1 Zipcode Modeling and 5 year Forecasts

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import statsmodels.api as sm
import pandas.tseries
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
import itertools
from matplotlib.pylab import rcParams
from statsmodels.graphics.tsaplots import plot_pacf
from statsmodels.graphics.tsaplots import plot_acf
from statsmodels.tsa.stattools import adfuller
from  matplotlib.ticker import FuncFormatter

df = pd.read_csv('zillow_data.csv')
```

executed in 3.13s, finished 15:55:30 2021-03-22

In [2]:

```
def forecast_accuracy(forecast, actual):
    mape = np.mean(np.abs(forecast - actual)/np.abs(actual))  # MAPE
    rmse = np.mean((forecast - actual)**2)**.5  # RMSE
    corr = np.corrcoef(forecast, actual)[0,1]   # corr

    return({'mape':mape, 'rmse':rmse,
            'corr':corr})
```

executed in 14ms, finished 15:55:30 2021-03-22

In [3]:

```
topzips = [32809,98203,80012,76131,49507]
```

executed in 14ms, finished 15:55:31 2021-03-22

In [4]:

```
final_zips = df.loc[df['RegionName'].isin(topzips)].drop(columns=['RegionID','City', 'CountyName', 'Metro',
                                              axis=1).groupby('RegionName').mean()
final_zips = pd.DataFrame(final_zips.reset_index())
columns = list(final_zips.T.iloc[0])
final_zips = final_zips.T
final_zips.columns = columns
final_zips.columns = final_zips.columns.astype(np.int).astype('str')
final_zips = final_zips[1:]
final_zips = final_zips.set_index(pd.to_datetime(final_zips.index))
final_zips = final_zips.applymap(lambda x: round(np.float(x),2))
final_zips
```

executed in 30ms, finished 15:55:31 2021-03-22

Out[4]:

	32809	49507	76131	80012	98203
1996-04-01	71700.0	49700.0	117400.0	111900.0	136800.0
1996-05-01	71700.0	51000.0	117300.0	112000.0	136500.0
1996-06-01	71800.0	52300.0	117300.0	112200.0	136300.0
1996-07-01	71800.0	53500.0	117300.0	112300.0	136300.0
1996-08-01	71800.0	54600.0	117600.0	112500.0	136300.0
...	...	...	...	...	...
2017-12-01	171400.0	106600.0	195800.0	307400.0	380100.0
2018-01-01	174800.0	107800.0	197100.0	311300.0	384300.0
2018-02-01	177800.0	108900.0	198700.0	314800.0	388900.0
2018-03-01	180900.0	110200.0	200600.0	318600.0	395700.0
2018-04-01	183400.0	111200.0	201900.0	321100.0	401300.0

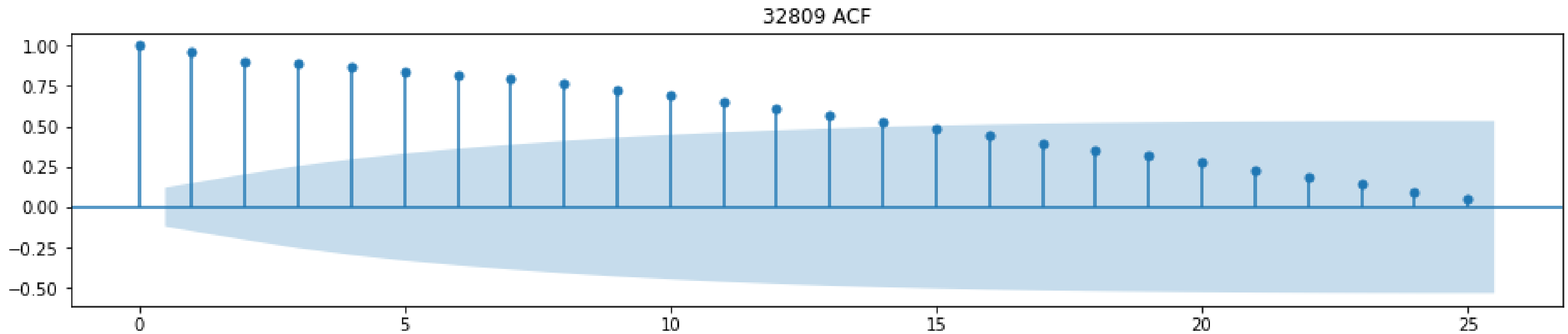
265 rows × 5 columns

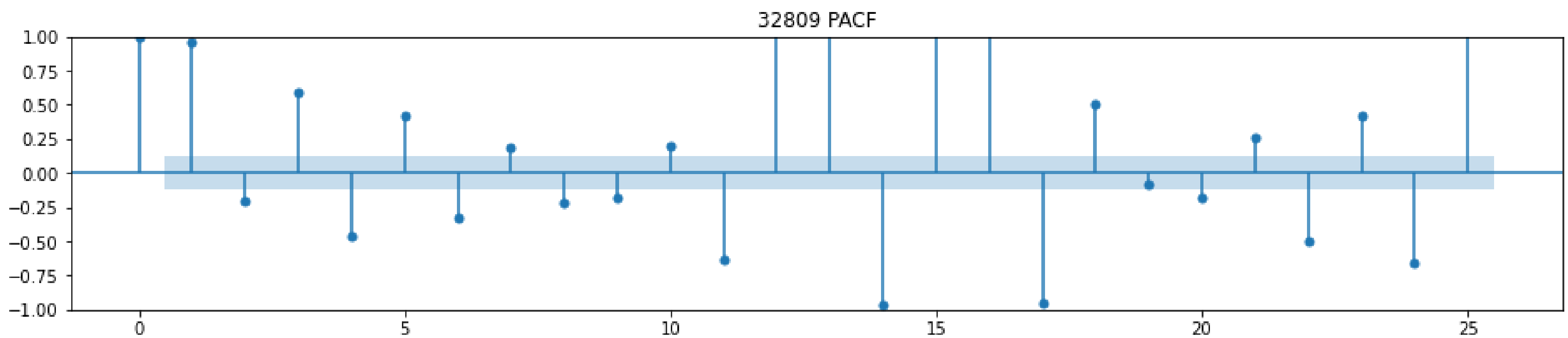
1.0.1 PACF and ACF plots for top 5 zipcodes

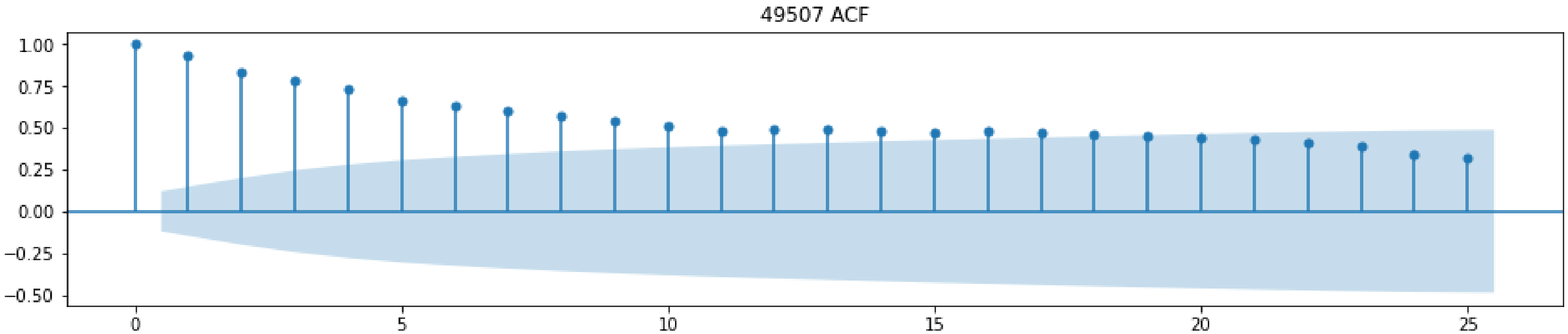
In [5]:

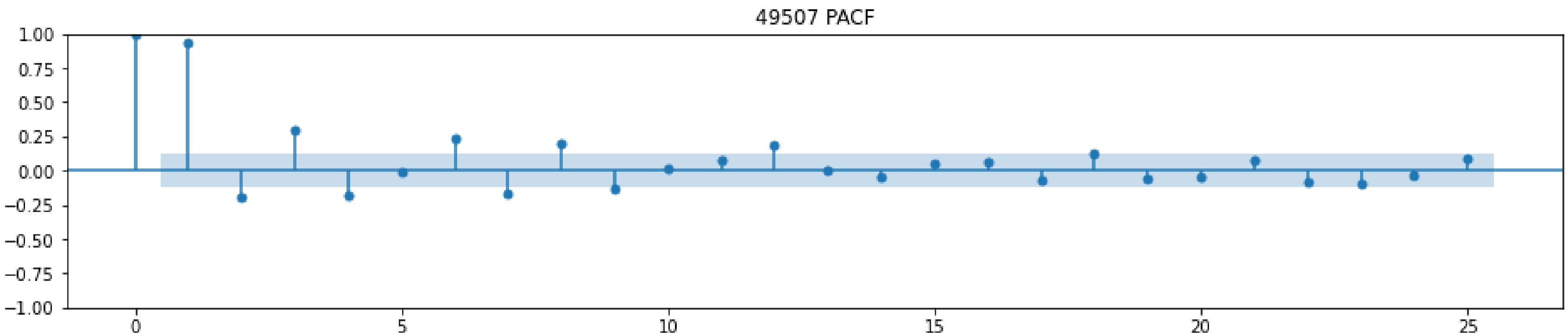
```
for idx,x in enumerate(final_zips.columns):
    y = final_zips[x].diff(periods=1).dropna()
    fig,ax = plt.subplots(figsize=(16,3))
    plot_acf(y,ax=ax,lags=25)
    plt.title('{} ACF'.format(x))
    fig1,ax1 = plt.subplots(figsize=(16,3))
    plot_pacf(y,ax=ax1,lags=25)
    plt.title('{} PACF'.format(x))
    plt.ylim(-1,1)
```

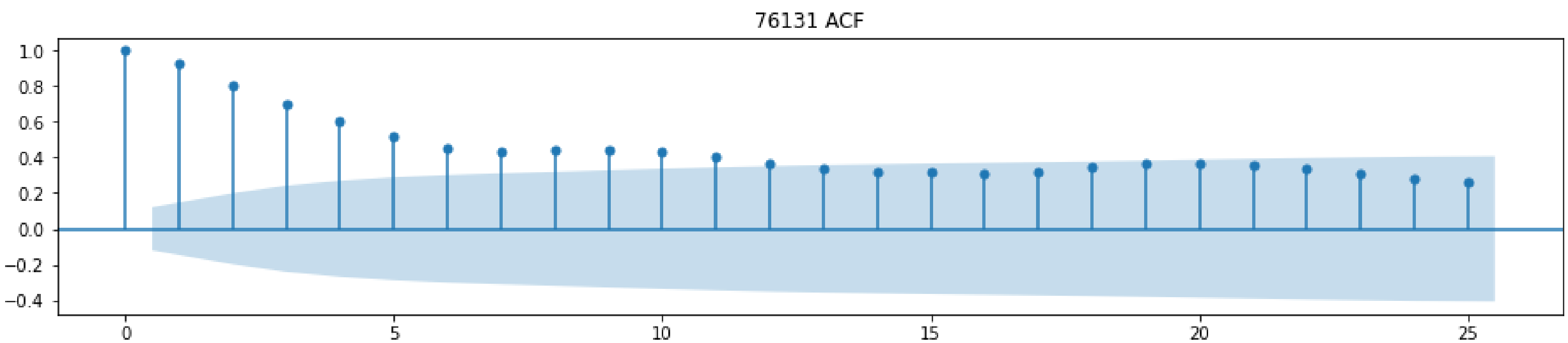
executed in 1.59s, finished 15:55:33 2021-03-22

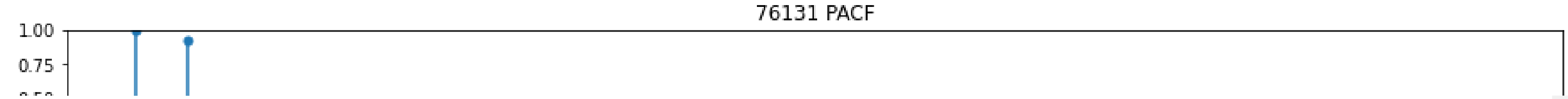


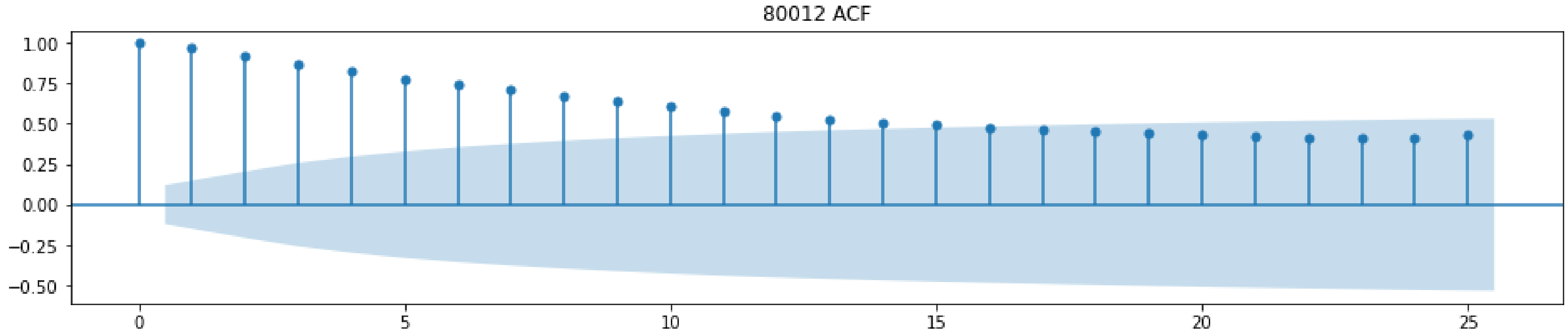


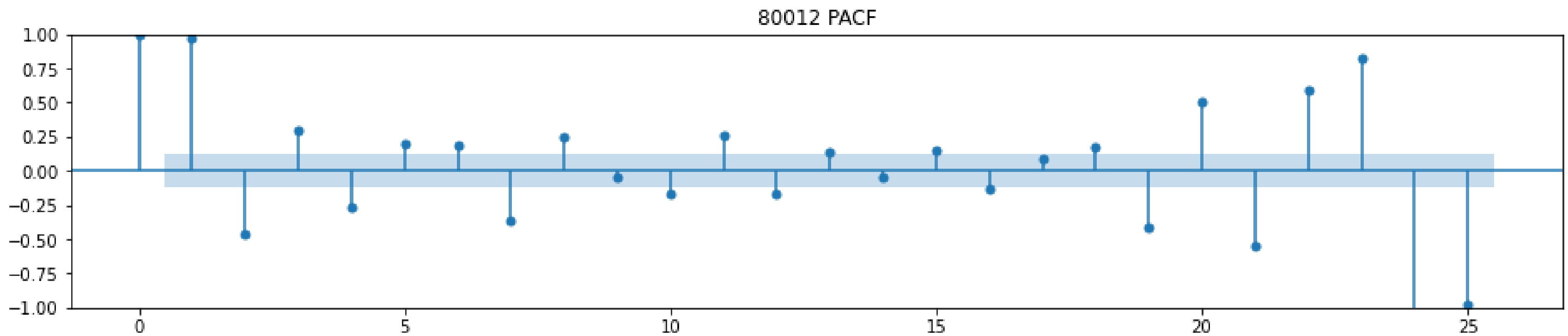

49507 ACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
5
10
15
20
25

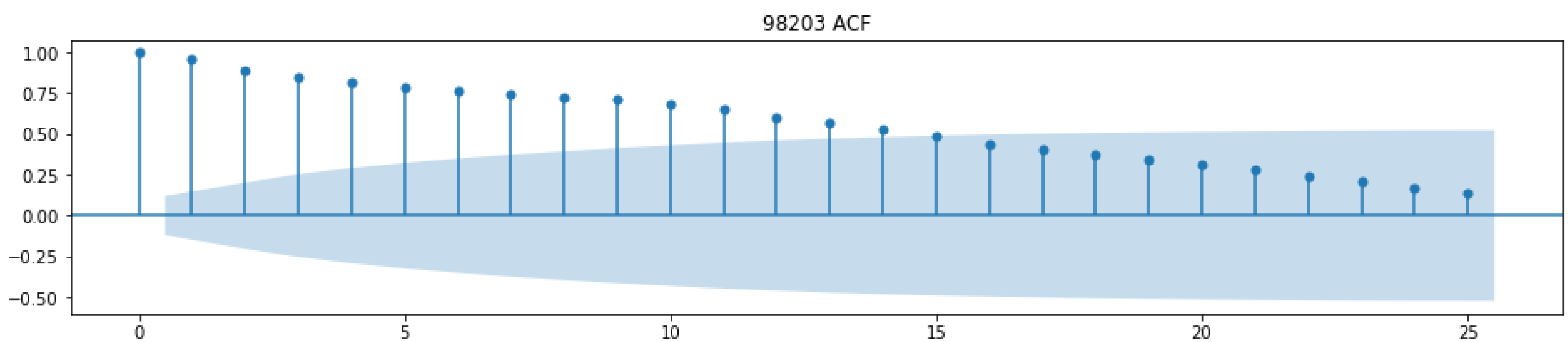

49507 PACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25

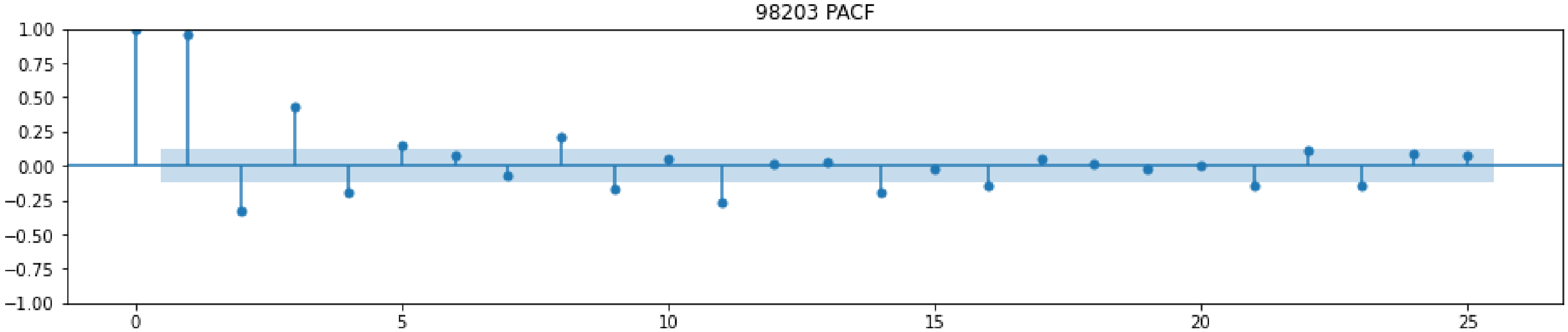

76131 ACF
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
0
5
10
15
20
25


76131 PACF
1.00
0.75


80012 ACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
5
10
15
20
25


80012 PACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0
5
10
15
20
25


98203 ACF
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
0
5
10
15
20
25



1.1 Zipcode 80012 Model and Forecast - Aurora, CO

In [6]:

```
mod_80012 = sm.tsa.statespace.SARIMAX(final_zips['80012'],
                                      order=(4,1,1),
                                      seasonal_order=(3,2,0,12),
                                      enforce_stationarity=False,
                                      enforce_invertibility=False,
                                         simple_differencing=False)

output_80012 = mod_80012.fit()
```

executed in 10.9s, finished 15:55:44 2021-03-22

In [7]:

```
output_80012.summary()
```

executed in 29ms, finished 15:55:44 2021-03-22

Out[7]:

SARIMAX Results

Dep. Variable:	80012	**No. Observations:**	265
Model:	SARIMAX(4, 1, 1)x(3, 2, [], 12)	**Log Likelihood**	-1533.833
Date:	Mon, 22 Mar 2021	**AIC**	3085.666
Time:	15:55:44	**BIC**	3115.351
Sample:	04-01-1996	**HQIC**	3097.679
	- 04-01-2018		
Covariance Type:	opg		

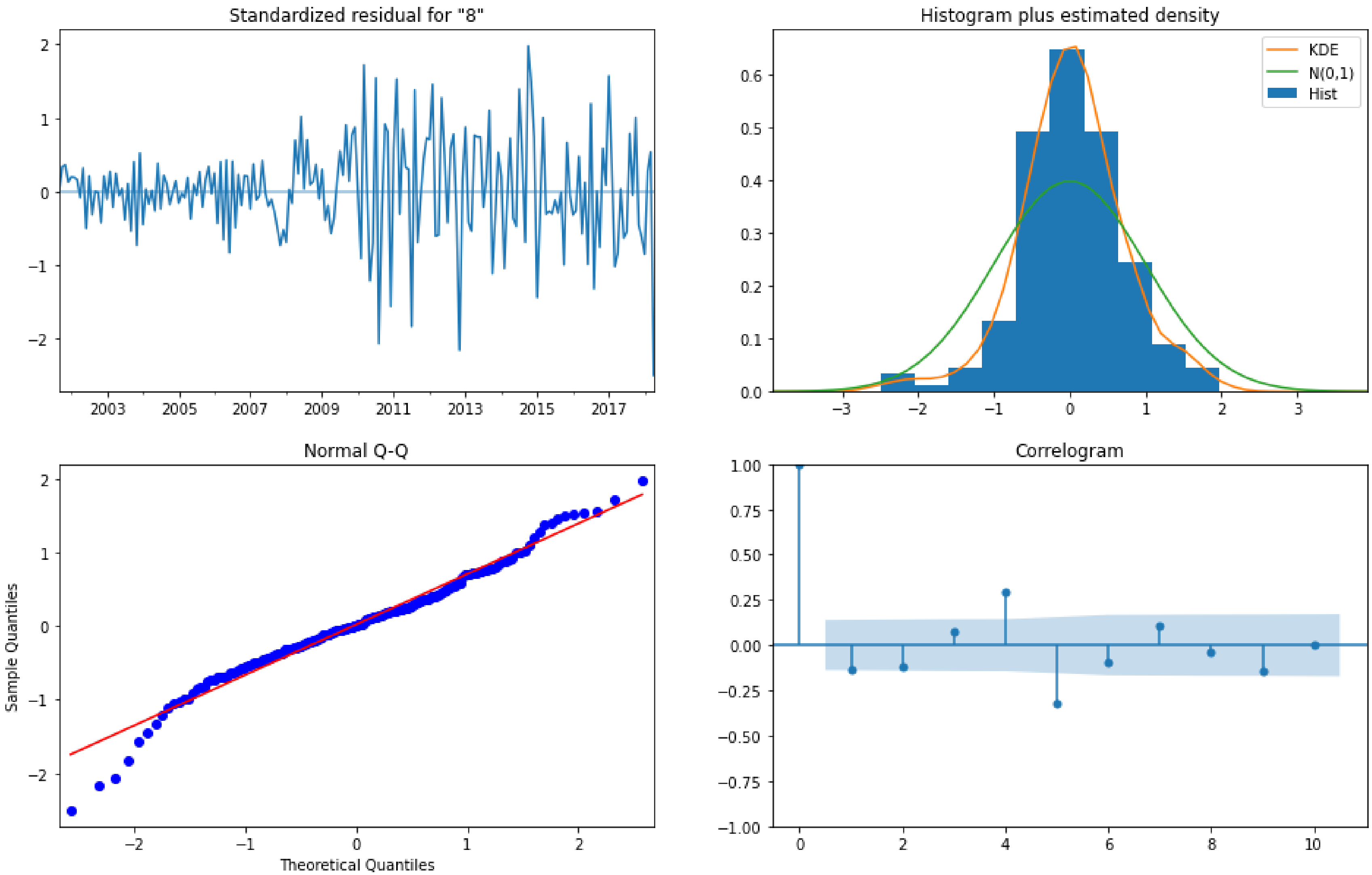
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	2.7195	0.155	17.503	0.000	2.415	3.024
ar.L2	-3.0269	0.331	-9.138	0.000	-3.676	-2.378
ar.L3	1.8294	0.316	5.788	0.000	1.210	2.449
ar.L4	-0.5332	0.132	-4.036	0.000	-0.792	-0.274
ma.L1	-0.9200	0.116	-7.899	0.000	-1.148	-0.692
ar.S.L12	-1.4986	0.115	-13.055	0.000	-1.724	-1.274
ar.S.L24	-1.2597	0.175	-7.213	0.000	-1.602	-0.917
ar.S.L36	-0.5783	0.146	-3.958	0.000	-0.865	-0.292
sigma2	4.559e+05	6.87e+04	6.634	0.000	3.21e+05	5.91e+05

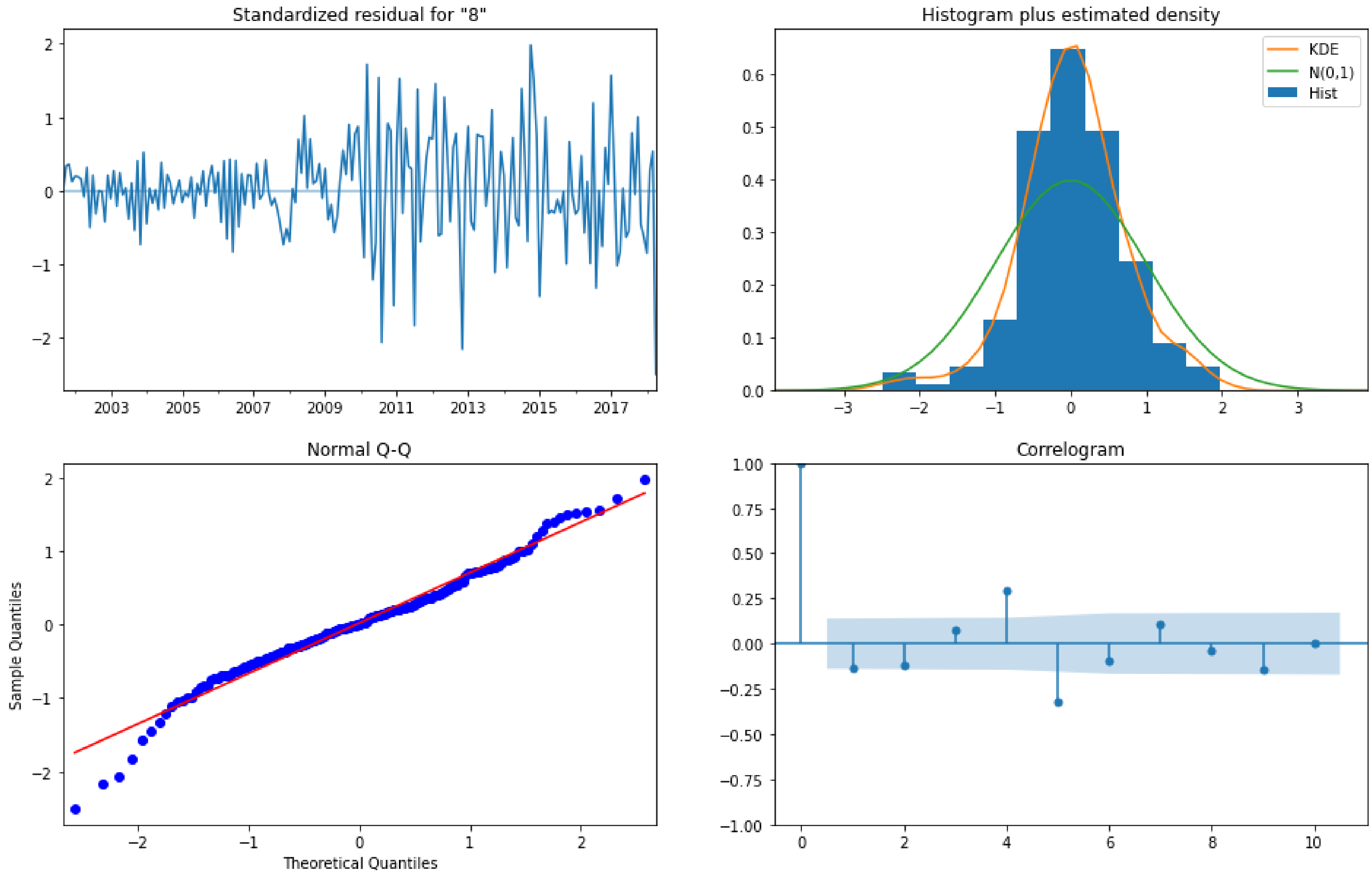
Ljung-Box (L1) (Q):	3.81	**Jarque-Bera (JB):**	18.99
Prob(Q):	0.05	**Prob(JB):**	0.00
Heteroskedasticity (H):	7.50	**Skew:**	-0.27
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.41

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 5.43e+14. Standard errors may be unstable.

In [8]:
```
output_80012.plot_diagnostics(figsize=(16,10))
```
executed in 1.10s, finished 15:55:46 2021-03-22

Out[8]:





Reviewing our plot diagnostics for Aurora:

Our top left plot (Standardized Residuals), show stationarity as they reflect that of a white noise model

Our top right plot (KDE vs Standard Normal distribution) shows our model meets the normal distribution of residuals as both lines have similar bell curves. Notably, our KDE has a smaller std and due to such has a higher peak in the center

Our bottom right qq plot, also shows our model residuals are normally distributed as the points lie across the 45 degree line, but have some values on the lower tails that are not on the line

Lastly, our residuals for the most part are not correlated to most of the prior lags except for the 4th and 5th. Otherwise clean.

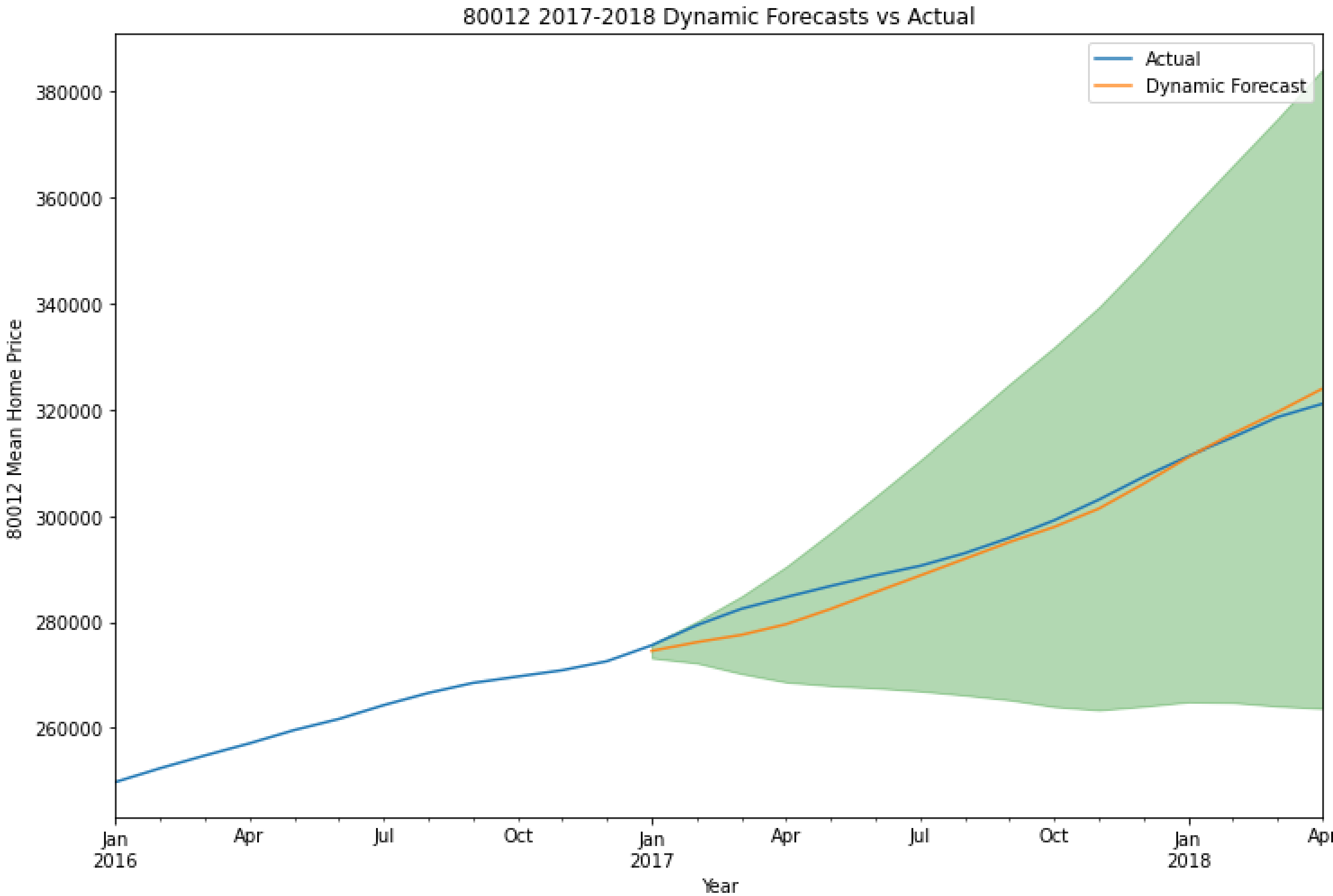
In [9]:
```
pred_80012 = output_80012.get_prediction(start=pd.to_datetime('2017'),dynamic=True)
pred_conf_80012 = pred_80012.conf_int()
```
executed in 13ms, finished 15:55:47 2021-03-22

In [10]:

```
# Plot real vs predicted values along with confidence interval
rcParams['figure.figsize'] = 12,8
ax = final_zips['2016':]['80012'].plot(kind='line',label='Actual')
pred_80012.predicted_mean.plot(kind='line',ax=ax,label='Dynamic Forecast',alpha=.9)

ax.fill_between(pred_conf_80012.index,
               pred_conf_80012.iloc[:,0],
               pred_conf_80012.iloc[:,1], color='g', alpha=.3)
# Set axes labels
plt.xlabel('Year')
plt.ylabel('80012 Mean Home Price')
plt.title('80012 2017-2018 Dynamic Forecasts vs Actual')
plt.legend()
```

executed in 340ms, finished 15:55:47 2021-03-22

Out[10]: <matplotlib.legend.Legend at 0x1ac977eb790>



In [11]:

```
forecast_80012 = pred_80012.predicted_mean
truth_80012 = final_zips['2017':]['80012']

# Compute the mean square error
mse_80012 = ((forecast_80012 -truth_80012)**2).mean()
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse_80012), 2)))
```

executed in 14ms, finished 15:55:48 2021-03-22

```
The Root Mean Squared Error of our forecasts is 2637.4
```

In [12]:
```
1- (np.sqrt(mse_80012) /truth_80012.mean())
```
executed in 14ms, finished 15:55:49 2021-03-22

Out[12]:
```
0.9911213511250386
```

In [13]:
```
forecast_accuracy(forecast_80012,truth_80012)
```
executed in 25ms, finished 15:55:49 2021-03-22

Out[13]:
```
{'mape': 0.00737873556040059,
 'rmse': 2637.4026483072594,
 'corr': 0.996720979722035}
```

In [14]:
```
1 - forecast_accuracy(forecast_80012,truth_80012)['mape']
```
executed in 14ms, finished 15:55:50 2021-03-22

Out[14]:
```
0.9926212644395994
```

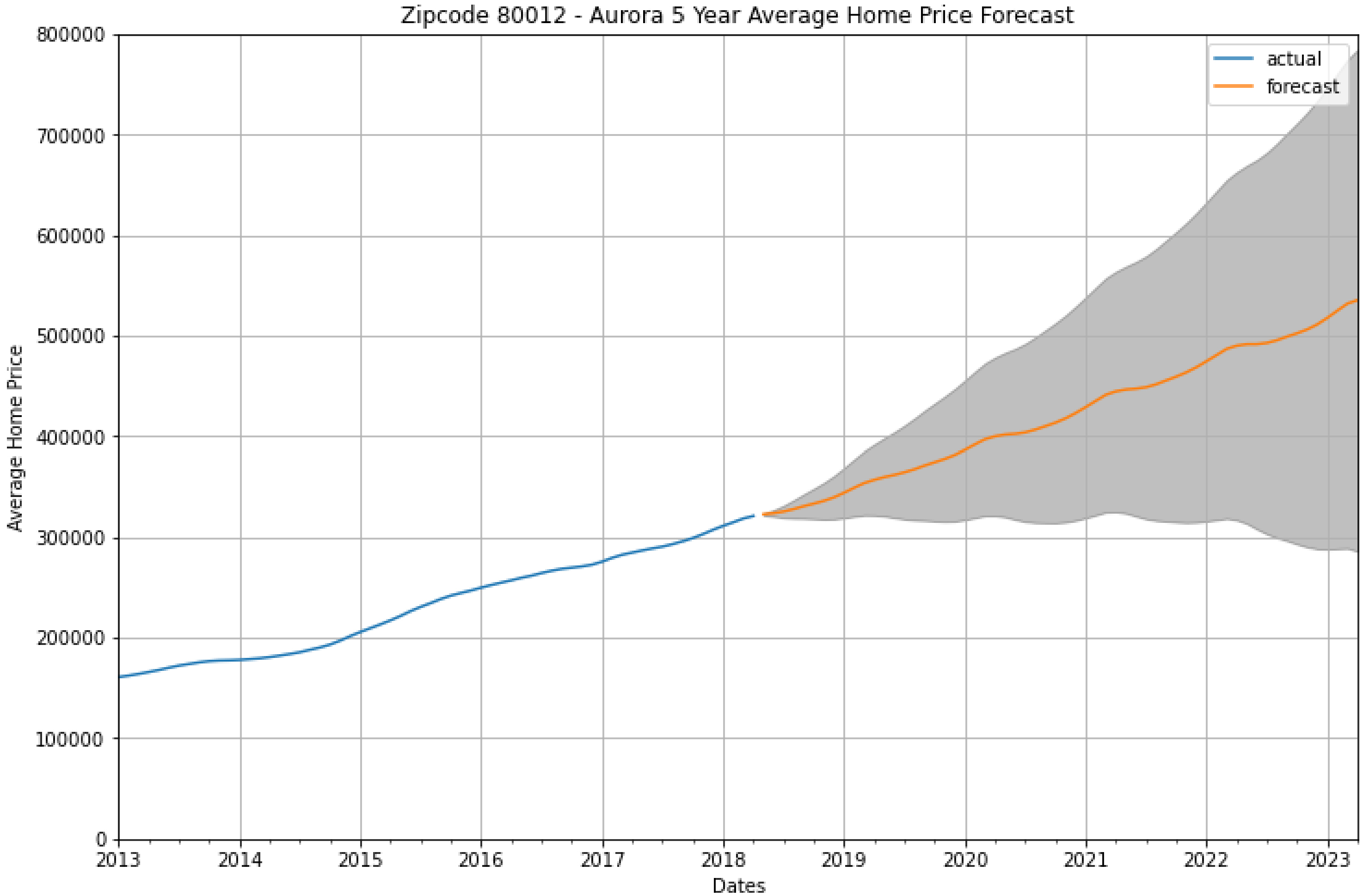
Our model predictions are 99% accurate for its forecast of the 2017-2018 values as per mape score

In [15]:
```
# Get forecast 60 steps ahead in future / 5 years
prediction_80012 = output_80012.get_forecast(steps=60)

# Get confidence intervals of forecasts
pred_conf_80012 = prediction_80012.conf_int(alpha=.10)
```
executed in 29ms, finished 15:55:50 2021-03-22

In [16]:

```
rcParams['figure.figsize'] = 12,8
ax = final_zips['2013':]['80012'].plot(label='actual')
prediction_80012.predicted_mean.plot(label='forecast')
ax.fill_between(pred_conf_80012.index,
                pred_conf_80012.iloc[:,0],
                pred_conf_80012.iloc[:,1],color='k',alpha=.25)
ax.set_xlabel('Dates')
ax.set_ylabel('Average Home Price')
plt.title('Zipcode 80012 - Aurora 5 Year Average Home Price Forecast')
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: int(x)))
plt.yticks(np.linspace(0,800000,num=9,dtype=int))
plt.ylim(0,800000)
plt.legend()
plt.grid(which='major')
plt.show()
```

executed in 370ms, finished 15:55:50 2021-03-22



Preliminarily we see that Aurora could provide returns up to 80% over the period.

1.2 Zipcode 32809 Model and Forecast - Sky Lake, Florida

In [17]:
```
mod_32809 = sm.tsa.statespace.SARIMAX(final_zips["32809"],
                                          order=(4,2,0),
                                          seasonal_order=(4,2,1,12),
                                          enforce_stationarity=False,
                                          enforce_invertibility=False)
output_32809 = mod_32809.fit()

```

executed in 12.5s, finished 15:56:04 2021-03-22

In [18]:
```
output_32809.summary()
```
executed in 29ms, finished 15:56:05 2021-03-22

Out[18]:

SARIMAX Results

Dep. Variable:	32809	**No. Observations:**	265
Model:	SARIMAX(4, 2, 0)x(4, 2, [1], 12)	**Log Likelihood**	-1482.188
Date:	Mon, 22 Mar 2021	**AIC**	2984.376
Time:	15:56:05	**BIC**	3016.687
Sample:	04-01-1996	**HQIC**	2997.469
	- 04-01-2018		
Covariance Type:	opg		

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5314	0.040	13.311	0.000	0.453	0.610
ar.L2	-0.7798	0.043	-18.019	0.000	-0.865	-0.695
ar.L3	0.4179	0.055	7.649	0.000	0.311	0.525
ar.L4	-0.1989	0.047	-4.197	0.000	-0.292	-0.106
ar.S.L12	-0.5648	0.056	-10.126	0.000	-0.674	-0.455
ar.S.L24	-0.2685	0.067	-4.017	0.000	-0.399	-0.137
ar.S.L36	-0.0988	0.053	-1.850	0.064	-0.204	0.006
ar.S.L48	0.1788	0.047	3.798	0.000	0.087	0.271
ma.S.L12	-1.0001	0.075	-13.401	0.000	-1.146	-0.854
sigma2	3.871e+05	1.93e-07	2.01e+12	0.000	3.87e+05	3.87e+05

Ljung-Box (L1) (Q):	0.03	**Jarque-Bera (JB):**	62.66
Prob(Q):	0.87	**Prob(JB):**	0.00
Heteroskedasticity (H):	3.57	**Skew:**	0.22

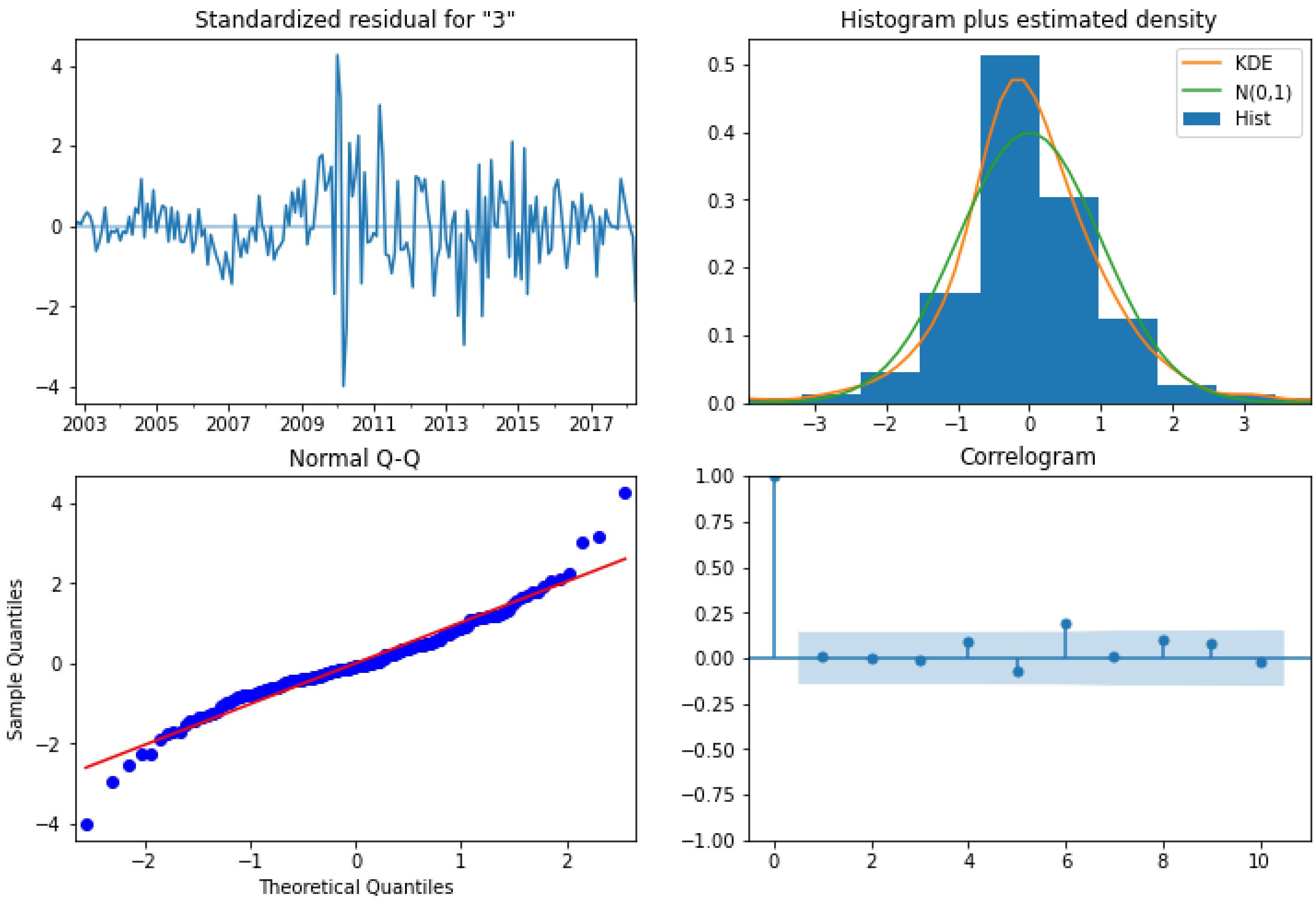
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.80

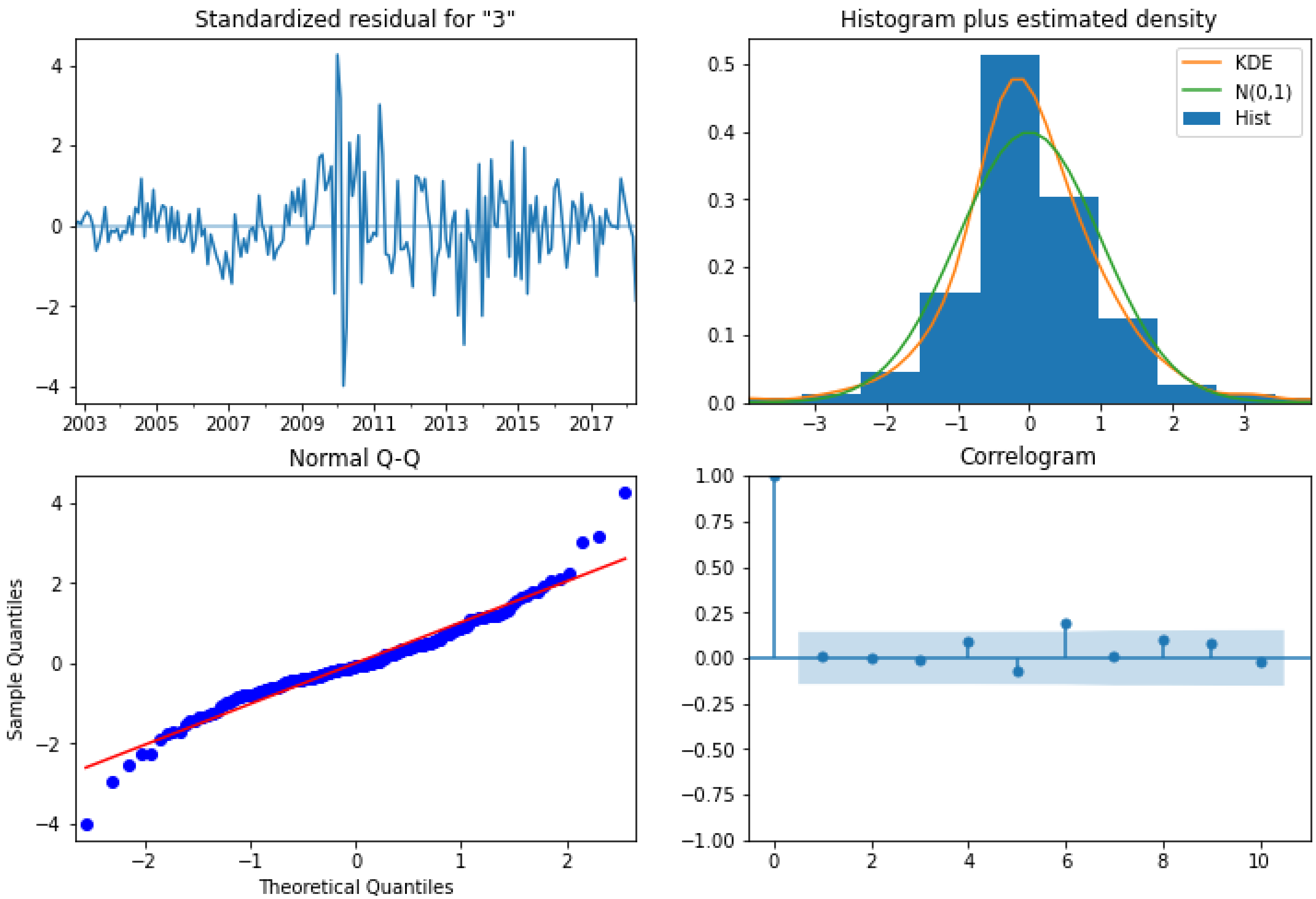
Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.87e+27. Standard errors may be unstable.

Note all of our coefficients are statistically significant at the .05 level except for the 3rd seasonal AR lag although it is close enough to leave in our model at .064

In [19]:
```
output_32809.plot_diagnostics()
```
executed in 1.18s, finished 15:56:07 2021-03-22

Out[19]:





Checking our diagnostic table to see if our residuals from our model meet our assumptions of stationarity and normality.

QQ plot bottom left indicates residuals are for the most part normally distributed except slightly off at the tails Our Standardized residuals in the top left plot represent that of a white noise model so passes the stationarity test Our top right plot, model KDE plot vs a N(0,1) normally distributed plot with mean 0 and std 1 are closely aligned although our KDE has notably slight positive skew Lastly, the correlogram show that our residuals have low correlation with lagged verisons of itself so all boxes here are checked for our model for Sky Lake

In [20]:

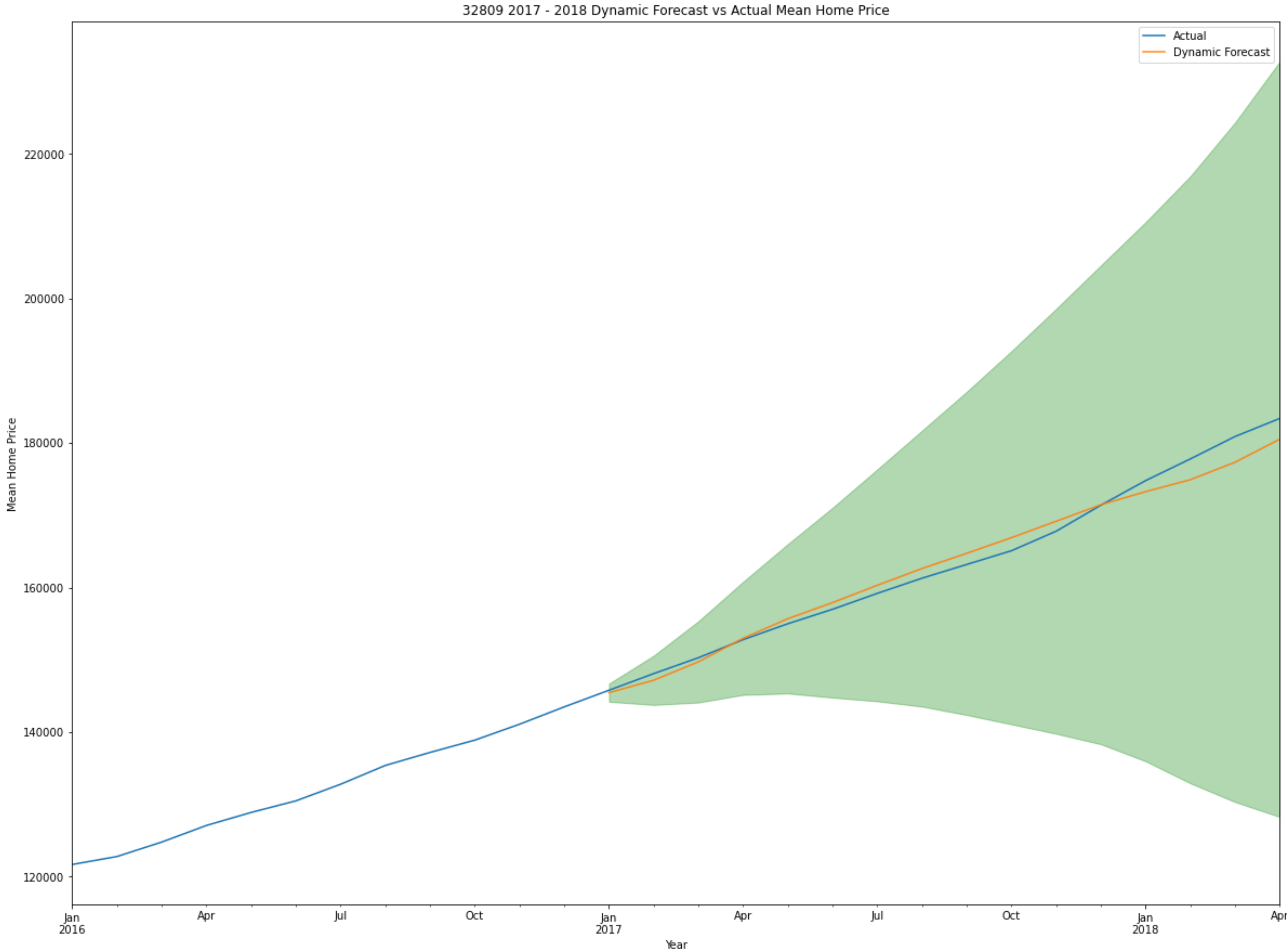
```
pred_32809 = output_32809.get_prediction(start=pd.to_datetime('2017'),dynamic=True)
pred_conf_32809 = pred_32809.conf_int()
```

executed in 14ms, finished 15:56:07 2021-03-22

In [21]:
```
# Plot real vs predicted values along with confidence interval
rcParams['figure.figsize'] = 20,15
ax = final_zips['2016':]['32809'].plot(kind='line',label='Actual')
pred_32809.predicted_mean.plot(kind='line',ax=ax,label='Dynamic Forecast',alpha=.9)

# Plot observed values

# Plot predicted values

# Plot the range for confidence intervals
ax.fill_between(pred_conf_32809.index,
                pred_conf_32809.iloc[:,0],
                pred_conf_32809.iloc[:,1], color='g', alpha=.3)
# Set axes labelsf
ax.set_xlabel('Year')
ax.set_ylabel('Mean Home Price')
plt.title("32809 2017 - 2018 Dynamic Forecast vs Actual Mean Home Price")
ax.legend()
```

executed in 417ms, finished 15:56:07 2021-03-22

Out[21]:
```
<matplotlib.legend.Legend at 0x1acaa336340>
```


32809 2017 - 2018 Dynamic Forecast vs Actual Mean Home Price
Actual
Dynamic Forecast
Mean Home Price
220000
200000
180000
160000
140000
120000
Jan 2016
Apr
Jul
Oct
Jan 2017
Apr
Jul
Oct
Jan 2018
Apr
Year

In [22]:
```
forecast_32809 = pred_32809.predicted_mean
truth_32809 = final_zips['2017':]['32809']

# Compute the mean square error
mse_32809 = ((forecast_32809 -truth_32809)**2).mean()
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse_32809), 2)))
```
executed in 14ms, finished 15:56:08 2021-03-22

```
The Root Mean Squared Error of our forecasts is 1671.57
```

In [23]:
```
forecast_accuracy(forecast_32809, truth_32809)
```
executed in 14ms, finished 15:56:09 2021-03-22

Out[23]:
```
{'mape': 0.008020124334683389,
 'rmse': 1671.5730130544912,
 'corr': 0.9913514068139596}
```

In [24]:
```
1 - forecast_accuracy(forecast_32809, truth_32809)['mape']

```
executed in 13ms, finished 15:56:10 2021-03-22

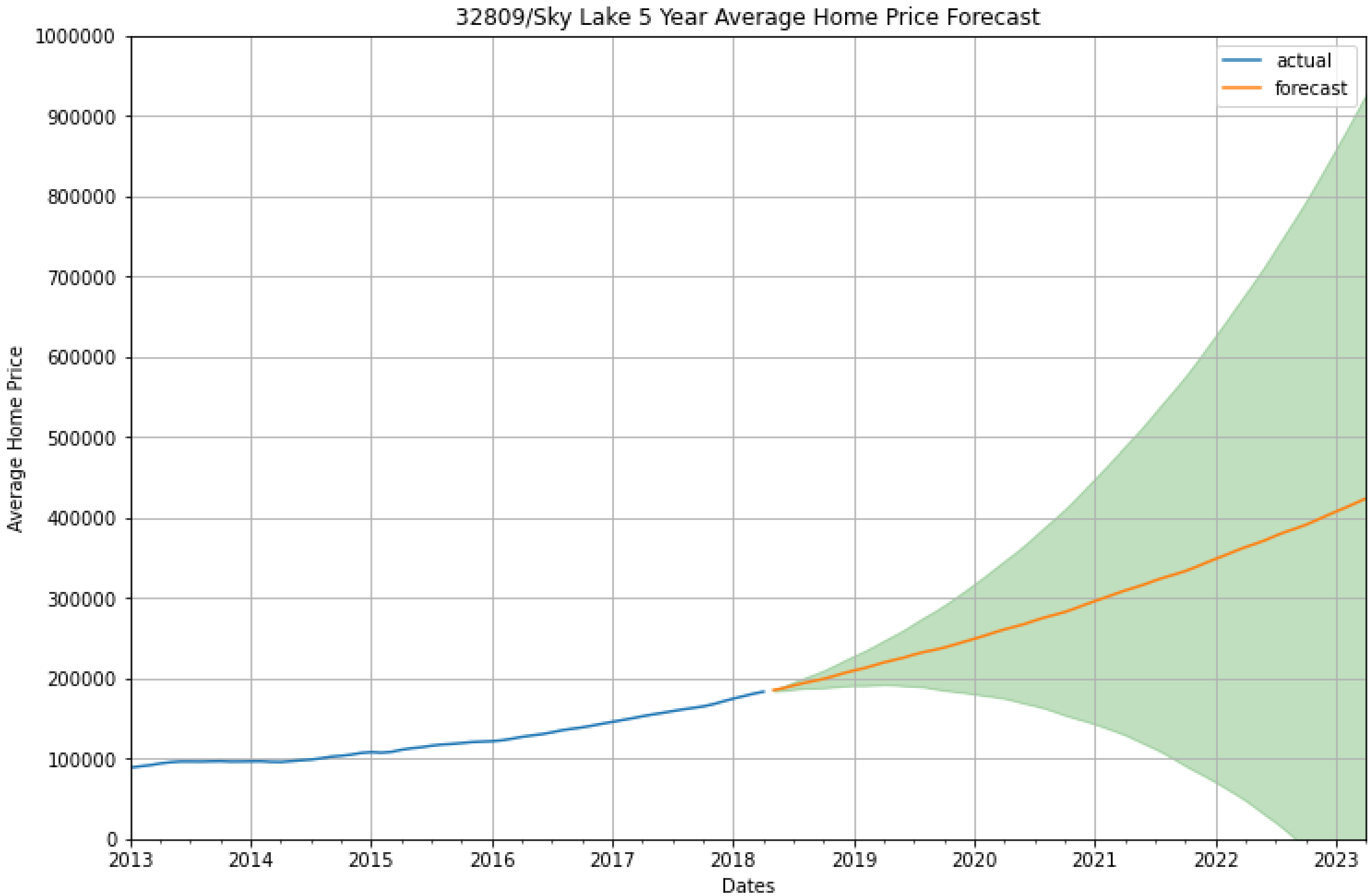
Out[24]:
```
0.9919798756653166
```

We can see here our model is accurate at predicting at the 99% level for a one year forecast. Notably this may not hold true for a 5 year period

In [25]:
```
# Get forecast 500 steps ahead in future
prediction_32809 = output_32809.get_forecast(steps=60)

# Get confidence intervals of forecasts
pred_conf_32809 = prediction_32809.conf_int(alpha=.1)
```
executed in 27ms, finished 15:56:11 2021-03-22

In [26]:
```
rcParams['figure.figsize'] = 12,8
ax = final_zips['2013':]['32809'].plot(label='actual')
prediction_32809.predicted_mean.plot(label='forecast')
ax.fill_between(pred_conf_32809.index,
               pred_conf_32809.iloc[:,0],
               pred_conf_32809.iloc[:,1],color='g',alpha=.25)
ax.set_xlabel('Dates')
ax.set_ylabel('Average Home Price')
plt.title('32809/Sky Lake 5 Year Average Home Price Forecast')
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: int(x)))
plt.yticks(np.linspace(0,1000000,num=11,dtype=int))
plt.ylim(0,1000000)
plt.legend()
plt.grid(which='major')
plt.show()
```

executed in 433ms, finished 15:56:11 2021-03-22



1.3 Zipcode 76131 Model and Forecast - Fort Worth, Texas

In [27]:
```
mod_76131 = sm.tsa.statespace.SARIMAX(final_zips['76131'],
                                            order=(0,2,3),
                                            seasonal_order=(1,2,3,12),
                                            enforce_stationarity=False,
                                            enforce_invertibility=False,
                                           simple_differencing=False)

output_76131 = mod_76131.fit()
```
executed in 9.43s, finished 15:56:22 2021-03-22

In [28]:
```
output_76131.summary()
```

executed in 27ms, finished 15:56:22 2021-03-22

Out[28]:

SARIMAX Results

Dep. Variable:	76131	**No. Observations:**	265
Model:	SARIMAX(0, 2, 3)x(1, 2, 3, 12)	**Log Likelihood**	-1424.531
Date:	Mon, 22 Mar 2021	**AIC**	2865.061
Time:	15:56:22	**BIC**	2891.408
Sample:	04-01-1996	**HQIC**	2875.724
	- 04-01-2018		
Covariance Type:	opg		

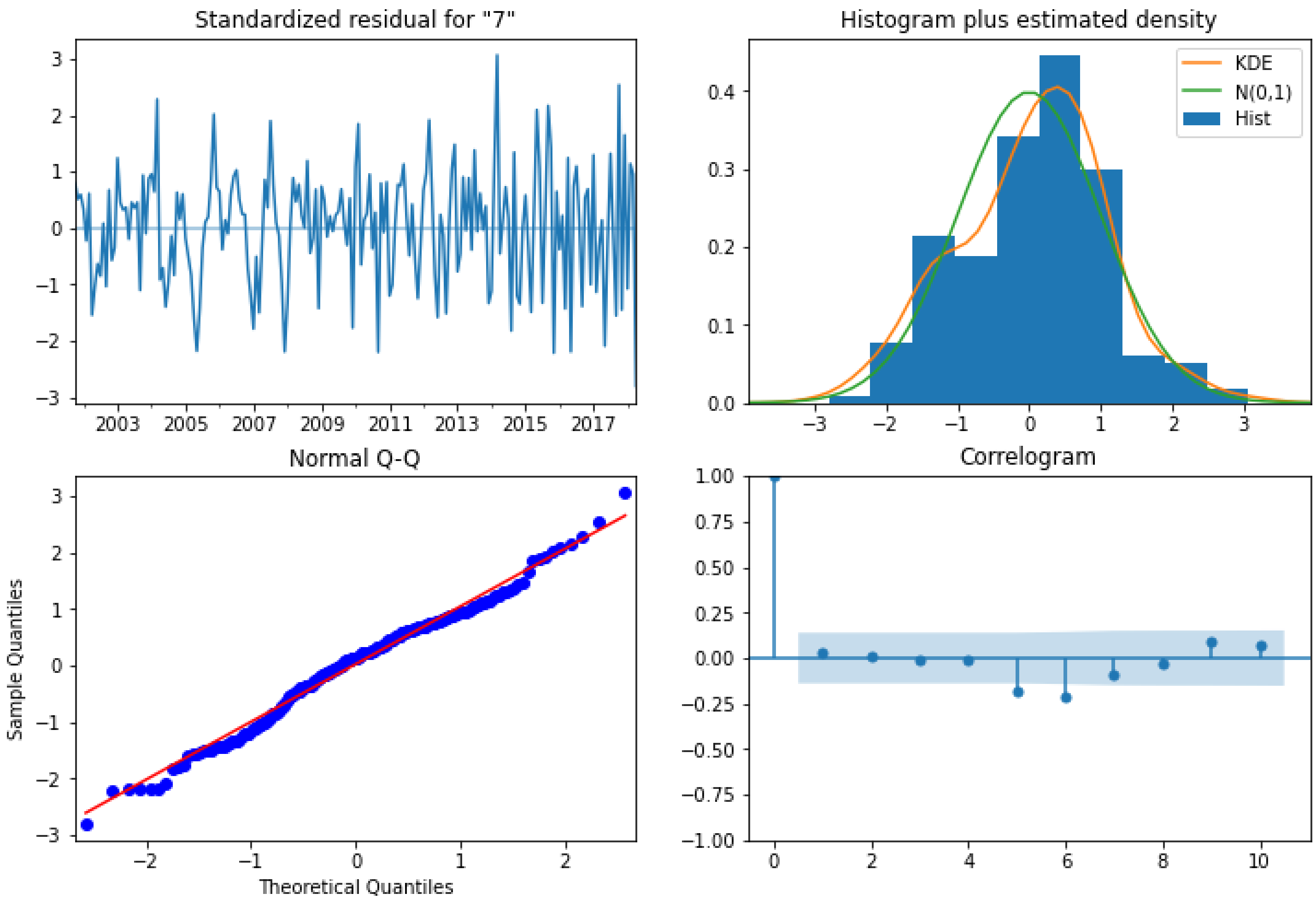
	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
ma.L1	0.3893	0.067	5.812	0.000	0.258	0.521
ma.L2	-0.1521	0.075	-2.031	0.042	-0.299	-0.005
ma.L3	-0.2597	0.066	-3.938	0.000	-0.389	-0.130
ar.S.L12	-0.1695	0.068	-2.490	0.013	-0.303	-0.036
ma.S.L12	-1.7587	0.132	-13.330	0.000	-2.017	-1.500
ma.S.L24	0.5991	0.175	3.417	0.001	0.255	0.943
ma.S.L36	0.2034	0.083	2.455	0.014	0.041	0.366
sigma2	7.027e+04	3.68e-06	1.91e+10	0.000	7.03e+04	7.03e+04

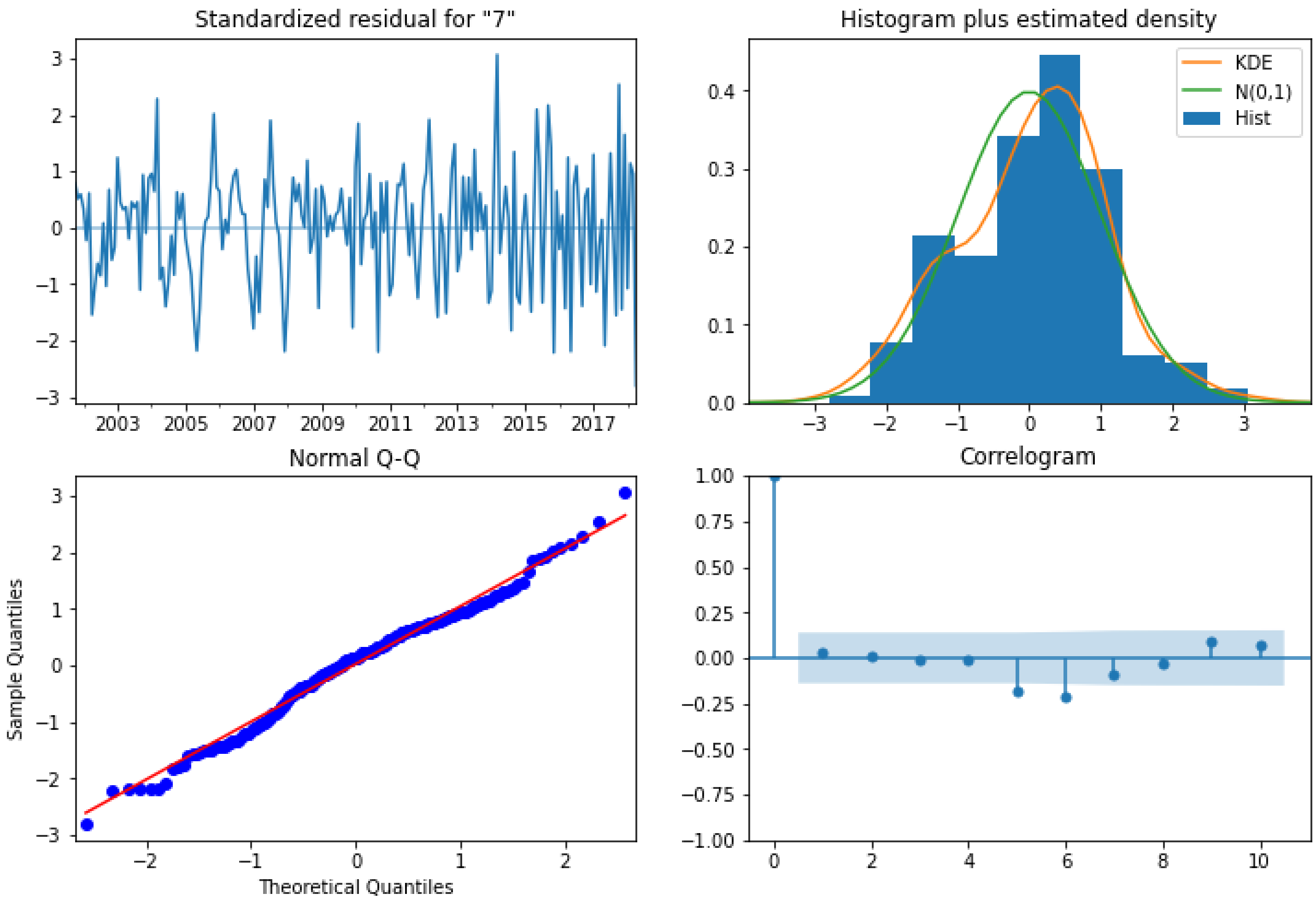
Ljung-Box (L1) (Q):	0.14	**Jarque-Bera (JB):**	0.79
Prob(Q):	0.71	**Prob(JB):**	0.67
Heteroskedasticity (H):	1.92	**Skew:**	-0.15
Prob(H) (two-sided):	0.01	**Kurtosis:**	2.97

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 5e+25. Standard errors may be unstable.

In [29]:
```
output_76131.plot_diagnostics()
```
executed in 1.11s, finished 15:56:25 2021-03-22

Out[29]:





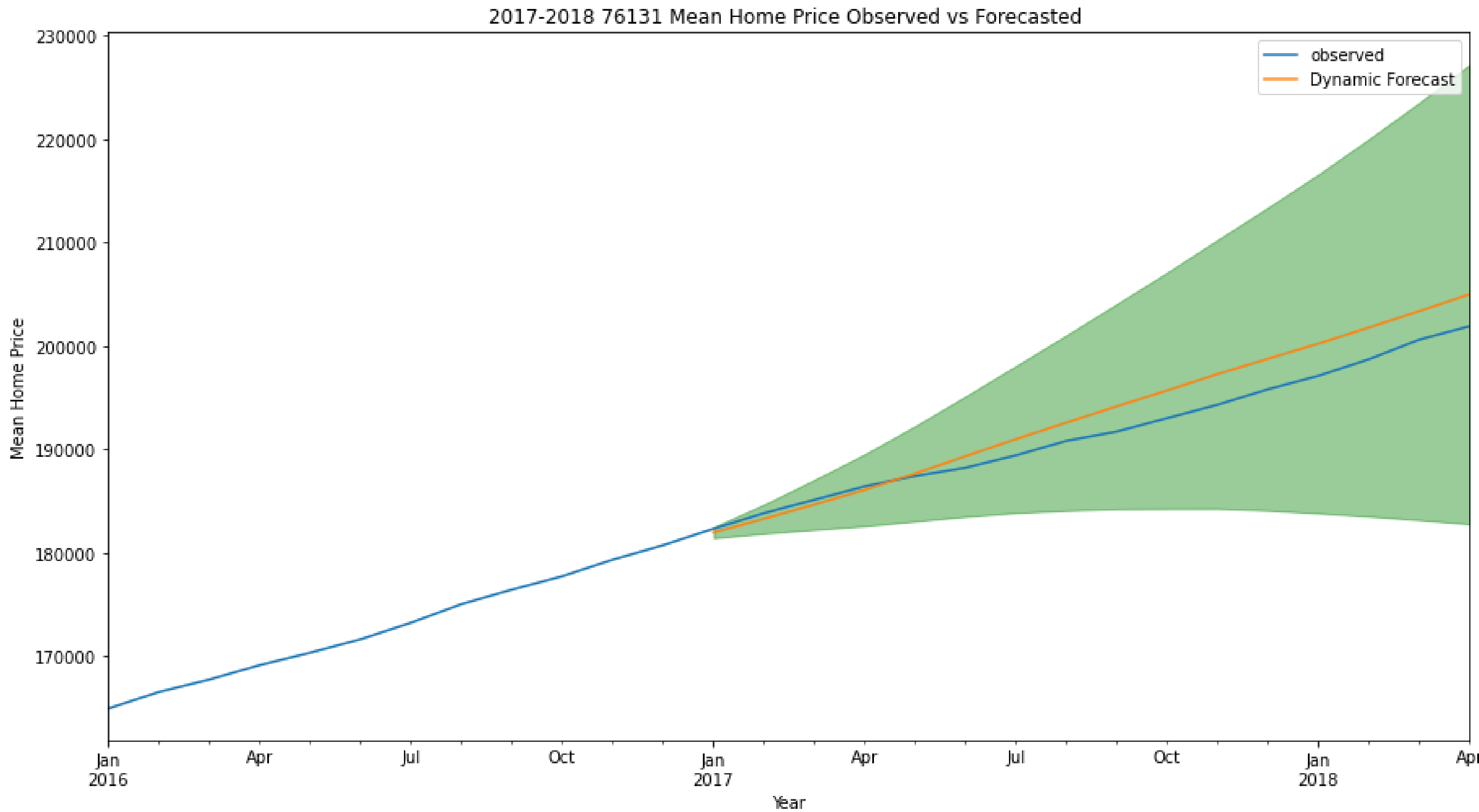
Our diagnostic tests point to our residuals being normally distributed and non correlated to past lags of itself. Our KDE plot is aligned for the most part with a standard normal distribution curve, and our residuals seem to represent that of a white noise model so reflecting stationarity

In [30]:
```
pred_76131 = output_76131.get_prediction(start=pd.to_datetime('2017'),dynamic=True)
pred_conf_76131 = pred_76131.conf_int()
```

executed in 29ms, finished 15:56:25 2021-03-22

In [31]:
```
# Plot real vs predicted values along with confidence interval
rcParams['figure.figsize'] = 15,8
ax = final_zips['2016':]['76131'].plot(kind='line',label='observed')
pred_76131.predicted_mean.plot(kind='line',ax=ax,label='Dynamic Forecast',alpha=.9)

ax.fill_between(pred_conf_76131.index,
                pred_conf_76131.iloc[:,0],
                pred_conf_76131.iloc[:,1], color='g', alpha=.4)

ax.set_xlabel('Year')
ax.set_ylabel('Mean Home Price')
plt.title("2017-2018 76131 Mean Home Price Observed vs Forecasted ")
ax.legend()
```
executed in 404ms, finished 15:56:26 2021-03-22

Out[31]:
```
<matplotlib.legend.Legend at 0x1acaa405250>
```



In [32]:

```
forecast_76131 = pred_76131.predicted_mean
truth_76131 = final_zips['2017':]['76131']

# Compute the mean square error
mse_76131 = ((forecast_76131 -truth_76131)**2).mean()
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse_76131), 2)))
```

executed in 14ms, finished 15:56:26 2021-03-22

```
The Root Mean Squared Error of our forecasts is 2159.91
```

In [33]:
```
forecast_accuracy(forecast_76131,truth_76131)
```
executed in 13ms, finished 15:56:26 2021-03-22

Out[33]:
```
{'mape': 0.009460773640675968,
 'rmse': 2159.907253918256,
 'corr': 0.9968044309476334}
```

In [34]:
```
1 - forecast_accuracy(forecast_76131,truth_76131)['mape']
```
executed in 13ms, finished 15:56:29 2021-03-22

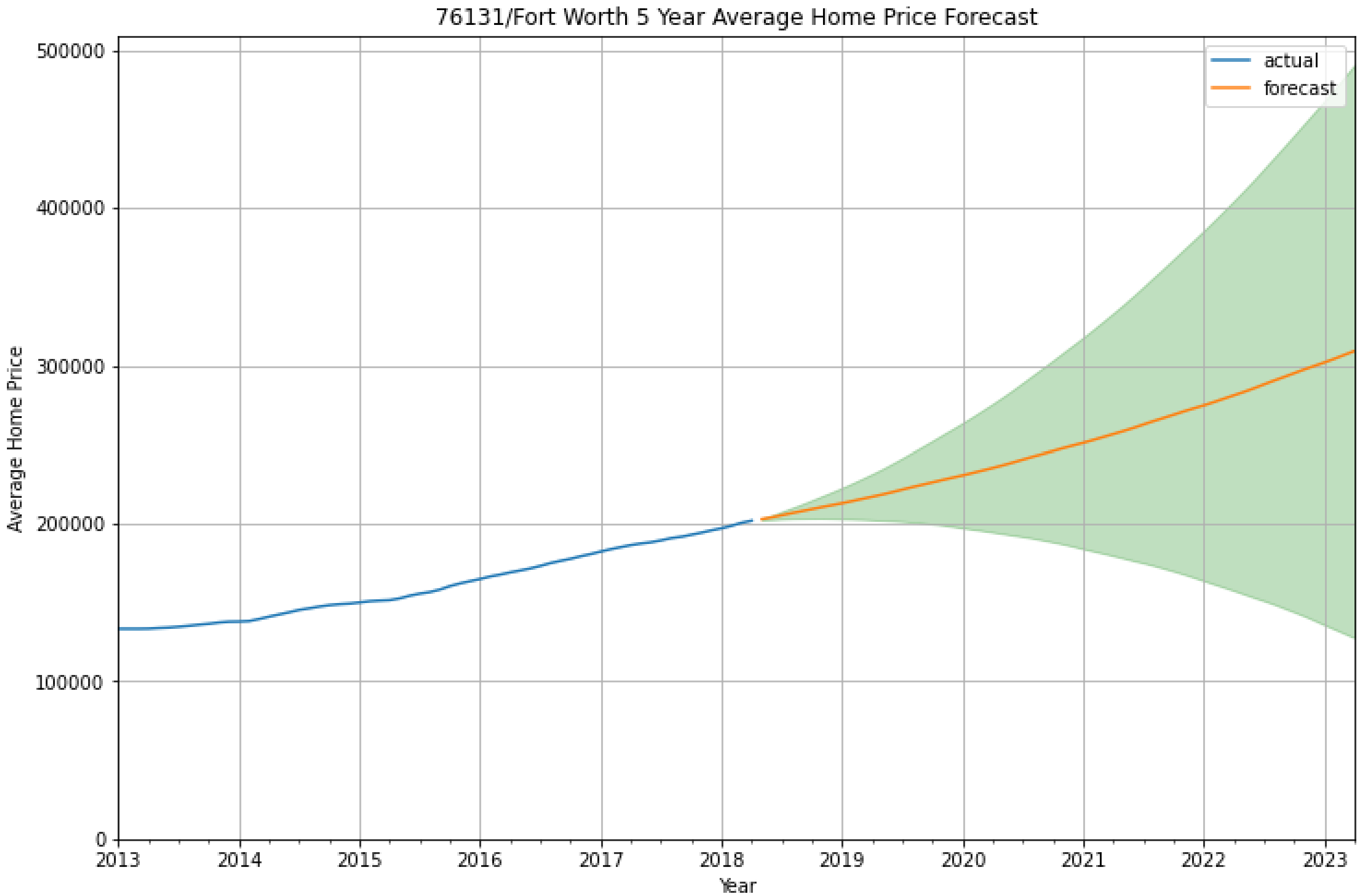
Out[34]:
```
0.990539226359324
```

Our model MAPE score tells us that our model forecasts was 98.7% accurate at predicting the average home prices for the 2017 - 2018 period

In [35]:
```
# Get forecast 500 steps ahead in future
prediction_76131 = output_76131.get_forecast(steps=60)

# Get confidence intervals of forecasts
pred_conf_76131 = prediction_76131.conf_int()
```
executed in 29ms, finished 15:56:29 2021-03-22

In [36]:
```
rcParams['figure.figsize'] = 12,8
ax = final_zips['2013':]['76131'].plot(label='actual')
prediction_76131.predicted_mean.plot(label='forecast')
ax.fill_between(pred_conf_76131.index,
               pred_conf_76131.iloc[:,0],
               pred_conf_76131.iloc[:,1],color='g',alpha=.25)
ax.set_xlabel('Year')
ax.set_ylabel('Average Home Price')
plt.title('76131/Fort Worth 5 Year Average Home Price Forecast')
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: int(x)))
plt.yticks(np.linspace(0,500000,num=6,dtype=int))
plt.legend()
plt.grid(which='major')
plt.show()
```

executed in 406ms, finished 15:56:30 2021-03-22



1.4 92803 Model and Forecast - Everett, Washington

In [37]:
```
mod_98203 = sm.tsa.statespace.SARIMAX(final_zips['98203'],
                                            order=(3,2,1),
                                            seasonal_order=(4,2,0,12),
                                            enforce_stationarity=False,
                                            enforce_invertibility=False,
                                           simple_differencing=False)

output_98203 = mod_98203.fit()
```
executed in 15.2s, finished 15:56:45 2021-03-22

In [38]:

```
output_98203.summary()
```

executed in 27ms, finished 15:56:45 2021-03-22

Out[38]:

SARIMAX Results

Dep. Variable:	98203	**No. Observations:**	265
Model:	SARIMAX(3, 2, 1)x(4, 2, [], 12)	**Log Likelihood**	-1521.691
Date:	Mon, 22 Mar 2021	**AIC**	3061.382
Time:	15:56:45	**BIC**	3090.510
Sample:	04-01-1996	**HQIC**	3073.184
	- 04-01-2018		
Covariance Type:	opg		

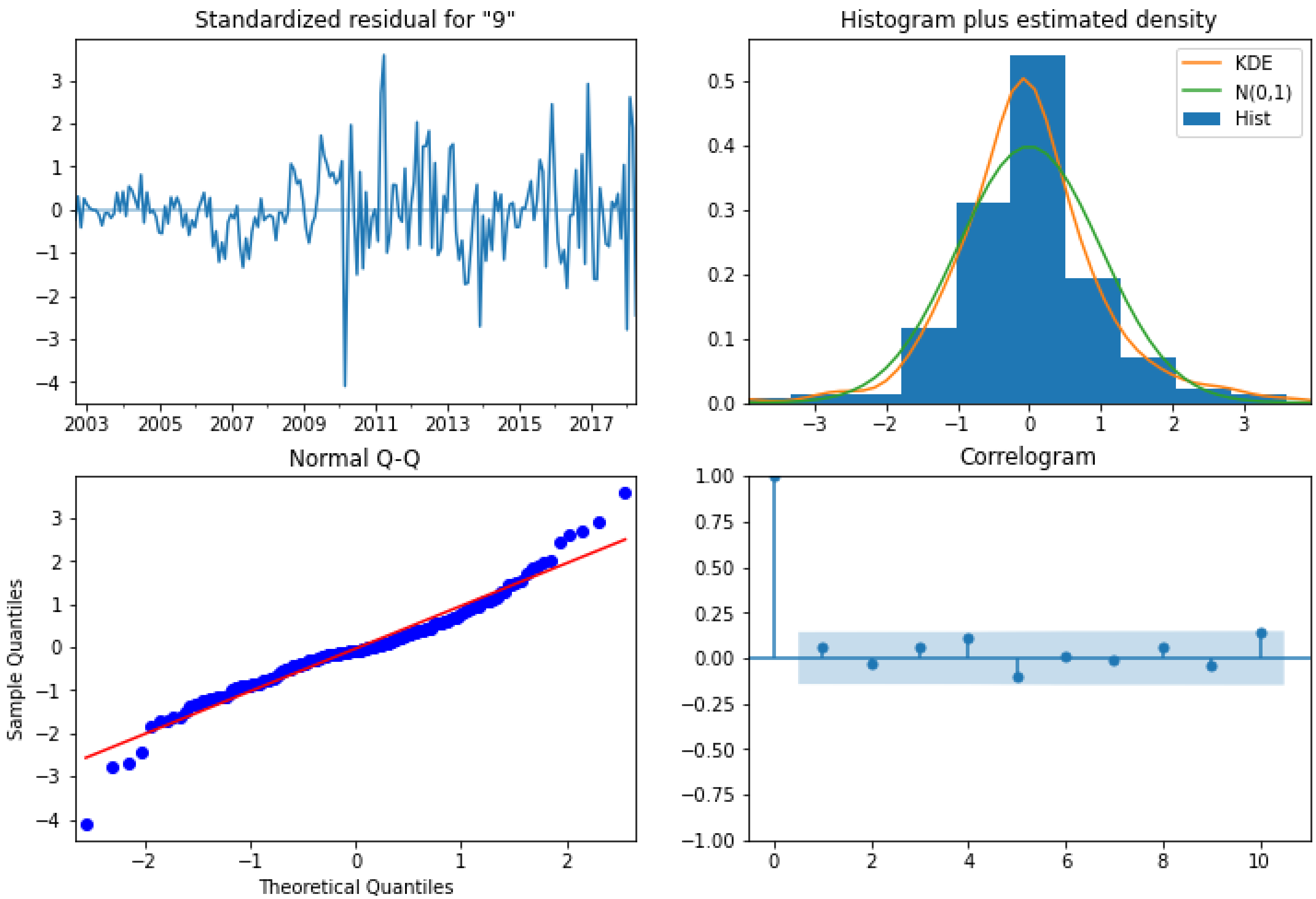
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	1.4668	0.056	26.293	0.000	1.358	1.576
ar.L2	-0.9346	0.096	-9.693	0.000	-1.124	-0.746
ar.L3	0.4367	0.063	6.949	0.000	0.313	0.560
ma.L1	-0.9996	0.078	-12.799	0.000	-1.153	-0.847
ar.S.L12	-1.2385	0.056	-22.060	0.000	-1.349	-1.129
ar.S.L24	-1.0069	0.091	-11.050	0.000	-1.186	-0.828
ar.S.L36	-0.8501	0.117	-7.236	0.000	-1.080	-0.620
ar.S.L48	-0.5040	0.090	-5.614	0.000	-0.680	-0.328
sigma2	6.305e+05	1.28e-07	4.92e+12	0.000	6.3e+05	6.3e+05

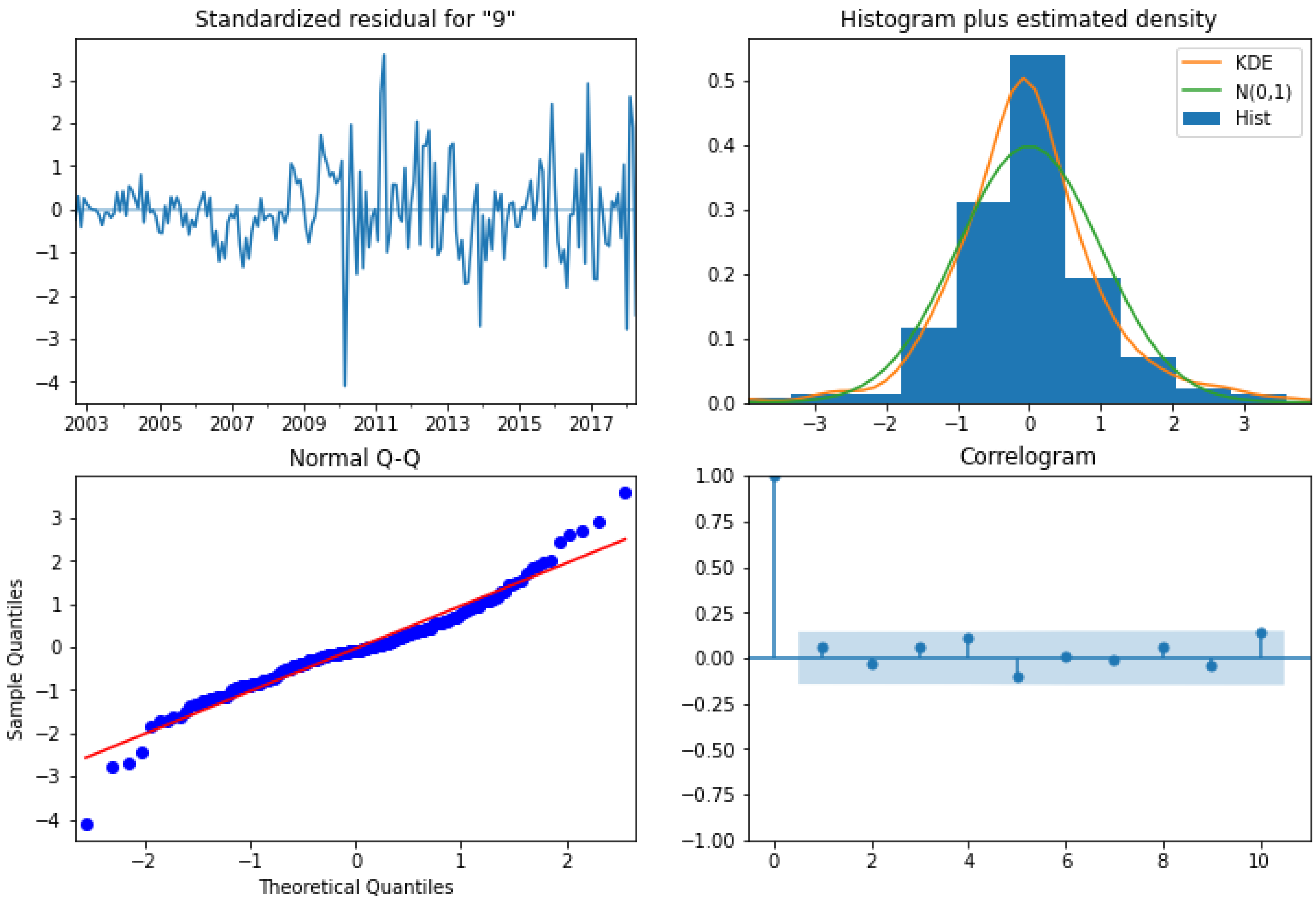
Ljung-Box (L1) (Q):	0.73	**Jarque-Bera (JB):**	49.40
Prob(Q):	0.39	**Prob(JB):**	0.00
Heteroskedasticity (H):	6.57	**Skew:**	0.12
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.50

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 4.79e+29. Standard errors may be unstable.

In [39]:
```
output_98203.plot_diagnostics()
```
executed in 1.41s, finished 15:56:46 2021-03-22

Out[39]:





Looking at our diagnostic plots

Our top left standardized residuals appear to be stationary reflecting a blend of a white noise and random walk model.

Our top right plot KDE vs standard normal distribution plot are similarly distributed although the KDE has a higher peak reflecting a smaller standard deviation but nonetheless is normally distributed.

Our bottom right QQ plot appears to be normally distributed as most points fall on the the 45 degree line

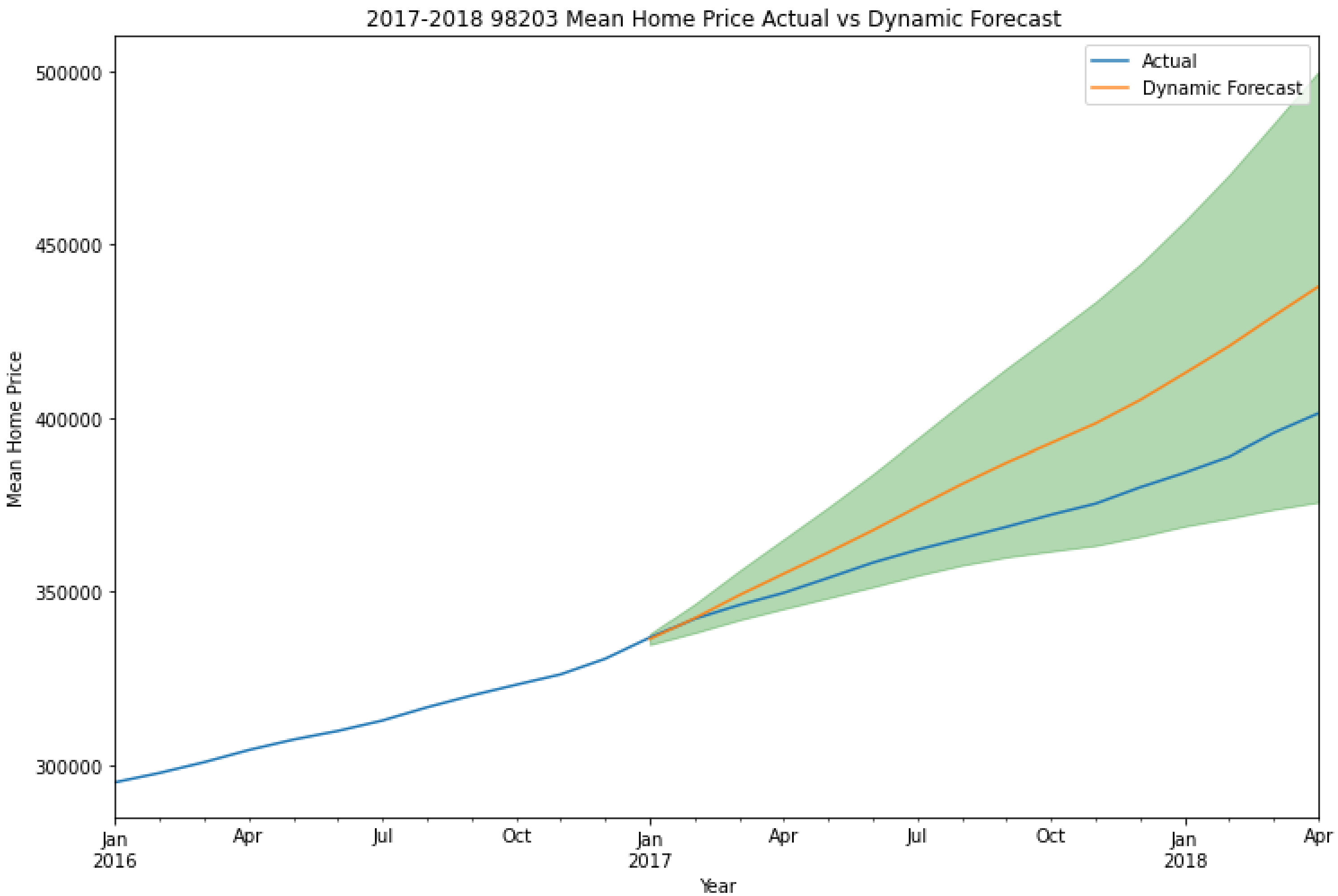
Lastly, our residuals do not appear to be correlated with past lags as none of the lags in the correlogram are above the stat sig blue shading

In [40]:
```
pred_98203 = output_98203.get_prediction(start=pd.to_datetime('2017'),dynamic=True)
pred_conf_98203 = pred_98203.conf_int()
```
executed in 14ms, finished 15:56:50 2021-03-22

In [41]:
```
rcParams['figure.figsize'] = 12,8
ax = final_zips['2016':]['98203'].plot(kind='line',label='Actual')
pred_98203.predicted_mean.plot(kind='line',ax=ax,label='Dynamic Forecast',alpha=.9)

# Plot observed values

# Plot predicted values

# Plot the range for confidence intervals
ax.fill_between(pred_conf_98203.index,
                pred_conf_98203.iloc[:,0],
                pred_conf_98203.iloc[:,1], color='g', alpha=.3)
# Set axes labelsf
ax.set_xlabel('Year')
ax.set_ylabel('Mean Home Price')
plt.title("2017-2018 98203 Mean Home Price Actual vs Dynamic Forecast ")
ax.legend()
```

executed in 345ms, finished 15:56:50 2021-03-22

Out[41]: <matplotlib.legend.Legend at 0x1aca8629310>



In [42]:

```
forecast_98203 = pred_98203.predicted_mean
truth_98203 = final_zips['2017':]['98203']

# Compute the mean square error
mse_98203 = ((forecast_98203 -truth_98203)**2).mean()
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse_98203), 2)))
```

executed in 14ms, finished 15:56:50 2021-03-22

```
The Root Mean Squared Error of our forecasts is 20596.94
```

In [43]:
```
forecast_accuracy(forecast_98203,truth_98203)
```
executed in 14ms, finished 15:56:50 2021-03-22

Out[43]:
```
{'mape': 0.044626299051938156,
 'rmse': 20596.935596267507,
 'corr': 0.9993114334119999}
```

In [44]:
```
1 - forecast_accuracy(forecast_98203,truth_98203)['mape']
```
executed in 14ms, finished 15:56:50 2021-03-22

Out[44]:
```
0.9553737009480618
```

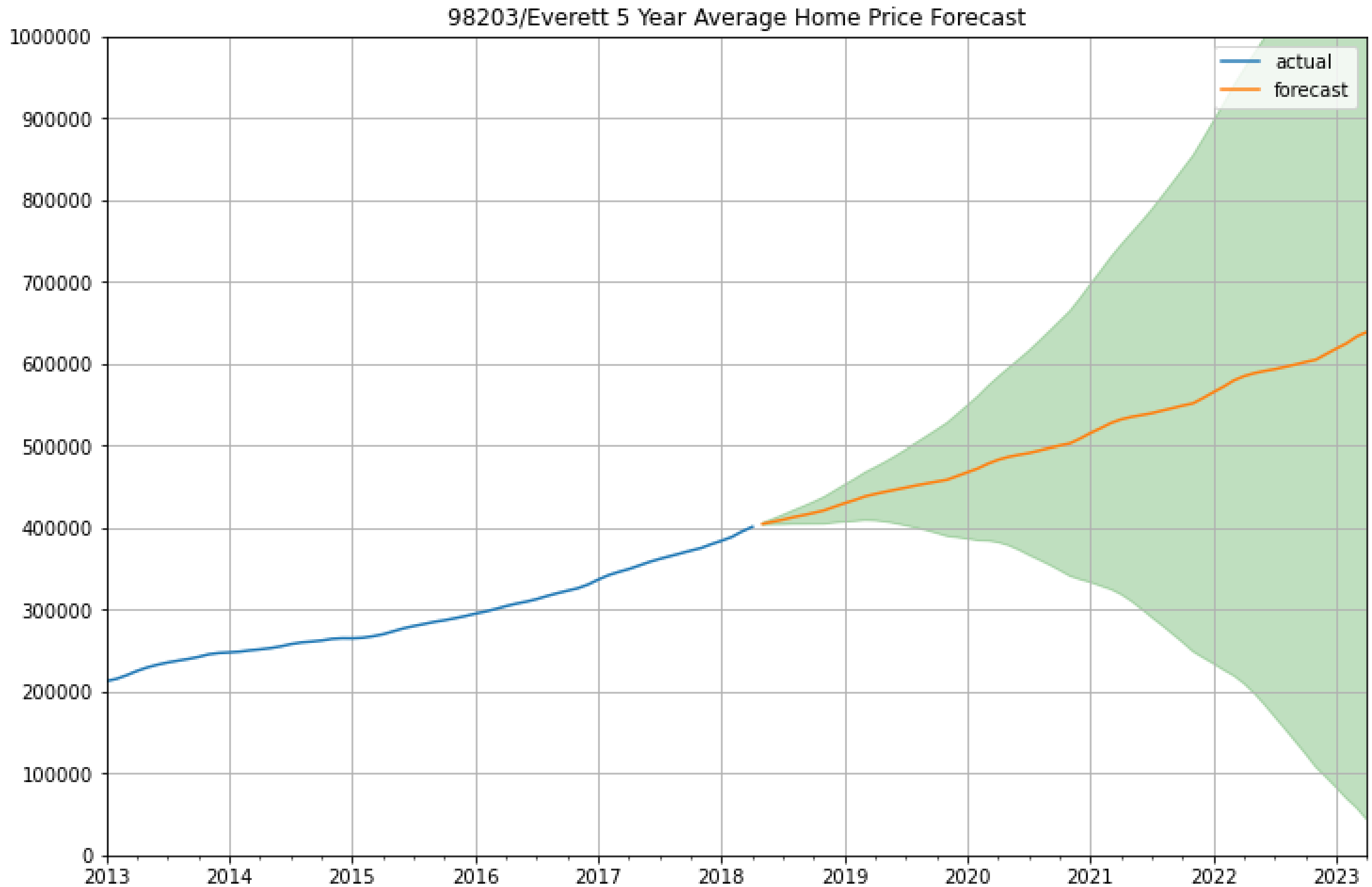
Our model predictions are 95.53% accurate as per our mape score for the 2017-2018 period. Notably, this is our lowest score and highest error per zipcode thus far

In [45]:
```
# Get forecast 500 steps ahead in future
prediction_98203 = output_98203.get_forecast(steps=60)

# Get confidence intervals of forecasts
pred_conf_98203 = prediction_98203.conf_int(alpha=.1)
```
executed in 44ms, finished 15:56:54 2021-03-22

In [46]:

```
rcParams['figure.figsize'] = 12,8
ax = final_zips['2013':]['98203'].plot(label='actual')
prediction_98203.predicted_mean.plot(label='forecast')
ax.fill_between(pred_conf_98203.index,
               pred_conf_98203.iloc[:,0],
               pred_conf_98203.iloc[:,1],color='g',alpha=.25)
plt.title('98203/Everett 5 Year Average Home Price Forecast')
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: int(x)))
plt.yticks(np.linspace(0,1000000,num=11,dtype=int))
plt.ylim(0,1000000)
plt.legend()
plt.grid(which='major')
plt.show()
```

executed in 438ms, finished 15:56:55 2021-03-22



1.5 49507 Zipcode Model and Forecast - Grand Rapids, Michigan

In [47]:

```
mod_MI = sm.tsa.statespace.SARIMAX(final_zips['49507'],
                                    order=(1,1,3),
                                    seasonal_order=(1,2,2,12),
                                    enforce_stationarity=False,
                                    enforce_invertibility=False,
                                   simple_differencing=False)

output_MI = mod_MI.fit()
```

executed in 5.96s, finished 15:57:01 2021-03-22

In [48]:

```
output_MI.summary()
```

executed in 44ms, finished 15:57:01 2021-03-22

Out[48]:

SARIMAX Results

Dep. Variable:	49507	**No. Observations:**	265
Model:	SARIMAX(1, 1, 3)x(1, 2, [1, 2], 12)	**Log Likelihood**	-1576.588
Date:	Mon, 22 Mar 2021	**AIC**	3169.176
Time:	15:57:01	**BIC**	3196.028
Sample:	04-01-1996	**HQIC**	3180.029
	- 04-01-2018		
Covariance Type:	opg		

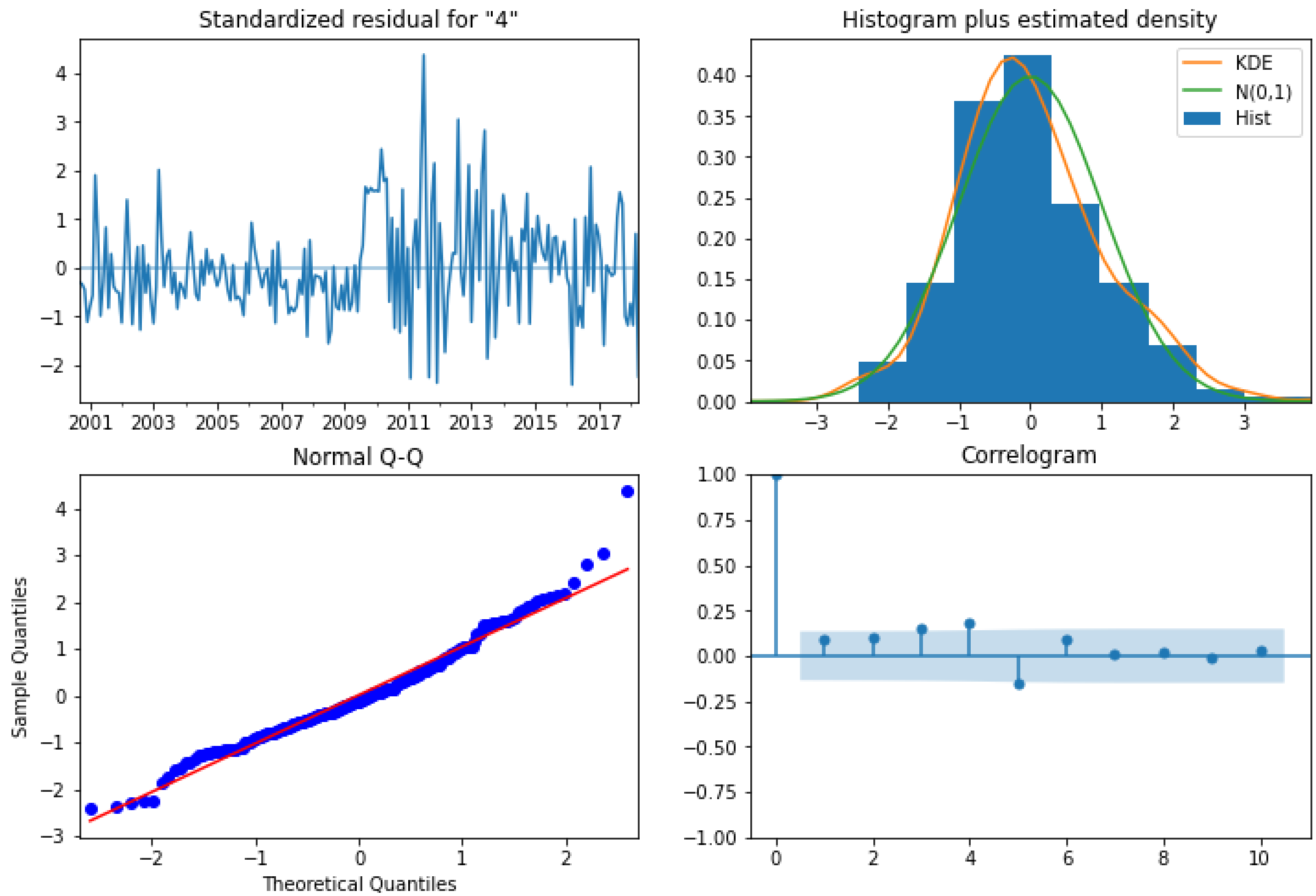
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9655	0.018	54.466	0.000	0.931	1.000
ma.L1	0.2114	0.058	3.632	0.000	0.097	0.326
ma.L2	-0.5018	0.052	-9.687	0.000	-0.603	-0.400
ma.L3	-0.3483	0.062	-5.574	0.000	-0.471	-0.226
ar.S.L12	0.0886	0.039	2.293	0.022	0.013	0.164
ma.S.L12	-1.7201	0.083	-20.843	0.000	-1.882	-1.558
ma.S.L24	0.8440	0.083	10.171	0.000	0.681	1.007
sigma2	1.412e+05	1.38e+04	10.258	0.000	1.14e+05	1.68e+05

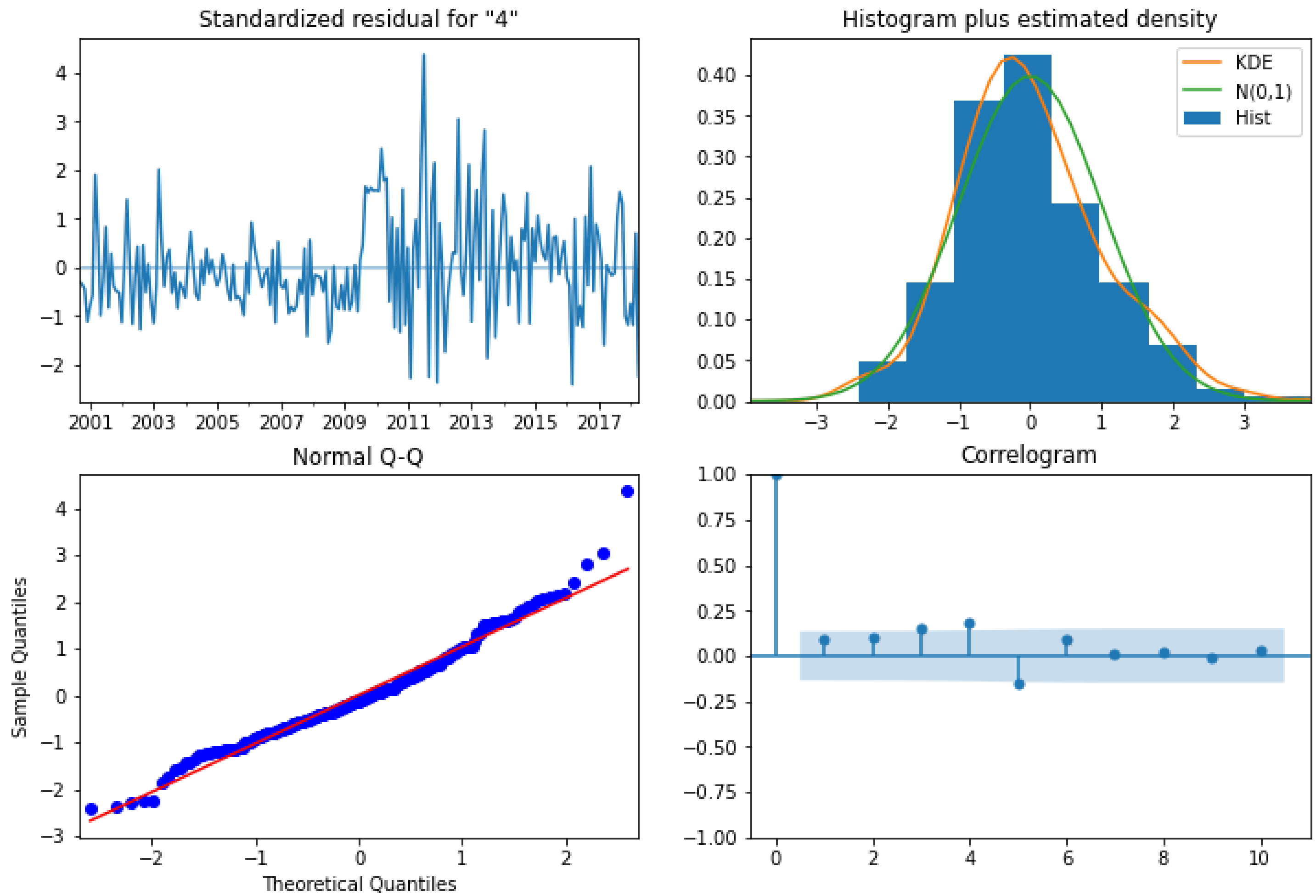
Ljung-Box (L1) (Q):	1.92	**Jarque-Bera (JB):**	26.46
Prob(Q):	0.17	**Prob(JB):**	0.00
Heteroskedasticity (H):	2.79	**Skew:**	0.63
Prob(H) (two-sided):	0.00	**Kurtosis:**	4.19

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [49]:
```
output_MI.plot_diagnostics()
```
executed in 1.71s, finished 15:57:03 2021-03-22

Out[49]:





Analyzing our diagnostic plots, our model passes tests for stationarity and normality of residuals.

The top left plot confirms stationarity as it reflects a white noise type model, while the qq plot and KDE vs N(0,1) plots show normally distributed residuals, notably with a slight positive skew.

Lastly, our residuals are not correlated to prior lags as represented by the bottom right correlogram

In [50]:
```
pred_MI = output_MI.get_prediction(start=pd.to_datetime('2017'),dynamic=True)
pred_conf_MI = pred_MI.conf_int()
```
executed in 14ms, finished 15:57:07 2021-03-22

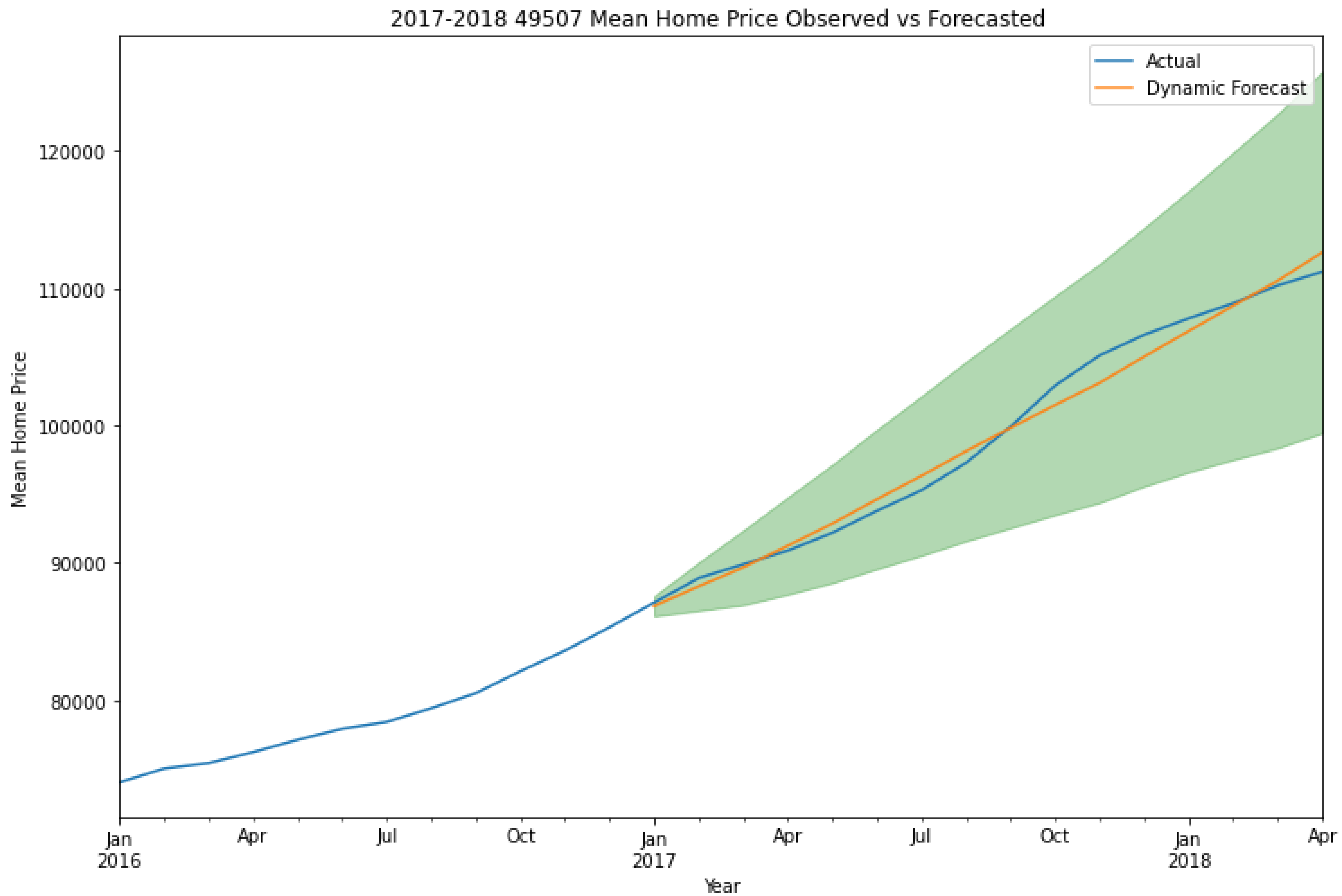
In [51]:

```
rcParams['figure.figsize'] = 12,8
ax = final_zips['2016':]['49507'].plot(kind='line',label='Actual')
pred_MI.predicted_mean.plot(kind='line',ax=ax,label='Dynamic Forecast',alpha=.9)

ax.fill_between(pred_conf_MI.index,
               pred_conf_MI.iloc[:,0],
               pred_conf_MI.iloc[:,1], color='g', alpha=.3)
# Set axes labelsf
ax.set_xlabel('Year')
ax.set_ylabel('Mean Home Price')
plt.title("2017-2018 49507 Mean Home Price Observed vs Forecasted ")
ax.legend()
```

executed in 359ms, finished 15:57:07 2021-03-22

Out[51]:

```
<matplotlib.legend.Legend at 0x1acd4f3abe0>
```



In [52]:

```
forecast_MI = pred_MI.predicted_mean
truth_MI = final_zips['2017':]['49507']

# Compute the mean square error
mse_MI = ((forecast_MI -truth_MI)**2).mean()
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse_MI), 2)))
```

executed in 12ms, finished 15:57:07 2021-03-22

```
The Root Mean Squared Error of our forecasts is 973.25
```

In [53]:

```
forecast_accuracy(forecast_MI,truth_MI)
```

executed in 13ms, finished 15:57:10 2021-03-22

Out[53]:

```
{'mape': 0.007929046716778668,
 'rmse': 973.2473729788235,
 'corr': 0.9928341179470681}
```

In [54]:

```
1- forecast_accuracy(forecast_MI,truth_MI)['mape']
```

executed in 13ms, finished 15:57:11 2021-03-22

Out[54]:

```
0.9920709532832214
```

Our model predictions vs actual for 2017-2018 mean home prices are 99.2% accurate according to our mape score and our RMSE is 973.25

In [55]:

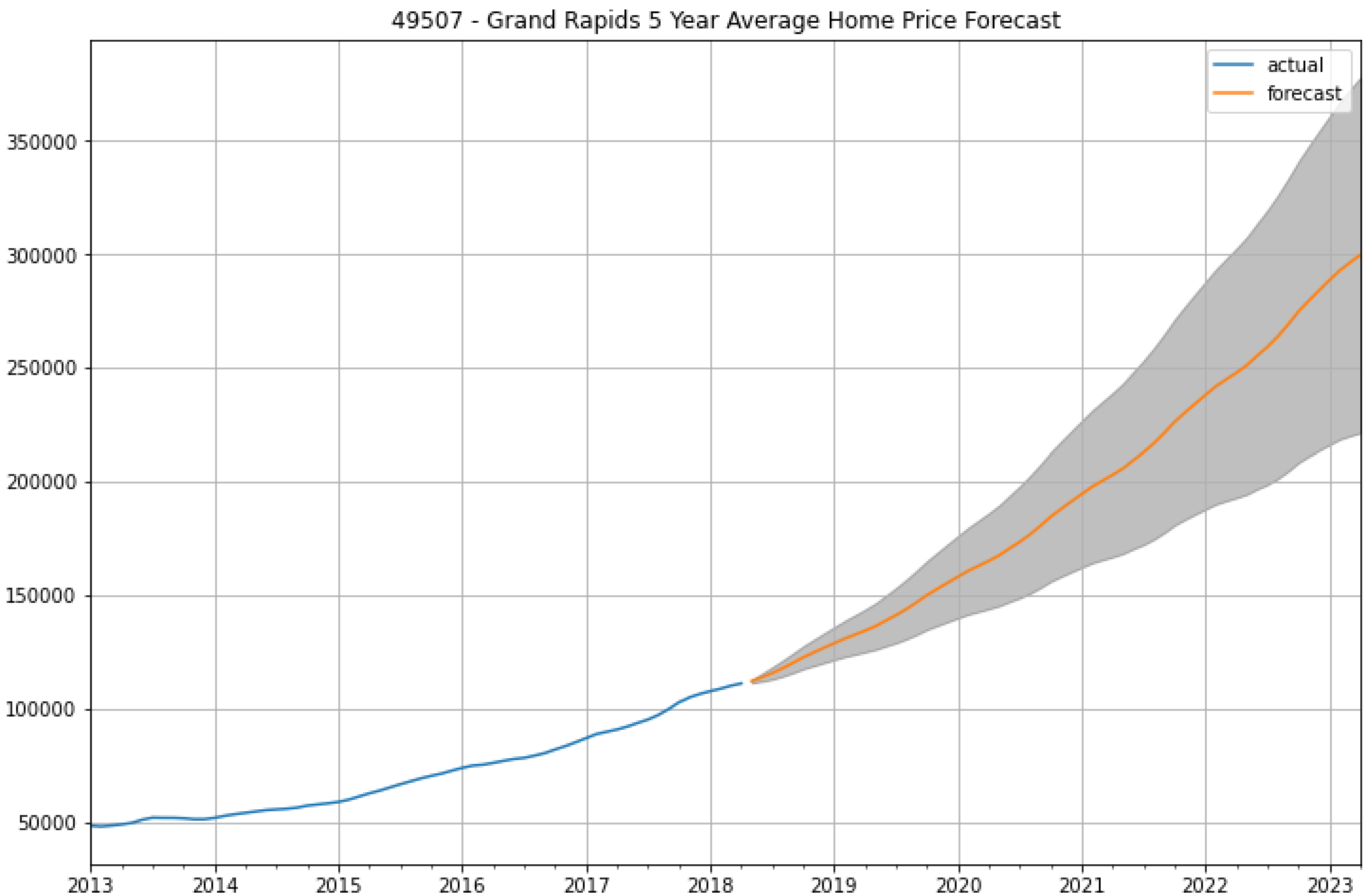
```
# Get forecast 500 steps ahead in future
prediction_MI = output_MI.get_forecast(steps=60)

# Get confidence intervals of forecasts
pred_conf_MI = prediction_MI.conf_int()
```

executed in 30ms, finished 15:57:12 2021-03-22

In [56]:

```
rcParams['figure.figsize'] = 12,8
ax = final_zips['2013':]['49507'].plot(label='actual')
prediction_MI.predicted_mean.plot(label='forecast')
ax.fill_between(pred_conf_MI.index,
               pred_conf_MI.iloc[:,0],
               pred_conf_MI.iloc[:,1],color='k',alpha=.25)
plt.title('49507 - Grand Rapids 5 Year Average Home Price Forecast')
plt.gca().yaxis.set_major_formatter(FuncFormatter(lambda x, _: int(x)))
plt.legend()
plt.grid(b=True, which='major')
plt.show()
plt.savefig('images/MI_forecast1')
```

executed in 490ms, finished 15:57:12 2021-03-22



<Figure size 864x576 with 0 Axes>

Surprisingly, Grand Rapids Forecasts seem to be the most promising compared to the other zipcodes. The plot above shows potential returns up to 200% over the 5 year period

1.5.1 RFR forecast

In [57]:

```
#importing data on US 10 year treasury rates i.e. risk free rate
rfr = pd.read_csv('DGS10.csv')
rfr.DATE = pd.to_datetime(rfr.DATE,infer_datetime_format=True)
rfr = rfr.set_index('DATE')
rfr = rfr[rfr.DGS10 != '.']
rfr.DGS10 = rfr.DGS10.apply(lambda x: np.float(x))
rfr = rfr.resample('MS',).mean()
rfr = rfr[3:-1]
rfr.DGS10 = rfr.DGS10/100
```

executed in 46ms, finished 15:57:12 2021-03-22

In [58]:
```
mod_rfr = sm.tsa.statespace.SARIMAX(rfr.DGS10,
                                            order=(1,0,1),
                                            seasonal_order=(0,0,0,12),
                                            enforce_stationarity=False,
                                            enforce_invertibility=False,
                                           simple_differencing=False)

output_rfr = mod_rfr.fit()
```

executed in 232ms, finished 15:57:13 2021-03-22

In [59]:
```
output_rfr.summary()
```

executed in 26ms, finished 15:57:13 2021-03-22

Out[59]:

SARIMAX Results

Dep. Variable:	DGS10	**No. Observations:**	265
Model:	SARIMAX(1, 0, 1)	**Log Likelihood**	1248.783
Date:	Mon, 22 Mar 2021	**AIC**	-2491.566
Time:	15:57:13	**BIC**	-2480.849
Sample:	04-01-1996	**HQIC**	-2487.259
	- 04-01-2018		
Covariance Type:	opg		

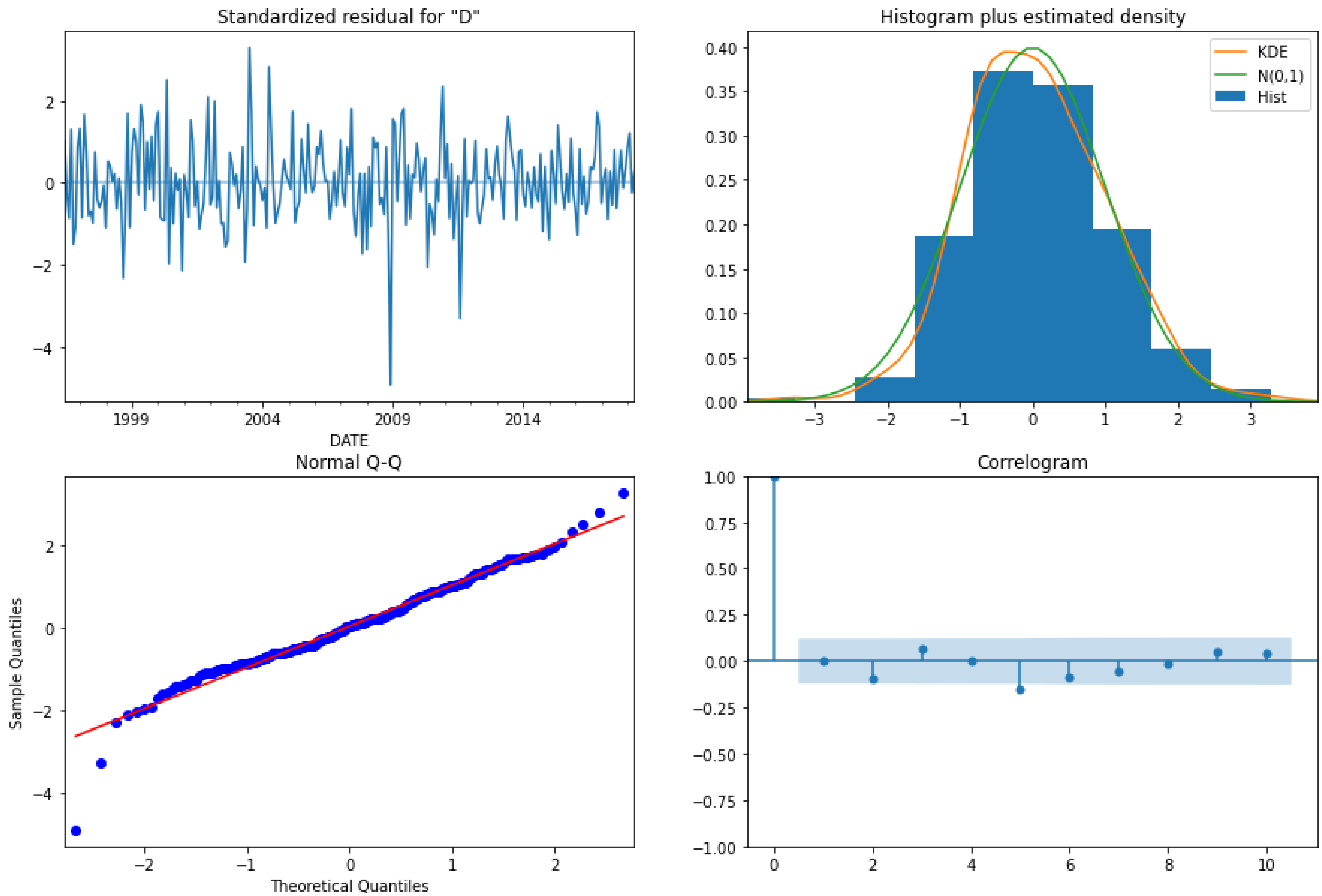
	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.9942	0.004	273.412	0.000	0.987	1.001
ma.L1	0.2031	0.053	3.802	0.000	0.098	0.308
sigma2	4.392e-06	2.78e-07	15.799	0.000	3.85e-06	4.94e-06

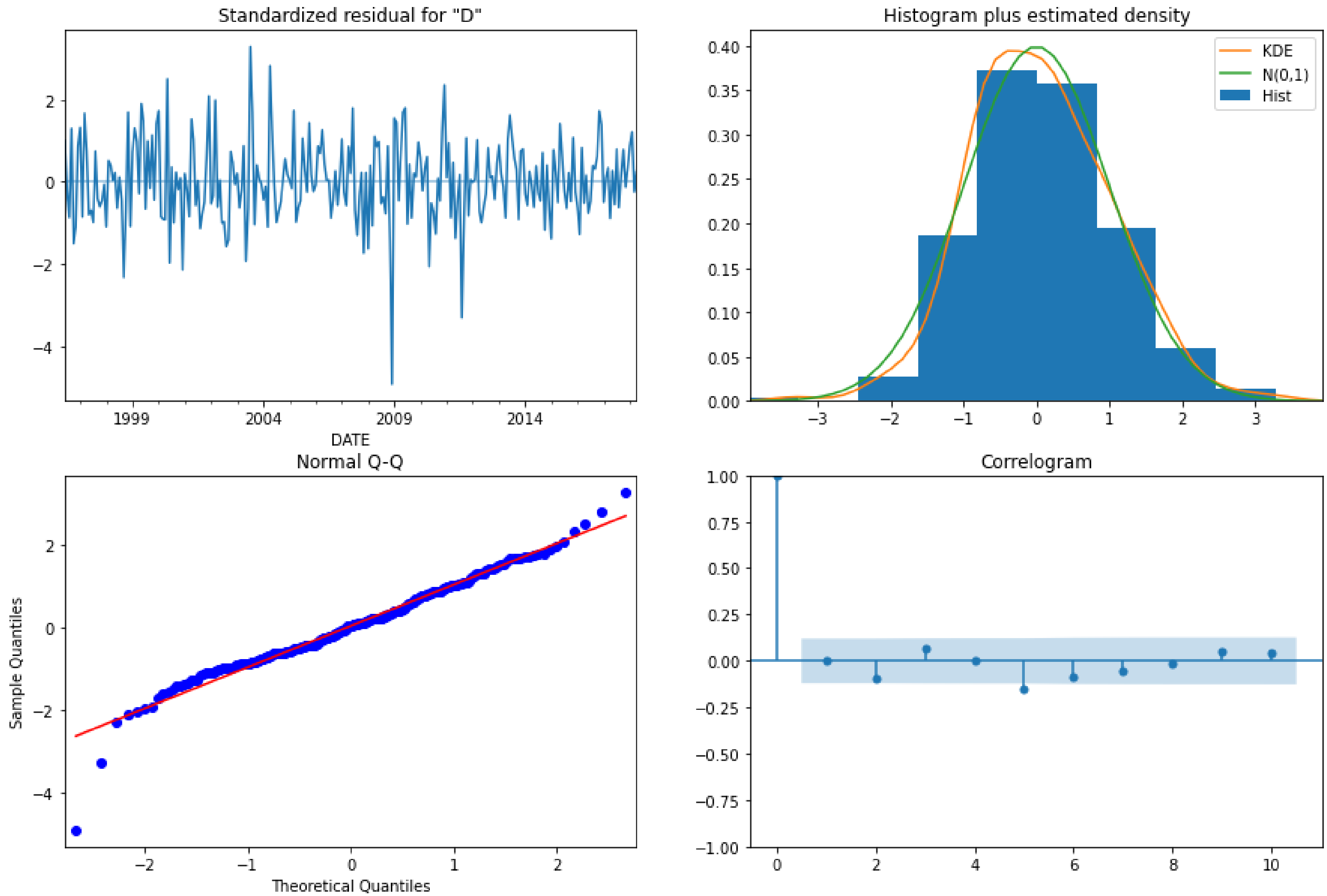
Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	52.25
Prob(Q):	0.95	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.52	**Skew:**	-0.25
Prob(H) (two-sided):	0.00	**Kurtosis:**	5.13

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).

In [60]:
```
output_rfr.plot_diagnostics(figsize=(15,10))
```
executed in 1.24s, finished 15:57:14 2021-03-22

Out[60]:





In [61]:

```
pred_rfr = output_rfr.get_prediction(start=pd.to_datetime('2017'),dynamic=True)
pred_conf_rfr = pred_rfr.conf_int()
```

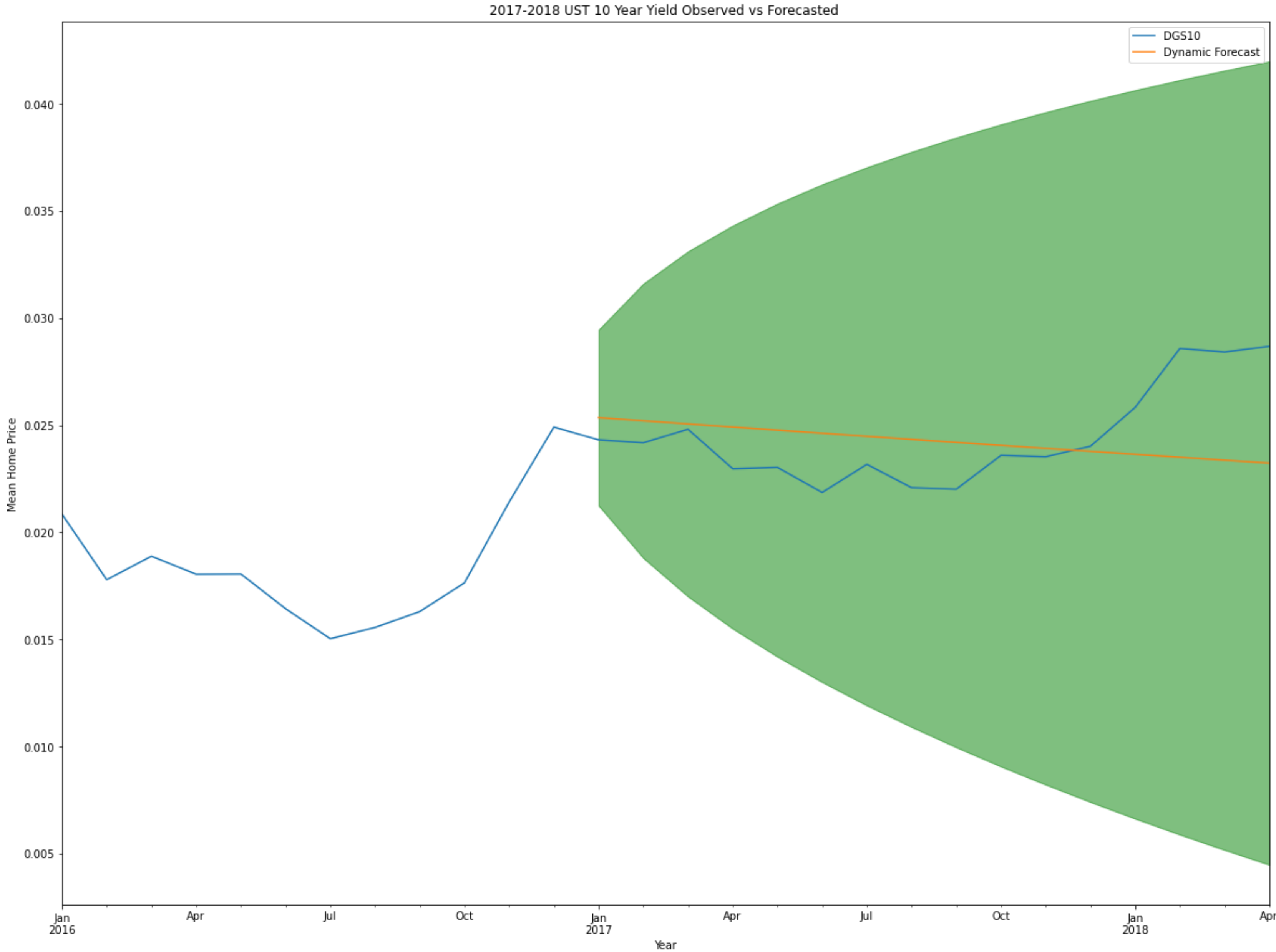
executed in 14ms, finished 15:57:14 2021-03-22

In [62]:

```
rcParams['figure.figsize'] = 20,15
ax = rfr['2016':].plot(kind='line',label='observed')
pred_rfr.predicted_mean.plot(kind='line',ax=ax,label='Dynamic Forecast',alpha=.9)

# Plot observed values

# Plot predicted values

# Plot the range for confidence intervals
ax.fill_between(pred_conf_rfr.index,
               pred_conf_rfr.iloc[:,0],
               pred_conf_rfr.iloc[:,1], color='g', alpha=.5)
# Set axes labelsf
ax.set_xlabel('Year')
ax.set_ylabel('Mean Home Price')
plt.title("2017-2018 UST 10 Year Yield Observed vs Forecasted ")
ax.legend()
```

executed in 363ms, finished 15:57:14 2021-03-22

Out[62]: <matplotlib.legend.Legend at 0x1acba7aa0d0>


2017-2018 UST 10 Year Yield Observed vs Forecasted
DGS10
Dynamic Forecast
Mean Home Price
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
Jan 2016
Apr
Jul
Oct
Jan 2017
Apr
Jul
Oct
Jan 2018
Apr
Year

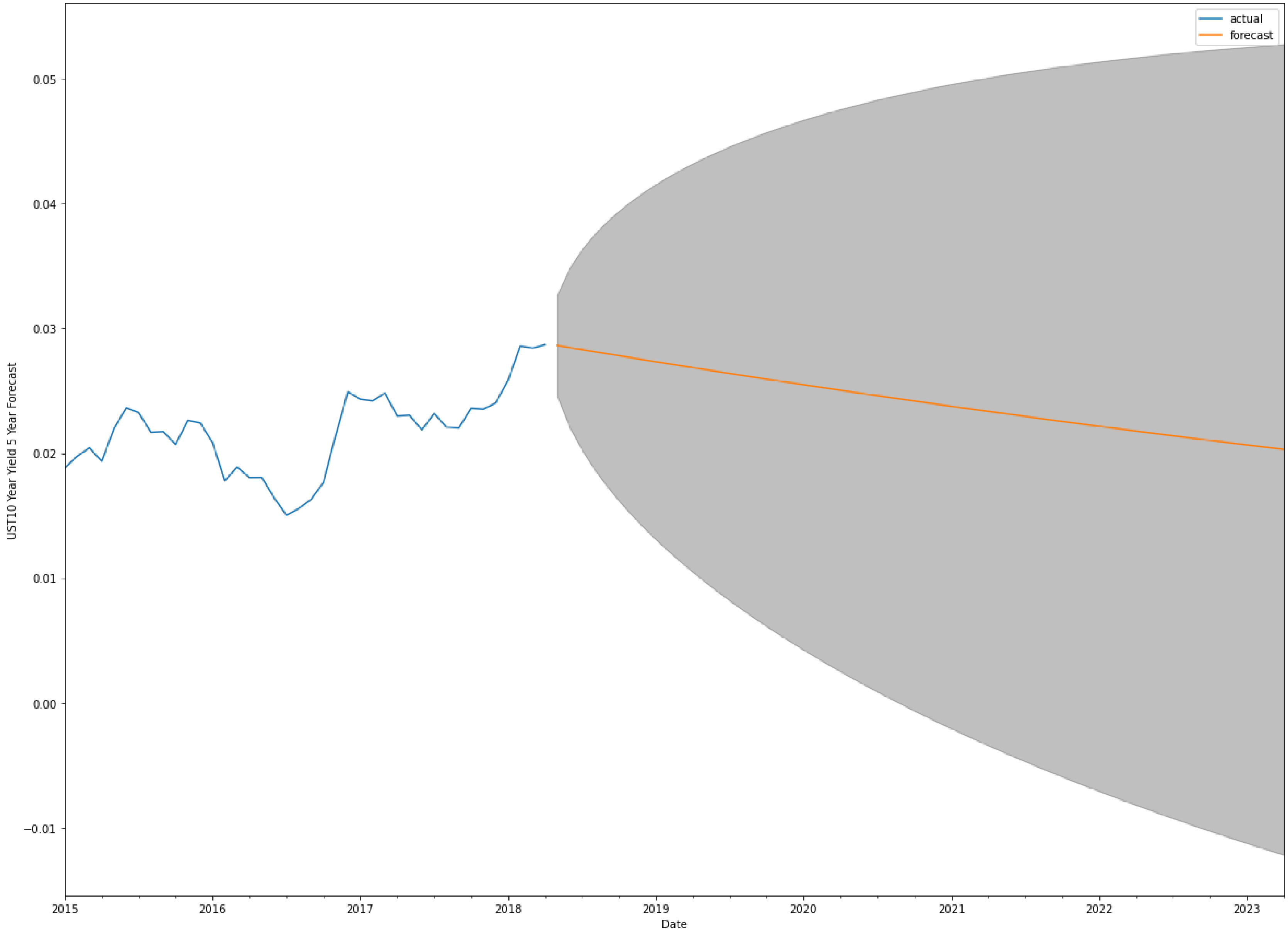
In [63]:
```
forecast_rfr = pred_rfr.predicted_mean
truth_rfr = rfr['2017':].DGS10

# Compute the mean square error
mse_rfr = ((forecast_rfr -truth_rfr)**2).mean()
print('The Root Mean Squared Error of our forecasts is {}'.format(round(np.sqrt(mse_rfr), 8)))
```
executed in 14ms, finished 15:57:14 2021-03-22

```
The Root Mean Squared Error of our forecasts is 0.00267493
```

In [64]:
```
# Get forecast 500 steps ahead in future
prediction_rfr = output_rfr.get_forecast(steps=60)

# Get confidence intervals of forecasts
pred_conf_rfr = prediction_rfr.conf_int()
```
executed in 14ms, finished 15:57:14 2021-03-22

In [65]:
```
rcParams['figure.figsize'] = 20,15
ax = rfr['2015':].DGS10.plot(label='actual')
prediction_rfr.predicted_mean.plot(label='forecast')
ax.fill_between(pred_conf_rfr.index,
                pred_conf_rfr.iloc[:,0],
                pred_conf_rfr.iloc[:,1],color='k',alpha=.25)
ax.set_xlabel('Date')
ax.set_ylabel('UST10 Year Yield 5 Year Forecast')
plt.legend()
plt.show()
```

executed in 373ms, finished 15:57:16 2021-03-22


actual
forecast
0.05
0.04
0.03
0.02
0.01
0.00
−0.01
UST10 Year Yield 5 Year Forecast
2015
2016
2017
2018
2019
2020
2021
2022
2023
Date

1.6 Results

In [66]:

```
zip_preds = list([("49507",prediction_MI),("32809",prediction_32809),("76131",prediction_76131), ("98203",pr
roi_yoy = [ ("49507", 'ROI_YoY'), ("32809", 'ROI_YoY'),("76131", 'ROI_YoY'),("98203", 'ROI_YoY'),("80012",
```

executed in 14ms, finished 15:57:26 2021-03-22

In [67]:

```
rfr_2023 = rfr.DGS10.append(prediction_rfr.predicted_mean)
rfr_2023 = rfr_2023['2013':]
rfr_2023
```

executed in 14ms, finished 15:57:26 2021-03-22

Out[67]:

```
2013-01-01    0.019148
2013-02-01    0.019842
2013-03-01    0.019575
2013-04-01    0.017591
2013-05-01    0.019282
                ...
2022-12-01    0.020780
2023-01-01    0.020659
2023-02-01    0.020539
2023-03-01    0.020419
2023-04-01    0.020301
Freq: MS, Length: 124, dtype: float64
```

In [68]:

```
index = prediction_32809.predicted_mean.index
```

executed in 14ms, finished 15:57:26 2021-03-22

In [69]:

```
index1 = final_zips['2013':].index
```

executed in 13ms, finished 15:57:27 2021-03-22

In [70]:

```
final_index = index1.append(index)
```

executed in 13ms, finished 15:57:30 2021-03-22

In [71]:

```
zips = list(final_zips.columns)
```

executed in 14ms, finished 15:57:31 2021-03-22

In [72]:
```
results = pd.DataFrame(data=None,index=final_index)
results['rfr'] = rfr_2023
for zipcode in zips:
    for item in zip_preds:
        if zipcode == item[0]:
            results[zipcode] = final_zips[zipcode].append(item[1].predicted_mean)
            results[zipcode,'ROI_YoY'] = np.zeros((124))
            for y in range(0,124):
                if y+12 < 124:
                    results[zipcode,'ROI_YoY'][y+12] = round((results[zipcode][y+12]-results[zipcode][y])/r
                else: break
            results[zipcode,'EMA_6MO_roi_y'] = results[zipcode,'ROI_YoY'].ewm(span=6,adjust=False).mean()
            results[zipcode,'EMA_6MO_std_y'] = results[zipcode,'ROI_YoY'].ewm(span=6,adjust=False).std()
            results[zipcode,'sharpe_ratio_annual'] = (results[zipcode,'EMA_6MO_roi_y']-results.rfr)/results
```
executed in 106ms, finished 15:57:32 2021-03-22

In [73]:
```
cols = list(results.columns)
cols = cols[0:][::5]
cols = cols[1:]
zipcode_sharpes = cols
zipcode_sharpes
```
executed in 13ms, finished 15:57:33 2021-03-22

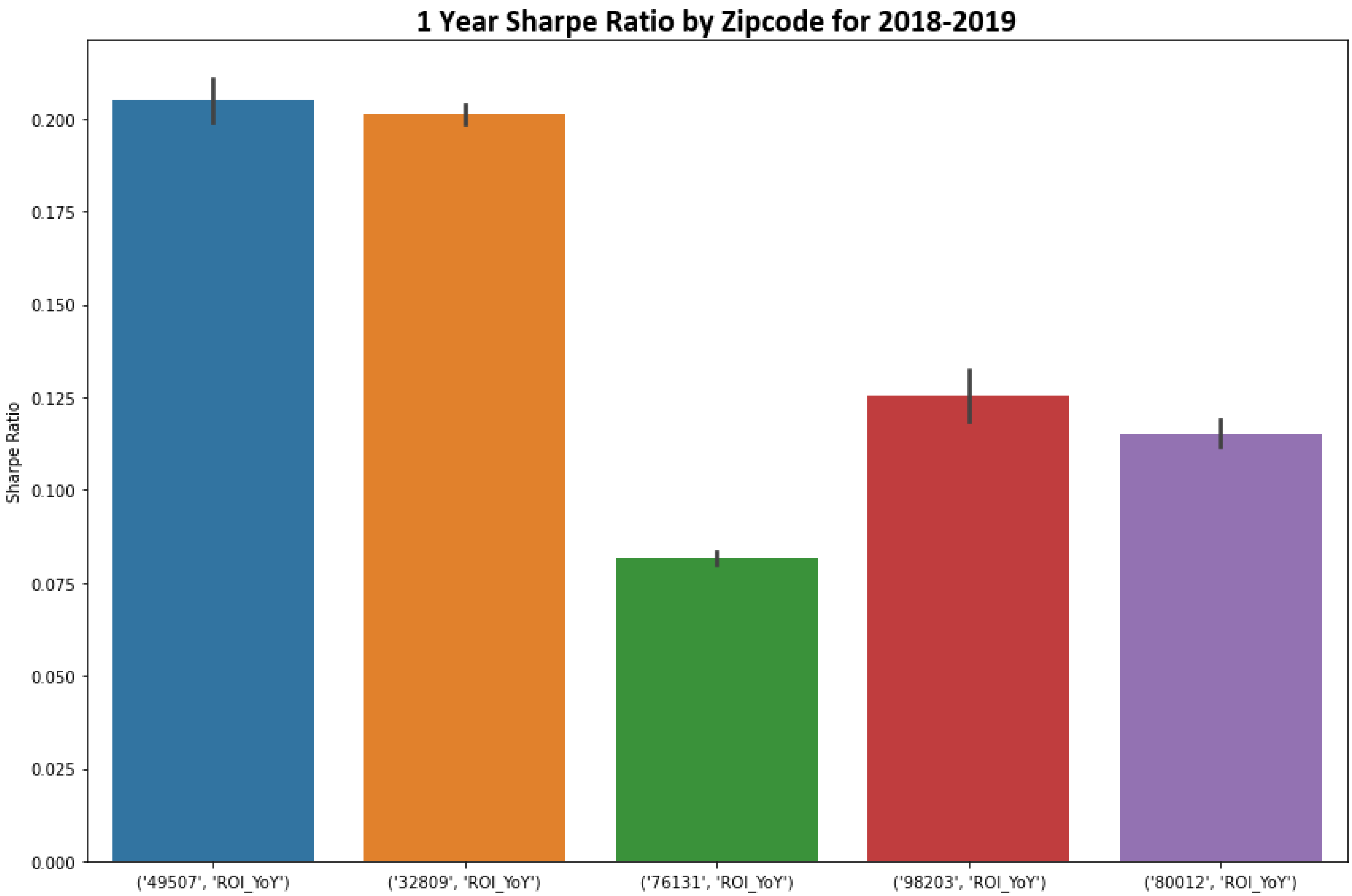
Out[73]:
```
[('32809', 'sharpe_ratio_annual'),
 ('49507', 'sharpe_ratio_annual'),
 ('76131', 'sharpe_ratio_annual'),
 ('80012', 'sharpe_ratio_annual'),
 ('98203', 'sharpe_ratio_annual')]
```

In [74]:
```
font = {'family' : 'Calibri',
        'weight' : 'bold',
        'size'   : 20}
```
executed in 14ms, finished 15:57:34 2021-03-22

In [75]:

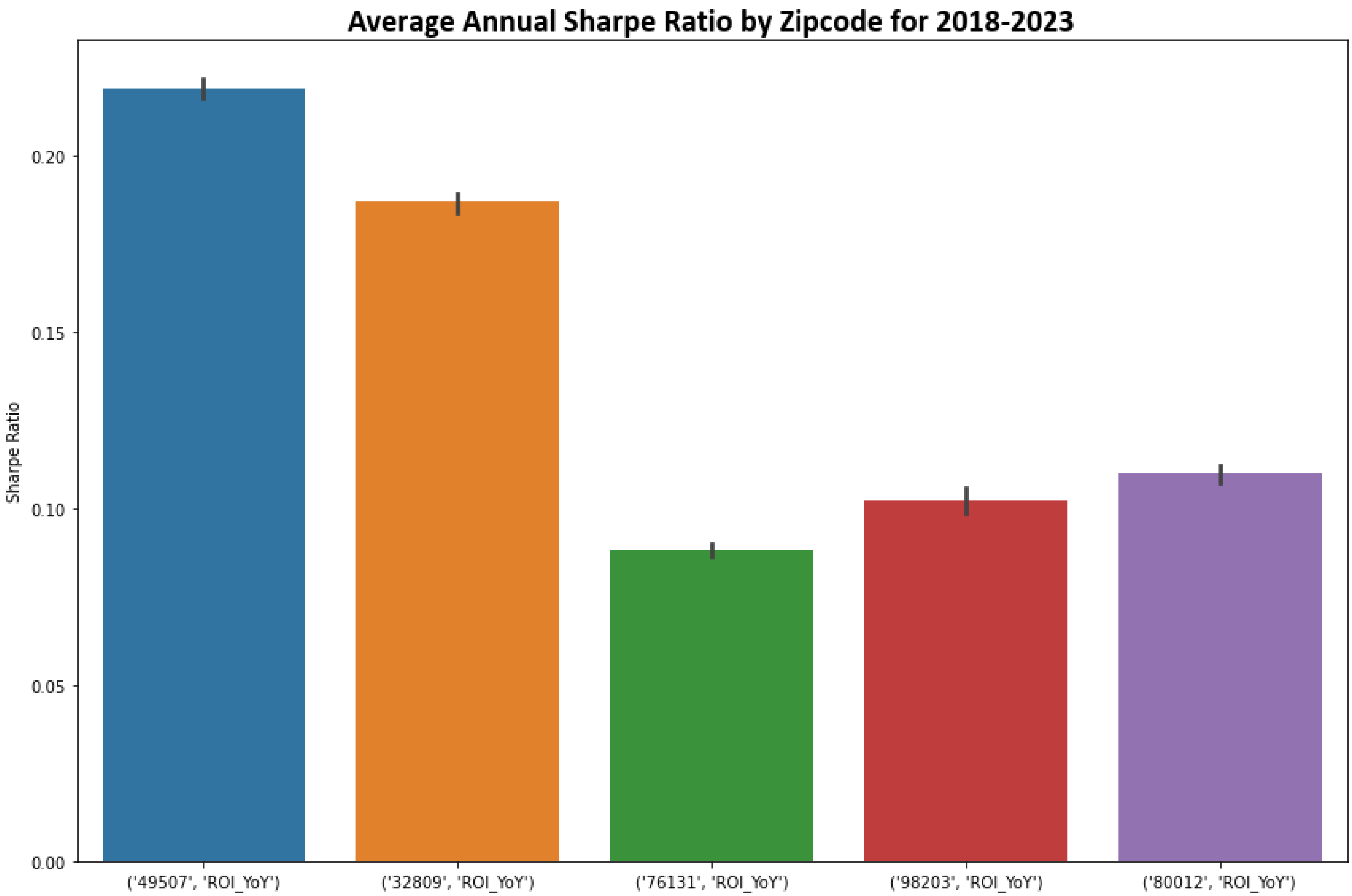
```
fig,ax = plt.subplots(figsize=(12,8))
s = sns.barplot(data=results['2018-04':'2019-04'][roi_yoy],ax=ax)
plt.title('1 Year Sharpe Ratio by Zipcode for 2018-2019',font=font)
plt.ylabel('Sharpe Ratio')
plt.tight_layout()
```

executed in 328ms, finished 15:57:35 2021-03-22



In [76]:
```
fig,ax = plt.subplots(figsize=(12,8))
s = sns.barplot(data=results['2018-04':'2023-04'][roi_yoy],ax=ax)
plt.title('Average Annual Sharpe Ratio by Zipcode for 2018-2023',font=font)
plt.ylabel('Sharpe Ratio')
plt.tight_layout()
```

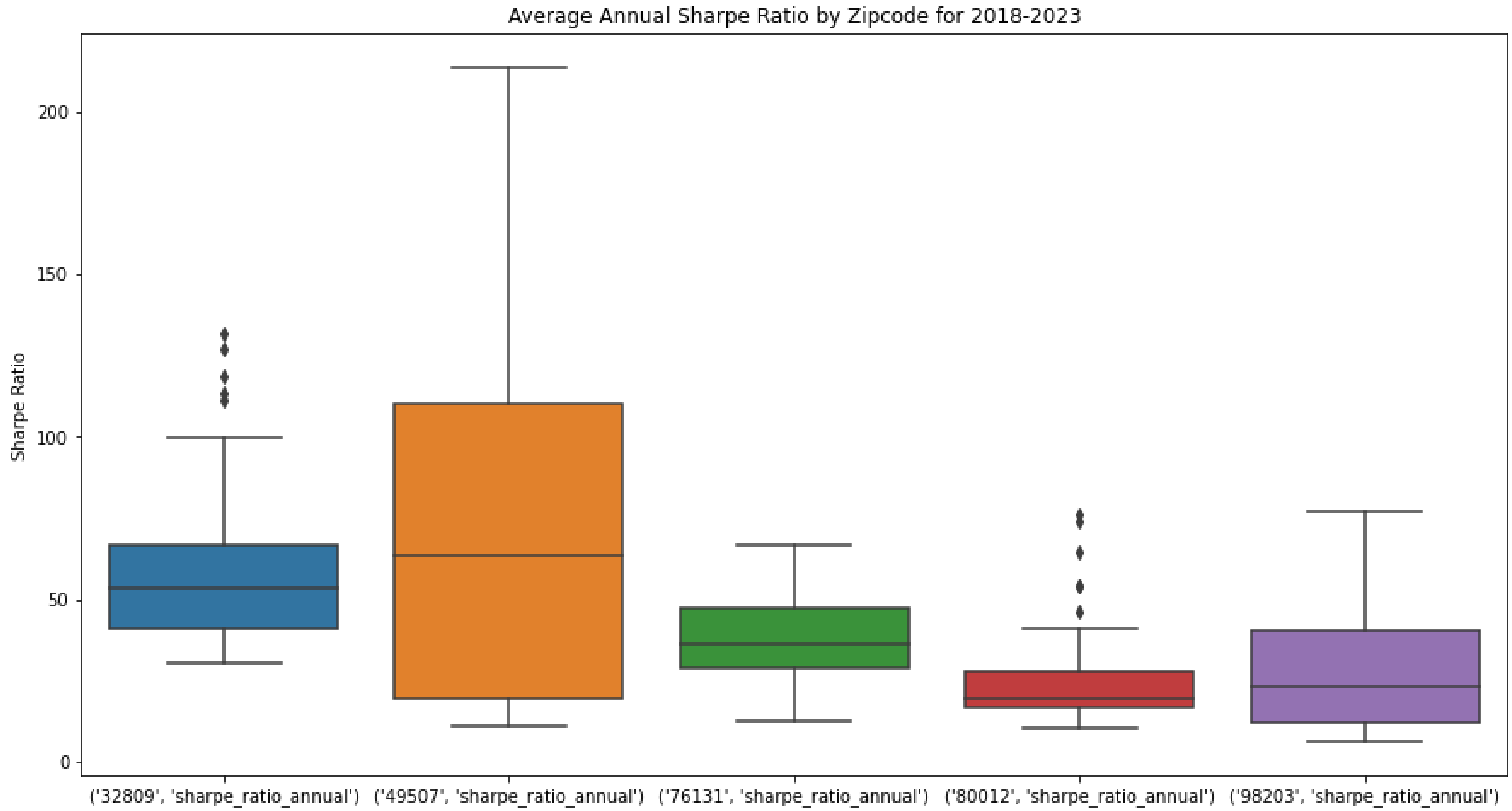
executed in 265ms, finished 15:57:37 2021-03-22



In [77]:

```
fig,ax = plt.subplots(figsize=(15,8))
sns.boxplot(data=results['2018-04':'2023-04'][zipcode_sharpes])
plt.title('Average Annual Sharpe Ratio by Zipcode for 2018-2023')
plt.ylabel('Sharpe Ratio')
plt.savefig('images/forecast_boxplot_5yr')
```

executed in 265ms, finished 15:57:38 2021-03-22



In [78]:

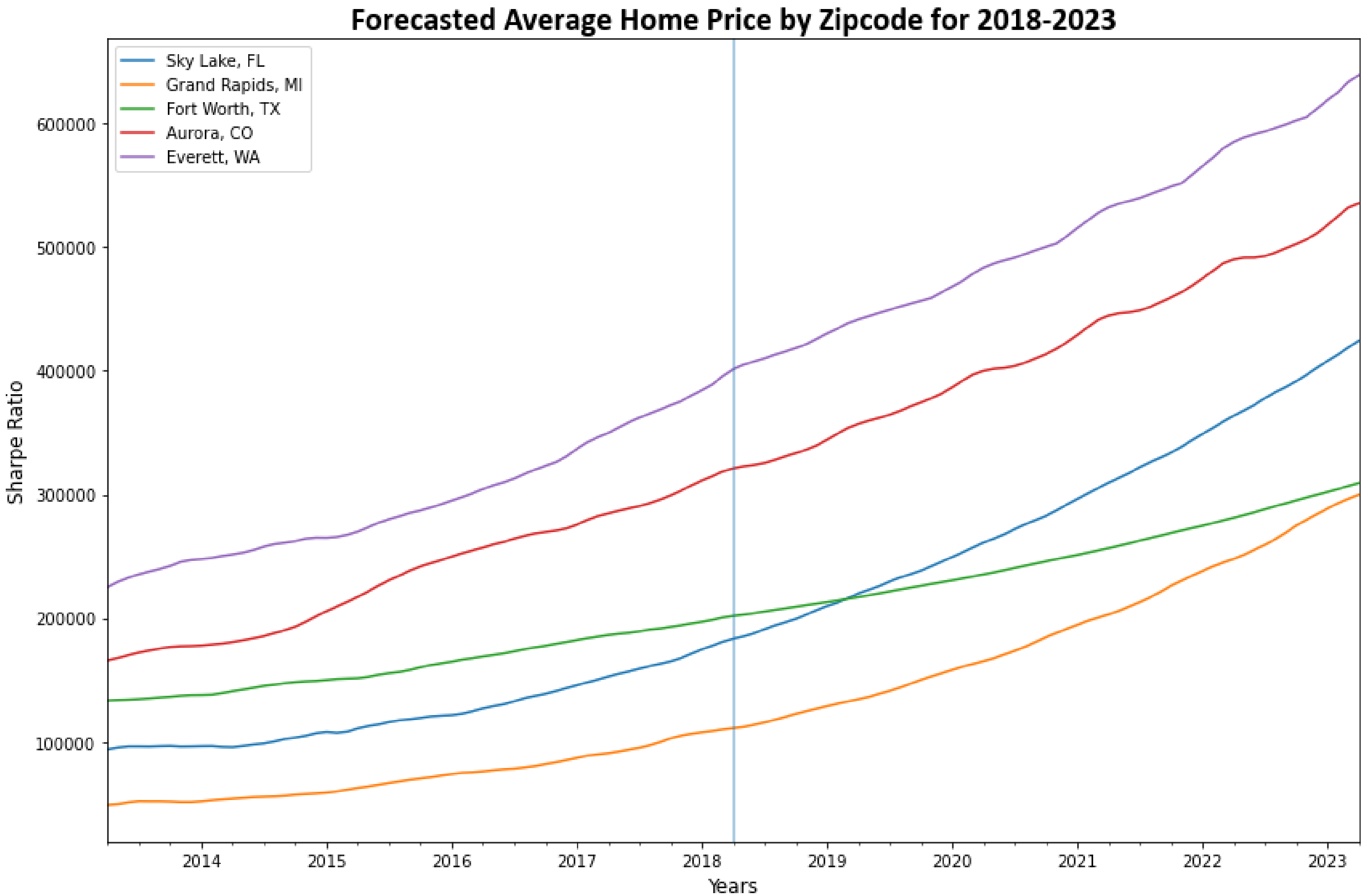
```
legend = ['Sky Lake, FL', 'Grand Rapids, MI', 'Fort Worth, TX', "Aurora, CO",'Everett, WA']
```

executed in 13ms, finished 15:57:39 2021-03-22

In [79]:

```
fig,ax=plt.subplots(figsize=(12,8))
results[final_zips.columns]['2013-04':].plot(ax=ax)
plt.legend(legend)
plt.title('Forecasted Average Home Price by Zipcode for 2018-2023',font=font)
plt.ylabel('Sharpe Ratio',fontsize=12)
plt.xlabel('Years',fontsize=12)
plt.axvline(x='2018-04',alpha=.5)
plt.tight_layout()
plt.savefig('images/price_forecasts')

```

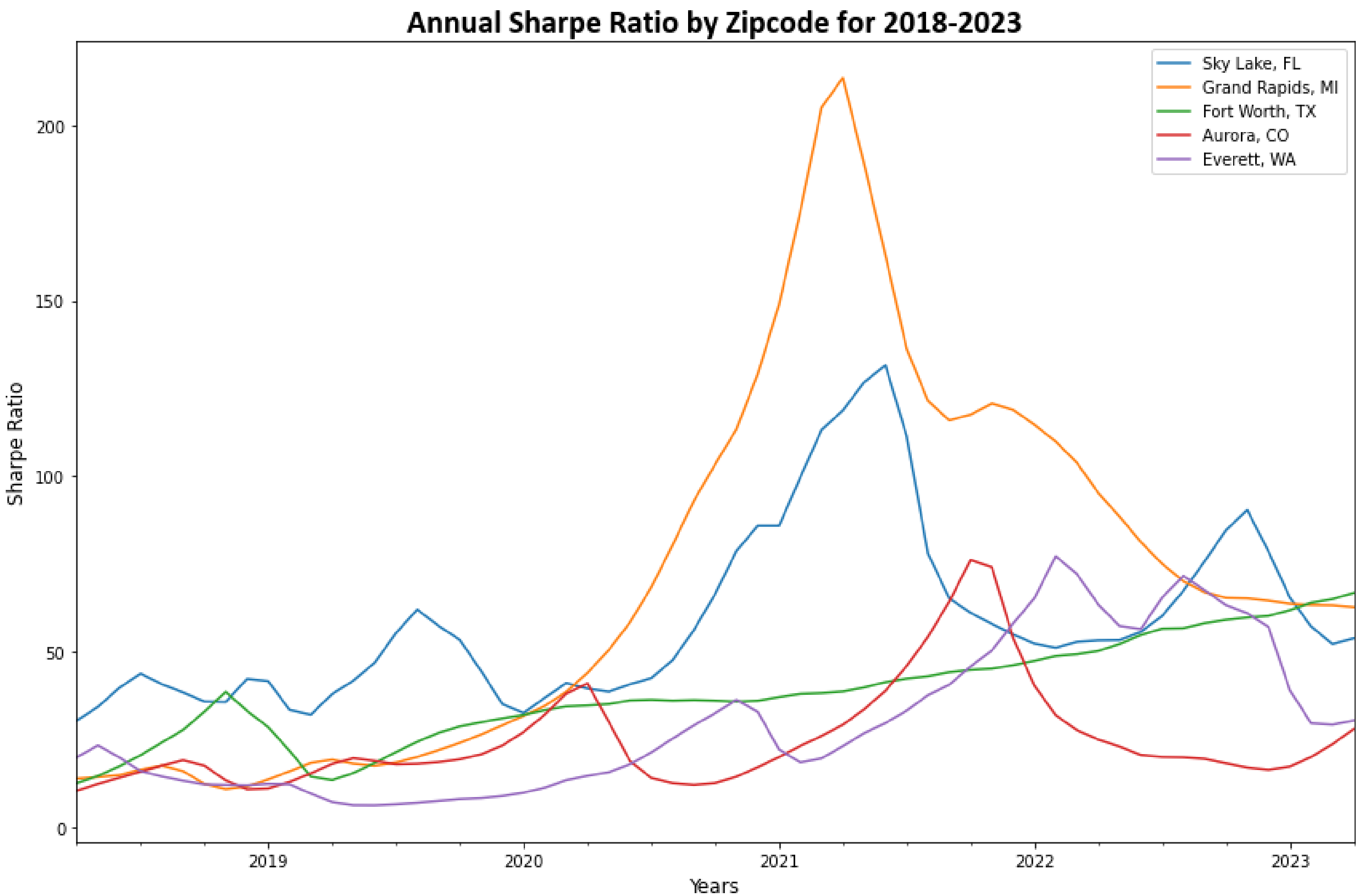
executed in 565ms, finished 15:57:40 2021-03-22


Forecasted Average Home Price by Zipcode for 2018-2023
Sky Lake, FL
Grand Rapids, MI
Fort Worth, TX
Aurora, CO
Everett, WA
Sharpe Ratio
100000
200000
300000
400000
500000
600000
Years
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023

In [80]:

```
fig,ax=plt.subplots(figsize=(12,8))
results[zipcode_sharpes]['2018-04':].plot(ax=ax)
plt.legend(legend)
plt.title('Annual Sharpe Ratio by Zipcode for 2018-2023',font=font)
plt.ylabel('Sharpe Ratio',fontsize=12)
plt.xlabel('Years',fontsize=12)
plt.tight_layout()
plt.savefig('images/sharpe_ratio_forecasts_5yr')
```

executed in 473ms, finished 15:57:42 2021-03-22



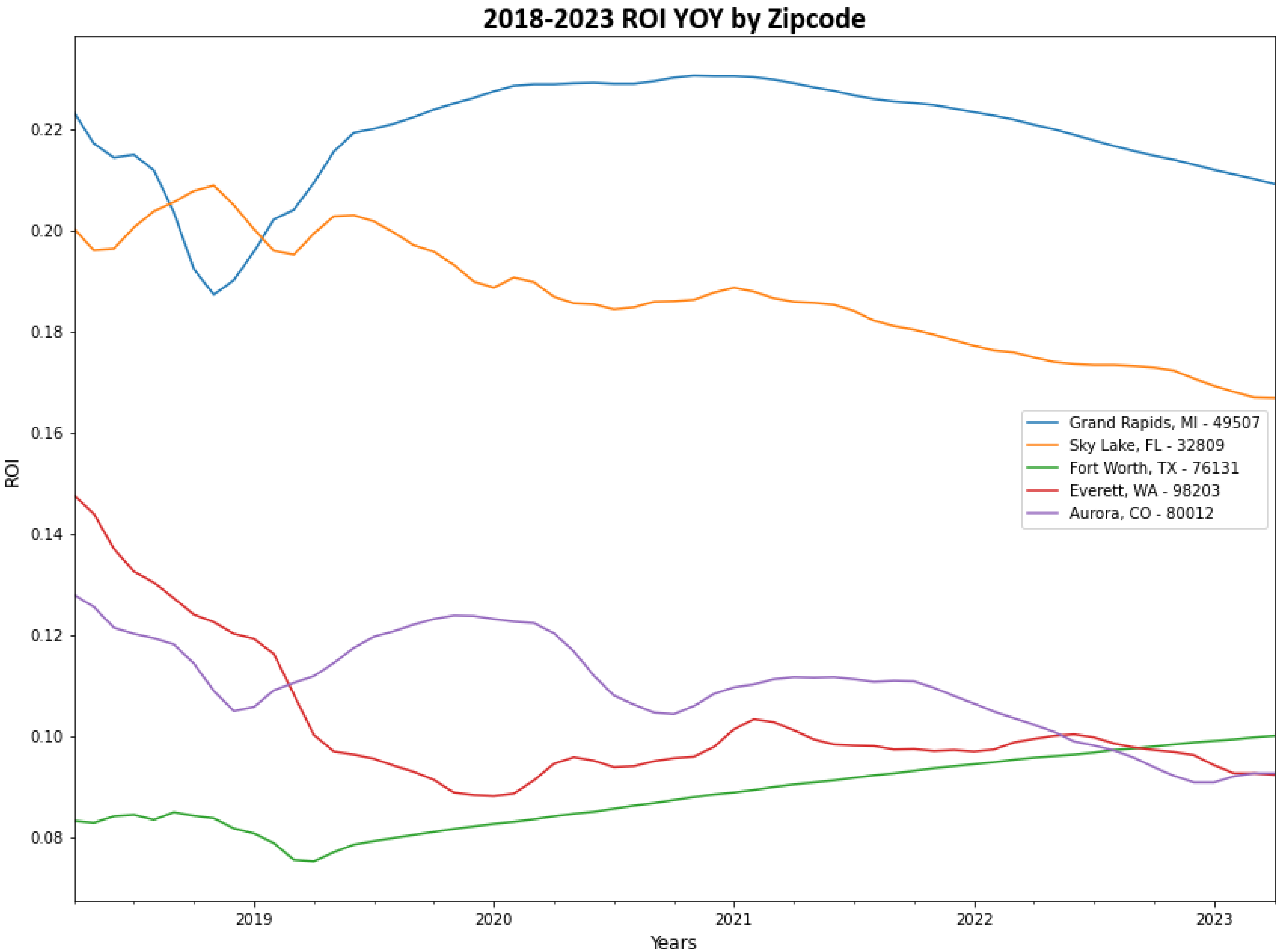
In [81]:
```
legend1 = ['Grand Rapids, MI - 49507','Sky Lake, FL - 32809', 'Fort Worth, TX - 76131', 'Everett, WA - 98203',"Aur
```

executed in 14ms, finished 15:57:42 2021-03-22

In [82]:

```
fig,ax = plt.subplots(figsize=(12,9))
results[roi_yoy]['2018-04':].plot(ax=ax)
plt.title('2018-2023 ROI YOY by Zipcode',font=font)
plt.ylabel('ROI',fontsize=12)
plt.xlabel('Years',fontsize=12)
plt.legend(legend1)
plt.tight_layout()
plt.savefig('images/ROI_5year_forecasts1')
```

executed in 517ms, finished 15:57:44 2021-03-22



In [83]:

```
avg_zips_list = final_zips.columns
```

executed in 13ms, finished 15:57:45 2021-03-22

In [84]:
```
results['2018-04']['32809'].values
```
executed in 15ms, finished 15:57:46 2021-03-22

Out[84]:
```
array([183400.])
```

In [85]:
```
df[df.RegionName.isin(topzips)]
```
executed in 30ms, finished 15:57:47 2021-03-22

Out[85]:

	RegionID	RegionName	City	State	Metro	CountyName	SizeRank	1996-04	1996-05	1996-06	...	2017-07	2017 0
781	93202	80012	Aurora	CO	Denver	Arapahoe	782	111900.0	112000.0	112200.0	...	290600	293
2422	99626	98203	Everett	WA	Seattle	Snohomish	2423	136800.0	136500.0	136300.0	...	362100	365
2689	91274	76131	Fort Worth	TX	Dallas-Fort Worth	Tarrant	2690	117400.0	117300.0	117300.0	...	189400	190
3052	79783	49507	Grand Rapids	MI	Grand Rapids	Kent	3053	49700.0	51000.0	52300.0	...	95300	97
4734	72235	32809	Sky Lake	FL	Orlando	Orange	4735	71700.0	71700.0	71800.0	...	159200	161

5 rows × 272 columns

In [86]:
```
zip_summary=pd.DataFrame(data=None)
ROI_5 = []
ROI_1=[]
ROI_3 = []
for z in avg_zips_list:
    ROI_1.append((results['2019-04'][z].values - results['2018-04'][z].values)/ results['2018-04'][z].value
    ROI_3.append((results['2021-04'][z].values - results['2018-04'][z].values)/ results['2018-04'][z].value
    ROI_5.append((results['2023-04'][z].values - results['2018-04'][z].values)/ results['2018-04'][z].value
zip_summary=pd.concat([pd.DataFrame(data=avg_zips_list,dtype='str'),pd.DataFrame(data = ROI_1,dtype=float),
                       pd.DataFrame(data=ROI_3,dtype=float),pd.DataFrame(data=ROI_5,dtype=float)],axis=1)
zip_summary.columns = ['Zipcode','1 Year ROI', '3 Year ROI', '5 Year ROI']
```
executed in 46ms, finished 15:57:48 2021-03-22

In [87]:
```
zip_summary.set_index('Zipcode',inplace=True)
```
executed in 13ms, finished 15:57:50 2021-03-22

In [88]:
```
zip_summary
```
executed in 14ms, finished 15:57:52 2021-03-22

Out[88]:

	1 Year ROI	3 Year ROI	5 Year ROI
Zipcode			
32809	0.199277	0.687731	1.313411
49507	0.209263	0.826214	1.695759
76131	0.075239	0.270982	0.531878
80012	0.111807	0.384553	0.667280
98203	0.100225	0.325942	0.592251

In [89]:
```
zip_summary = zip_summary*100
```
executed in 13ms, finished 15:57:53 2021-03-22

In [90]:
```
zip_summary
```
executed in 14ms, finished 15:57:54 2021-03-22

Out[90]:

	1 Year ROI	3 Year ROI	5 Year ROI
Zipcode			
32809	19.927666	68.773075	131.341128
49507	20.926289	82.621367	169.575863
76131	7.523901	27.098230	53.187808
80012	11.180724	38.455272	66.728049
98203	10.022461	32.594237	59.225062

In [91]:
```
zip_sharpes=pd.DataFrame(data=None)
SR_1 = []
SR_3=[]
SR_5 = []
for z in avg_zips_list:
    SR_1.append((((results['2019-04'][z].values - results['2018-04'][z].values)/ results['2018-04'][z].valu
    SR_3.append((((results['2021-04'][z].values - results['2018-04'][z].values)/ results['2018-04'][z].valu
    SR_5.append((((results['2023-04'][z].values - results['2018-04'][z].values)/ results['2018-04'][z].valu

zip_sharpes=pd.concat([pd.DataFrame(data=avg_zips_list,dtype='str'),pd.DataFrame(data = SR_1,dtype=float),
                       pd.DataFrame(data=SR_3,dtype=float),pd.DataFrame(data=SR_5,dtype=float)],axis=1)
zip_sharpes.columns = ['zipcode','1 Year Sharpe Ratio', '3 Year Sharpe Ratio', '5 Year Sharpe Ratio']
```

executed in 77ms, finished 15:57:55 2021-03-22

In [92]:
```
zip_sharpes.set_index('zipcode',inplace=True)
```

executed in 13ms, finished 15:57:56 2021-03-22

In [93]:
```
zip_sharpes
```

executed in 14ms, finished 15:57:57 2021-03-22

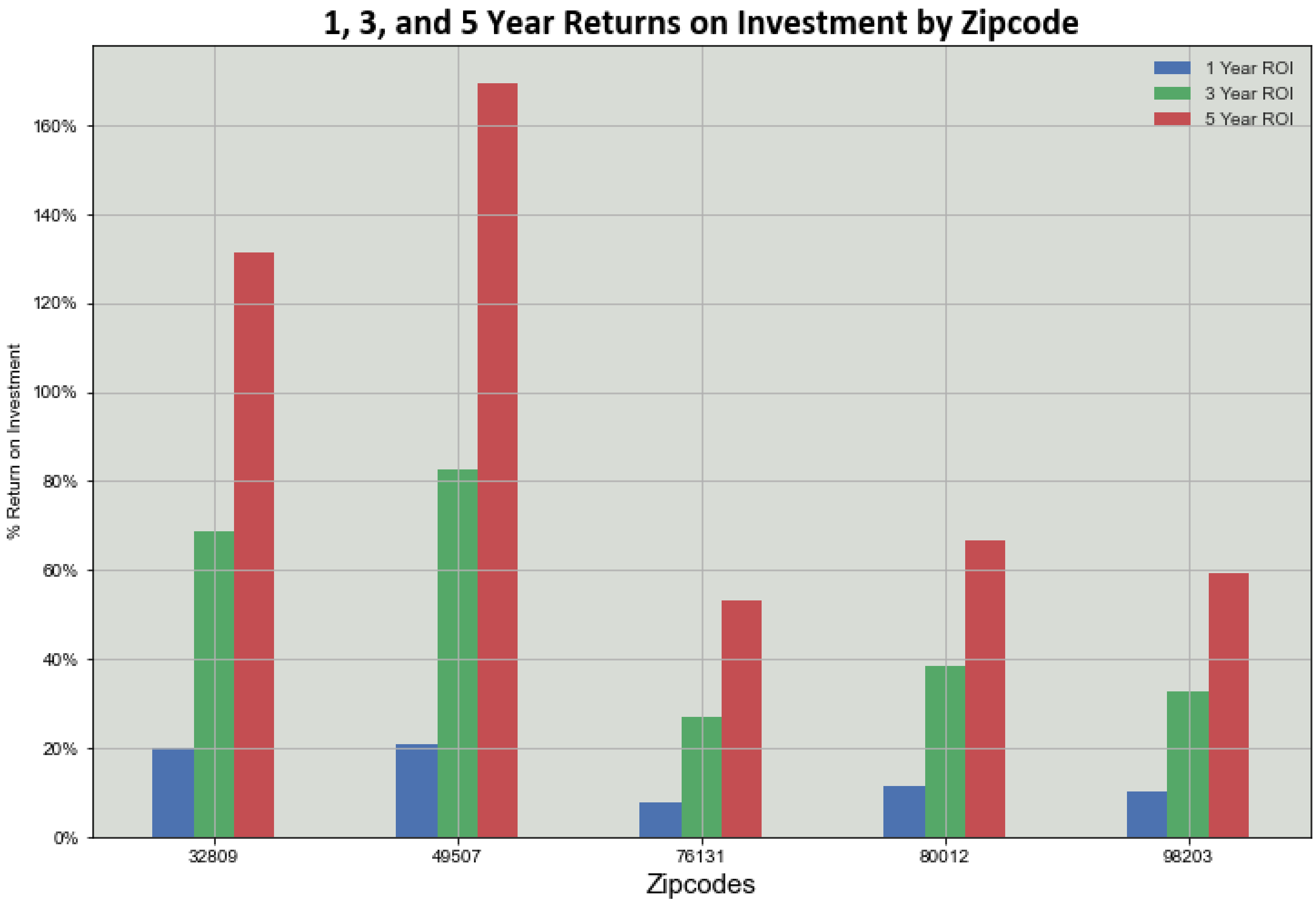
Out[93]:

	1 Year Sharpe Ratio	3 Year Sharpe Ratio	5 Year Sharpe Ratio
zipcode			
32809	36.798441	87.636488	112.848365
49507	16.076224	62.448357	157.689890
76131	14.412070	64.116214	72.961911
80012	11.283999	51.213698	65.102097
98203	5.424561	17.684846	41.029424

In [94]:

```
import matplotlib.ticker as mtick
fig, ax = plt.subplots()
plt.style.use('seaborn')
ax.yaxis.set_major_formatter(mtick.PercentFormatter())
ax.set_facecolor('#d8dcd6')
zip_summary.plot(kind='bar',figsize=(12,8), ax=ax)
plt.title('1, 3, and 5 Year Returns on Investment by Zipcode',font=font)
plt.xticks(rotation=0)
plt.xlabel('Zipcodes',size=15)
plt.ylabel('% Return on Investment')

plt.savefig('images/ROI_summary_barplot')
```

executed in 365ms, finished 15:57:59 2021-03-22


1, 3, and 5 Year Returns on Investment by Zipcode
1 Year ROI
3 Year ROI
5 Year ROI
% Return on Investment
0%
20%
40%
60%
80%
100%
120%
140%
160%
32809
49507
76131
80012
98203
Zipcodes

In [95]:

```python

fig, ax = plt.subplots()
plt.style.use('seaborn')
ax.set_facecolor('#d8dcd6')
zip_sharpes.plot(kind='bar',figsize=(12,8), ax=ax)
plt.title('1, 3, and 5 Year Sharpe Ratios by Zipcode',font=font)
plt.xticks(rotation=0)
plt.xlabel('Zipcodes',size=15)
plt.ylabel('Sharpe Ratio')

plt.savefig('images/ROI_summary_barplot')
```

executed in 361ms, finished 15:58:00 2021-03-22