IST-718: Lab #2

PROFESSOR FOX

Mario Ruiz
MJRUIZ@SYR.EDU | IST-718

Introduction

Property is often thought as one of the safer investment opportunities compared to various alternatives. This often brings to question: how does one analyze or predict investment property? Can linear regression, or time series analysis be implemented? 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 Preparation:

Multiple datasets were implemented, and version controlled with the codebase:

- https://github.com/jeff1evesque/ist-718-lab/blob/master/data/

Specifically, a main `Zip_Zhvi_SingleFamilyResidence.csv` 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. However, additional FBI crime data was aggregated based on state county:

- maryland.xls
- new-hampshire.xls
- virginia.xls
- district-of-columbia.xls

This allowed the former data to be filtered based on specific crime criteria. In the case of this study, the sum of all offense types divided by the county population (i.e. crime ratio). The ratio was not allowed to exceed 3%. Any county that exceed this value was omitted from the data aggregation. However, since there was only one row of data in the DC FBI dataset, its use was omitted, and possibly left as future exercise. Next labor force data was collected from the Department of Labor:

- laucnty17.xlsx

When joined with the latter combined data, the composition allowed further filtering based on unemployment rate. For this study, unemployment was not allowed to exceed 3.5%. Finally, the adjusted data was filtered on select locations to reduce the modeling:

- Maryland
- Virginia
- Washington D.C.
- Massachusetts
- New Hampshire

Furthermore, it is important to note that any `NaN` cell from the previous datasets were converted to zero. Then, the `zipcodes` package was used to introduce an additional zipcode column:

```
def get_zipcode(city, state):
    result = zipcodes.filter_by(
        zipcodes.list_all(),
        active=True,
        city=city,
        state=state
    )
    if result and result[0] and result[0]['zip_code']:
        return(result[0]['zip_code'])
    else:
        return(0)
df['zip_code'] = df[['City', 'State']].apply(
    lambda x: get_zipcode(
        x['City'].upper(),
        x['State'].upper()
    ),
    axis=1
)
```

Results

Time series models were generated for metro areas in Arkansas:

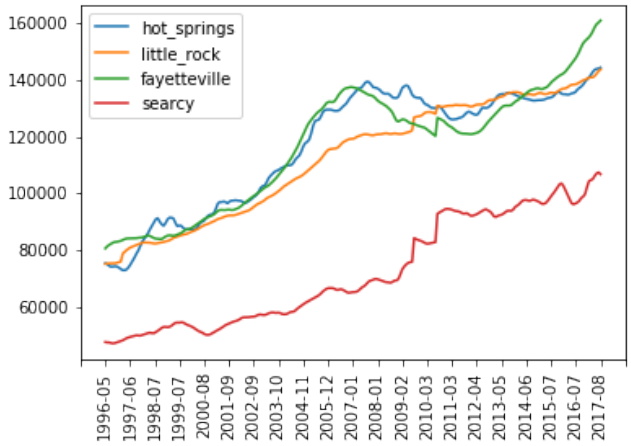


Figure 1. Time series plots for metro areas in Arkansas. The code used to generate the plot above can be reviewed in Appendix A below.

It appears that Fayetteville, Arkansas has the greatest increase of property value, followed by a visually difficult decision between Little Rock and Hot Springs Arkansas. Next, an ARIMA 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:

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	239
Model:	ARIMA(5, 1, 0)	Log Likelihood	-1992.196
Method:	css-mle	S.D. of innovations	1006.221
Date:	Sun, 18 Nov 2018	AIC	3998.391
Time:	21:41:45	BIC	4022.726
Sample:	02-01-1997 - 12-01-2016	HQIC	4008.197

	coef	std err	z	P>|z|	[0.025	0.975]
const	1010.1910	484.735	2.084	0.038	60.127	1960.255
ar.L1.D.y	0.7894	0.065	12.235	0.000	0.663	0.916
ar.L2.D.y	-0.0996	0.082	-1.211	0.227	-0.261	0.062
ar.L3.D.y	0.1331	0.082	1.625	0.106	-0.027	0.294
ar.L4.D.y	0.0433	0.082	0.527	0.598	-0.118	0.204
ar.L5.D.y	0.0044	0.064	0.068	0.946	-0.121	0.130

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.1035	-0.0000j	1.1035	-0.0000
AR.2	0.3177	-2.1123j	2.1360	-0.2262
AR.3	0.3177	+2.1123j	2.1360	0.2262
AR.4	-5.8238	-3.4003j	6.7437	-0.4159
AR.5	-5.8238	+3.4003j	6.7437	0.4159

Figure 2. Descriptive statistics for an overall ARIMA model between 01/1997 and 01/2017 (train set). The code used to generate the plot above can be reviewed in Appendix B below.

An overall residual (Figure 3), and kernel density estimation (kde) plot (Figure 4) were generated using the same train dataset:

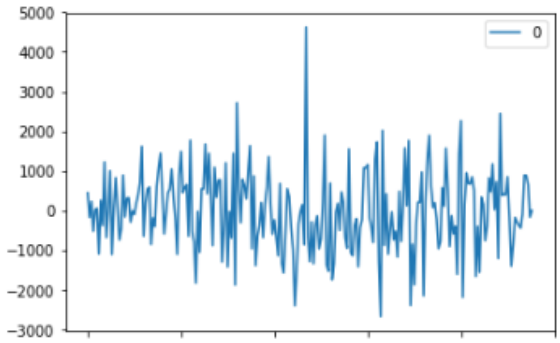


Figure 3. Residual plot for an overall ARIMA model. The code used to generate the plot above can be reviewed in Appendix C below.

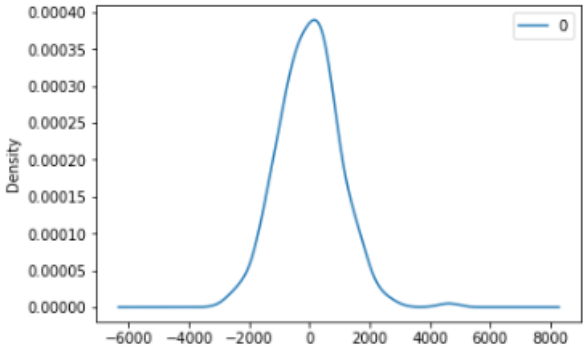


Figure 4. KDE plot for an overall ARIMA model. The code used to generate the plot above can be reviewed in Appendix C below.

Descriptive statistics for the overall ARIMA model:

	0
count	239.000000
mean	-2.069194
std	1008.618450
min	-2682.753851
25%	-675.706720
50%	9.156721
75%	595.122398
max	4625.965351

Figure 5. Descriptive statistics for the overall ARIMA model. The code used to generate the table above can be reviewed in Appendix C below.

The predicted values with error were computed for the remaining months in 2017, which were not in the train dataset (Figure 6), as well as for the 2018 months (Figure 7):

```
========================================
date: 2017-2
----------------------------------------
predicted=388813.576701, expected=39105
0.000000
prediction difference: 0.005719

========================================
predicted=388813.576701

========================================
date: 2017-3
----------------------------------------
predicted=390443.689396, expected=39290
0.000000
prediction difference: 0.006252
========================================
predicted=390443.689396

========================================
date: 2017-4
----------------------------------------
predicted=392003.290988, expected=39580
0.000000
```

```
prediction difference: 0.009592

========================================
predicted=392003.290988

========================================
date: 2017-5
----------------------------------------
predicted=393503.316512, expected=39885
0.000000
prediction difference: 0.013405

========================================
predicted=393503.316512

========================================
date: 2017-6
----------------------------------------
predicted=394958.772728, expected=40085
0.000000
prediction difference: 0.014697

========================================
predicted=394958.772728

========================================
date: 2017-7
----------------------------------------
predicted=396374.708561, expected=40225
0.000000
prediction difference: 0.014606

========================================
predicted=396374.708561

========================================
date: 2017-8
----------------------------------------
predicted=397752.815045, expected=40340
0.000000
prediction difference: 0.013999

========================================
Test MSE: 22164768.684165
```

Figure 6. ARIMA prediction and error rate, from the remaining 2017 months not in train. The code used to generate the plot above can be reviewed in Appendix D below.

```
========================================
date: 2017-9
----------------------------------------
predicted=409223.452509

========================================
date: 2017-10
----------------------------------------
predicted=413533.867822

========================================
date: 2017-11
----------------------------------------
predicted=417515.922014

========================================
date: 2017-12
----------------------------------------
predicted=421472.969033

========================================
date: 2018-1
----------------------------------------
predicted=425233.650838

========================================
date: 2018-2
----------------------------------------
predicted=428722.288873

========================================
date: 2018-3
----------------------------------------
predicted=431986.176158

========================================
date: 2018-4
----------------------------------------
predicted=435068.894530

========================================
date: 2018-5
----------------------------------------
predicted=437986.877565

========================================
date: 2018-6
----------------------------------------
predicted=440749.383756

========================================
date: 2018-7
----------------------------------------
predicted=443369.575329

========================================
date: 2018-8
----------------------------------------
predicted=445861.222357

========================================
date: 2018-9
----------------------------------------
predicted=448234.798712
```

```
========================================
date: 2018-10
----------------------------------------
predicted=450501.831974

========================================
date: 2018-11
----------------------------------------
predicted=452672.004600

========================================
date: 2018-12
----------------------------------------
predicted=454753.750478
```

Figure 7. ARIMA predictions for 2018 months. The code used to generate the plot above can be reviewed in Appendix D below.

To visualize the difference between the predicted and actual values:

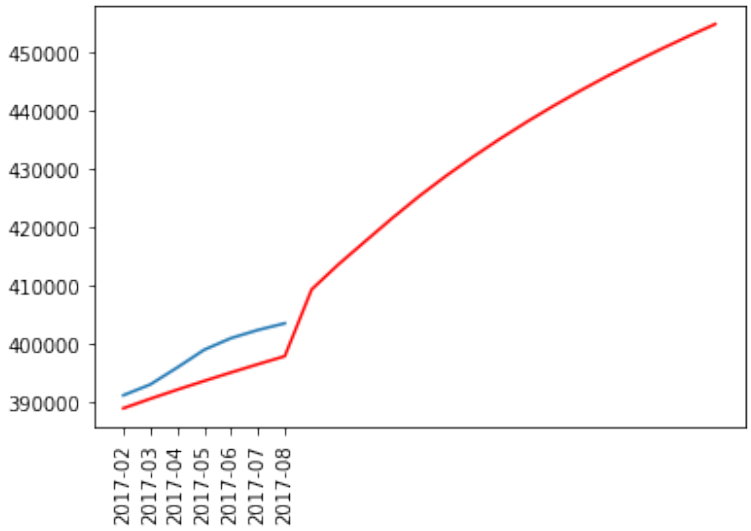


Figure 8. Comparison of the predicted (red) vs. actual mean (blue). The code used to generate the plot above can be reviewed in Appendix E below.

Upon reviewing the above train case, it is evident that none of the difference between the predicted with the actual test data exceed 1.5%. Therefore, the implementation of `ARIMA(5,1,0)` for the entire population, was extended for each individual zip code. However, since the internal `statsmodels.tsa.arima_model` does not allow `d > 2`[1], the differencing was indirectly applied, by adapting the actual dataset only when the base implementation would not succeed.

Each zipcode data was passed to a custom `compute_arima` function. However, a direct implementation could sometimes fail. Specifically, the `statsmodels` package

[1] https://www.statsmodels.org/dev/_modules/statsmodels/tsa/arima_model.html

could fail with `LinAlgError: SVD did not converge`[2] with unusual combination of data with arima arguments. This potential of this arising was reconciled by incrementing the difference factor until the error was eliminated.

A computed rolling prediction successfully generated n predictions after the train. Conveniently, earlier edge cases did not arise. Therefore, each computation succeeded with a standard non-iterative differencing strategy.

```
predictions: [441994.60672424873, 447533.39722298674, 452893.74103927833,
457767.27660728176, 462337.7806300529, 466176.1912943393, 468413.750851138
44, 470550.8847704935, 473187.8653109091, 475525.0143412446, 478662.498983
5472, 482899.99798691145, 487237.503724659, 491175.01066866284, 494912.514
4486477, 499150.0146820069, 504687.51469225716, 511125.0146084265, 517462.
5145216945, 523000.0144826223, 527837.5144814998, 532075.0144836435, 53521
2.5144869951]
```

Figure 10. Rolling prediction for a given zipcode in the aggregated list. Appendix F and Appendix G demonstrate the code to iterate all zipcode in the provided dataset.

An aggregated plot of the full original dataset duration indicates similar trends:

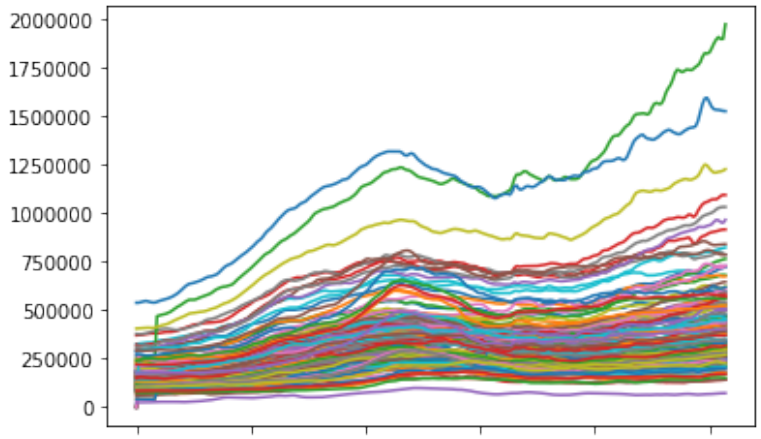


Figure 11. Zipcode trend for select zipcodes. The code used to generate the plot above can be reviewed in Appendix H below.

Based on the above plot, an iterative differencing would likely not provide large benefits. However, it is possible to further smooth the corresponding models. After models were generated from the train, 23 iterative months were predicted. The first seven months of this iterative process coincided with the test data and was used to validate the strength of the model. The successive 16 months were used for future forecasting.

[2] https://github.com/scipy/scipy/issues/4524

The most accurate model was found to be Grottoes, VA (24441). The rolling prediction for successive months starting 2017-02:

1. $161,484.18/0.01
2. $163,610.26/0.006
3. $166,112.94/0.005
4. $167,715.31/0.007
5. $167,951.84/0.01
6. $168,296.66/0.01
7. $169,441.690.008
8. $170,185.82
9. $169,725.05
10. $168,858.37
11. $168,488.13
12. $168,717.51
13. $168,446.43
14. $168,174.83
15. $168,702.87
16. $168,930.72
17. $168,958.52
18. $169,086.27
19. $168,713.98
20. $167,941.67
21. $168,869.34
22. $171,397.01
23. $175,124.68

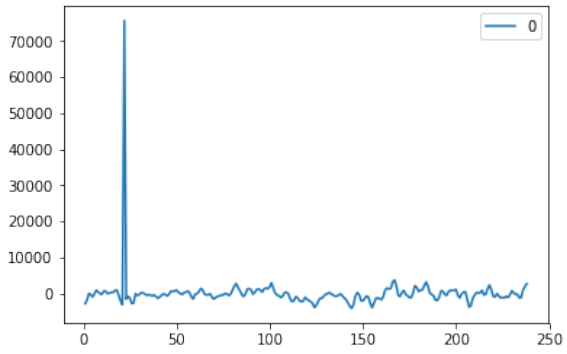


Figure 11. Residuals plot for most accurate model (24441). The code used to generate the plot above can be reviewed in Appendix I below.

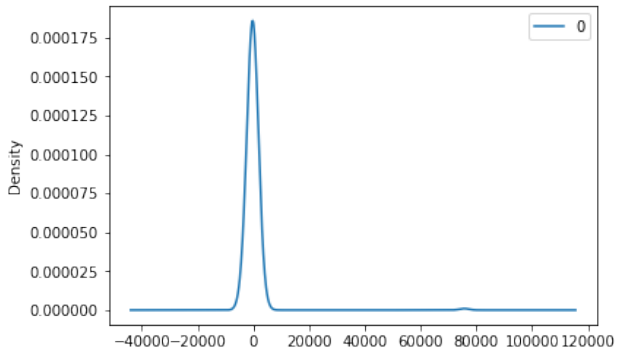


Figure 13. KDE plot for most accurate model (24441). The code used to generate the plot above can be reviewed in Appendix I below.

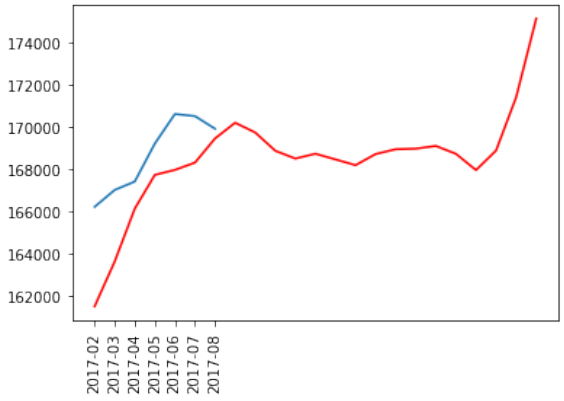


Figure 12. Most accurate model (red), against actual (blue) (01002). The code used to generate the plot above can be reviewed in Appendix I below.

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	238
Model:	ARIMA(5, 1, 0)	Log Likelihood	-2369.261
Method:	css-mle	S.D. of innovations	5094.435
Date:	Wed, 21 Nov 2018	AIC	4752.522
Time:	01:33:07	BIC	4776.828
Sample:	1	HQIC	4762.318

	coef	std err	z	P>|z|	[0.025	0.975]
const	327.6580	387.529	0.846	0.399	-431.884	1087.200
ar.L1.D.y	0.0301	0.065	0.465	0.642	-0.097	0.157
ar.L2.D.y	0.0207	0.065	0.320	0.749	-0.106	0.147
ar.L3.D.y	0.0263	0.064	0.408	0.684	-0.100	0.153
ar.L4.D.y	0.0371	0.064	0.577	0.565	-0.089	0.163
ar.L5.D.y	0.0357	0.064	0.555	0.579	-0.090	0.162

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.6773	-0.0000j	1.6773	-0.0000
AR.2	0.4120	-1.8837j	1.9283	-0.2157
AR.3	0.4120	+1.8837j	1.9283	0.2157
AR.4	-1.7709	-1.1664j	2.1205	-0.4073
AR.5	-1.7709	+1.1664j	2.1205	0.4073

Figure 14. Model summary for most accurate model (24441). The code used to generate the plot above can be reviewed in Appendix I below.

The second most accurate model was found to be Dayton, VA (22821). The rolling prediction for successive months starting 2017-02:

1. $198,827.60/0.01
2. $198,677.87/0.0006
3. $199,964.44/0.002
4. $201476.82/0.007
5. $201,596.99/0.006
6. $202,217.43/0.01
7. $203,639.40/0.02
8. $205,962.23
9. $206,785.41
10. $204,108.72
11. $201,832.09
12. $204,355.50
13. $208978.92
14. $212,402.35
15. $215,625.79
16. $219,249.22
17. $221,972.66
18. $222,996.10
19. $224,019.54
20. $224,742.97
21. $224,466.41
22. $224,489.85
23. $225,413.29

Note: the first seven months (test) above, provide a prediction significance against the corresponding test value.

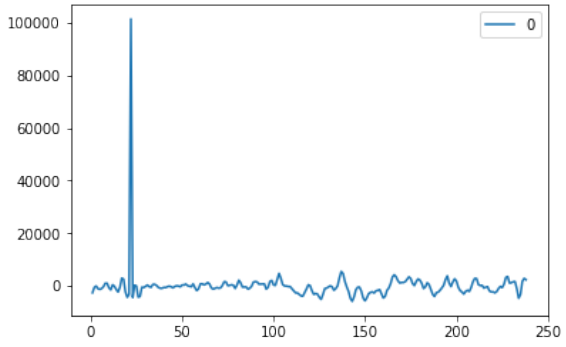


Figure 15. Residuals plot for most accurate model (22821). The code used to generate the plot above can be reviewed in Appendix I below.

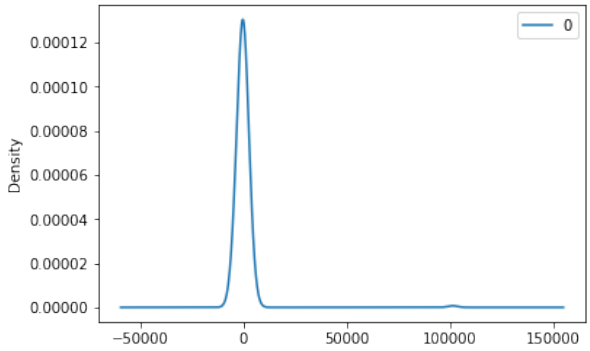


Figure 17. KDE plot for most accurate model (22821). The code used to generate the plot above can be reviewed in Appendix I below.

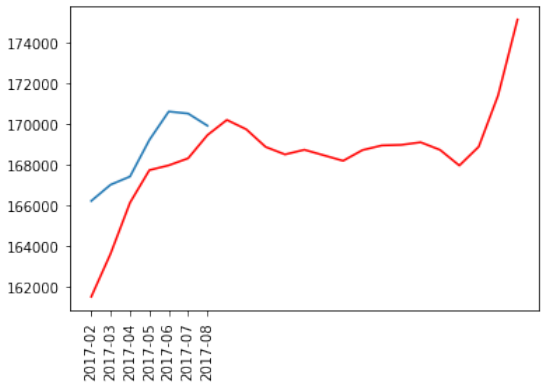
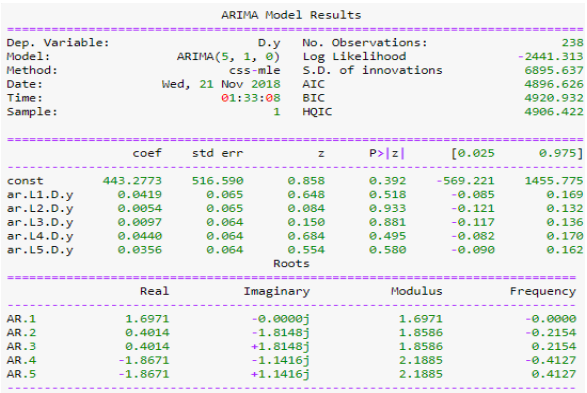


Figure 16. Most accurate model (red), against actual (blue) (22821). The code used to generate the plot above can be reviewed in Appendix I below.



ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	238
Model:	ARIMA(5, 1, 0)	Log Likelihood	-2441.313
Method:	css-mle	S.D. of innovations	6895.637
Date:	Wed, 21 Nov 2018	AIC	4896.626
Time:	01:33:08	BIC	4920.932
Sample:	1	HQIC	4906.422

	coef	std err	z	P>|z|	[0.025	0.975]
const	443.2773	516.590	0.858	0.392	-569.221	1455.775
ar.L1.D.y	0.0419	0.065	0.648	0.518	-0.085	0.169
ar.L2.D.y	0.0054	0.065	0.084	0.933	-0.121	0.132
ar.L3.D.y	0.0097	0.064	0.150	0.881	-0.117	0.136
ar.L4.D.y	0.0440	0.064	0.684	0.495	-0.082	0.170
ar.L5.D.y	0.0356	0.064	0.554	0.580	-0.090	0.162

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.6971	-0.0000j	1.6971	-0.0000
AR.2	0.4014	-1.8148j	1.8586	-0.2154
AR.3	0.4014	+1.8148j	1.8586	0.2154
AR.4	-1.8671	-1.1416j	2.1885	-0.4127
AR.5	-1.8671	+1.1416j	2.1885	0.4127

Figure 18. Model summary for most accurate model (22821). The code used to generate the plot above can be reviewed in Appendix I below.

The third most accurate model was found to be Elkton, VA (22827). The rolling prediction for successive months starting 2017-02:

1. $175,100.62/0.03
2. $174,356.88/0.01
3. $174,299.22/0.002
4. $175,067.27/0.008
5. $175,429.34/0.009
6. $175,691.15/0.003
7. $176,753.34/0.005
8. $177,813.93
9. $178,073.80
10. $177,533.66
11. $176,793.45
12. $177,853.19
13. $180,512.94
14. $183,972.70
15. $187,432.47
16. $190,192.23
17. $192,251.99
18. $195,111.76
19. $198,171.53
20. $199,131.29
21. $198,691.06
22. $198,450.82
23. $198,410.59

Note: the first seven months (test) above, provide a prediction significance against the corresponding test value.

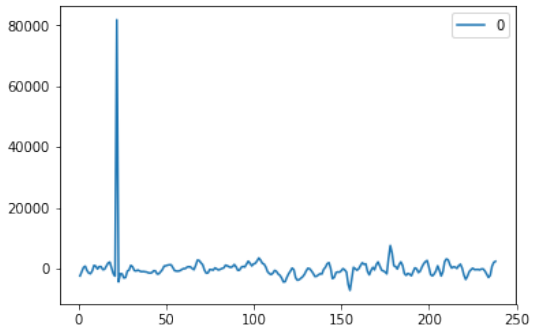


Figure 19. Residuals plot for most accurate model (22827). The code used to generate the plot above can be reviewed in Appendix I below.

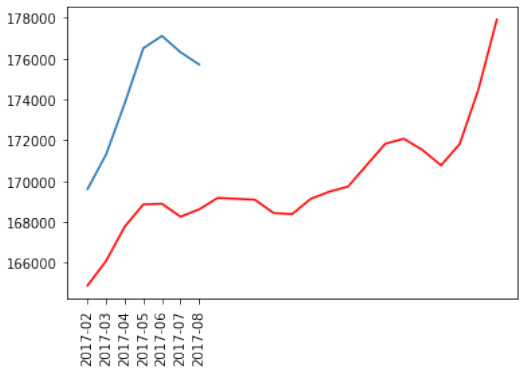


Figure 20. Most accurate model (red), against actual (blue) (22827). The code used to generate the plot above can be reviewed in Appendix I below.

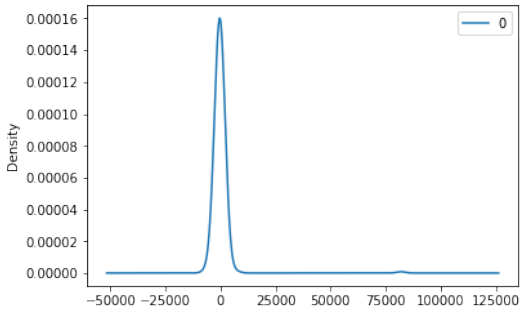


Figure 21. KDE plot for most accurate model (22827). The code used to generate the plot above can be reviewed in Appendix I below.

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	238
Model:	ARIMA(5, 1, 0)	Log Likelihood	-2391.553
Method:	css-mle	S.D. of innovations	5594.612
Date:	Wed, 21 Nov 2018	AIC	4797.105
Time:	01:33:10	BIC	4821.411
Sample:	1	HQIC	4806.901

	coef	std err	z	P>|z|	[0.025	0.975]
const	347.8992	445.693	0.781	0.436	-525.643	1221.442
ar.L1.D.y	0.0659	0.065	1.018	0.310	-0.061	0.193
ar.L2.D.y	0.0335	0.065	0.518	0.605	-0.093	0.160
ar.L3.D.y	0.0250	0.065	0.387	0.699	-0.102	0.152
ar.L4.D.y	0.0338	0.064	0.525	0.600	-0.092	0.160
ar.L5.D.y	0.0303	0.064	0.471	0.638	-0.096	0.156

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.6809	-0.0000j	1.6809	-0.0000
AR.2	0.4560	-1.9495j	2.0022	-0.2134
AR.3	0.4560	+1.9495j	2.0022	0.2134
AR.4	-1.8550	-1.2095j	2.2144	-0.4080
AR.5	-1.8550	+1.2095j	2.2144	0.4080

Figure 22. Model summary for most accurate model (22827). The code used to generate the plot above can be reviewed in Appendix I below.

The fourth most accurate model was found to be Ringe, NH (03461). The rolling prediction for successive months starting 2017-02:

1. $207,893.76/0.01
2. $207,011.27/0.008
3. $206,969.19/0.002
4. $207,273.43/0.01
5. $208,090.60/0.01
6. $209,303.86/0.01
7. $210,618.83/0.001
8. $212,134.11
9. $214,449.07
10. $216,363.91
11. $217,078.79
12. $217,393.64
13. $217,908.49
14. $218,723.33
15. $219,938.17
16. $221,153.02
17. $222,367.86
18. $224,982.70
19. $227,997.55
20. $229,712.39
21. $230,627.24
22. $230,142.08
23. $228,856.92

Note: the first seven months (test) above, provide a prediction significance against the corresponding test value.

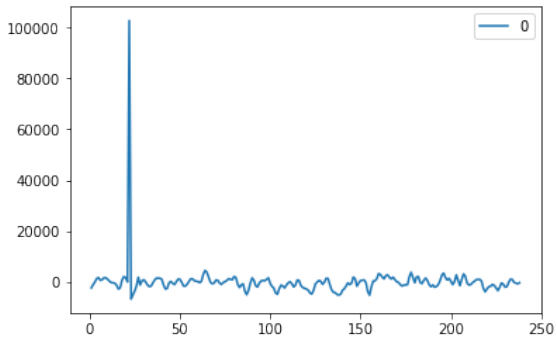


Figure 23. Residuals plot for most accurate model (03461). The code used to generate the plot above can be reviewed in Appendix I below.

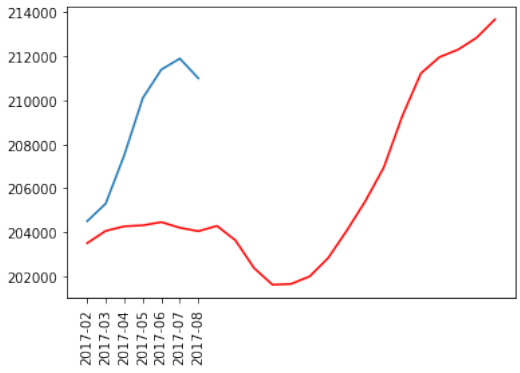


Figure 24. Most accurate model (red), against actual (blue) (03461). The code used to generate the plot above can be reviewed in Appendix I below.

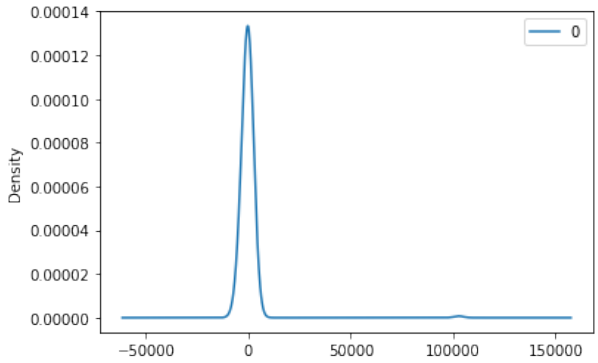


Figure 25. KDE plot for most accurate model (03461). The code used to generate the plot above can be reviewed in Appendix I below.

ARIMA Model Results

Dep. Variable:	D.y	No. Observations:	238
Model:	ARIMA(5, 1, 0)	Log Likelihood	-2442.852
Method:	css-mle	S.D. of innovations	6940.125
Date:	Wed, 21 Nov 2018	AIC	4899.705
Time:	01:33:11	BIC	4924.010
Sample:	1	HQIC	4909.500

	coef	std err	z	P>|z|	[0.025	0.975]
const	439.9160	553.140	0.795	0.427	-644.219	1524.050
ar.L1.D.y	0.0880	0.065	1.361	0.175	-0.039	0.215
ar.L2.D.y	0.0793	0.065	1.224	0.222	-0.048	0.206
ar.L3.D.y	0.0454	0.065	0.700	0.484	-0.082	0.172
ar.L4.D.y	0.0061	0.065	0.095	0.924	-0.120	0.133
ar.L5.D.y	-0.0310	0.064	-0.483	0.629	-0.157	0.095

Roots

	Real	Imaginary	Modulus	Frequency
AR.1	1.7911	-0.8555j	1.9849	-0.0709
AR.2	1.7911	+0.8555j	1.9849	0.0709
AR.3	-0.6659	-1.8824j	1.9967	-0.3041
AR.4	-0.6659	+1.8824j	1.9967	0.3041
AR.5	-2.0527	-0.0000j	2.0527	-0.5000

Figure 26. Model summary for most accurate model (03461). The code used to generate the plot above can be reviewed in Appendix I below.

When reviewing the top four performing models, it is evident that the first seven months for each model, never exceeded 3% difference between from the corresponding test value. Therefore, it is safe to assume the corresponding models are valid, and the successive 16 months are interesting.

Conclusions

In this study, Figure 1 shows amongst four Arkansas metro area, each depicting a similar trend, while Fayetteville having much greater recent value. Next, an overall ARIMA rolling model showed (Figure 6) a highly accurate model. Specifically, at each interval-step, the predicted value contained less than a 1% error. Therefore, this approach was adopted when attempting to aggregate the dataset by zipcode, then predicting successive months of prediction.

When reviewing the above results, it was found that the top four zipcodes aggregated by crime, unemployment, and by location (MD, VA, D.C., MA, NH) were three in Virginia, and one New Hampshire. Though the results are highly accurate, and succeeds to model the defined constraints, it would have been more interesting to model the most profit returning zipcode. Adjusting the necessary code to make this determination is trivial. Specifically, measuring a series of prediction with the greatest positive rolling difference could be one way to capture this information.

Overall, this study succeeded at demonstrating concepts of timeseries forecasting using ARIMA. Figures 11 – 26 visually demonstrate this. Furthermore, understanding other approaches, and algorithms could complement similar studies, by improving either the performance or modeling accuracy. However, adding an additional data source with respect to school district performance, or accessibility to public transportation by zipcode could likely improve the model.

Appendix A

Arkansas Metro Area

```
# metro areas
hot_springs = df.loc[(df['Metro'] == 'hot springs') & (df['State'] == 'ar')]
little_rock = df.loc[(df['Metro'] == 'little rock') & (df['State'] == 'ar')]
fayetteville = df.loc[(df['Metro'] == 'fayetteville') & (df['State'] == 'ar')]
searcy = df.loc[(df['Metro'] == 'searcy') & (df['State'] == 'ar')]

# timeseries plot
fig, ax = plt.subplots()
ax.plot(hot_springs[date_columns].mean(), linestyle='solid')
ax.plot(little_rock[date_columns].mean(), linestyle='solid')
ax.plot(fayetteville[date_columns].mean(), linestyle='solid')
ax.plot(searcy[date_columns].mean(), 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(('hot_springs', 'little_rock', 'fayetteville', 'searcy'))

# show overall plot
plt.show()
```

Appendix B

Overall ARIMA model descriptive statistics between 01/1997 through 01/2017:

```
# 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 + 1):test_stop].columns.tolist()

# transpose dataframe: left column data, right column value
df_train = df[train_columns].median().T
df_test = df[test_columns].median().T

# build arima model:
model = ARIMA(df_train, order=(5,1,0))
model_fit = model.fit()
print(model_fit.summary())
```

Appendix C

Residual plot for overall ARIMA model:

```
# plot residual errors
residuals = DataFrame(model_fit.resid)
residuals.plot()
plt.show()
```

Appendix D

Residual, KDE plot and descriptive for ARIMA model:

```
# plot residual errors
def residuals_plot(model_fit):
    residuals = DataFrame(model_fit.resid)
    residuals.plot()
    plt.show()

    # plot kernel density estimation
    residuals.plot(kind='kde')
    plt.show()

    # descriptibe statistics
    print(residuals.describe())

residuals_plot(model_fit)
```

Appendix D

Error rate for ARIMA model, with remaining 2017 months through 2018 predictions:

```
history = [x for x in df_train]
predictions = list()
iterations = (12-len(df_test)) + 18
for t in range(iterations):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    if t > 10:
        year = 2018
        month = (t+2) % 12
        if month == 0:
            month = 12
    else:
        year = 2017
        month = t+2
        if month == 0:
            month = 12
    print('date: {}-{:01d}'.format(year, month))
    try:
        obs = df_test[t]
        print('predicted={:03f}, expected={:03f}'.format(float(yhat), obs))
        print('prediction difference: {:03f}'.format(abs(1-float(yhat)/obs)))
        error = mean_squared_error(df_test, predictions)
        print('Test MSE: {:03f}\n\n'.format(error))
    except:
        obs = yhat
        print('predicted={:03f}'.format(float(yhat)))

    history.append(obs)
```

Appendix E

Comparison between predicted vs actual mean value:

```
# plot rolling prediction
def rolling_plot(data, predictions):
    plt.plot(data)
    plt.plot(predictions, color='red')
    plt.xticks(rotation=90)
    plt.show()

rolling_plot(df_test, predictions)
```

Appendix F

Generic arima function using rolling prediction:

```
def compute_arima(
    data=df_train,
    p=5,
    q=0,
    d=0,
    delta=(12-len(df_test)) + 18,
    alpha=0.05,
    residuals_plot=False,
    summary=False
):
    history = [x for x in data]
    predictions = list()
    model_fit = False

    try:
        model = ARIMA(difference(history, delta), order=(p,q,d))
        model_fit = model.fit(disp=0)
        print('standard fit used')
    except Exception as e:
        print('stationary differences will be used')
        print('original error: {}'.format(e))

    if not model_fit:
        for delta in range(10):
            stationary = difference(history, delta)
            stationary.index = history[1:]
            result = adfuller(stationary)
            print('stationary fit: {}, p: {}'.format(delta, result[1]))

            if (result[1] <= 0.05):
                try:
```

```
                model = ARIMA(stationary, order=(p,q,d))
                model_fit = model.fit(disp=0)
                break
            except Exception as e:
                print('bad condition {}: stationarity not adequate'.format(delta))
                print('original error: {}'.format(e))
                continue

if model_fit:
    output = model_fit.forecast(steps=delta, alpha=alpha)[0]
    if residuals_plot:
        residuals_plot(model_fit)
    if summary:
        print(model_fit.summary())

for yhat in output:
    inverted = inverse_difference(history, yhat, interval=delta)
    history.append(inverted)
    predictions.append(inverted)

print('predictions: {}'.format(predictions))

return(predictions)
```

Appendix G

Implementation of custom ARIMA rolling forecast model from Appendix F:

```
# iterate columns
results = []
for column in df_zipcode_clean:
    predictions = compute_arima(df_zipcode_clean[column], q=1)
    results.append({
        'zip_code': df_zipcode_clean[column].name,
        'predictions': predictions
    })
```

Associated helper functions for Appendix G:

```
# stationarity test
def difference(dataset, interval):
    diff = list()
    for i in range(1, len(dataset)):
        value = dataset[i] - dataset[i - interval]
        diff.append(value)
    return pd.Series(diff)

# invert differenced value
def inverse_difference(history, yhat, interval=1):
    return yhat + history[-interval]
```

Appendix G

Zipcode trend for select zipcodes:

```
# iterate columns
results = []
for column in df_zipcode_clean:
    predictions = compute_arima(df_zipcode_clean[column], p=5)
    results.append({
        'zip_code': df_zipcode_clean[column].name,
        'predictions': predictions
    })
```

Appendix I

Top four zipcodes:

```
# best 4 models
for model in sorted_results[:4]:
    # get data
    zipcode = model['zipcode']
    data_zipcode = df_zipcode_clean[[zipcode]]
    data_train = data_zipcode.T[train_columns].T

    # compute_arima
    predictions = compute_arima(
        data_train.iloc[:,0],
        q=1,
        rplot=True,
        summary=True
    )

    # plot predictions
    rolling_plot(data_zipcode.T[test_columns].T, predictions)
```