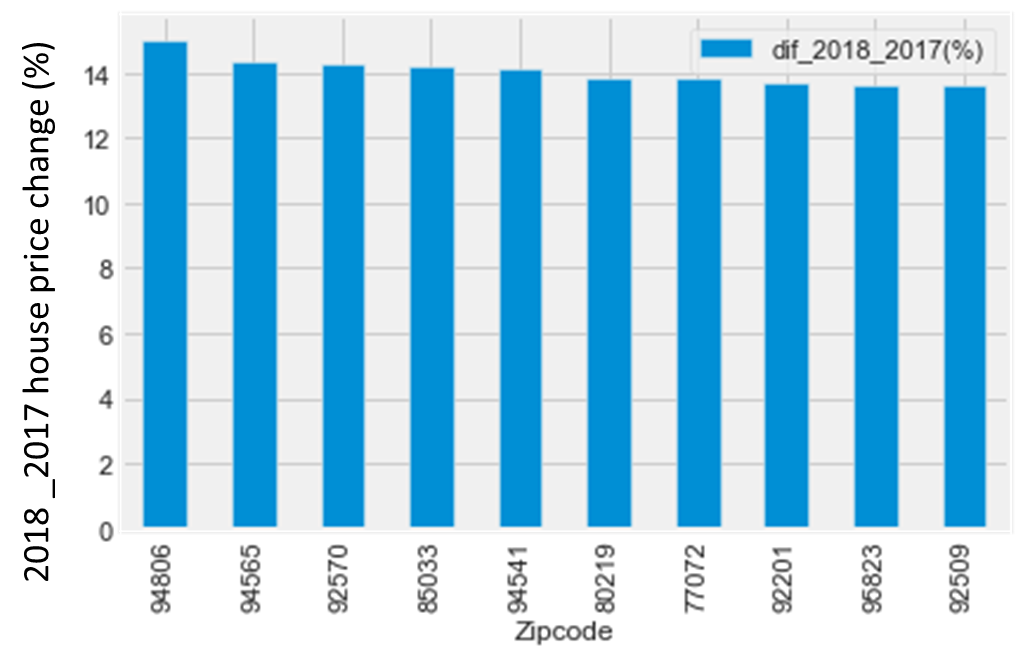
Description automatically generated**Subsequently, utilizing the aforementioned model, predictions for the housing prices of individual zip codes were made.The median house values in the year 2018 were imported into a dataframe. The partial data of this dataframe is presented in **Table 1** and **Supplementary Table 1**. The code used to generate the plot above can be reviewed in Appendix F below.

**Table 1**. Predicted house media price for individual zip code in 2018.

**7. Identify three zip codes provide the best investment opportunity for SREIT.**

To identify the top three zip codes with best investment opportunities in 2018 for SREIT. The original were first filtered with most populated zip code, and with the available zip codes available in the original dataset, there were total 321 zip code house price record data were used. Within those data, the median value of 2018 year predicted price were generated with the Prophet time series model prediction.

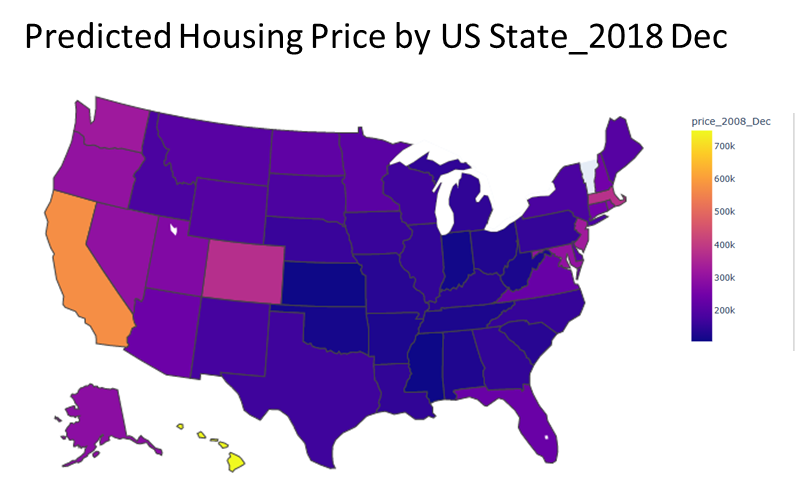
House price increase rate from year 2018 to year 2017 were calculated. As shown in **Figure 8** and **Supplementary Table 1**, and the top three zip code with the great investment potential in 2018 would be zip code 94806 (San Pablo, CA), 94565 (Pittsburge, CA), and 92570(Hutto, TX). The predicted 2018 price increased from 2017 is 15.020%, 14.364%, and 14.262%, respectively. The code used to generate the plot above can be reviewed in Appendix G below.

****

**Figure 8**. Bar graph shows top ten zip code shows the high house price increasement.

In addition, the Syracuse city local zip code house price in next year, were also considered, and the three zip codes would be Zip code13212, 13215, and 13219.

**8.** **Develop a geographic visualization that show the predicted house price in Dec 2018 by state.**

****The predicted house price in 2018 by different state were shown in **Supplementary Table2**. Data were grouped by state, **Figure 9** shows the predicted house price in Dec 2018 in different state. From this map, it seems that in 2018, the house price in HI and CA are much higher than that in other states. The code used to generate the plot above can be reviewed in Appendix H below.

**Figure 9**. House price prediction in Dec 2018 by state

**Conclusions**

In this study, Figure 1-2 show different zip code areas house price, and amongst four Arkansas metro area, each depicting a similar trend, while Fayetteville having much greater recent value. Next, a base Prophet model (Figure 3) with median house price showed a highly accurate model. Prophet model with seasonality did not provide a better model. Interesting, the Prophet model with additional feature – unemployment rate (Multivariant Time series Forecasting) performed the best model and improved the prediction. Specifically, The MAE for the multivariate model is $ 4386.48; The MAPE for the multivariate model is 2.23%.

To narrow down our target, most populated zip codes were focused, and with the prediction of house price in 2018, and calculate the house price the increasement percentage from 2018 to 2017, it was found that the top three zipcodes are zip code 94806 (San Pablo, CA), 94565 (Pittsburge, CA), and 92570(Hutto, TX). Two from CA, and one from TX. In addition, by Dec 2018, median house price in HI and CA are the higher than other state.

This study effectively demonstrated the fundamental principles of time series forecasting using Prophet. It is worth noting that incorporating alternative methodologies and algorithms could potentially enhance the model's accuracy and overall performance. Moreover, the integration of supplementary data sources, such as school district performance or public transportation accessibility by zip code, may result in further improvement of the forecasting model.

**Supplementary Table 1**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Zipcode | Med\_price\_2017 | Med\_price\_2018 | dif\_2018\_2017 | dif\_2018\_2017(%) |
| 94806 | 427400 | 491594.5514 | 64194.5514 | 15.01978 |
| 94565 | 388100 | 443848.2179 | 55748.2179 | 14.3644 |
| 92570 | 299200 | 341873.089 | 42673.089 | 14.2624 |
| 85033 | 152500 | 174094.137 | 21594.137 | 14.16009 |
| 94541 | 581600 | 663839.9936 | 82239.9936 | 14.1403 |
| 80219 | 269000 | 306195.5287 | 37195.5287 | 13.82733 |
| 77072 | 131300 | 149413.4304 | 18113.4304 | 13.79545 |
| 92201 | 249000 | 283042.8013 | 34042.8013 | 13.67181 |
| 95823 | 255100 | 289885.2359 | 34785.2359 | 13.63592 |
| 92509 | 346500 | 393648.458 | 47148.458 | 13.60706 |
| 94509 | 368200 | 417712.1077 | 49512.1077 | 13.44707 |
| 77433 | 209500 | 237644.5088 | 28144.5088 | 13.43413 |
| 33157 | 291300 | 330400.3014 | 39100.3014 | 13.42269 |
| 85301 | 160600 | 182093.7717 | 21493.7717 | 13.38342 |
| 94538 | 842100 | 954742.7457 | 112642.7457 | 13.37641 |
| 94544 | 580700 | 657143.956 | 76443.956 | 13.1641 |
| 33024 | 249500 | 282274.5566 | 32774.5566 | 13.13609 |
| 33142 | 143700 | 162462.6377 | 18762.6377 | 13.05681 |
| 33463 | 232100 | 262384.5238 | 30284.5238 | 13.04805 |
| 94080 | 882600 | 997463.8677 | 114863.8677 | 13.01426 |
| 95051 | 1218600 | 1376881.379 | 158281.379 | 12.98879 |
| 94015 | 885900 | 1000246.524 | 114346.524 | 12.90739 |
| 95355 | 300600 | 339086.8483 | 38486.8483 | 12.80334 |
| 77379 | 219500 | 247526.3306 | 28026.3306 | 12.76826 |
| 95206 | 239900 | 270475.8065 | 30575.8065 | 12.74523 |
| 92571 | 280600 | 315947.9827 | 35347.9827 | 12.59729 |
| 33125 | 266600 | 299937.2418 | 33337.2418 | 12.50459 |
| 89115 | 152600 | 171538.457 | 18938.457 | 12.41052 |
| 92507 | 331600 | 372583.8868 | 40983.8868 | 12.35944 |
| 30058 | 114800 | 128932.2831 | 14132.2831 | 12.31035 |
| 77373 | 137100 | 153936.8874 | 16836.8874 | 12.28073 |
| 95127 | 673100 | 755593.9628 | 82493.9628 | 12.25583 |
| 95828 | 276900 | 310829.4003 | 33929.4003 | 12.2533 |
| 30331 | 133200 | 149473.2559 | 16273.2559 | 12.21716 |
| 30044 | 172500 | 193565.9907 | 21065.9907 | 12.21217 |
| 95112 | 736800 | 826588.1644 | 89788.1644 | 12.18623 |
| 90004 | 1509300 | 1691655.164 | 182355.164 | 12.0821 |
| 33012 | 297900 | 333664.5273 | 35764.5273 | 12.00555 |
| 95014 | 1992200 | 2230048.574 | 237848.574 | 11.93899 |
| 94536 | 967300 | 1082519.283 | 115219.283 | 11.91143 |
| 95687 | 389300 | 435651.623 | 46351.623 | 11.9064 |
| 97701 | 404600 | 452078.475 | 47478.475 | 11.73467 |
| 34953 | 204000 | 227927.074 | 23927.074 | 11.72896 |
| 77449 | 159500 | 178165.9995 | 18665.9995 | 11.70282 |
| 33064 | 192800 | 215328.267 | 22528.267 | 11.68479 |
| 30349 | 121400 | 135561.5922 | 14161.5922 | 11.66523 |
| 94533 | 342900 | 382834.4129 | 39934.4129 | 11.64608 |
| 85041 | 169200 | 188847.9615 | 19647.9615 | 11.61227 |
| 30281 | 145600 | 162462.2627 | 16862.2627 | 11.58122 |
| 60804 | 158400 | 176733.9901 | 18333.9901 | 11.57449 |
| 33015 | 322300 | 359435.8598 | 37135.8598 | 11.52214 |
| 33311 | 169000 | 188299.5574 | 19299.5574 | 11.41986 |
| 77084 | 158700 | 176781.5389 | 18081.5389 | 11.39353 |
| 89123 | 276800 | 308261.5874 | 31461.5874 | 11.36618 |
| 94587 | 811600 | 903204.6447 | 91604.6447 | 11.28692 |
| 94112 | 924700 | 1028956.693 | 104256.693 | 11.27465 |
| 94513 | 543000 | 603644.3988 | 60644.3988 | 11.1684 |
| 95020 | 695000 | 772211.8386 | 77211.8386 | 11.10962 |
| 90042 | 698600 | 776017.2976 | 77417.2976 | 11.08178 |
| 80013 | 314600 | 349430.2037 | 34830.2037 | 11.07127 |
| 92407 | 295000 | 327603.7433 | 32603.7433 | 11.05212 |
| 77396 | 153500 | 170431.4857 | 16931.4857 | 11.03028 |
| 92336 | 416100 | 461829.1697 | 45729.1697 | 10.98995 |
| 33027 | 382800 | 424528.4096 | 41728.4096 | 10.90084 |
| 91762 | 398100 | 441401.5147 | 43301.5147 | 10.87704 |
| 30083 | 118300 | 131148.003 | 12848.003 | 10.86053 |
| 32822 | 170500 | 188930.1254 | 18430.1254 | 10.80946 |
| 33411 | 322900 | 357549.8253 | 34649.8253 | 10.73082 |
| 30135 | 151200 | 167401.2846 | 16201.2846 | 10.71514 |
| 94501 | 967500 | 1070685.275 | 103185.275 | 10.66514 |
| 95035 | 953000 | 1054313.053 | 101313.053 | 10.63096 |
| 77095 | 209700 | 231942.756 | 22242.756 | 10.60694 |
| 89108 | 178900 | 197826.818 | 18926.818 | 10.57955 |
| 92563 | 400300 | 442485.7732 | 42185.7732 | 10.53854 |
| 91331 | 444100 | 490789.8368 | 46689.8368 | 10.51336 |
| 33186 | 328900 | 363043.8832 | 34143.8832 | 10.38124 |
| 95111 | 670700 | 740105.7473 | 69405.7473 | 10.34826 |
| 94122 | 1277900 | 1410041.691 | 132141.691 | 10.34053 |
| 80634 | 273900 | 302186.4226 | 28286.4226 | 10.32728 |
| 85281 | 250000 | 275760.4365 | 25760.4365 | 10.30417 |
| 92503 | 356500 | 393201.4765 | 36701.4765 | 10.29494 |
| 89110 | 175200 | 193216.1616 | 18016.1616 | 10.2832 |
| 92882 | 446700 | 492592.3504 | 45892.3504 | 10.27364 |
| 30052 | 182300 | 200921.9423 | 18621.9423 | 10.215 |
| 89031 | 212300 | 233874.4865 | 21574.4865 | 10.16226 |
| 91766 | 376600 | 414615.0738 | 38015.0738 | 10.09428 |
| 91911 | 448800 | 493393.1572 | 44593.1572 | 9.936087 |
| 60640 | 727700 | 799108.8564 | 71408.8564 | 9.812953 |
| 95123 | 852300 | 935836.0866 | 83536.0866 | 9.801254 |
| 85225 | 243500 | 267364.6734 | 23864.6734 | 9.800687 |
| 80015 | 376800 | 413442.5368 | 36642.5368 | 9.724665 |
| 7087 | 378600 | 415241.4775 | 36641.4775 | 9.67815 |
| 85008 | 190100 | 208472.7113 | 18372.7113 | 9.664761 |
| 92346 | 327600 | 358878.763 | 31278.763 | 9.547852 |
| 30041 | 322600 | 353082.497 | 30482.497 | 9.449007 |
| 32244 | 138900 | 152007.4754 | 13107.4754 | 9.436627 |
| 34744 | 204600 | 223892.7268 | 19292.7268 | 9.429485 |
| 32818 | 177300 | 193938.1151 | 16638.1151 | 9.38416 |
| 60634 | 256000 | 279995.5581 | 23995.5581 | 9.373265 |
| 60647 | 407000 | 445087.2447 | 38087.2447 | 9.358045 |
| 95624 | 399800 | 437150.6025 | 37350.6025 | 9.342322 |
| 92880 | 521400 | 570087.3917 | 48687.3917 | 9.33782 |
| 91402 | 445600 | 486963.1561 | 41363.1561 | 9.282575 |
| 95758 | 363500 | 397193.2595 | 33693.2595 | 9.269122 |
| 84118 | 184800 | 201542.9643 | 16742.9643 | 9.060046 |
| 90805 | 419000 | 456839.3609 | 37839.3609 | 9.030874 |
| 60402 | 194100 | 211542.7014 | 17442.7014 | 8.986451 |
| 92562 | 405500 | 441898.6657 | 36398.6657 | 8.976243 |
| 91761 | 426500 | 464656.1032 | 38156.1032 | 8.946331 |
| 91710 | 460300 | 501244.8492 | 40944.8492 | 8.895253 |
| 91343 | 567600 | 618078.5125 | 50478.5125 | 8.893325 |
| 33414 | 375100 | 408272.1922 | 33172.1922 | 8.84356 |
| 85122 | 152600 | 166053.4872 | 13453.4872 | 8.816178 |
| 91706 | 429000 | 466697.5077 | 37697.5077 | 8.787298 |
| 91744 | 433100 | 470905.903 | 37805.903 | 8.729139 |
| 30062 | 314500 | 341789.3913 | 27289.3913 | 8.677072 |
| 92592 | 438700 | 476654.7047 | 37954.7047 | 8.651631 |
| 7055 | 298100 | 323816.7403 | 25716.7403 | 8.626884 |
| 91730 | 430000 | 467034.2286 | 37034.2286 | 8.612611 |
| 90706 | 489300 | 531394.9291 | 42094.9291 | 8.603092 |
| 93033 | 421500 | 457589.4158 | 36089.4158 | 8.562139 |
| 60632 | 167200 | 181488.9034 | 14288.9034 | 8.545995 |
| 91702 | 431700 | 468537.2758 | 36837.2758 | 8.533073 |
| 37211 | 237400 | 257595.755 | 20195.755 | 8.507058 |
| 32825 | 215900 | 234243.7378 | 18343.7378 | 8.496405 |
| 91950 | 406200 | 440674.8769 | 34474.8769 | 8.487168 |
| 92105 | 405300 | 439666.9333 | 34366.9333 | 8.479382 |
| 90650 | 443800 | 481399.6734 | 37599.6734 | 8.472211 |
| 30040 | 264600 | 286966.8759 | 22366.8759 | 8.45309 |
| 33458 | 423800 | 459532.5627 | 35732.5627 | 8.431468 |
| 90022 | 416300 | 451396.6551 | 35096.6551 | 8.430616 |
| 60618 | 416000 | 451033.5573 | 35033.5573 | 8.421528 |
| 90731 | 593000 | 642581.9726 | 49581.9726 | 8.36121 |
| 91910 | 503200 | 545039.8272 | 41839.8272 | 8.314751 |
| 92126 | 567500 | 614566.2688 | 47066.2688 | 8.293616 |
| 97007 | 426000 | 461267.9117 | 35267.9117 | 8.278853 |
| 97229 | 560500 | 606684.6689 | 46184.6689 | 8.239905 |
| 89121 | 193500 | 209437.0015 | 15937.0015 | 8.236176 |
| 80134 | 454300 | 491590.4238 | 37290.4238 | 8.208326 |
| 60657 | 1073900 | 1161713.118 | 87813.118 | 8.177029 |
| 91732 | 490000 | 529983.1647 | 39983.1647 | 8.15983 |
| 90631 | 584400 | 631903.0723 | 47503.0723 | 8.12852 |
| 91342 | 488200 | 527883.2973 | 39683.2973 | 8.128492 |
| 30004 | 430000 | 464783.2612 | 34783.2612 | 8.089131 |
| 30188 | 212600 | 229745.792 | 17145.792 | 8.064813 |
| 92154 | 444200 | 479994.4058 | 35794.4058 | 8.058173 |
| 34711 | 239100 | 258299.0927 | 19199.0927 | 8.029733 |
| 8701 | 368500 | 397806.8561 | 29306.8561 | 7.953014 |
| 91335 | 515100 | 555982.9229 | 40882.9229 | 7.93689 |
| 91709 | 645100 | 696237.5921 | 51137.5921 | 7.92708 |
| 30024 | 331400 | 357665.7675 | 26265.7675 | 7.925699 |
| 33511 | 204600 | 220741.2941 | 16141.2941 | 7.889196 |
| 85032 | 255700 | 275823.6128 | 20123.6128 | 7.870009 |
| 92057 | 481700 | 519547.6924 | 37847.6924 | 7.857109 |
| 60073 | 144500 | 155844.6354 | 11344.6354 | 7.850959 |
| 93727 | 236300 | 254799.3759 | 18499.3759 | 7.828767 |
| 30127 | 176100 | 189838.5359 | 13738.5359 | 7.801554 |
| 93722 | 225800 | 243389.6581 | 17589.6581 | 7.789928 |
| 7093 | 370100 | 398924.1306 | 28824.1306 | 7.788201 |
| 30096 | 218000 | 234865.4404 | 16865.4404 | 7.736441 |
| 90250 | 573300 | 617614.3651 | 44314.3651 | 7.729699 |
| 97006 | 397500 | 428153.1944 | 30653.1944 | 7.711495 |
| 90660 | 444600 | 478831.0678 | 34231.0678 | 7.699296 |
| 92115 | 500100 | 538452.057 | 38352.057 | 7.668878 |
| 30043 | 216000 | 232478.7865 | 16478.7865 | 7.629068 |
| 60614 | 1525300 | 1640935.956 | 115635.956 | 7.581194 |
| 90640 | 526000 | 565816.9179 | 39816.9179 | 7.569756 |
| 93065 | 561100 | 603168.6819 | 42068.6819 | 7.497537 |
| 93312 | 271200 | 291197.9491 | 19997.9491 | 7.373875 |
| 93030 | 503700 | 540710.6803 | 37010.6803 | 7.347763 |
| 30047 | 198700 | 213298.5802 | 14598.5802 | 7.347046 |
| 90026 | 875700 | 939725.9576 | 64025.9576 | 7.311403 |
| 91770 | 552800 | 593007.1472 | 40207.1472 | 7.273362 |
| 85364 | 126900 | 136108.1046 | 9208.1046 | 7.25619 |
| 92021 | 470000 | 503719.1677 | 33719.1677 | 7.174291 |
| 85308 | 273100 | 292583.5681 | 19483.5681 | 7.134225 |
| 60641 | 284900 | 305072.8844 | 20172.8844 | 7.08069 |
| 2301 | 272200 | 291366.1718 | 19166.1718 | 7.041209 |
| 32210 | 120700 | 129190.879 | 8490.879 | 7.034697 |
| 60623 | 141000 | 150847.3571 | 9847.3571 | 6.983941 |
| 93536 | 322800 | 345066.1393 | 22266.1393 | 6.897813 |
| 34787 | 278300 | 297478.0035 | 19178.0035 | 6.891126 |
| 2155 | 553200 | 591118.9866 | 37918.9866 | 6.854481 |
| 78640 | 183800 | 196085.3588 | 12285.3588 | 6.684091 |
| 30022 | 421500 | 449492.8326 | 27992.8326 | 6.641241 |
| 48197 | 198400 | 211534.8533 | 13134.8533 | 6.62039 |
| 84095 | 387900 | 413370.2097 | 25470.2097 | 6.566179 |
| 96706 | 639200 | 680862.7862 | 41662.7862 | 6.517958 |
| 37013 | 193300 | 205773.4688 | 12473.4688 | 6.452907 |
| 83709 | 222300 | 236563.3414 | 14263.3414 | 6.416258 |
| 2148 | 423700 | 450578.9164 | 26878.9164 | 6.343856 |
| 60085 | 109400 | 116234.6193 | 6834.6193 | 6.247367 |
| 28277 | 382400 | 406214.1531 | 23814.1531 | 6.22755 |
| 37075 | 257100 | 273085.8779 | 15985.8779 | 6.217767 |
| 60629 | 159500 | 169324.4746 | 9824.4746 | 6.159545 |
| 43081 | 228200 | 242246.2547 | 14046.2547 | 6.155239 |
| 30263 | 152200 | 161547.8553 | 9347.8553 | 6.141823 |
| 32218 | 157100 | 166644.3598 | 9544.3598 | 6.07534 |
| 29445 | 165300 | 175324.4204 | 10024.4204 | 6.06438 |
| 60639 | 221600 | 234990.4475 | 13390.4475 | 6.042621 |
| 95747 | 449500 | 476657.7189 | 27157.7189 | 6.041762 |
| 28027 | 198500 | 210450.0654 | 11950.0654 | 6.020184 |
| 37122 | 279200 | 295853.4957 | 16653.4957 | 5.964719 |
| 60651 | 160800 | 170314.0997 | 9514.0997 | 5.916729 |
| 60625 | 423100 | 448060.1671 | 24960.1671 | 5.899354 |
| 98052 | 827500 | 874986.1205 | 47486.1205 | 5.738504 |
| 96818 | 876700 | 926991.653 | 50291.653 | 5.736472 |
| 32765 | 292000 | 308614.0959 | 16614.0959 | 5.689759 |
| 28269 | 186300 | 196819.9028 | 10519.9028 | 5.646754 |
| 98012 | 521800 | 551247.6988 | 29447.6988 | 5.643484 |
| 60016 | 269300 | 284382.2417 | 15082.2417 | 5.600535 |
| 29483 | 180700 | 190653.8509 | 9953.8509 | 5.508495 |
| 32808 | 106100 | 111944.3303 | 5844.3303 | 5.508323 |
| 84043 | 309100 | 326123.6451 | 17023.6451 | 5.507488 |
| 78130 | 197900 | 208791.3969 | 10891.3969 | 5.503485 |
| 37167 | 190900 | 201086.0698 | 10186.0698 | 5.335814 |
| 95630 | 516800 | 544196.167 | 27396.167 | 5.301116 |
| 95608 | 381300 | 401464.2181 | 20164.2181 | 5.288282 |
| 37064 | 412000 | 433758.6277 | 21758.6277 | 5.28122 |
| 22193 | 322200 | 339188.0052 | 16988.0052 | 5.272503 |
| 37128 | 221400 | 233036.0165 | 11636.0165 | 5.255653 |
| 30144 | 202900 | 213436.709 | 10536.709 | 5.193055 |
| 28078 | 276900 | 291245.0267 | 14345.0267 | 5.18058 |
| 83646 | 234200 | 246009.4835 | 11809.4835 | 5.042478 |
| 37129 | 208600 | 218898.6478 | 10298.6478 | 4.937032 |
| 96797 | 679900 | 713466.1043 | 33566.1043 | 4.936918 |
| 77840 | 204800 | 214871.0558 | 10071.0558 | 4.917508 |
| 60608 | 231200 | 242549.6315 | 11349.6315 | 4.90901 |
| 43026 | 234500 | 246005.4583 | 11505.4583 | 4.906379 |
| 74012 | 153300 | 160779.2517 | 7479.2517 | 4.878833 |
| 84015 | 201700 | 211451.8202 | 9751.8202 | 4.834814 |
| 7047 | 398400 | 417080.3708 | 18680.3708 | 4.688848 |
| 32828 | 281900 | 294940.2127 | 13040.2127 | 4.625829 |
| 33647 | 304900 | 318655.8234 | 13755.8234 | 4.511585 |
| 2360 | 340000 | 355319.0831 | 15319.0831 | 4.505613 |
| 27587 | 281600 | 294255.5824 | 12655.5824 | 4.49417 |
| 22030 | 585600 | 610995.1269 | 25395.1269 | 4.3366 |
| 22407 | 250600 | 261419.3006 | 10819.3006 | 4.317359 |
| 14850 | 250100 | 260631.1164 | 10531.1164 | 4.210762 |
| 29681 | 221400 | 230612.6332 | 9212.6332 | 4.161081 |
| 84404 | 179200 | 186556.4153 | 7356.4153 | 4.105142 |
| 43123 | 165400 | 172017.9674 | 6617.9674 | 4.001189 |
| 20906 | 383900 | 398791.3086 | 14891.3086 | 3.878955 |
| 28215 | 148700 | 154382.1958 | 5682.1958 | 3.821248 |
| 22191 | 303700 | 314858.1711 | 11158.1711 | 3.674077 |
| 29485 | 197500 | 204752.4371 | 7252.4371 | 3.67212 |
| 72712 | 194500 | 201502.0787 | 7002.0787 | 3.60004 |
| 19720 | 177900 | 184055.5297 | 6155.5297 | 3.460107 |
| 78045 | 169000 | 174739.8066 | 5739.8066 | 3.396335 |
| 43230 | 200000 | 206763.5269 | 6763.5269 | 3.381763 |
| 63376 | 192300 | 198757.9582 | 6457.9582 | 3.358273 |
| 45011 | 185900 | 192009.844 | 6109.844 | 3.286629 |
| 27406 | 114800 | 118505.8916 | 3705.8916 | 3.228129 |
| 38401 | 152000 | 156844.0641 | 4844.0641 | 3.186884 |
| 44060 | 175300 | 180689.8134 | 5389.8134 | 3.074623 |
| 21234 | 204000 | 210166.2281 | 6166.2281 | 3.022661 |
| 44256 | 212600 | 218909.971 | 6309.971 | 2.968001 |
| 99301 | 192300 | 197897.797 | 5597.797 | 2.910971 |
| 27610 | 149100 | 153395.9211 | 4295.9211 | 2.881235 |
| 19134 | 113800 | 116851.1073 | 3051.1073 | 2.681114 |
| 6902 | 568000 | 583101.3731 | 15101.3731 | 2.658692 |
| 20147 | 606700 | 622802.5309 | 16102.5309 | 2.654118 |
| 73160 | 133300 | 136689.1573 | 3389.1573 | 2.542504 |
| 20874 | 515200 | 528200.8003 | 13000.8003 | 2.523447 |
| 29072 | 196300 | 201011.5144 | 4711.5144 | 2.40016 |
| 23464 | 268200 | 274231.011 | 6031.011 | 2.248699 |
| 10314 | 514700 | 525413.0088 | 10713.0088 | 2.081408 |
| 23462 | 231500 | 236153.4194 | 4653.4194 | 2.010116 |
| 43130 | 129700 | 132269.1247 | 2569.1247 | 1.980821 |
| 37042 | 124100 | 126434.2711 | 2334.2711 | 1.88096 |
| 21122 | 309400 | 315049.1628 | 5649.1628 | 1.825844 |
| 23322 | 343600 | 349431.0842 | 5831.0842 | 1.697056 |
| 87121 | 135700 | 138002.8432 | 2302.8432 | 1.69701 |
| 22192 | 435900 | 443218.4566 | 7318.4566 | 1.67893 |
| 10701 | 456100 | 463364.9919 | 7264.9919 | 1.592851 |
| 44035 | 92100 | 93515.20656 | 1415.20656 | 1.536598 |
| 60620 | 137400 | 139465.8934 | 2065.8934 | 1.503561 |
| 60609 | 135700 | 137637.4336 | 1937.4336 | 1.427733 |
| 23320 | 285800 | 289876.9773 | 4076.9773 | 1.426514 |
| 20878 | 652300 | 661380.064 | 9080.064 | 1.392007 |
| 21740 | 152600 | 154706.6666 | 2106.6666 | 1.380515 |
| 8753 | 266800 | 270011.5992 | 3211.5992 | 1.203748 |
| 10312 | 509100 | 514961.3204 | 5861.3204 | 1.15131 |
| 28314 | 110800 | 112000.1939 | 1200.1939 | 1.083207 |
| 7002 | 352100 | 355016.967 | 2916.967 | 0.828448 |
| 17603 | 146100 | 147181.9947 | 1081.9947 | 0.740585 |
| 87120 | 186300 | 187409.0944 | 1109.0944 | 0.595327 |
| 46307 | 201400 | 202383.8727 | 983.8727 | 0.488517 |
| 21117 | 273400 | 274631.5 | 1231.5 | 0.450439 |
| 72401 | 118700 | 119225.0591 | 525.0591 | 0.442341 |
| 53215 | 98800 | 99007.79288 | 207.79288 | 0.210317 |
| 40475 | 155300 | 155555.6669 | 255.6669 | 0.164628 |
| 87114 | 208800 | 208942.03 | 142.03 | 0.068022 |
| 19124 | 120400 | 120437.178 | 37.178 | 0.030879 |
| 47906 | 182900 | 182813.3887 | -86.6113 | -0.04735 |
| 19111 | 176300 | 176183.6509 | -116.3491 | -0.06599 |
| 6010 | 182200 | 181446.5327 | -753.4673 | -0.41354 |
| 79912 | 171400 | 170472.7915 | -927.2085 | -0.54096 |
| 79938 | 130900 | 129967.5139 | -932.4861 | -0.71237 |
| 79936 | 114000 | 113071.2382 | -928.7618 | -0.8147 |
| 19120 | 119700 | 118328.3412 | -1371.6588 | -1.14591 |
| 7305 | 261900 | 258606.4313 | -3293.5687 | -1.25757 |
| 78520 | 77400 | 76117.23403 | -1282.76597 | -1.65732 |
| 60619 | 149100 | 146564.1571 | -2535.8429 | -1.70077 |
| 78521 | 71700 | 70274.15176 | -1425.84824 | -1.98863 |
| 79924 | 91200 | 89066.43495 | -2133.56505 | -2.33944 |
| 92345 | 227100 | 219898.6611 | -7201.3389 | -3.171 |
| 60617 | 122600 | 118684.2441 | -3915.7559 | -3.19393 |
| 90280 | 413300 | 396657.1935 | -16642.8065 | -4.02681 |
| 92404 | 254100 | 242125.2372 | -11974.7628 | -4.71262 |
| 60628 | 112900 | 105385.2273 | -7514.7727 | -6.65613 |
| 93306 | 187200 | 174390.3513 | -12809.6487 | -6.84276 |
| 90262 | 409900 | 374610.4136 | -35289.5864 | -8.60932 |
| 92376 | 303300 | 266145.4316 | -37154.5684 | -12.2501 |
| 92335 | 317900 | 276318.615 | -41581.385 | -13.08 |
| 90044 | 397000 | 335474.2467 | -61525.7533 | -15.4977 |
| 90037 | 427600 | 357104.3676 | -70495.6324 | -16.4863 |
| 92324 | 272100 | 224850.8409 | -47249.1591 | -17.3646 |
| 92553 | 267800 | 218004.7984 | -49795.2016 | -18.5942 |
| 90003 | 361400 | 287180.098 | -74219.902 | -20.5368 |
| 93307 | 159100 | 125347.8213 | -33752.1787 | -21.2144 |

**Supplementary Table 2**

State price\_2008\_Dec

0 AK 290624.887

1 AL 129409.082

2 AR 126863.087

3 AZ 239688.622

4 CA 572128.541

5 CO 372992.446

6 CT 251239.503

7 DE 205904.569

8 FL 235010.886

9 GA 153125.784

10 HI 746944.256

11 IA 161692.172

12 ID 186350.539

13 IL 153277.740

14 IN 112596.076

15 KS 107891.853

16 KY 142804.012

17 LA 151972.657

18 MA 378283.485

19 MD 303163.917

20 ME 204078.356

21 MI 150648.258

22 MN 209381.069

23 MO 140831.063

24 MS 106502.079

25 MT 207108.352

26 NC 156643.442

27 ND 218698.309

28 NE 161647.614

29 NH 248778.195

30 NJ 328048.035

31 NM 177612.018

32 NV 299415.326

33 NY 186522.255

34 OH 130882.527

35 OK 117913.034

36 OR 302498.018

37 PA 156255.171

38 RI 285581.490

39 SC 140385.085

40 SD 186867.540

41 TN 125525.111

42 TX 169015.761

43 UT 274032.485

44 VA 231316.516

45 WA 323330.541

46 WI 168896.700

47 WV 124189.228

48 WY 202158.164

**Appendix A**

**Time series plots of house price by zip code**

# read data and check the summary of the data

df = pd.read\_csv("Zip\_Zhvi\_SingleFamilyResidence.csv")

df.describe()

# clean up the dataframe

# Rename RegionName to ZipCode and Change Zip Code to String

df.rename(columns={"RegionName":"ZipCode"}, inplace=True)

df["ZipCode"]=df["ZipCode"].map(lambda x: "{:.0f}".format(x))

df["RegionID"]=df["RegionID"].map(lambda x: "{:.0f}".format(x))

df.head()

# 2.2 visulize the price of all the zip code using seaborn.

# group by zipcode

df\_zipcode = df.groupby('ZipCode').agg(np.median).dropna().T

# remove columns: column 0 indicates an NaN column

df\_zipcode\_clean = df\_zipcode.drop(['SizeRank'], axis=0)

# df\_zipcode\_clean = df\_zipcode\_clean.drop([0], axis=1)

df\_zipcode\_clean.plot(legend=None)

plt.grid(False)

plt.show()

**Appendix B**

**Arkansas Metro Area**

# 2.1 develop time serier plots for the following Arkansaa metro areas: Hot Springs, Little Rock, Fayetteville, Searcy

options = ['Fayetteville','Hot Springs', 'Little Rock', 'Searcy']

AR\_df = df.loc[(df['State'] == 'AR') & df['Metro'].isin(options)]

# fill the NA value with the mean of the group

# generate individual dataframe

AR\_Fay = AR\_df.loc[AR\_df['Metro'] == 'Fayetteville']

AR\_Hot = AR\_df.loc[AR\_df['Metro'] == 'Hot Springs']

AR\_Little = AR\_df.loc[AR\_df['Metro'] == 'Little Rock']

AR\_Searcy = AR\_df.loc[AR\_df['Metro'] == 'Searcy']

AR\_Fay = AR\_Fay.fillna(AR\_Fay.mean())

AR\_Hot = AR\_Hot.fillna(AR\_Hot.mean())

AR\_Little = AR\_Little.fillna(AR\_Little.mean())

AR\_Searcy = AR\_Searcy.fillna(AR\_Searcy.mean())

# clean data, delete not significant columns

# delet RegionID, ZipCode, City, State,CountyName, SizeRank,Metro

cols = [0,1,2,3,4,5,6]

AR\_Fay.drop(df.columns[cols],axis=1,inplace=True)

AR\_Hot.drop(df.columns[cols],axis=1,inplace=True)

AR\_Little.drop(df.columns[cols],axis=1,inplace=True)

AR\_Searcy.drop(df.columns[cols],axis=1,inplace=True)

AR\_Fay.loc['mean'] = AR\_Fay.mean()

AR\_Hot.loc['mean'] = AR\_Hot.mean()

AR\_Little.loc['mean'] = AR\_Little.mean()

AR\_Searcy.loc['mean'] = AR\_Searcy.mean()

# timeseries plot

fig, ax = plt.subplots()

ax.plot(AR\_Fay.iloc[-1], linestyle='solid')

ax.plot(AR\_Hot.iloc[-1], linestyle='solid')

ax.plot(AR\_Little.iloc[-1], linestyle='solid')

ax.plot(AR\_Searcy.iloc[-1], linestyle='solid')

# decrease ticks

xmin, xmax = ax.get\_xlim()

ax.set\_xticks(np.round(np.linspace(xmin, xmax, 23), 2))

# rotate ticks + show legend

plt.xticks(rotation=90)

plt.gca().legend(('fayetteville','hot\_springs', 'little\_rock', 'searcy'))

plt.grid(False)

# show overall plot

plt.show()

**Appendix C**

**Baseline Prophet time series model**

# transforms the dataset into a time series modeling dataset.

# Prophet requirs at least two colums as input: a ds column and a y colum

# the ds column has the time information. currently we have the date as the index

# the y column has the time series values. in this case, we are predicting the house price, we will use median price

# train: collapse colums by median

# train: collapse column by median

train\_start = df.columns.get\_loc('1997-01')

train\_stop = df.columns.get\_loc('2017-01')

test\_stop = df.columns.get\_loc('2017-09')

train\_columns = df.iloc[:, train\_start:train\_stop].columns.tolist()

test\_columns = df.iloc[:, (train\_stop ):test\_stop].columns.tolist()

# remove rows with 0's beginning (1997-01) with trainset

date\_columns = df.iloc[:, train\_start:test\_stop].columns.tolist()

df[date\_columns] = df[date\_columns].replace(0, np.nan)

df[date\_columns] = df[date\_columns].dropna()

#

# transpose dataframe: left column data, right column value

#

# date1 val1

# date2 val2

# ... ...

# daten valn

#

df\_train = df[train\_columns].median().T

df\_test = df[test\_columns].median().T

df\_train = pd.DataFrame({'ds': df\_train.index, 'y':df\_train.values})

df\_test = pd.DataFrame({'ds': df\_test.index, 'y':df\_test.values})

# check the shape of the dataset

print(df\_train.shape)

print(df\_test.shape)

# check the minimum and maximum values for the train and test dataset separately gave us the starting and ending dates

print('The start time of the training dataset is ',df\_train['ds'].min())

print('The end time of the training dataset is ',df\_train['ds'].max())

print('The start time of the testing dataset is ',df\_test['ds'].min())

print('The end time of the testing dataset is ',df\_test['ds'].max())

### baseline model

# In this step, we will build a univariate baseline model using the default prophet hyperparameters, and fit the model using the training dataset

# Prophet automatically fits dailym weekly, and yearly seasonalities if the time series is more than two cycles long.

# use the default hyperparameters to initiate the Prophet model

model\_baseline = Prophet()

# Fit the model on the training dataset

model\_baseline.fit(df\_train)

# we will continue with the default values for the baseline model and force the yearly seasonality in the next model to see the impact of the yearly seasonality

# to make a forecast, we first need to create a future dataframe, (2017-2 to 2017-8 test, 2017-9 to 2018-12 prediction)

# there are total 23 period

# create the time range for the forcast

future\_baseline = model\_baseline.make\_future\_dataframe(periods = 24, freq = 'MS') # frequency is Month, use 'MS', instead of 'M'

# make prediction

forecast\_baseline = model\_baseline.predict(future\_baseline)

# Visualize the forecast

model\_baseline.plot(forecast\_baseline);

# the black dots are the actual values

# the blue line is the prediction

# visulize the forecast components

# from the component plot, we can see the the house price has an overall upward trend.

# the monthly seasonality shows that the price tends to be lower at the beginning of year, and higher at June, and much high at Dec.

model\_baseline.plot\_components(forecast\_baseline);

test1 = forecast\_baseline.iloc[[240, 241,242,243,244,245,246,247]]

test1['y'] = df\_test['y'].tolist()

# baseline model performance

# next let's check the model performance

# the forcast dataframe does not include the actual values, so we need to merger the forecast dataframe with the test dataframe to compare the actual value with the predicted values.

# merge actual and predicted values

# performance\_baseline = pd.merge(df\_test, forcast\_baseline[['ds','yhat','yhat\_lower','yhat\_upper']][-23:], on = 'ds')

performance\_baseline = test1

# check MAE value

performance\_baseline\_MAE = mean\_absolute\_error(performance\_baseline['y'], performance\_baseline['yhat'])

print(f'The MAPE for the baseline model is {performance\_baseline\_MAE}')

# check MAPE value

performance\_baseline\_MAPE = mean\_absolute\_percentage\_error(performance\_baseline['y'], performance\_baseline['yhat'])

print(f'The MAPE for the baseline model is {performance\_baseline\_MAPE}')

### Check the prediction media value of house price in 2018

pred\_table = forecast\_baseline.tail(16)

pred\_table1 = pred\_table[['ds', 'yhat\_lower', 'yhat\_upper', 'yhat']]

pred\_table1.rename (columns = {'ds':'date', 'yhat\_lower': 'predict\_lower', 'yhat\_upper': 'predict\_upper', 'yhat': 'predict value'}, inplace =True)

**Appendix D**

**Baseline Prophet time series model** **with seasonality**

# the baseline already give a good estimations, now we tune the model to make the estimations better

# we will force the model to consider the yearly seasonality.

# add seasonality

model\_season = Prophet(yearly\_seasonality =True, weekly\_seasonality =True)

# Fit the model on the training data set

model\_season.fit(df\_train)

# create the time range for the forcast

future\_season = model\_season.make\_future\_dataframe(periods = 24, freq = 'MS') # frequency is Month, use 'MS', instead of 'M'

# make prediction

forecast\_season = model\_season.predict(future\_season)

# Visualize the forecast

model\_season.plot(forecast\_season);

model\_season.plot\_components(forecast\_season);

test2 = forecast\_season.iloc[[241,242,243,244,245,246,247]]

test2['y'] = df\_test['y'].tolist()

test2

# merge actual and predicted values

# performance\_baseline = pd.merge(df\_test, forcast\_baseline[['ds','yhat','yhat\_lower','yhat\_upper']][-23:], on = 'ds')

performance\_season = test2

# check MAE value

performance\_season\_MAE = mean\_absolute\_error(performance\_season['y'], performance\_season['yhat'])

print(f'The MAPE for the season model is {performance\_season\_MAE}')

# check MAPE value

performance\_season\_MAPE = mean\_absolute\_percentage\_error(performance\_season['y'], performance\_season['yhat'])

print(f'The MAPE for the season model is {performance\_season\_MAPE}')

**Appendix E**

**Baseline Prophet time series model** **with unemployment rate**

### Multivariate Model

# In this step, we will add additional data

# require data from Bureau of Labor Statistics gathering unemployment from 1997 to 2023 monthly.

# read data https://data.bls.gov/pdq/SurveyOutputServlet

unemployment = pd.read\_csv('unemployment.csv')

unemployment\_train= unemployment[unemployment['Year']<2017]

# merge unemployment with train data

df\_train\_1 = pd.concat([unemployment\_train, df\_train], axis =1)

df\_train\_1 = df\_train\_1.drop(df\_train\_1.columns[[0,1,2]], axis=1)

unemployment\_test= unemployment[unemployment['Year']==2017]

# unemployment\_test = unemployment\_test.iloc[1:, :]

df\_test['Value'] = unemployment\_test['Value'].tolist()

df\_test\_1 = df\_test.copy()

data = pd.concat([df\_train\_1,df\_test\_1], axis = 0)

data.rename({'ds':'date'}, axis =1, inplace =True)

## Here we added unemployment rate as an additional predictor using the add\_regressor function. standardize = False means the regressor will not be standardized

# unemployment is chosen as a convenient example of illustrating the process of building multivariate model

model\_multivariate = Prophet()

# add regressor

model\_multivariate.add\_regressor('Value', standardize = False)

#Fit the model on the training data set

model\_multivariate.fit(df\_train\_1)

unemploydata\_full = pd.read\_csv('dataa.csv')

# when make forecasts for the multivariate model, we need to make sure that the regressors have values for the forecast periods, so we used left join and appended value data to the future dataframe

# in the case that the forecast is for the future without the regressor data, separate model need to buillt for the regressors to get the predictions for the future dates

# create the time range for the forcast

future\_multivariate = model\_multivariate.make\_future\_dataframe(periods = 24, freq = 'MS') # frequency is Month, use 'MS', instead of 'M'

# Append the regressor values

test = pd.concat([future\_multivariate, unemploydata\_full], axis =1)

# # Fill the missing values with the previous value

# future\_multivariate = test.fillna(method = 'ffill')

# # # check the data

test.tail(10)

# make prediction

forecast\_multivariate = model\_multivariate.predict(test)

# visu

model\_multivariate.plot(forecast\_multivariate);

model\_multivariate.plot\_components(forecast\_multivariate);

test\_v = forecast\_multivariate.iloc[[240, 241,242,243,244,245,246,247]]

test\_v['y'] = df\_test['y'].tolist()

test\_v

## multivariate model performance

# next let's check the model performance

# the forcast dataframe does not include the actual values, so we need to merger the forecast dataframe with the test dataframe to compare the actual value with the predicted values.

# merge actual and predicted values

# performance\_baseline = pd.merge(df\_test\_1, forcast\_baseline[['ds','yhat','yhat\_lower','yhat\_upper']][-23:], on = 'ds')

performance\_mul = test\_v

# check MAE value

performance\_mul\_MAE = mean\_absolute\_error(performance\_mul['y'], performance\_mul['yhat'])

print(f'The MAPE for the multivariate model is {performance\_mul\_MAE}')

# check MAPE value

performance\_mul\_MAPE = mean\_absolute\_percentage\_error(performance\_mul['y'], performance\_mul['yhat'])

print(f'The MAPE for the multivariate model is {performance\_mul\_MAPE}')

**Appendix F**

**Develop model for forecasting average median housing value for 2018 by zip code**

zipcode = list(df\_zipcode\_clean.columns.values)

##################################################################

#### This is significant to predict individual house price in 2018 by zip code

## import the median value of house price in 2018 into a dataframe

cols = ['Zipcode', 'Med\_price\_2018']

zipcode = list(df\_zipcode\_clean.columns.values)

# zipcode50 = zipcode[0:50]

dat = pd.DataFrame(columns = cols)

for i in zipcode:

model\_baseline1 = Prophet()

inputdata = df\_zipcode\_clean[i]

df = pd.DataFrame(inputdata).reset\_index()

df1 = df.set\_axis(['ds','y'], axis=1, inplace=False)

# Fit the model on the training dataset

model\_baseline1.fit(df1)

# create the time range for the forcast

future\_baseline = model\_baseline1.make\_future\_dataframe(periods = 15, freq = 'MS') # frequency is Month, use 'MS', instead of 'M'

# make prediction

forecast\_baseline = model\_baseline1.predict(future\_baseline)

# Visualize the forecast

# model\_baseline1.plot(forecast\_baseline);

### Check the prediction media value of house price in 2018

pred\_table = forecast\_baseline.tail(12)

pred\_table1 = pred\_table[['ds', 'yhat']]

pred\_table1.rename (columns = {'ds':'date', 'yhat': 'predict value'}, inplace =True)

price\_2008 = pred\_table1['predict value'].median()

dat = dat.append({'Zipcode': str(i), 'Med\_price\_2018':price\_2008},ignore\_index=True)

**Appendix G**

**Identify three zip codes provide the best investment opportunity for SREIT.**

### down sample with selecting the top 500 populated zip code

## read the top 500 populated zip code

pp\_zip = pd.read\_csv('population zip code.csv')

pp\_zip\_list = pp\_zip['zip'].tolist()

pp\_zip\_list

## select the zip code for downsampling

str\_list = [str(i) for i in pp\_zip\_list]

seleted\_cols = list(filter(lambda col: col in str\_list,df\_zipcode\_clean.columns))

df\_zipcode\_500 = df\_zipcode\_clean.loc[:,seleted\_cols]

df\_zipcode\_500

#### This is significant to predict individual house price in 2018 by zip code

## import the median value of house price in 2018 into a dataframe

cols = ['Zipcode', 'Med\_price\_2018']

zipcode = list(df\_zipcode\_500.columns.values)

dat = pd.DataFrame(columns = cols)

for i in zipcode:

model\_baseline1 = Prophet()

inputdata = df\_zipcode\_clean[i]

df = pd.DataFrame(inputdata).reset\_index()

df1 = df.set\_axis(['ds','y'], axis=1, inplace=False)

# Fit the model on the training dataset

model\_baseline1.fit(df1)

# create the time range for the forcast

future\_baseline = model\_baseline1.make\_future\_dataframe(periods = 15, freq = 'MS') # frequency is Month, use 'MS', instead of 'M'

# make prediction

forecast\_baseline = model\_baseline1.predict(future\_baseline)

# Visualize the forecast

# model\_baseline1.plot(forecast\_baseline);

### Check the prediction media value of house price in 2018

pred\_table = forecast\_baseline.tail(12)

pred\_table1 = pred\_table[['ds', 'yhat']]

pred\_table1.rename (columns = {'ds':'date', 'yhat': 'predict value'}, inplace =True)

price\_2008 = pred\_table1['predict value'].median()

dat = dat.append({'Zipcode': str(i), 'Med\_price\_2018':price\_2008},ignore\_index=True)

## now get the median price from 2017. and merege data to caluculate the increase price (%) from 2017 to 2018

last\_9\_rows = df\_zipcode\_500.tail(9)

medians = last\_9\_rows.median()

print(medians)

# export data

dat.to\_csv('2018predicted.csv')

dfa.to\_csv('2017.csv')

combined2017\_2018 = pd.read\_csv('2017\_2018combined.csv')

combined2017\_2018['dif\_2018\_2017'] = combined2017\_2018['Med\_price\_2018'] - combined2017\_2018['Med\_price\_2017']

combined2017\_2018['dif\_2018\_2017(%)'] = 100 \* combined2017\_2018['dif\_2018\_2017']/combined2017\_2018['Med\_price\_2017']

#### the top 3 zip code is 94806, 94565, and 926570

# visulize the the top ten zip code

sorted\_combined2017\_2018a = sorted\_combined2017\_2018[['Zipcode', 'dif\_2018\_2017(%)']]

sorted\_combined2017\_2018b = sorted\_combined2017\_2018a.set\_index('Zipcode')

sorted\_combined2017\_2018b[0:10].plot(kind="bar")

**Appendix H**

**Develop a geographic visualization that show the predicted house price in Dec, 2018 by state.**

### predict house price in 2018 by state

##################################################################

##################################################################

#### This is significant to predict individual house price in 2018 by zip code

## import the median value of house price in 2018 into a dataframe

cols = ['State', 'price\_2008\_Dec']

state = list(df\_state.columns.values)

dat = pd.DataFrame(columns = cols)

for i in state:

model\_baseline1 = Prophet()

inputdata = df\_state[i]

df = pd.DataFrame(inputdata).reset\_index()

df1 = df.set\_axis(['ds','y'], axis=1, inplace=False)

# Fit the model on the training dataset

model\_baseline1.fit(df1)

# create the time range for the forcast

future\_baseline = model\_baseline1.make\_future\_dataframe(periods = 15, freq = 'MS') # frequency is Month, use 'MS', instead of 'M'

# make prediction

forecast\_baseline = model\_baseline1.predict(future\_baseline)

# Visualize the forecast

# model\_baseline1.plot(forecast\_baseline);

### Check the prediction media value of house price in 2018

pred\_table = forecast\_baseline.tail(12)

pred\_table1 = pred\_table[['ds', 'yhat']]

pred\_table1.rename (columns = {'ds':'date', 'yhat': 'predict value'}, inplace =True)

price\_2008\_Dec = pred\_table1['predict value'].iloc[-1]

dat = dat.append({'State': str(i), 'price\_2008\_Dec':price\_2008\_Dec},ignore\_index=True)

import plotly.express as px

# read in your dataframe (replace 'your\_dataframe.csv' with the name of your actual file)

# dat

# create a choropleth map using the 'state' and 'median price' columns in the dataframe

fig = px.choropleth(dat, locations='State', locationmode='USA-states', color='price\_2008\_Dec', scope='usa',

title='Predicted Housing Price by US State\_2018 Dec')

# display the resulting visualization

fig.show()