House Price Prediction

Submitted By:

Nilanjana Chatterjee

Rahul Jala

Alli Manideep Yadav

In [30]:

```
import itertools
import pandas as pd
import numpy as np

#for modeling
import statsmodels.api as sm
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
from statsmodels.tsa.statespace.sarimax import SARIMAX
from statsmodels.tsa.stattools import adfuller
from statsmodels.tsa.seasonal import seasonal_decompose as sd
from sklearn.metrics import mean_squared_error
from sklearn.linear_model import LassoLarsCV

#for visualizations
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from matplotlib.pylab import rcParams
import time
```

In [31]:
```
#Import data from zillow https://www.zillow.com/research/data/.
#After that we have checked the head (first 5 rows) of the data set

dataSet = pd.read_csv('County_zillow_month.csv')
dataSet.head()
```

Out[31]:

	RegionID	SizeRank	RegionName	RegionType	StateName	State	Metro	StateCodeFIPS	MunicipalCodeFIPS	2000-01-31	...	2022-02-28	2022-03-31	2022-04-30	2022-05-31	2022-06-30	2022-07-31	2022-08-31	2022-09-30	2022-10-31	2022-11-30
0	3101	0	Los Angeles County	county	CA	CA	Los Angeles-Long Beach-Anaheim, CA	6	37	216545.0	...	832178.0	846832.0	864584.0	879885.0	881568.0	881064.0	867874.0	857204.0	845637.0	843637.0
1	139	1	Cook County	county	IL	IL	Chicago-Naperville-Elgin, IL-IN-WI	17	31	174229.0	...	299175.0	301732.0	303877.0	307321.0	310111.0	312096.0	312582.0	313144.0	313615.0	314365.0
2	1090	2	Harris County	county	TX	TX	Houston-The Woodlands-Sugar Land, TX	48	201	115634.0	...	266079.0	270805.0	275917.0	280922.0	284786.0	287343.0	288902.0	289270.0	290016.0	290595.0
3	2402	3	Maricopa County	county	AZ	AZ	Phoenix-Mesa-Chandler, AZ	4	13	142913.0	...	448385.0	457624.0	469152.0	478023.0	484372.0	481959.0	475364.0	465435.0	460400.0	455305.0
4	2841	4	San Diego County	county	CA	CA	San Diego-Chula Vista-Carlsbad, CA	6	73	225245.0	...	864585.0	887571.0	906792.0	919951.0	919417.0	908115.0	897043.0	890055.0	884988.0	880863.0

5 rows × 284 columns

In [32]:
```
# As we want to find the house price in the State of New Jersey. We would need to filter the data for NJ from DataSet.
# We need only timeseries on NJ counties hence removing other column to get a clean dataset.
nj=dataSet[(dataSet['State']=='NJ')]
nj1 = nj.drop(['RegionID','SizeRank','RegionType','StateName','State','Metro','StateCodeFIPS','MunicipalCodeFIPS'], axis ='columns')
nj1.head()
```

Out[32]:

	RegionName	2000-01-31	2000-02-29	2000-03-31	2000-04-30	2000-05-31	2000-06-30	2000-07-31	2000-08-31	2000-09-30	...	2022-02-28	2022-03-31	2022-04-30	2022-05-31	2022-06-30	2022-07-31	2022-08-31	2022-09-30	2022-10-31	2022-11-30
55	Bergen County	258624.0	259696.0	260852.0	263059.0	265262.0	268219.0	270680.0	273143.0	275297.0	...	596313.0	600605.0	605905.0	610075.0	614951.0	619399.0	621760.0	622560.0	620656.0	622297.0
71	Middlesex County	179676.0	180280.0	181067.0	182583.0	184532.0	186564.0	188911.0	190786.0	193166.0	...	429944.0	434852.0	440191.0	446549.0	451035.0	455874.0	457419.0	459886.0	461451.0	465069.0
78	Essex County	201868.0	203028.0	204085.0	205670.0	206938.0	208373.0	210270.0	212628.0	214807.0	...	522030.0	528628.0	536057.0	544644.0	551314.0	556778.0	558547.0	559054.0	561264.0	565603.0
99	Hudson County	185856.0	186979.0	187821.0	189586.0	191389.0	192471.0	193805.0	194960.0	197023.0	...	552417.0	558681.0	566875.0	574035.0	580728.0	585279.0	587686.0	585634.0	587725.0	593026.0
106	Monmouth County	216931.0	217725.0	218605.0	220437.0	222564.0	224896.0	227670.0	232104.0	236327.0	...	581711.0	589237.0	598660.0	610382.0	619669.0	626352.0	628840.0	630735.0	631369.0	634207.0

5 rows × 276 columns

In [33]:
```
# Reshape from Wide to Long Format & Data Processing. Basically we need the date row wise instead of column wise.
# Melt funtion is responsible for transposing the dataframe.

def melt_data(df):
    melted = pd.melt(df, id_vars=['RegionName'], var_name='Month', value_name = 'MeanValue')
    melted['Month'] = pd.to_datetime(melted['Month'], format = '%Y-%m')
    melted = melted.dropna(subset=['MeanValue'])
    return melted
```

In [34]:
```
dfm = melt_data(nj1)
print(dfm.head())
print(dfm.info())
```

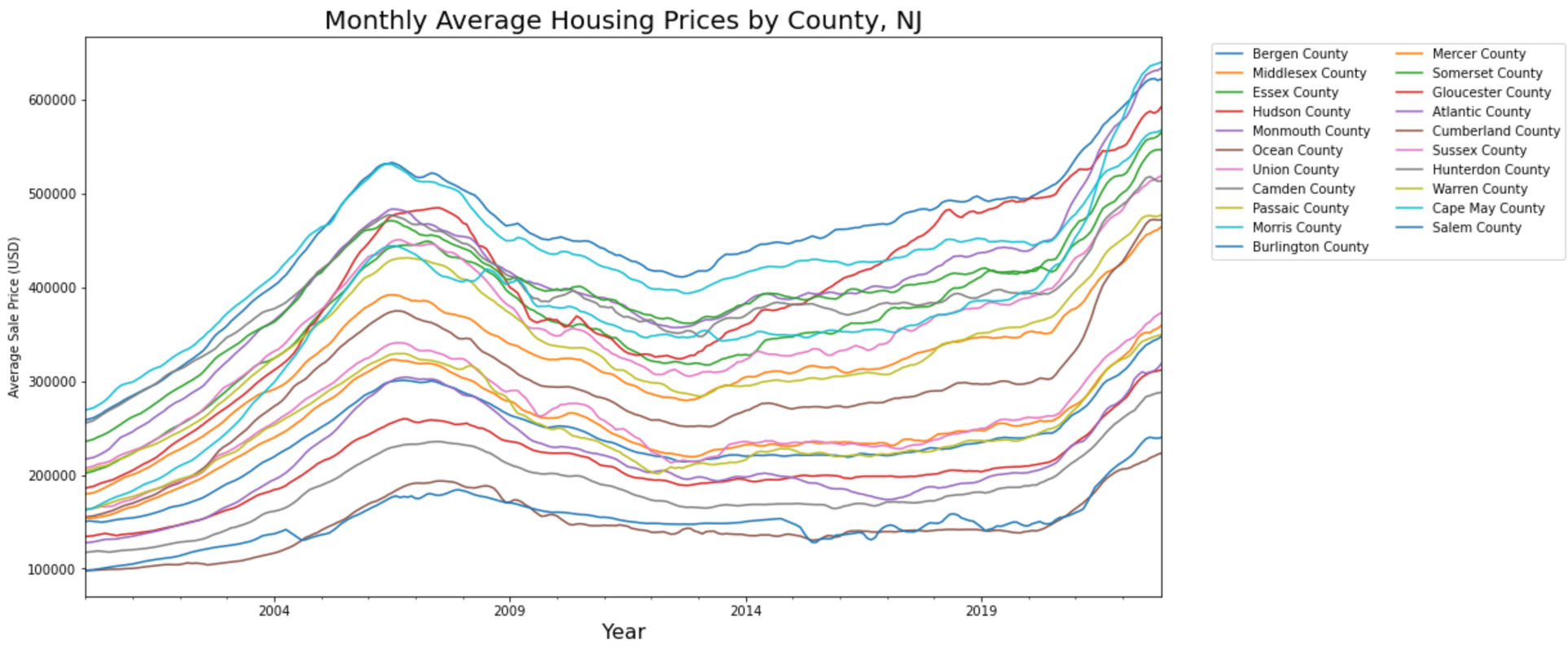
```
         RegionName      Month  MeanValue
0     Bergen County 2000-01-31   258624.0
1  Middlesex County 2000-01-31   179676.0
2      Essex County 2000-01-31   201868.0
3     Hudson County 2000-01-31   185856.0
4   Monmouth County 2000-01-31   216931.0
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5766 entries, 0 to 5774
Data columns (total 3 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   RegionName  5766 non-null   object
 1   Month       5766 non-null   datetime64[ns]
 2   MeanValue   5766 non-null   float64
dtypes: datetime64[ns](1), float64(1), object(1)
memory usage: 180.2+ KB
None
```

In [35]:
```
# setting Month as index so that We can create graphical presentation of the timeseries.
dfm.set_index('Month', inplace = True)
dfm.head()
```

Out[35]:

Month	RegionName	MeanValue
2000-01-31	Bergen County	258624.0
2000-01-31	Middlesex County	179676.0
2000-01-31	Essex County	201868.0
2000-01-31	Hudson County	185856.0
2000-01-31	Monmouth County	216931.0

In [36]:
```
#EDA and Visualization. Creating line graph comparison of the county Monthly Average house prices.

for county in dfm.RegionName.unique():
    temp_df = dfm[dfm.RegionName == county]
    temp_df['MeanValue'].plot(figsize = (15,8), label=county)

plt.legend(bbox_to_anchor=(1.04,1), loc='upper left', ncol=2)
plt.xlabel("Year", fontsize = 16)
plt.ylabel("Average Sale Price (USD)")
plt.title('Monthly Average Housing Prices by County, NJ', fontsize = 20);
```



In [37]:
```
# SARIMA models on all NJ County
county_dfs = []
county_list = dfm.RegionName.unique()
for county in county_list:
    county_dfs.append(pd.DataFrame(dfm[dfm['RegionName']==county][['MeanValue']].copy()))
#p: Seasonal autoregressive order.
#d: Seasonal difference order.
#q: Seasonal moving average order.
#m: The number of time steps for a single seasonal period.
# Define the p, d and q parameters to take any value between 0 and 2
p = d = q = range(0,2)
# Generate all different combinations of p, d and q triplets
pdq = list(itertools.product(p,d,q))
# Generate all different combinations of seasonal p, d and q triplets
pdqs = [(x[0], x[1], x[2], 12) for x in list(itertools.product(p, d, q))]
```

In [38]:
```
#Run SARIMA
#The implementation is called SARIMAX instead of SARIMA because the "X" addition to the method name means
#that the implementation also supports exogenous variables.
start=time.time()
ans = []

for df, name in zip(county_dfs, county_list):
    for p1 in pdq:
        for p2 in pdqs:
            try:
                mod = sm.tsa.statespace.SARIMAX(df,
                                                order = p1,
                                                seasonal_order = p2,
                                                enforce_stationarity = False,
                                                enforce_invertibility = False)
                output = mod.fit()
                ans.append([name, p1, p2, output.aic])
                print('Result for {}'.format(name) + ' SARIMAX {} x {}12 : AIC Calculated = {}'
                      .format(p1, p2, output.aic))
            except:
                continue
```

```
   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    2     23      43      1     1     0   2.135D-07   1.071D+01
  F =   10.710037520693547

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
Result for Bergen County SARIMAX (0, 0, 0) x (0, 1, 1, 12)12 : AIC Calculated = 5894.5206363814505
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            2     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  1.11557D+01    |proj g|=  1.22515D-03

           * * *
```

The Akaike Information Critera (AIC) is a widely used measure of a statistical model. It basically quantifies 1) the goodness of fit, and 2) the simplicity/parsimony, of the model into a single statistic.

When comparing two models, the one with the lower AIC is generally "better".

In [39]:
```
result = pd.DataFrame(ans, columns = ['name','pdq','pdqs','AIC'])
#Return the best set of parameters based on AIC
best_parameter = result.loc[result.groupby("name")["AIC"].idxmin()]

with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    print(best_parameter)
```

	name	pdq	pdqs	AIC
959	Atlantic County	(1, 1, 1)	(1, 1, 1, 12)	4168.300597
63	Bergen County	(1, 1, 1)	(1, 1, 1, 12)	4344.584383
703	Burlington County	(1, 1, 1)	(1, 1, 1, 12)	4022.838606
511	Camden County	(1, 1, 1)	(1, 1, 1, 12)	4018.001440
1270	Cape May County	(1, 1, 0)	(1, 1, 0, 12)	4440.896047
1023	Cumberland County	(1, 1, 1)	(1, 1, 1, 12)	4189.474484
191	Essex County	(1, 1, 1)	(1, 1, 1, 12)	4373.683934
895	Gloucester County	(1, 1, 1)	(1, 1, 1, 12)	4034.136611
255	Hudson County	(1, 1, 1)	(1, 1, 1, 12)	4626.550609
1151	Hunterdon County	(1, 1, 1)	(1, 1, 1, 12)	4408.808521
767	Mercer County	(1, 1, 1)	(1, 1, 1, 12)	4159.632993
127	Middlesex County	(1, 1, 1)	(1, 1, 1, 12)	4206.663239
319	Monmouth County	(1, 1, 1)	(1, 1, 1, 12)	4313.665524
639	Morris County	(1, 1, 1)	(1, 1, 1, 12)	4322.464568
383	Ocean County	(1, 1, 1)	(1, 1, 1, 12)	4184.994240
575	Passaic County	(1, 1, 1)	(1, 1, 1, 12)	4174.329506
1343	Salem County	(1, 1, 1)	(1, 1, 1, 12)	4373.533279
831	Somerset County	(1, 1, 1)	(1, 1, 1, 12)	4361.910767
1087	Sussex County	(1, 1, 1)	(1, 1, 1, 12)	4328.902215
447	Union County	(1, 1, 1)	(1, 1, 1, 12)	4380.970839
1215	Warren County	(1, 1, 1)	(1, 1, 1, 12)	4245.339974

In [40]:
```
# Output the whole cell
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# plug the optimal parameter values into a new SARIMAX model. This is to check that model does not have
# any unexpected errors with optimal parameters.

for name, pdq, pdqs, df in zip(best_parameter['name'], best_parameter['pdq'], best_parameter['pdqs'], county_dfs):

    SARIMAX_MODEL = sm.tsa.SARIMAX(df,
                            order = pdq,
                            seasonal_order = pdqs,
                            enforce_stationarity = False,
                            enforce_invertibility = False)

    output = SARIMAX_MODEL.fit()
    print('SARIMAX Model Result for {}'.format(name))
    print(output.summary().tables[1])


# Fit the model and print results
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.94747D+00    |proj g|=  1.93868D-01

At iterate    5    f=  7.90316D+00    |proj g|=  4.57387D-02

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
At iterate   10    f=  7.90307D+00    |proj g|=  2.03891D-04

At iterate   15    f=  7.90306D+00    |proj g|=  1.22891D-03

At iterate   20    f=  7.90297D+00    |proj g|=  1.70119D-02

At iterate   25    f=  7.88316D+00    |proj g|=  7.88999D-02

At iterate   30    f=  7.88106D+00    |proj g|=  5.60490D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     31     36      1     0     0   3.412D-05   7.881D+00
  F =   7.8810625140986934

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Atlantic County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8126      0.046     17.723      0.000       0.723       0.902
ma.L1          0.0742      0.090      0.824      0.410      -0.102       0.251
ar.S.L12      -0.5817      0.056    -10.389      0.000      -0.691      -0.472
ma.S.L12      -0.0097      0.018     -0.535      0.592      -0.045       0.026
sigma2       2.28e+06   1.89e+05     12.037      0.000    1.91e+06    2.65e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.70605D+00    |proj g|=  3.80994D-01


/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
At iterate    5    f=  7.63306D+00    |proj g|=  4.33096D-02

At iterate   10    f=  7.63030D+00    |proj g|=  3.43314D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     12     15      1     0     0   1.424D-04   7.630D+00
  F =   7.6302967977782332

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Bergen County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8379      0.049     17.111      0.000       0.742       0.934
ma.L1          0.0676      0.094      0.719      0.472      -0.117       0.252
ar.S.L12      -0.4360      0.057     -7.640      0.000      -0.548      -0.324
ma.S.L12      -0.0479      0.043     -1.127      0.260      -0.131       0.035
sigma2      1.592e+06   1.41e+05     11.256      0.000    1.31e+06    1.87e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.02822D+00    |proj g|=  5.34154D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
At iterate    5    f=  7.93399D+00    |proj g|=  4.57795D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5      9     11      1     0     0   1.525D-04   7.934D+00
  F =   7.9339707892830980

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Burlington County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7695      0.074     10.370      0.000       0.624       0.915
ma.L1          0.1473      0.092      1.603      0.109      -0.033       0.327
ar.S.L12      -0.6228      0.074     -8.389      0.000      -0.768      -0.477
ma.S.L12      -0.0173      0.043     -0.399      0.690      -0.102       0.068
sigma2      3.517e+06   4.28e+05      8.219      0.000    2.68e+06    4.36e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.55408D+00    |proj g|=  8.89204D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated
frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated
frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
 This problem is unconstrained.
```

```
At iterate    5    f=  8.46271D+00    |proj g|=  8.12953D-02

At iterate   10    f=  8.45523D+00    |proj g|=  2.15456D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     13     15      1     0     0   6.467D-05   8.455D+00
  F =   8.4552208960414053

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Camden County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7722      0.073     10.567      0.000       0.629       0.915
ma.L1         -0.1825      0.094     -1.934      0.053      -0.367       0.002
ar.S.L12      -0.5030      0.055     -9.177      0.000      -0.610      -0.396
ma.S.L12      -0.0495      0.053     -0.936      0.349      -0.153       0.054
sigma2      1.042e+07   5.71e+05     18.237      0.000     9.3e+06    1.15e+07
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            3     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.92671D+00    |proj g|=  2.97227D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
At iterate    5    f=  7.86543D+00    |proj g|=  3.66114D-04

At iterate   10    f=  7.86510D+00    |proj g|=  4.07097D-02

At iterate   15    f=  7.84444D+00    |proj g|=  2.31249D-01

At iterate   20    f=  7.83731D+00    |proj g|=  1.26231D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    3     20     26      1     0     0   1.262D-05   7.837D+00
  F =   7.8373092441552528

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Cape May County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8938      0.024     36.509      0.000       0.846       0.942
ar.S.L12      -0.6173      0.045    -13.822      0.000      -0.705      -0.530
sigma2      1.931e+06   1.58e+05     12.239      0.000    1.62e+06    2.24e+06
==============================================================================

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.88051D+00    |proj g|=  7.31078D-01

At iterate    5    f=  7.61048D+00    |proj g|=  5.15481D-03

At iterate   10    f=  7.61046D+00    |proj g|=  1.87763D-04

At iterate   15    f=  7.61045D+00    |proj g|=  4.86511D-03

At iterate   20    f=  7.60835D+00    |proj g|=  6.22124D-02

At iterate   25    f=  7.59097D+00    |proj g|=  1.65363D-02

At iterate   30    f=  7.59090D+00    |proj g|=  1.55765D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     30     35      1     0     0   1.558D-05   7.591D+00
  F =   7.5908986188576302

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Cumberland County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8932      0.024     37.038      0.000       0.846       0.940
ma.L1          0.1618      0.060      2.687      0.007       0.044       0.280
ar.S.L12      -0.5618      0.048    -11.651      0.000      -0.656      -0.467
ma.S.L12      -0.0171      0.030     -0.567      0.571      -0.076       0.042
sigma2      1.195e+06   9.41e+04     12.693      0.000    1.01e+06    1.38e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.01361D+00    |proj g|=  1.39409D-01
```

```
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.


At iterate    5    f=  7.97640D+00    |proj g|=  2.99191D-03

At iterate   10    f=  7.97638D+00    |proj g|=  4.48700D-03

At iterate   15    f=  7.97519D+00    |proj g|=  6.59970D-02

At iterate   20    f=  7.94982D+00    |proj g|=  1.53388D-01

At iterate   25    f=  7.94722D+00    |proj g|=  2.25121D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     25     31      1     0     0   2.251D-05   7.947D+00
  F =   7.9472197077704747

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Essex County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7452      0.048     15.380      0.000       0.650       0.840
ma.L1          0.2212      0.045      4.880      0.000       0.132       0.310
ar.S.L12      -0.6113      0.053    -11.545      0.000      -0.715      -0.507
ma.S.L12      -0.0091      0.024     -0.375      0.707      -0.056       0.038
sigma2      2.633e+06   2.36e+05     11.161      0.000    2.17e+06     3.1e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.35140D+00    |proj g|=  2.95483D-01
```

```
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.


At iterate    5    f=  7.29562D+00    |proj g|=  3.96329D-02

At iterate   10    f=  7.29444D+00    |proj g|=  1.76437D-04

At iterate   15    f=  7.29444D+00    |proj g|=  1.62379D-03

At iterate   20    f=  7.29407D+00    |proj g|=  2.08746D-02

At iterate   25    f=  7.28731D+00    |proj g|=  6.47484D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     29     34      1     0     0   1.065D-05   7.287D+00
  F =   7.2872753462820850

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Gloucester County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8561      0.043     20.094      0.000       0.773       0.940
ma.L1         -0.0640      0.064     -1.003      0.316      -0.189       0.061
ar.S.L12      -0.4568      0.065     -6.975      0.000      -0.585      -0.328
ma.S.L12      -0.0321      0.034     -0.959      0.338      -0.098       0.034
sigma2      6.098e+05    4.1e+04     14.862      0.000    5.29e+05     6.9e+05
==============================================================================

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.69602D+00    |proj g|=  4.85344D-01

At iterate    5    f=  7.57400D+00    |proj g|=  1.01259D-01

At iterate   10    f=  7.57151D+00    |proj g|=  1.74044D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     11     15      1     0     0   1.740D-04   7.572D+00
  F =   7.5715081925689267
CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Hudson County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8590      0.045     19.044      0.000       0.771       0.947
ma.L1          0.1485      0.069      2.159      0.031       0.014       0.283
ar.S.L12      -0.5214      0.081     -6.458      0.000      -0.680      -0.363
ma.S.L12      -0.0175      0.035     -0.497      0.619      -0.086       0.051
sigma2      1.445e+06   1.73e+05      8.348      0.000    1.11e+06    1.78e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.87258D+00    |proj g|=  1.88861D-01

At iterate    5    f=  7.84095D+00    |proj g|=  3.99296D-02
```

```
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.


At iterate   10    f=  7.84084D+00    |proj g|=  1.33931D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     11     15      1     0     0   2.700D-04   7.841D+00
  F =   7.8408446696271730

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Hunterdon County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8394      0.059     14.281      0.000       0.724       0.955
ma.L1          0.0488      0.106      0.460      0.646      -0.159       0.257
ar.S.L12      -0.5085      0.071     -7.143      0.000      -0.648      -0.369
ma.S.L12      -0.0180      0.027     -0.677      0.499      -0.070       0.034
sigma2      2.652e+06   2.89e+05      9.172      0.000    2.09e+06    3.22e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.47673D+00    |proj g|=  4.82285D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
At iterate    5    f=  7.31048D+00    |proj g|=  3.63602D-02

At iterate   10    f=  7.30911D+00    |proj g|=  2.73365D-04

At iterate   15    f=  7.30911D+00    |proj g|=  1.94848D-03

At iterate   20    f=  7.30871D+00    |proj g|=  2.47139D-02

At iterate   25    f=  7.29925D+00    |proj g|=  1.57599D-01

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     29     33      1     0     0   1.824D-04   7.296D+00
  F =   7.2960701935591903

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Mercer County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8659      0.028     30.393      0.000       0.810       0.922
ma.L1          0.0180      0.064      0.279      0.780      -0.108       0.144
ar.S.L12      -0.4998      0.051     -9.765      0.000      -0.600      -0.399
ma.S.L12      -0.0408      0.034     -1.187      0.235      -0.108       0.027
sigma2      6.203e+05   4.23e+04     14.670      0.000    5.37e+05    7.03e+05
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.62615D+00    |proj g|=  3.24122D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
At iterate    5    f=  7.57444D+00    |proj g|=  8.31412D-02

At iterate   10    f=  7.57337D+00    |proj g|=  3.15944D-04

At iterate   15    f=  7.57336D+00    |proj g|=  3.30770D-03

At iterate   20    f=  7.57305D+00    |proj g|=  3.96847D-02

At iterate   25    f=  7.55411D+00    |proj g|=  1.81029D-01

At iterate   30    f=  7.54480D+00    |proj g|=  3.41501D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     34     37      1     0     0   4.343D-06   7.545D+00
  F =   7.5447872592530656

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
SARIMAX Model Result for Middlesex County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8358      0.038     21.811      0.000       0.761       0.911
ma.L1          0.0352      0.061      0.573      0.567      -0.085       0.155
ar.S.L12      -0.6098      0.049    -12.439      0.000      -0.706      -0.514
ma.S.L12      -0.0348      0.033     -1.056      0.291      -0.099       0.030
sigma2      1.079e+06   8.44e+04     12.788      0.000    9.14e+05    1.24e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.01589D+00    |proj g|=  1.01412D+00

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
At iterate    5    f=  7.91329D+00    |proj g|=  3.72458D-02

At iterate   10    f=  7.91257D+00    |proj g|=  6.16075D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     12     16      1     0     0   1.580D-04   7.913D+00
  F =   7.9125650301851351

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Monmouth County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7813      0.059     13.240      0.000       0.666       0.897
ma.L1          0.1613      0.092      1.760      0.078      -0.018       0.341
ar.S.L12      -0.6099      0.078     -7.793      0.000      -0.763      -0.457
ma.S.L12      -0.0089      0.042     -0.210      0.833      -0.091       0.074
sigma2      3.372e+06   4.21e+05      8.010      0.000    2.55e+06     4.2e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.37238D+00    |proj g|=  2.13474D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
At iterate    5    f=  7.33211D+00    |proj g|=  4.74294D-02

At iterate   10    f=  7.33145D+00    |proj g|=  2.29360D-04

At iterate   15    f=  7.33144D+00    |proj g|=  7.12039D-03

At iterate   20    f=  7.33046D+00    |proj g|=  8.11149D-02

At iterate   25    f=  7.31824D+00    |proj g|=  9.51673D-02

At iterate   30    f=  7.31661D+00    |proj g|=  2.00788D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     30     36      1     0     0   2.008D-05   7.317D+00
  F =   7.3166120192210826

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Morris County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8479      0.038     22.476      0.000       0.774       0.922
ma.L1          0.0117      0.072      0.161      0.872      -0.130       0.154
ar.S.L12      -0.5007      0.062     -8.027      0.000      -0.623      -0.378
ma.S.L12      -0.0265      0.043     -0.624      0.532      -0.110       0.057
sigma2      6.509e+05   4.43e+04     14.685      0.000    5.64e+05    7.38e+05
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.63063D+00    |proj g|=  2.59993D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
At iterate    5    f=  7.56102D+00    |proj g|=  2.22430D-02

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5      9     12      1     0     0   1.162D-04   7.561D+00
  F =   7.5605465405227701

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Ocean County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8418      0.042     19.976      0.000       0.759       0.924
ma.L1          0.1197      0.080      1.487      0.137      -0.038       0.277
ar.S.L12      -0.5050      0.101     -5.020      0.000      -0.702      -0.308
ma.S.L12      -0.0149      0.066     -0.227      0.820      -0.143       0.114
sigma2      1.297e+06   8.14e+04     15.938      0.000    1.14e+06    1.46e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.65107D+00    |proj g|=  8.57074D-02

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated
frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated
frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
 This problem is unconstrained.
```

```
At iterate    5    f=  7.64206D+00    |proj g|=  1.47974D-02

At iterate   10    f=  7.64195D+00    |proj g|=  1.65037D-04

At iterate   15    f=  7.64193D+00    |proj g|=  4.70202D-03

At iterate   20    f=  7.63996D+00    |proj g|=  5.17026D-02

At iterate   25    f=  7.62722D+00    |proj g|=  1.84714D-02

At iterate   30    f=  7.62678D+00    |proj g|=  9.26343D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     30     32      1     0     0   9.263D-06   7.627D+00
  F =   7.6267782556156805

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
SARIMAX Model Result for Passaic County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.6780      0.053     12.885      0.000       0.575       0.781
ma.L1          0.1879      0.062      3.006      0.003       0.065       0.310
ar.S.L12      -0.4195      0.068     -6.172      0.000      -0.553      -0.286
ma.S.L12      -0.1047      0.036     -2.945      0.003      -0.174      -0.035
sigma2      1.294e+06   1.07e+05     12.067      0.000    1.08e+06     1.5e+06
==============================================================================

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.90366D+00    |proj g|=  4.87036D-01

At iterate    5    f=  7.86446D+00    |proj g|=  4.17481D-03

At iterate   10    f=  7.86444D+00    |proj g|=  1.30037D-04

At iterate   15    f=  7.86444D+00    |proj g|=  2.17789D-03

At iterate   20    f=  7.86428D+00    |proj g|=  2.66477D-02

At iterate   25    f=  7.85547D+00    |proj g|=  1.07359D-01

At iterate   30    f=  7.85255D+00    |proj g|=  2.43380D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     32     39      1     0     0   3.691D-07   7.853D+00
  F =   7.8525494815537638

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
SARIMAX Model Result for Salem County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.7880      0.048     16.394      0.000       0.694       0.882
ma.L1          0.0501      0.070      0.719      0.472      -0.086       0.187
ar.S.L12      -0.4815      0.052     -9.272      0.000      -0.583      -0.380
ma.S.L12      -0.0319      0.035     -0.908      0.364      -0.101       0.037
sigma2      2.137e+06   1.58e+05     13.564      0.000    1.83e+06    2.45e+06
==============================================================================

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.04185D+00    |proj g|=  3.20143D-01

At iterate    5    f=  8.01757D+00    |proj g|=  5.83822D-02

At iterate   10    f=  8.01681D+00    |proj g|=  1.18235D-04

At iterate   15    f=  8.01679D+00    |proj g|=  3.50779D-03

At iterate   20    f=  8.01540D+00    |proj g|=  3.62868D-02

At iterate   25    f=  7.99796D+00    |proj g|=  2.55634D-02

At iterate   30    f=  7.99783D+00    |proj g|=  1.36962D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     30     36      1     0     0   1.370D-05   7.998D+00
  F =   7.9978336748657979

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Somerset County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8119      0.047     17.384      0.000       0.720       0.903
ma.L1         -0.0307      0.074     -0.415      0.678      -0.176       0.114
ar.S.L12      -0.5238      0.053     -9.930      0.000      -0.627      -0.420
ma.S.L12      -0.0326      0.024     -1.386      0.166      -0.079       0.013
sigma2      2.946e+06   2.21e+05     13.348      0.000    2.51e+06    3.38e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.90080D+00    |proj g|=  1.31198D+00
```

```
At iterate    5    f=  7.75661D+00    |proj g|=  9.86754D-02

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.


At iterate   10    f=  7.74853D+00    |proj g|=  4.10133D-04

At iterate   15    f=  7.74853D+00    |proj g|=  1.65853D-03

At iterate   20    f=  7.74829D+00    |proj g|=  2.46879D-02

At iterate   25    f=  7.72109D+00    |proj g|=  2.02348D-01

At iterate   30    f=  7.70063D+00    |proj g|=  5.30803D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     34     36      1     0     0   1.070D-06   7.701D+00
  F =   7.7006181343252020

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
SARIMAX Model Result for Sussex County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8079      0.037     21.892      0.000       0.736       0.880
ma.L1          0.0754      0.070      1.077      0.281      -0.062       0.212
ar.S.L12      -0.6489      0.044    -14.755      0.000      -0.735      -0.563
ma.S.L12      -0.0295      0.022     -1.347      0.178      -0.073       0.013
sigma2      1.521e+06   1.13e+05     13.491      0.000     1.3e+06    1.74e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.07490D+00    |proj g|=  1.38161D-01
```

```
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

At iterate    5    f=  8.05649D+00    |proj g|=  2.90166D-03

At iterate   10    f=  8.05649D+00    |proj g|=  1.34590D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     10     12      1     0     0   1.346D-04   8.056D+00
  F =   8.0564873988801775

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
SARIMAX Model Result for Union County
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.8530      0.047     18.318      0.000       0.762       0.944
ma.L1          0.0806      0.095      0.848      0.397      -0.106       0.267
ar.S.L12      -0.6398      0.067     -9.502      0.000      -0.772      -0.508
ma.S.L12      -0.0264      0.038     -0.702      0.482      -0.100       0.047
sigma2      4.635e+06   4.74e+05      9.780      0.000    3.71e+06    5.56e+06
==============================================================================
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.15662D+00    |proj g|=  3.69787D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated
frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated
frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
 This problem is unconstrained.
```

```
At iterate    5    f=  8.12671D+00    |proj g|=  2.08743D-02

At iterate   10    f=  8.12615D+00    |proj g|=  9.43132D-05

At iterate   15    f=  8.12615D+00    |proj g|=  3.10003D-03

At iterate   20    f=  8.12532D+00    |proj g|=  2.68100D-02

At iterate   25    f=  8.11169D+00    |proj g|=  2.83213D-02

At iterate   30    f=  8.11066D+00    |proj g|=  1.89958D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     32     35      1     0     0   1.677D-06   8.111D+00
  F =   8.1106566517039145

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
SARIMAX Model Result for Warren County
```

	coef	std err	z	P>|z|	[0.025	0.975]
ar.L1	0.5948	0.088	6.742	0.000	0.422	0.768
ma.L1	-0.0490	0.094	-0.518	0.604	-0.234	0.136
ar.S.L12	-0.4180	0.047	-8.875	0.000	-0.510	-0.326
ma.S.L12	-0.1275	0.039	-3.236	0.001	-0.205	-0.050
sigma2	3.943e+06	1.61e+05	24.540	0.000	3.63e+06	4.26e+06

In [41]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"

# Making Prediction post 2022 and compare/testing with real values
summary = pd.DataFrame()
county = []
mse = []
models = []
for name, pdq, pdqs, df in zip(best_parameter['name'], best_parameter['pdq'], best_parameter['pdqs'], county_dfs):

    SARIMAX_MODEL = sm.tsa.SARIMAX(df,
                                   order = pdq,
                                   seasonal_order = pdqs,
                                   enforce_stationarity = False,
                                   enforce_invertibility = False)

    output = SARIMAX_MODEL.fit()
    models.append(output)

    #get dynamic predictions starting 2022-06-30
    pred_dynamic = output.get_prediction(start=pd.to_datetime('2022-06-30'), dynamic = True, full_results = True)
    pred_dynamic_conf = pred_dynamic.conf_int()
    county_predicted = pred_dynamic.predicted_mean
    county_actual = df['2022-06-30':]['MeanValue']

    #Plot the dynamic forecast with confidence intervals as above
    ax = df['2022':].plot(label='observed', figsize=(10, 8))
    pred_dynamic.predicted_mean.plot(label='Dynamic Forecast', ax=ax)

    ax.fill_between(pred_dynamic_conf.index,
                    pred_dynamic_conf.iloc[:, 0],
                    pred_dynamic_conf.iloc[:, 1], color='g', alpha=.3)

    ax.fill_betweenx(ax.get_ylim(), pd.to_datetime('2022-06-30'), county_predicted.index[-1], alpha=.1, zorder=-1)

    ax.set_xlabel('Date')
    ax.set_ylabel('Average House Price')
    plt.title('For '+ name)
    plt.legend()
    plt.show()

    ##print(type(county_predicted))
    ##print(county_actual)
    sqrt_mse = np.sqrt(((county_predicted - county_actual)**2).mean())
    print('\033[1m' + 'The Mean Squared Error of our forecasts for {} county is {}'.format(name, round(sqrt_mse, 2)))

    county.append(name)
    mse.append(sqrt_mse)

summary['County'] = county
summary['Sqrt_MSE'] = mse
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.94747D+00    |proj g|=  1.93868D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.


At iterate    5    f=  7.90316D+00    |proj g|=  4.57387D-02

At iterate   10    f=  7.90307D+00    |proj g|=  2.03891D-04

At iterate   15    f=  7.90306D+00    |proj g|=  1.22891D-03

At iterate   20    f=  7.90297D+00    |proj g|=  1.70119D-02

At iterate   25    f=  7.88316D+00    |proj g|=  7.88999D-02

At iterate   30    f=  7.88106D+00    |proj g|=  5.60490D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     31     36      1     0     0   3.412D-05   7.881D+00
  F =   7.8810625140986934

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x14e7dbf40>

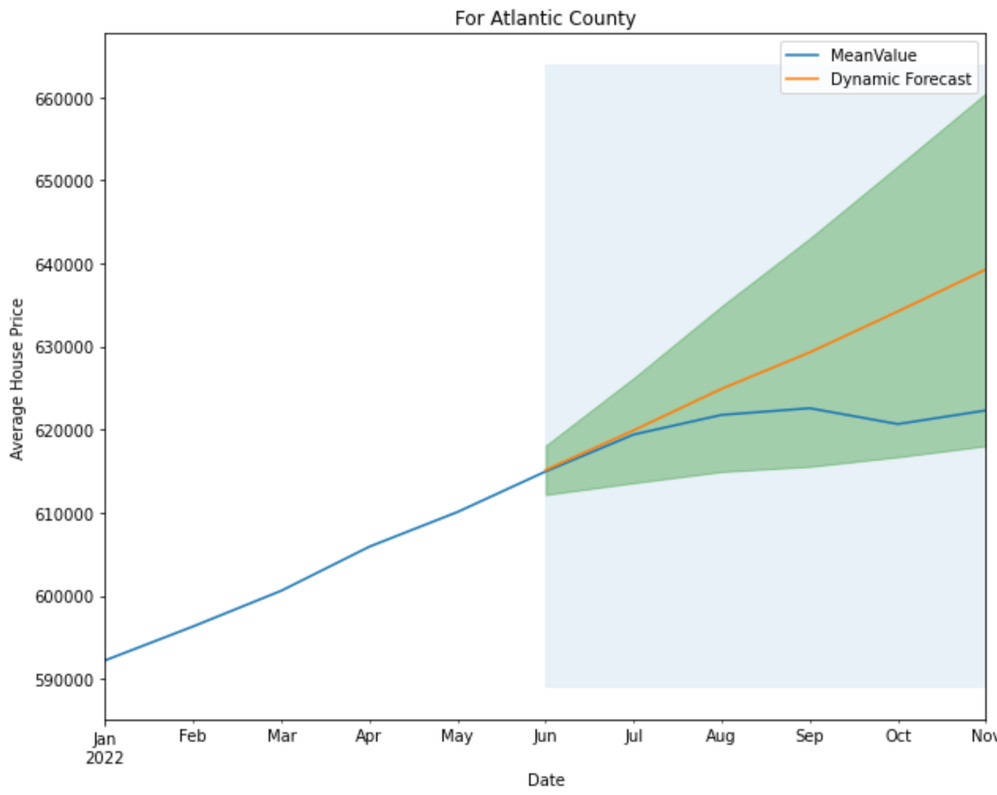
Out[41]: <matplotlib.collections.PolyCollection at 0x155b71c40>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Atlantic County')

Out[41]: <matplotlib.legend.Legend at 0x14c8f7be0>



The Mean Squared Error of our forecasts for Atlantic County county is 9372.02
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.70605D+00 |proj g|= 3.80994D-01

At iterate 5 f= 7.63306D+00 |proj g|= 4.33096D-02

At iterate 10 f= 7.63030D+00 |proj g|= 3.43314D-04

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     12     15      1     0     0   1.424D-04   7.630D+00
  F =   7.6302967977782332

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x1322a2610>

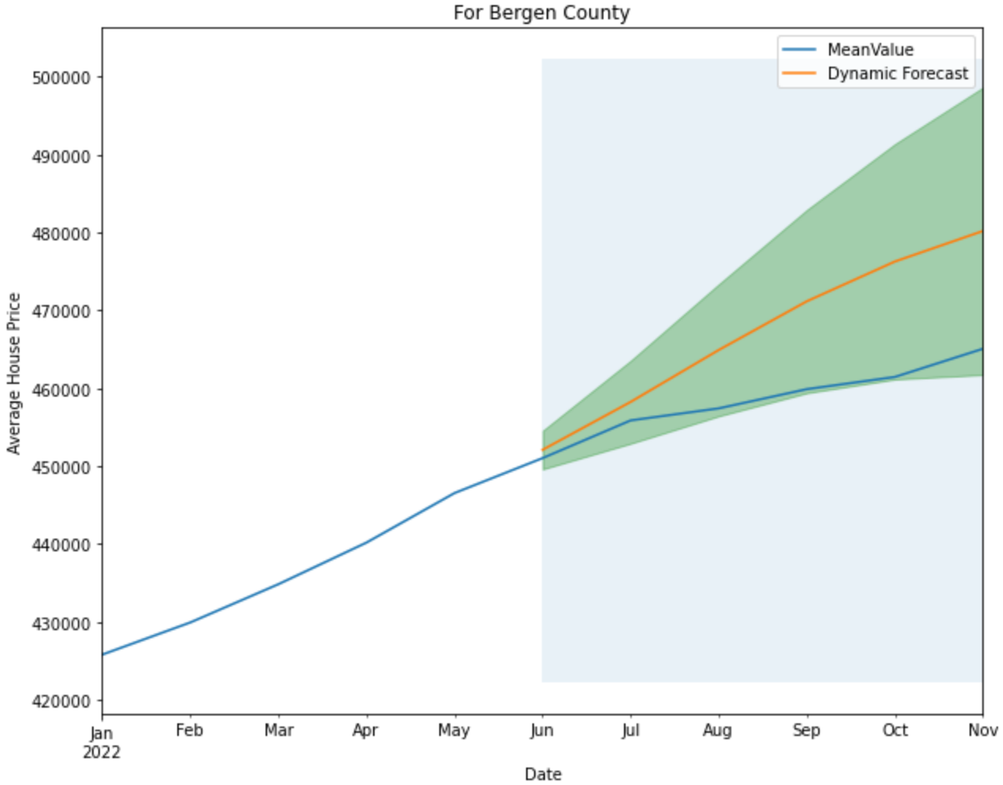
Out[41]: <matplotlib.collections.PolyCollection at 0x12cb6c490>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Bergen County')

Out[41]: <matplotlib.legend.Legend at 0x12c3cec40>


For Bergen County
MeanValue
Dynamic Forecast
Average House Price
500000
490000
480000
470000
460000
450000
440000
430000
420000
Jan
2022
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Date

```
The Mean Squared Error of our forecasts for Bergen County county is 10310.67
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.02822D+00    |proj g|=  5.34154D-01

At iterate    5    f=  7.93399D+00    |proj g|=  4.57795D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5      9     11      1     0     0   1.525D-04   7.934D+00
  F =   7.9339707892830980

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
```

```
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x13e8733d0>

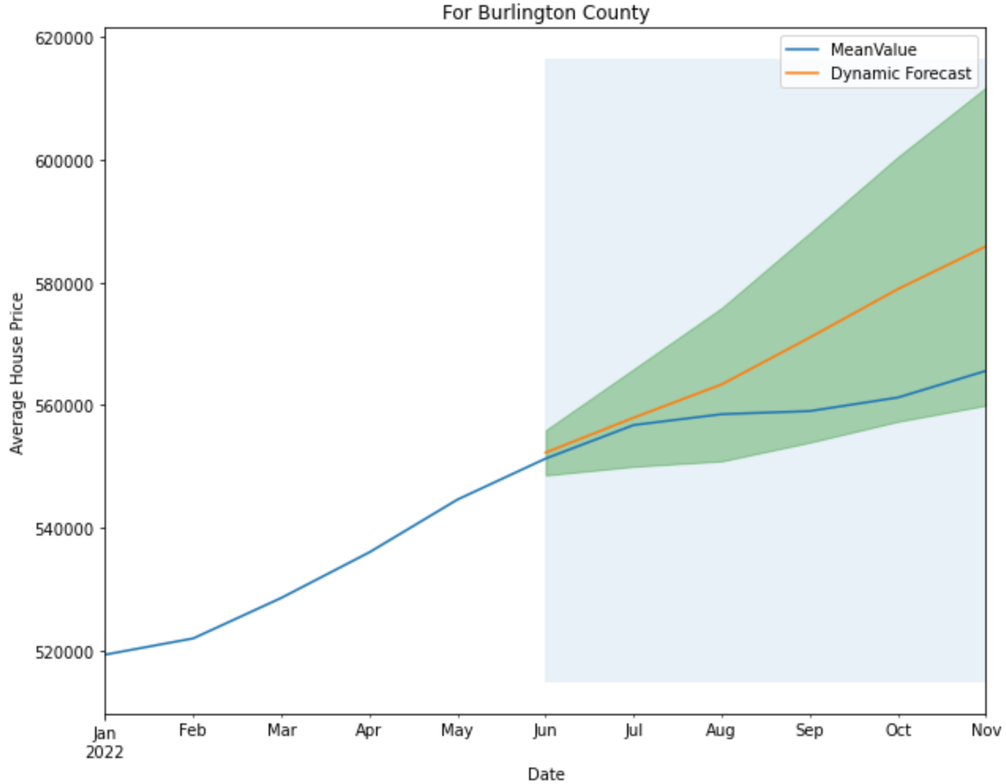
Out[41]: <matplotlib.collections.PolyCollection at 0x13e873ca0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Burlington County')

Out[41]: <matplotlib.legend.Legend at 0x125019220>



The Mean Squared Error of our forecasts for Burlington County county is 12200.21
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 8.55408D+00 |proj g|= 8.89204D-01

At iterate 5 f= 8.46271D+00 |proj g|= 8.12953D-02

At iterate 10 f= 8.45523D+00 |proj g|= 2.15456D-03

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
 This problem is unconstrained.

```
           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     13     15      1     0     0   6.467D-05   8.455D+00
  F =   8.4552208960414053

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x12f415910>

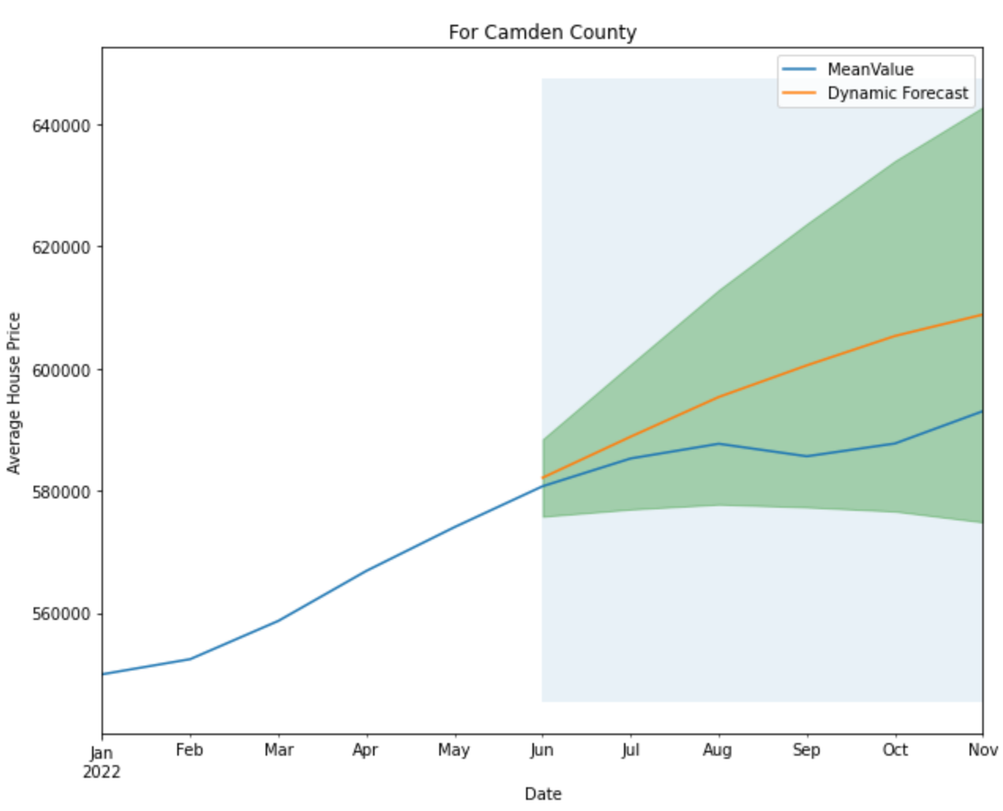
Out[41]: <matplotlib.collections.PolyCollection at 0x14cb007c0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Camden County')

Out[41]: <matplotlib.legend.Legend at 0x15257cd00>



```
The Mean Squared Error of our forecasts for Camden County county is 11940.9
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            3     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.92671D+00    |proj g|=  2.97227D-01

At iterate    5    f=  7.86543D+00    |proj g|=  3.66114D-04

At iterate   10    f=  7.86510D+00    |proj g|=  4.07097D-02

At iterate   15    f=  7.84444D+00    |proj g|=  2.31249D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
ency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.


At iterate   20    f=  7.83731D+00    |proj g|=  1.26231D-05

           * * *

           Tit   = total number of iterations
           Tnf   = total number of function evaluations
           Tnint = total number of segments explored during Cauchy searches
           Skip  = number of BFGS updates skipped
           Nact  = number of active bounds at final generalized Cauchy point
           Projg = norm of the final projected gradient
           F     = final function value

              * * *

              N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
               3     20     26      1     0     0   1.262D-05   7.837D+00
             F =   7.8373092441552528

           CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

           /Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
           be removed in a future version.
             date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x15278dcd0>

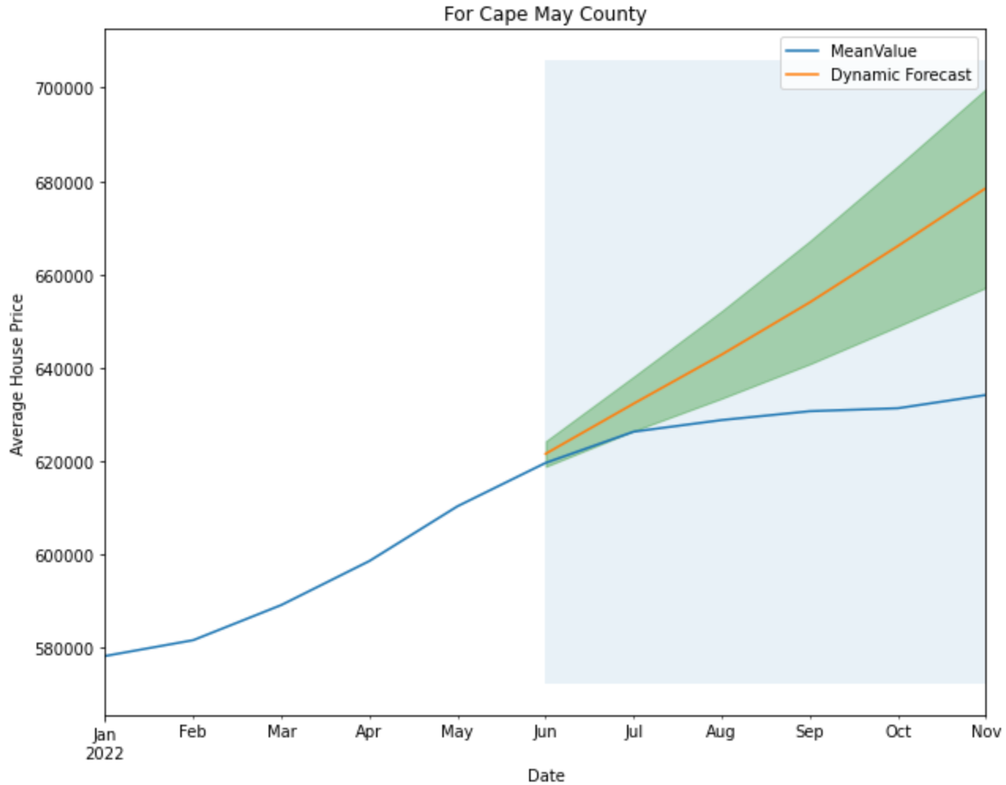
Out[41]: <matplotlib.collections.PolyCollection at 0x150708c10>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Cape May County')

Out[41]: <matplotlib.legend.Legend at 0x15278d640>



The Mean Squared Error of our forecasts for Cape May County county is 25658.03
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.88051D+00 |proj g|= 7.31078D-01

At iterate 5 f= 7.61048D+00 |proj g|= 5.15481D-03

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   10    f=  7.61046D+00    |proj g|=  1.87763D-04

At iterate   15    f=  7.61045D+00    |proj g|=  4.86511D-03

At iterate   20    f=  7.60835D+00    |proj g|=  6.22124D-02

At iterate   25    f=  7.59097D+00    |proj g|=  1.65363D-02

At iterate   30    f=  7.59090D+00    |proj g|=  1.55765D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     30     35      1     0     0   1.558D-05   7.591D+00
  F =   7.5908986188576302

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x1507efeb0>

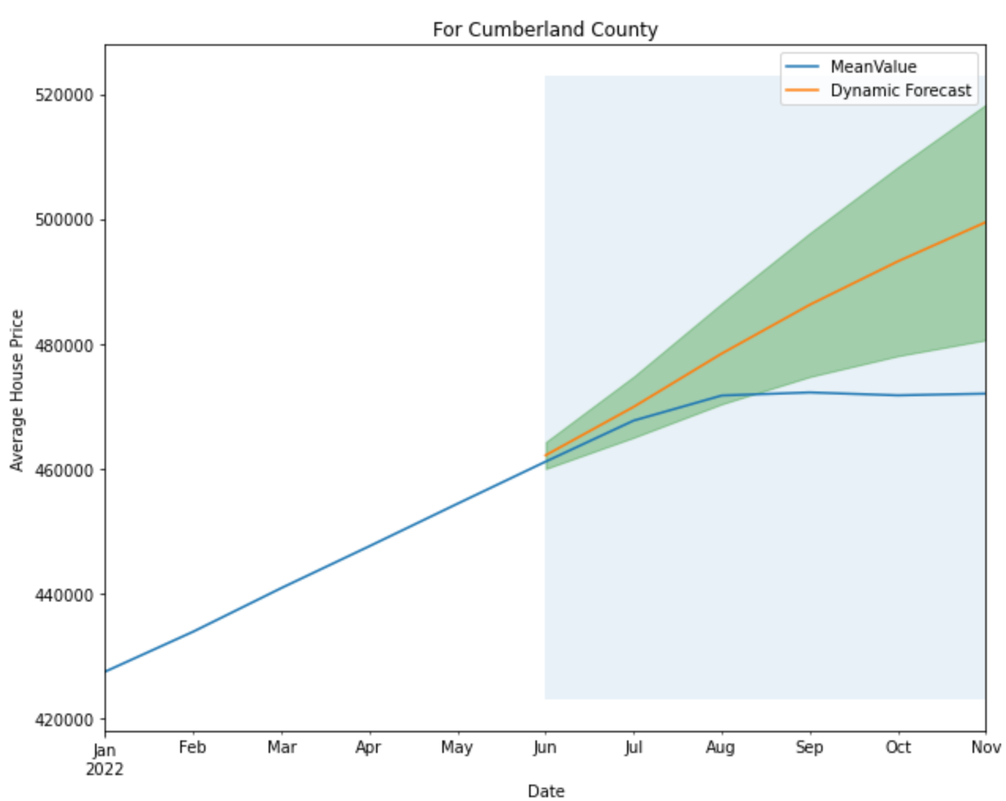
Out[41]: <matplotlib.collections.PolyCollection at 0x1507ef2e0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Cumberland County')

Out[41]: <matplotlib.legend.Legend at 0x152494340>



The Mean Squared Error of our forecasts for Cumberland County county is 15631.62
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

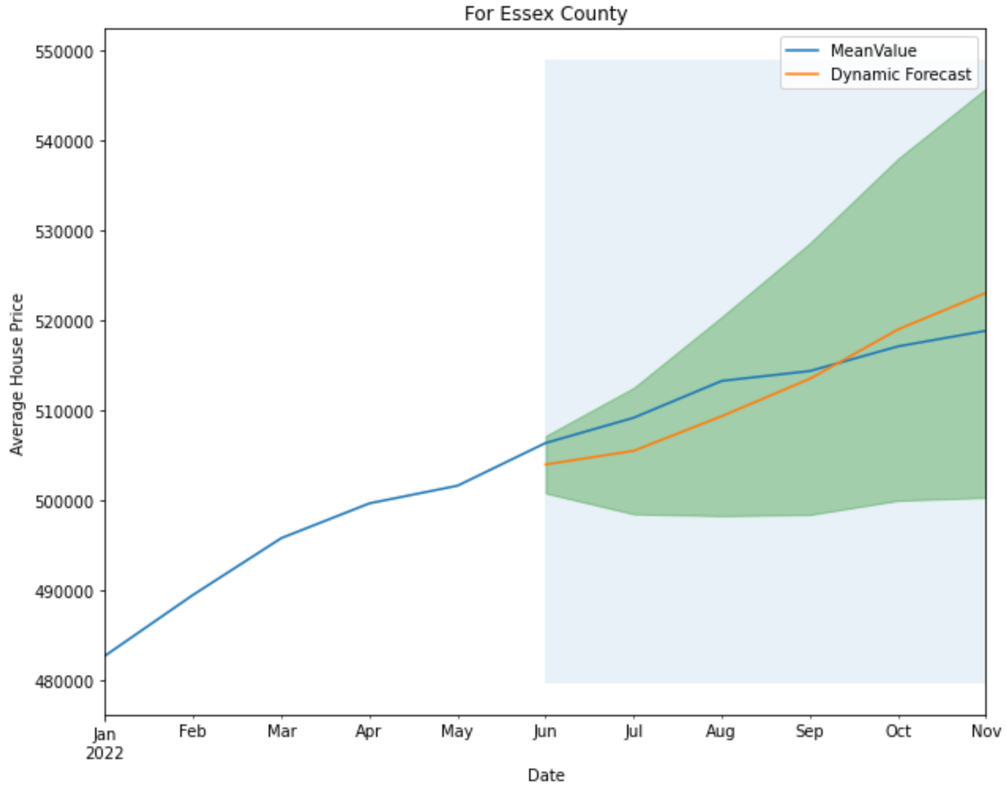
At X0 0 variables are exactly at the bounds

At iterate 0 f= 8.01361D+00 |proj g|= 1.39409D-01

At iterate 5 f= 7.97640D+00 |proj g|= 2.99191D-03

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   10    f=  7.97638D+00    |proj g|=  4.48700D-03

At iterate   15    f=  7.97519D+00    |proj g|=  6.59970D-02

At iterate   20    f=  7.94982D+00    |proj g|=  1.53388D-01

At iterate   25    f=  7.94722D+00    |proj g|=  2.25121D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     25     31      1     0     0   2.251D-05   7.947D+00
  F =   7.9472197077704747

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x13c955730>

Out[41]: <matplotlib.collections.PolyCollection at 0x12cadb310>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Essex County')

Out[41]: <matplotlib.legend.Legend at 0x12caf1d00>
```



The Mean Squared Error of our forecasts for Essex County county is 3066.63
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.35140D+00 |proj g|= 2.95483D-01

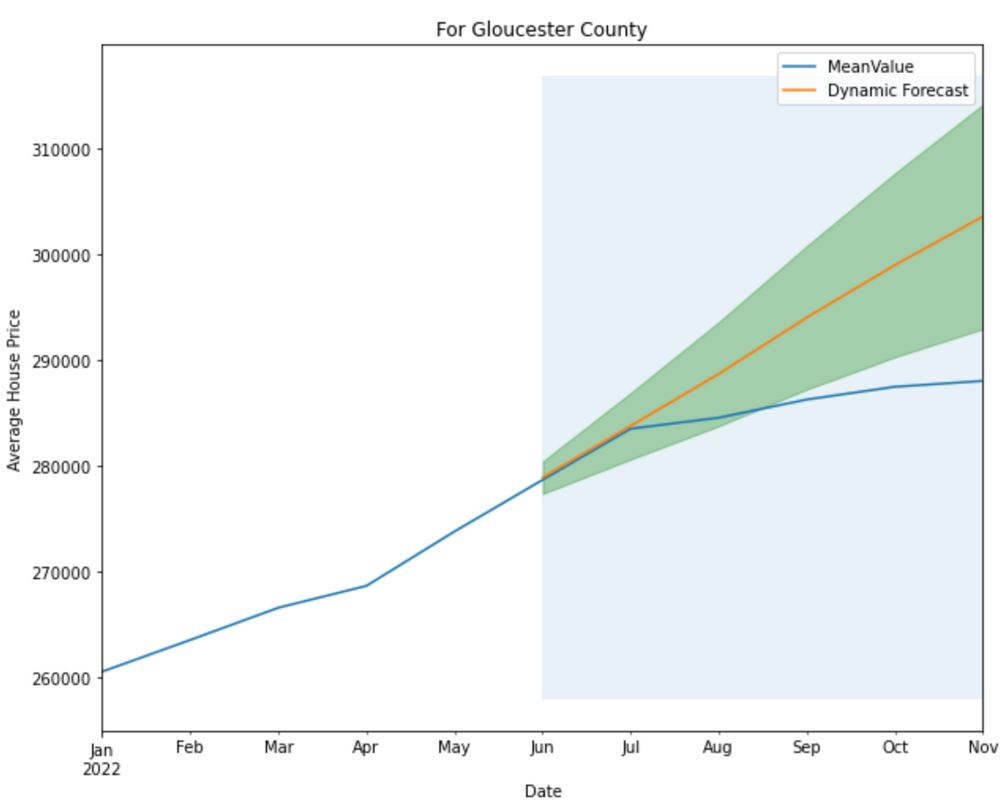
At iterate 5 f= 7.29562D+00 |proj g|= 3.96329D-02

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   10    f=  7.29444D+00    |proj g|=  1.76437D-04

At iterate   15    f=  7.29444D+00    |proj g|=  1.62379D-03

At iterate   20    f=  7.29407D+00    |proj g|=  2.08746D-02

At iterate   25    f=  7.28731D+00    |proj g|=  6.47484D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     29     34      1     0     0   1.065D-05   7.287D+00
  F =   7.2872753462820850

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
```

```
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

```
Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x13e82aa00>

Out[41]: <matplotlib.collections.PolyCollection at 0x132273910>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Gloucester County')

Out[41]: <matplotlib.legend.Legend at 0x14e7e79d0>
```



The Mean Squared Error of our forecasts for Gloucester County county is 8666.25
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.69602D+00 |proj g|= 4.85344D-01

At iterate 5 f= 7.57400D+00 |proj g|= 1.01259D-01

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   10    f=  7.57151D+00    |proj g|=  1.74044D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     11     15      1     0     0   1.740D-04   7.572D+00
  F =   7.5715081925689267

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x1541e66d0>

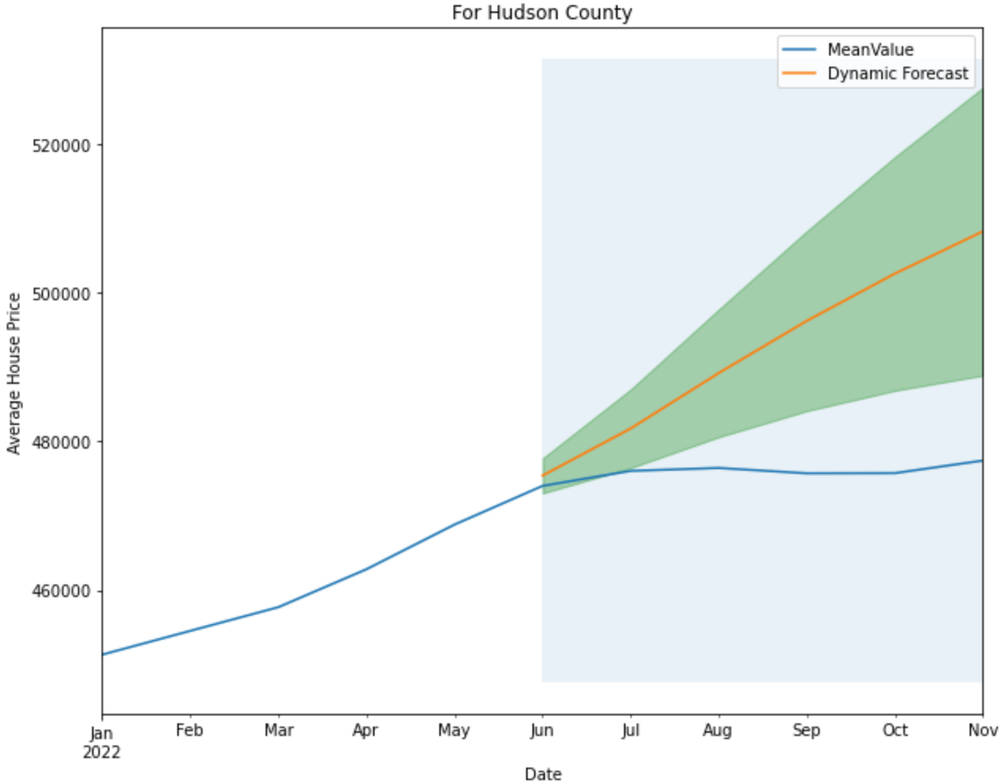
Out[41]: <matplotlib.collections.PolyCollection at 0x13ab00c10>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Hudson County')

Out[41]: <matplotlib.legend.Legend at 0x1541f0640>



The Mean Squared Error of our forecasts for Hudson County county is 19515.92
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.87258D+00 |proj g|= 1.88861D-01

At iterate 5 f= 7.84095D+00 |proj g|= 3.99296D-02

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   10    f=  7.84084D+00    |proj g|=  1.33931D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     11     15      1     0     0   2.700D-04   7.841D+00
  F =   7.8408446696271730

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x13227d4f0>

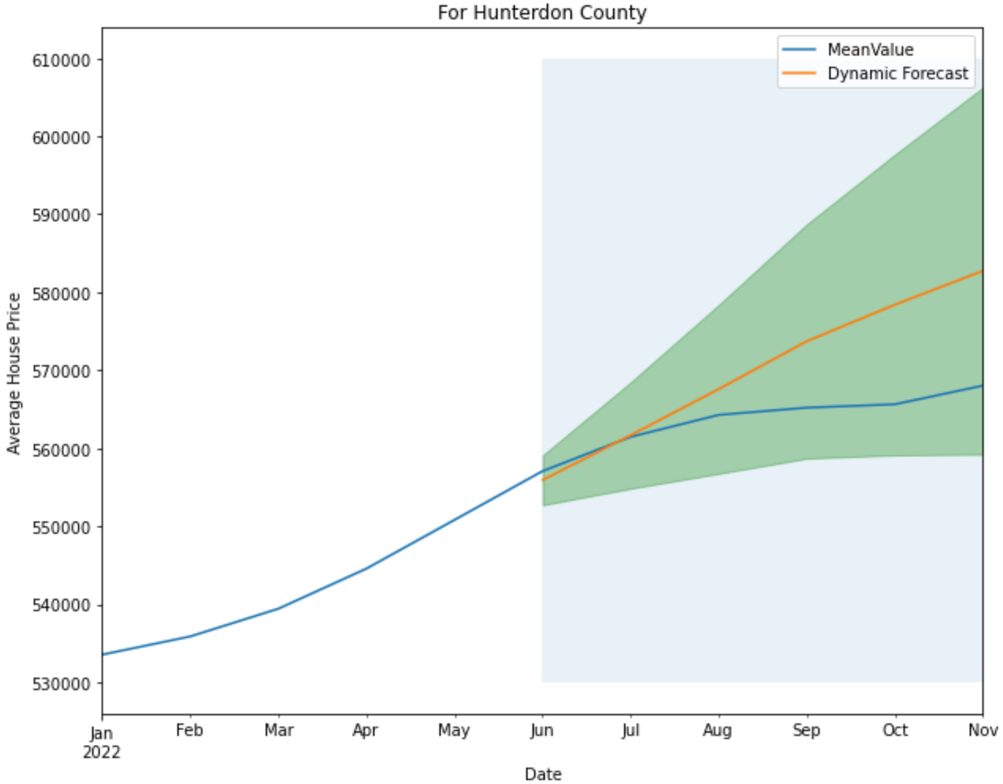
Out[41]: <matplotlib.collections.PolyCollection at 0x132288df0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Hunterdon County')

Out[41]: <matplotlib.legend.Legend at 0x132288850>



The Mean Squared Error of our forecasts for Hunterdon County county is 8797.71
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.47673D+00 |proj g|= 4.82285D-01

At iterate 5 f= 7.31048D+00 |proj g|= 3.63602D-02

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   10    f=  7.30911D+00    |proj g|=  2.73365D-04

At iterate   15    f=  7.30911D+00    |proj g|=  1.94848D-03

At iterate   20    f=  7.30871D+00    |proj g|=  2.47139D-02

At iterate   25    f=  7.29925D+00    |proj g|=  1.57599D-01

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     29     33      1     0     0   1.824D-04   7.296D+00
  F =   7.2960701935591903

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x1322af3d0>

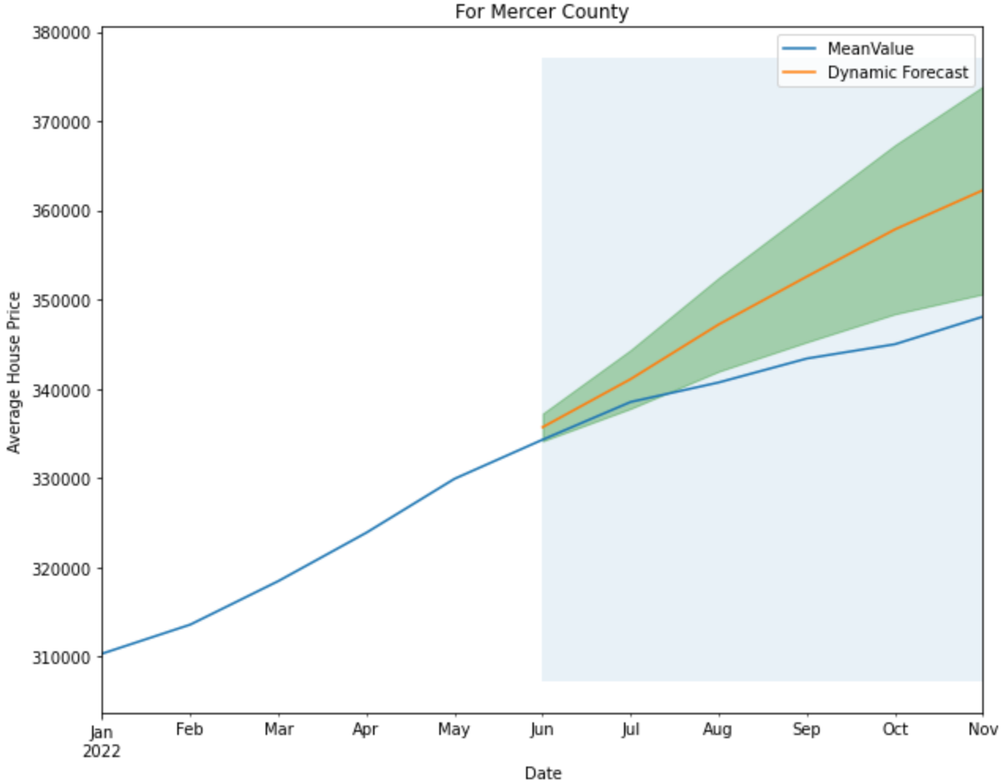
Out[41]: <matplotlib.collections.PolyCollection at 0x1322d8d00>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Mercer County')

Out[41]: <matplotlib.legend.Legend at 0x1322af7f0>



The Mean Squared Error of our forecasts for Mercer County county is 9142.4
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

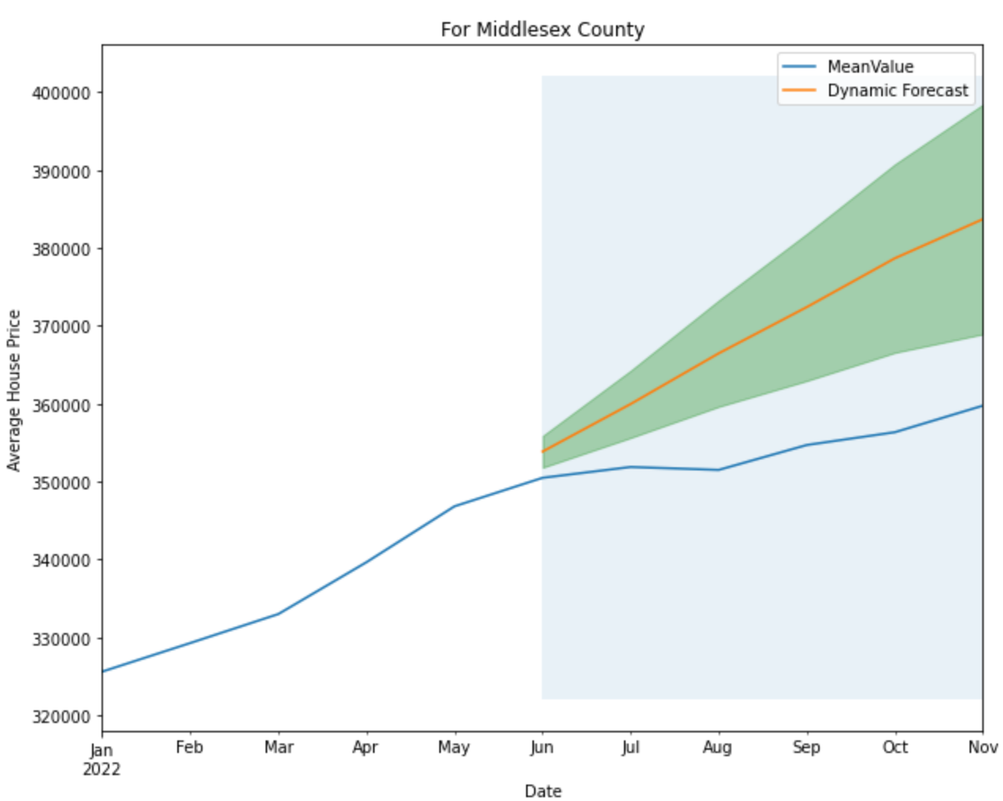
At iterate 0 f= 7.62615D+00 |proj g|= 3.24122D-01

At iterate 5 f= 7.57444D+00 |proj g|= 8.31412D-02

At iterate 10 f= 7.57337D+00 |proj g|= 3.15944D-04

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   15    f=  7.57336D+00    |proj g|=  3.30770D-03

At iterate   20    f=  7.57305D+00    |proj g|=  3.96847D-02

At iterate   25    f=  7.55411D+00    |proj g|=  1.81029D-01

At iterate   30    f=  7.54480D+00    |proj g|=  3.41501D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     34     37      1     0     0   4.343D-06   7.545D+00
  F =   7.5447872592530656

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x152467ac0>

Out[41]: <matplotlib.collections.PolyCollection at 0x1524670d0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Middlesex County')

Out[41]: <matplotlib.legend.Legend at 0x140202100>
```



The Mean Squared Error of our forecasts for Middlesex County county is 16772.25
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 8.01589D+00 |proj g|= 1.01412D+00

At iterate 5 f= 7.91329D+00 |proj g|= 3.72458D-02

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   10    f=  7.91257D+00    |proj g|=  6.16075D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     12     16      1     0     0   1.580D-04   7.913D+00
  F =   7.9125650301851351

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x15247adf0>

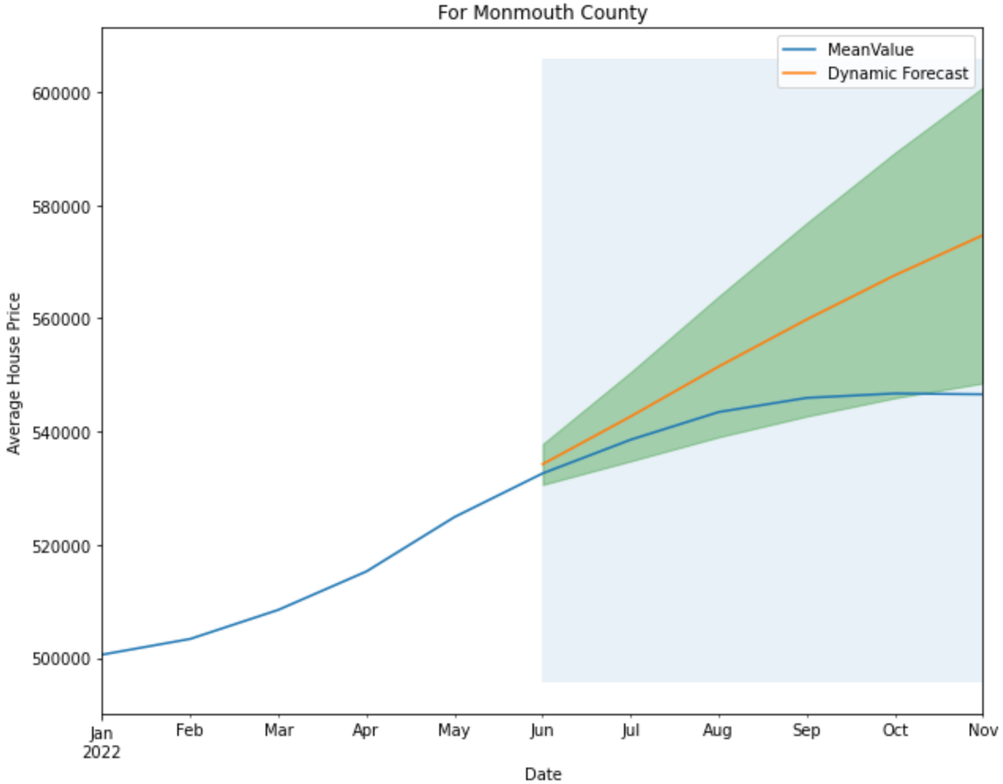
Out[41]: <matplotlib.collections.PolyCollection at 0x14ef37280>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Monmouth County')

Out[41]: <matplotlib.legend.Legend at 0x1524a4a90>



The Mean Squared Error of our forecasts for Monmouth County county is 15822.9
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

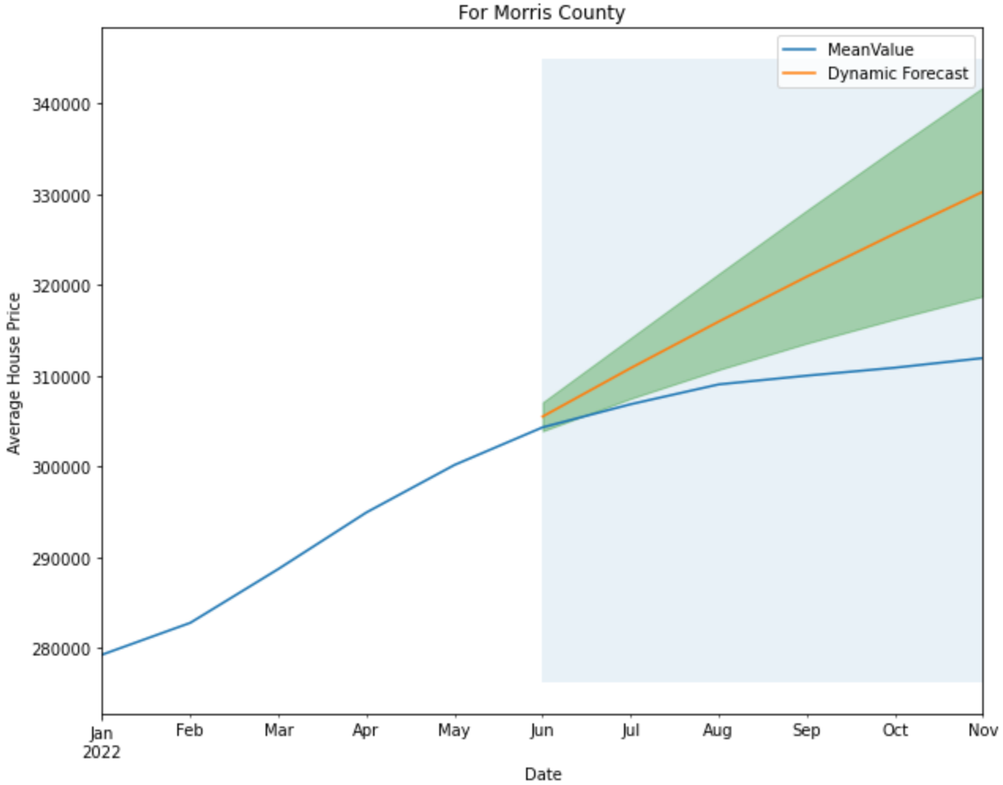
At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.37238D+00 |proj g|= 2.13474D-01

At iterate 5 f= 7.33211D+00 |proj g|= 4.74294D-02

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   10    f=  7.33145D+00    |proj g|=  2.29360D-04

At iterate   15    f=  7.33144D+00    |proj g|=  7.12039D-03

At iterate   20    f=  7.33046D+00    |proj g|=  8.11149D-02

At iterate   25    f=  7.31824D+00    |proj g|=  9.51673D-02

At iterate   30    f=  7.31661D+00    |proj g|=  2.00788D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     30     36      1     0     0   2.008D-05   7.317D+00
  F =   7.3166120192210826

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x14e811b50>

Out[41]: <matplotlib.collections.PolyCollection at 0x1525b4250>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Morris County')

Out[41]: <matplotlib.legend.Legend at 0x13ab00760>
```


For Morris County
MeanValue
Dynamic Forecast
340000
330000
320000
310000
300000
290000
280000
Average House Price
Jan
2022
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Date

```
The Mean Squared Error of our forecasts for Morris County county is 11084.61
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  7.63063D+00    |proj g|=  2.59993D-01

At iterate    5    f=  7.56102D+00    |proj g|=  2.22430D-02

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5      9     12      1     0     0   1.162D-04   7.561D+00
  F =   7.5605465405227701

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH
```

```
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x132295f70>

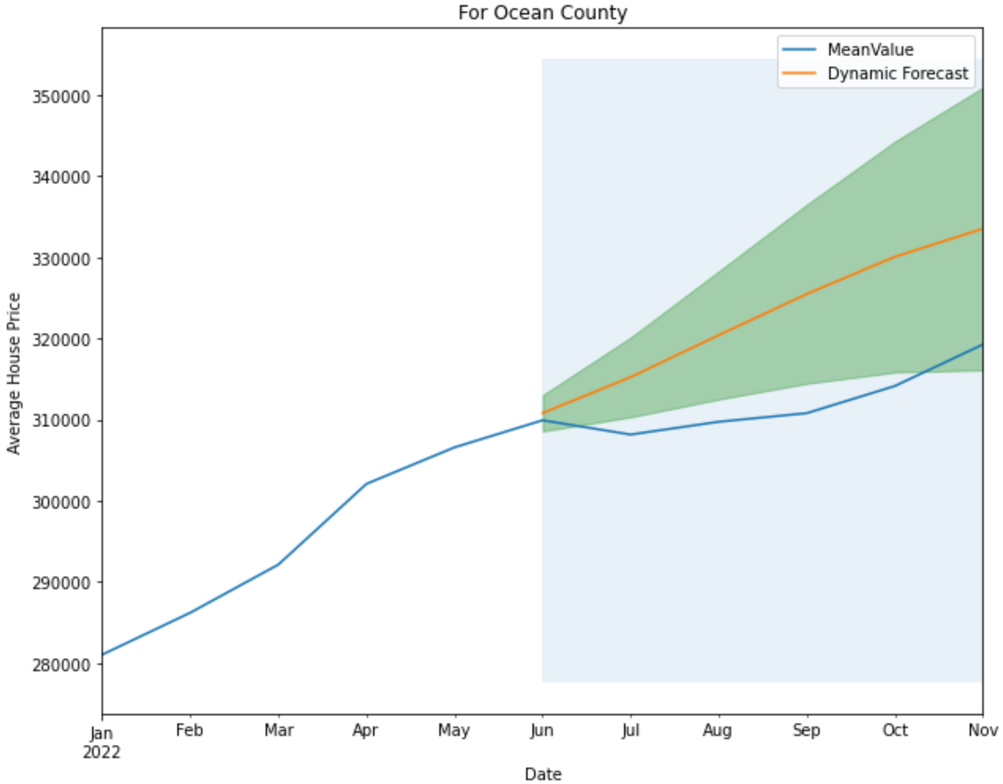
Out[41]: <matplotlib.collections.PolyCollection at 0x12c064be0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Ocean County')

Out[41]: <matplotlib.legend.Legend at 0x14020c520>



The Mean Squared Error of our forecasts for Ocean County county is 11818.45
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.65107D+00 |proj g|= 8.57074D-02

At iterate 5 f= 7.64206D+00 |proj g|= 1.47974D-02

At iterate 10 f= 7.64195D+00 |proj g|= 1.65037D-04

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
 This problem is unconstrained.

```
At iterate   15    f=  7.64193D+00    |proj g|=  4.70202D-03

At iterate   20    f=  7.63996D+00    |proj g|=  5.17026D-02

At iterate   25    f=  7.62722D+00    |proj g|=  1.84714D-02

At iterate   30    f=  7.62678D+00    |proj g|=  9.26343D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     30     32      1     0     0   9.263D-06   7.627D+00
  F =   7.6267782556156805

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x13c955970>

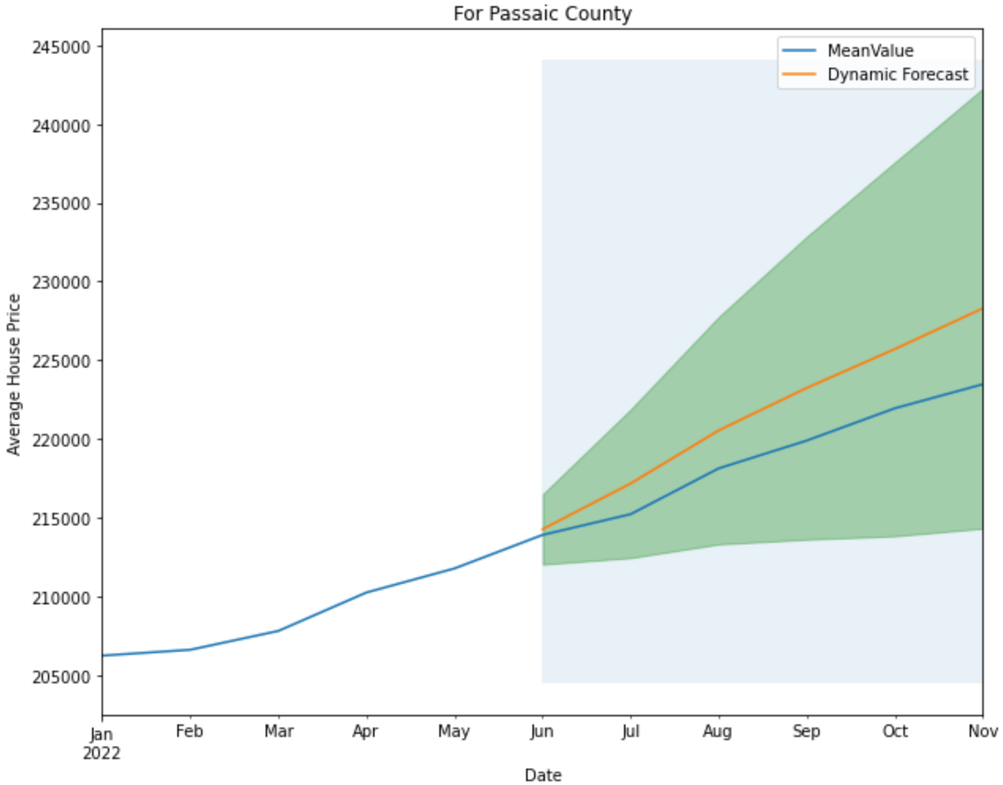
Out[41]: <matplotlib.collections.PolyCollection at 0x13e854cd0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Passaic County')

Out[41]: <matplotlib.legend.Legend at 0x14ef8c8e0>



The Mean Squared Error of our forecasts for Passaic County county is 3117.28
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.90366D+00 |proj g|= 4.87036D-01

At iterate 5 f= 7.86446D+00 |proj g|= 4.17481D-03

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   10    f=  7.86444D+00    |proj g|=  1.30037D-04

At iterate   15    f=  7.86444D+00    |proj g|=  2.17789D-03

At iterate   20    f=  7.86428D+00    |proj g|=  2.66477D-02

At iterate   25    f=  7.85547D+00    |proj g|=  1.07359D-01

At iterate   30    f=  7.85255D+00    |proj g|=  2.43380D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     32     39      1     0     0   3.691D-07   7.853D+00
  F =   7.8525494815537638

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x1250771f0>

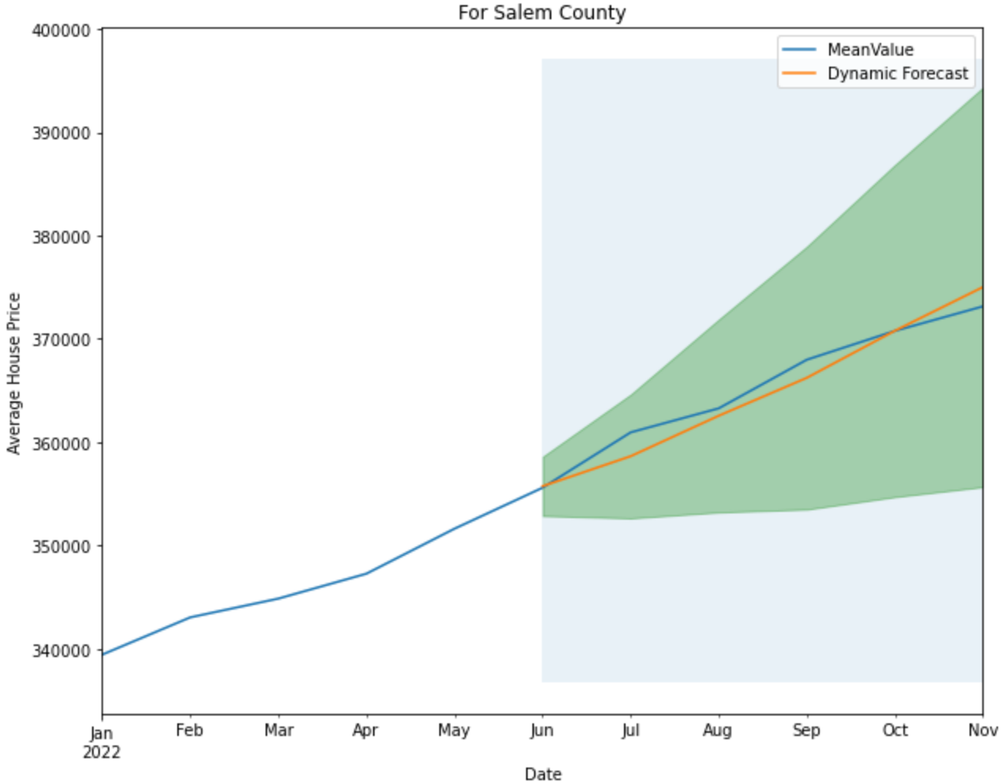
Out[41]: <matplotlib.collections.PolyCollection at 0x12fedc4f0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Salem County')

Out[41]: <matplotlib.legend.Legend at 0x14aaa8fd0>



The Mean Squared Error of our forecasts for Salem County county is 1439.35
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 8.04185D+00 |proj g|= 3.20143D-01

At iterate 5 f= 8.01757D+00 |proj g|= 5.83822D-02

At iterate 10 f= 8.01681D+00 |proj g|= 1.18235D-04

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   15    f=  8.01679D+00    |proj g|=  3.50779D-03

At iterate   20    f=  8.01540D+00    |proj g|=  3.62868D-02

At iterate   25    f=  7.99796D+00    |proj g|=  2.55634D-02

At iterate   30    f=  7.99783D+00    |proj g|=  1.36962D-05

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     30     36      1     0     0   1.370D-05   7.998D+00
  F =   7.9978336748657979

CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x13e86f400>

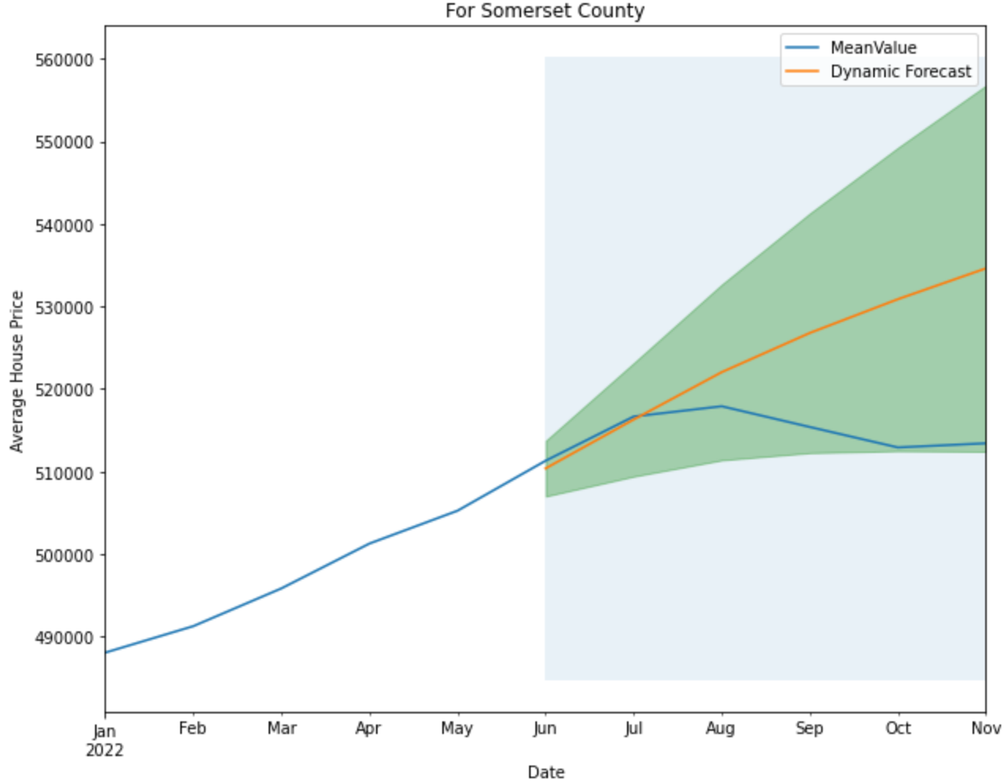
Out[41]: <matplotlib.collections.PolyCollection at 0x13aad7790>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Somerset County')

Out[41]: <matplotlib.legend.Legend at 0x14e7eef10>



The Mean Squared Error of our forecasts for Somerset County county is 12371.83
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 7.90080D+00 |proj g|= 1.31198D+00

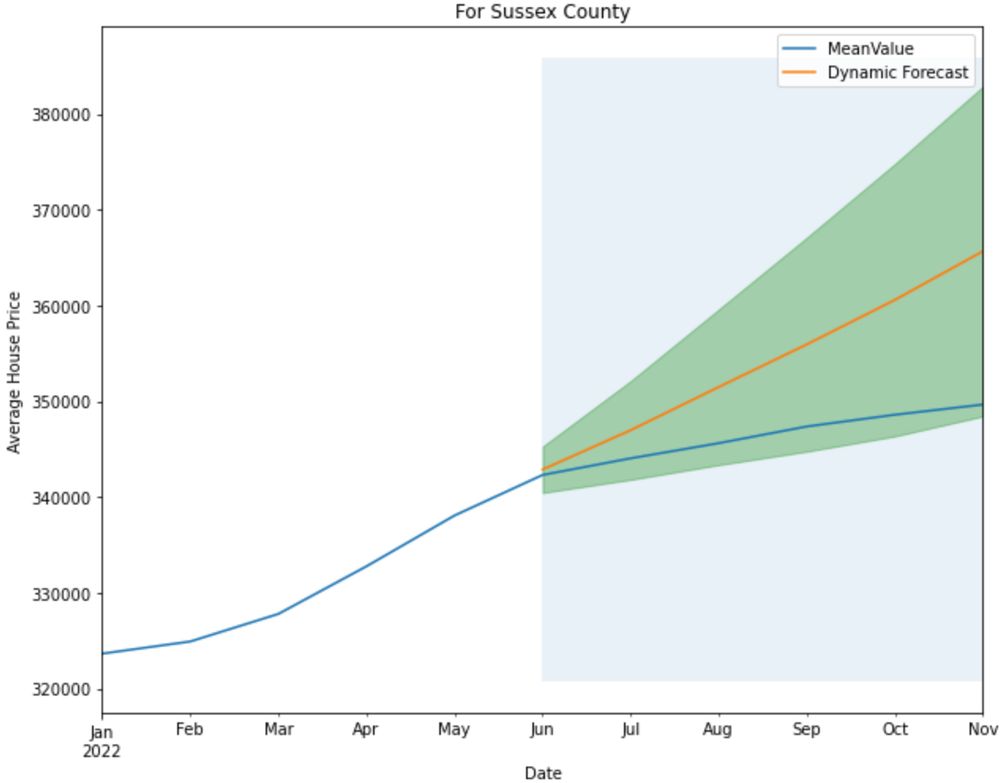
At iterate 5 f= 7.75661D+00 |proj g|= 9.86754D-02

At iterate 10 f= 7.74853D+00 |proj g|= 4.10133D-04

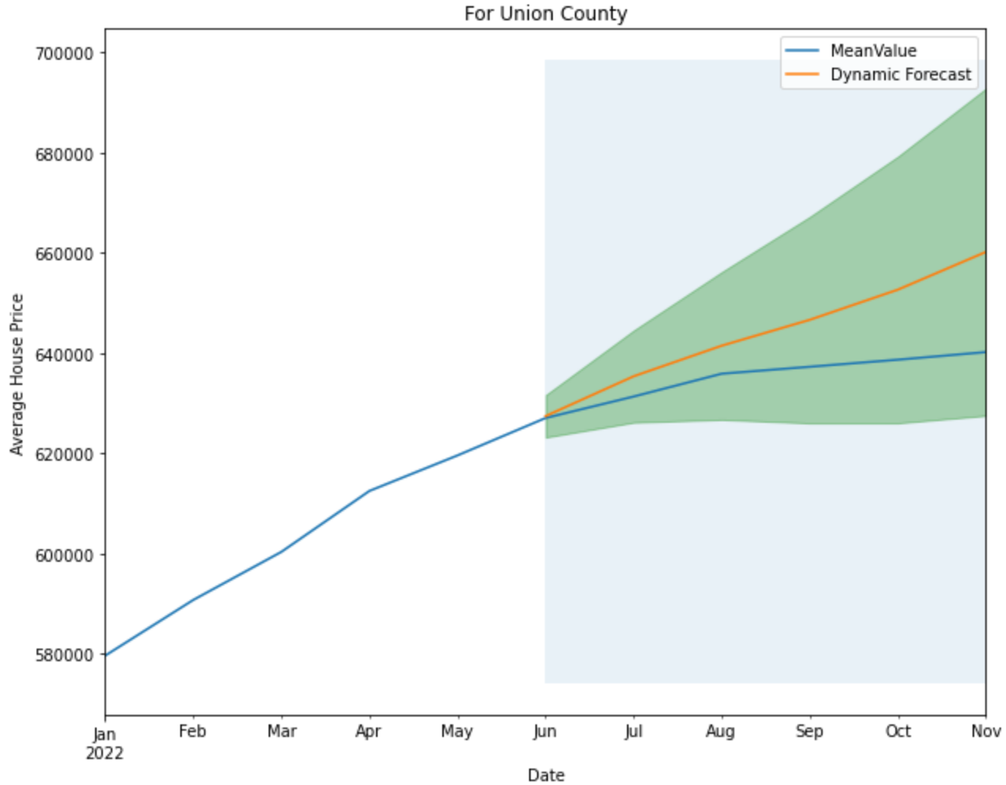
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequency M will be used.
  warnings.warn('No frequency information was'
 This problem is unconstrained.

```
At iterate   15    f=  7.74853D+00    |proj g|=  1.65853D-03

At iterate   20    f=  7.74829D+00    |proj g|=  2.46879D-02

At iterate   25    f=  7.72109D+00    |proj g|=  2.02348D-01

At iterate   30    f=  7.70063D+00    |proj g|=  5.30803D-03

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     34     36      1     0     0   1.070D-06   7.701D+00
  F =   7.7006181343252020

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
be removed in a future version.
  date_key = Timestamp(key, freq=base_index.freq)
```

```
Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x124e6a9a0>

Out[41]: <matplotlib.collections.PolyCollection at 0x13231aca0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Sussex County')

Out[41]: <matplotlib.legend.Legend at 0x124e6ac40>
```


For Sussex County
MeanValue
Dynamic Forecast
380000
370000
360000
350000
340000
330000
320000
Average House Price
Jan
2022
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Date

```
The Mean Squared Error of our forecasts for Sussex County county is 9281.57
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         0 variables are exactly at the bounds

At iterate    0    f=  8.07490D+00    |proj g|=  1.38161D-01

At iterate    5    f=  8.05649D+00    |proj g|=  2.90166D-03

At iterate   10    f=  8.05649D+00    |proj g|=  1.34590D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     10     12      1     0     0   1.346D-04   8.056D+00
 F =   8.0564873988801775

          CONVERGENCE: REL_REDUCTION_OF_F_<=_FACTR*EPSMCH

          /Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
          ency M will be used.
            warnings.warn('No frequency information was'
          /Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:524: ValueWarning: No frequency information was provided, so inferred frequ
          ency M will be used.
            warnings.warn('No frequency information was'
           This problem is unconstrained.
          /Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:132: FutureWarning: The 'freq' argument in Timestamp is deprecated and will
          be removed in a future version.
            date_key = Timestamp(key, freq=base_index.freq)

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x124f5b6a0>

Out[41]: <matplotlib.collections.PolyCollection at 0x130ecd4c0>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Union County')

Out[41]: <matplotlib.legend.Legend at 0x13231a7f0>
```



The Mean Squared Error of our forecasts for Union County county is 11027.95
RUNNING THE L-BFGS-B CODE

*** * ***

Machine precision = 2.220D-16
N = 5 M = 10

At X0 0 variables are exactly at the bounds

At iterate 0 f= 8.15662D+00 |proj g|= 3.69787D-01

At iterate 5 f= 8.12671D+00 |proj g|= 2.08743D-02

/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:581: ValueWarning: A date index has been provided, but it has no associated frequency information and so will be ignored when e.g. forecasting.
  warnings.warn('A date index has been provided, but it has no'
 This problem is unconstrained.

```
At iterate   10    f=  8.12615D+00    |proj g|=  9.43132D-05

At iterate   15    f=  8.12615D+00    |proj g|=  3.10003D-03

At iterate   20    f=  8.12532D+00    |proj g|=  2.68100D-02

At iterate   25    f=  8.11169D+00    |proj g|=  2.83213D-02

At iterate   30    f=  8.11066D+00    |proj g|=  1.89958D-04

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5     32     35      1     0     0   1.677D-06   8.111D+00
  F =   8.1106566517039145

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL
```

Out[41]: <AxesSubplot:xlabel='Month'>

Out[41]: <matplotlib.collections.PolyCollection at 0x12c3ced90>

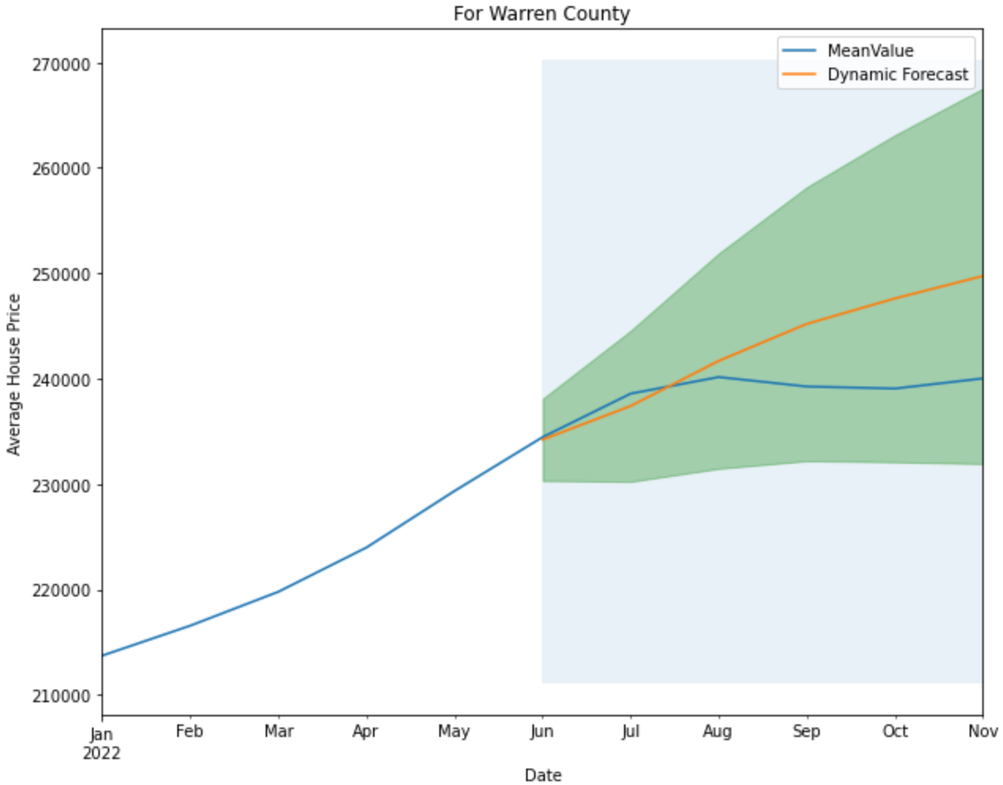
Out[41]: <matplotlib.collections.PolyCollection at 0x12ddbbe20>

Out[41]: Text(0.5, 0, 'Date')

Out[41]: Text(0, 0.5, 'Average House Price')

Out[41]: Text(0.5, 1.0, 'For Warren County')

Out[41]: <matplotlib.legend.Legend at 0x150717640>



The Mean Squared Error of our forecasts for Warren County county is 5868.99

In [42]:
```
summary
```

Out[42]:

	County	Sqrt_MSE
0	Atlantic County	9372.017216
1	Bergen County	10310.674222
2	Burlington County	12200.212958
3	Camden County	11940.898071
4	Cape May County	25658.031677
5	Cumberland County	15631.616939
6	Essex County	3066.634473
7	Gloucester County	8666.254671
8	Hudson County	19515.924784
9	Hunterdon County	8797.713752
10	Mercer County	9142.401576
11	Middlesex County	16772.253120
12	Monmouth County	15822.901074
13	Morris County	11084.606797
14	Ocean County	11818.449875
15	Passaic County	3117.282044
16	Salem County	1439.347281
17	Somerset County	12371.827175
18	Sussex County	9281.568901
19	Union County	11027.945333
20	Warren County	5868.991999

In [43]:
```
#Final Model
forecast = pd.DataFrame()
current = []
forecast_2Yr = []
forecast_3Yr =[]
forecast_5Yr = []
forecast_10Yr =[]
forecast_15Yr = []
forecast_20Yr =[]
forecast_30Yr = []

conf_3Yr_L=[]
conf_3Yr_U=[]
conf_5Yr_L=[]
conf_5Yr_U=[]
for cnty, output, df in zip(county, models, county_dfs):
    pred_2 = output.get_forecast(steps = 24)
    pred_conf_2 = pred_2.conf_int()
    forecast_2 = pred_2.predicted_mean.to_numpy()[-1]

    pred_3 = output.get_forecast(steps = 36)
    pred_conf_3 = pred_3.conf_int()
    forecast_3 = pred_3.predicted_mean.to_numpy()[-1]

    pred_5 = output.get_forecast(steps = 60)
    pred_conf_5 = pred_5.conf_int()
    forecast_5 = pred_5.predicted_mean.to_numpy()[-1]

    pred_10 = output.get_forecast(steps = 120)
    pred_conf_10 = pred_10.conf_int()
    forecast_10 = pred_10.predicted_mean.to_numpy()[-1]

    pred_15 = output.get_forecast(steps = 180)
    pred_conf_15 = pred_15.conf_int()
    forecast_15 = pred_15.predicted_mean.to_numpy()[-1]

    pred_20 = output.get_forecast(steps = 240)
    pred_conf_20 = pred_20.conf_int()
    forecast_20 = pred_20.predicted_mean.to_numpy()[-1]

    pred_30 = output.get_forecast(steps = 360)
    pred_conf_30 = pred_30.conf_int()
    forecast_30 = pred_30.predicted_mean.to_numpy()[-1]

    current.append(df['2022-11']['MeanValue'][0])

    forecast_2Yr.append(forecast_2)
    forecast_3Yr.append(forecast_3)
    conf_3Yr_L.append((pred_conf_3.iloc[:, 0]).iloc[-1])
    conf_3Yr_U.append((pred_conf_3.iloc[:, 1]).iloc[-1])
    forecast_5Yr.append(forecast_5)
    conf_5Yr_L.append((pred_conf_5.iloc[:, 0]).iloc[-1])
    conf_5Yr_U.append((pred_conf_5.iloc[:, 1]).iloc[-1])
    forecast_10Yr.append(forecast_10)
    forecast_15Yr.append(forecast_15)
    forecast_20Yr.append(forecast_20)
    forecast_30Yr.append(forecast_30)


forecast['County'] = county
forecast['Current Value'] = current
```

```
forecast['2 Years Value'] = forecast_2Yr
forecast['3 Years Value'] = forecast_3Yr
forecast['5 Years Value'] = forecast_5Yr
forecast['10 Years Value'] = forecast_10Yr
forecast['15 Years Value'] = forecast_15Yr
forecast['20 Years Value'] = forecast_20Yr
forecast['30 Years Value'] = forecast_30Yr

forecast['3 Years Lower'] = conf_3Yr_L
forecast['3 Years Upper'] = conf_3Yr_U
forecast['5 Years Lower'] = conf_5Yr_L
forecast['5 Years Upper'] = conf_5Yr_U

forecast['2Yr-ROI']=((forecast['2 Years Value'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)
forecast['3Yr-ROI']=((forecast['3 Years Value'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)
forecast['5Yr-ROI']=((forecast['5 Years Value'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)
forecast['10Yr-ROI']=((forecast['10 Years Value'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)
forecast['15Yr-ROI']=((forecast['15 Years Value'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)
forecast['20Yr-ROI']=((forecast['20 Years Value'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)
forecast['30Yr-ROI']=((forecast['30 Years Value'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)


forecast['3Yr-ROI-Lower']=((forecast['3 Years Lower'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)
forecast['3Yr-ROI-Upper']=((forecast['3 Years Upper'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)
forecast['5Yr-ROI-Lower']=((forecast['5 Years Lower'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)
forecast['5Yr-ROI-Upper']=((forecast['5 Years Upper'] - forecast['Current Value'])/forecast['Current Value']).map('{:,.2f}'.format)

with pd.option_context('display.max_rows', None, 'display.max_columns', None):  # more options can be specified also
    print(forecast)
```

```
/var/folders/55/nnkwdj2178z85z6pk884kg40mpppbj/T/ipykernel_76707/1541991435.py:45: FutureWarning: Indexing a DataFrame with a datetimelike index using a single string to slice the rows, like `frame[string]`, is deprecated and will be removed in a future version. Use `frame.loc[string]` instead.
  current.append(df['2022-11']['MeanValue'][0])
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
```

	County	Current Value	2 Years Value	3 Years Value
0	Atlantic County	622297.0	700893.535512	740289.840527
1	Bergen County	465069.0	564331.110143	613872.007608
2	Burlington County	565603.0	669701.787309	722274.489986
3	Camden County	593026.0	683284.408752	728409.406636
4	Cape May County	634207.0	725345.756486	768261.003945
5	Cumberland County	471976.0	566337.903321	612128.152202
6	Essex County	518810.0	604619.807020	647626.986618
7	Gloucester County	288005.0	338057.524423	362911.294494
8	Hudson County	477392.0	543385.824058	576167.782186
9	Hunterdon County	568032.0	654971.275099	698407.109063
10	Mercer County	348080.0	437992.133579	482976.176587
11	Middlesex County	359738.0	450686.873124	496587.219942
12	Monmouth County	546582.0	630189.265797	670797.868726
13	Morris County	311950.0	375955.383990	407774.185378
14	Ocean County	319267.0	427518.312652	481816.818835
15	Passaic County	223478.0	278630.046288	305955.457495
16	Salem County	373152.0	457197.889991	499194.711551
17	Somerset County	513431.0	591147.059233	630096.028475
18	Sussex County	349646.0	415294.763864	448577.293265
19	Union County	640165.0	787738.523730	861446.257909
20	Warren County	240023.0	313944.024968	350718.856724

	5 Years Value	10 Years Value	15 Years Value	20 Years Value
0	8.181675e+05	1.013055e+06	1.208001e+06	1.402942e+06
1	7.123119e+05	9.585717e+05	1.204849e+06	1.451125e+06
2	8.251252e+05	1.082688e+06	1.340433e+06	1.598161e+06
3	8.190267e+05	1.045482e+06	1.271923e+06	1.498364e+06
4	8.561697e+05	1.075258e+06	1.294147e+06	1.513054e+06
5	7.019519e+05	9.266818e+05	1.151489e+06	1.376291e+06
6	7.331463e+05	9.470420e+05	1.160975e+06	1.374904e+06
7	4.136294e+05	5.401638e+05	6.666668e+05	7.931704e+05
8	6.407707e+05	8.024921e+05	9.642554e+05	1.126017e+06
9	7.842407e+05	9.990701e+05	1.213942e+06	1.428813e+06
10	5.726799e+05	7.970040e+05	1.021339e+06	1.245673e+06
11	5.864568e+05	8.115124e+05	1.036711e+06	1.261898e+06
12	7.574804e+05	9.731074e+05	1.188330e+06	1.403586e+06
13	4.726876e+05	6.346604e+05	7.965823e+05	9.585057e+05
14	5.884794e+05	8.556006e+05	1.122801e+06	1.389998e+06
15	3.597296e+05	4.943823e+05	6.290558e+05	7.637290e+05
16	5.828446e+05	7.920528e+05	1.001273e+06	1.210493e+06
17	7.056579e+05	8.951079e+05	1.084664e+06	1.274215e+06
18	5.132425e+05	6.752333e+05	8.373998e+05	9.995461e+05
19	1.007190e+06	1.371835e+06	1.736623e+06	2.101396e+06
20	4.236407e+05	6.061009e+05	7.885760e+05	9.710508e+05

	30 Years Value	3 Years Lower	3 Years Upper	5 Years Lower
0	1.792826e+06	600330.014901	8.802497e+05	543406.482380
1	1.943678e+06	477464.493243	7.502795e+05	440094.797199
2	2.113621e+06	569577.224031	8.749718e+05	530386.079932
3	1.951246e+06	530530.093271	9.262887e+05	436476.552887
4	1.950864e+06	586443.961286	9.500780e+05	481346.849310
5	1.825896e+06	445066.867863	7.791894e+05	356563.332213
6	1.802764e+06	517547.204955	7.777068e+05	482688.827694
7	1.046178e+06	281486.775944	4.443358e+05	249641.628176
8	1.449540e+06	424398.450721	7.279371e+05	334709.767259
9	1.858555e+06	526974.203796	8.698400e+05	442334.370792
10	1.694341e+06	391439.190474	5.745132e+05	387900.227172
11	1.712273e+06	396058.565165	5.971159e+05	388542.427313
12	1.834093e+06	511077.920386	8.305178e+05	447454.569686
13	1.282353e+06	322311.061075	4.932373e+05	301682.153850

14	1.924393e+06	352121.332129	6.115123e+05	329090.351871
15	1.033076e+06	231527.076053	3.803838e+05	217834.641498
16	1.628933e+06	375531.614806	6.228578e+05	340914.783710
17	1.653319e+06	484692.430243	7.754996e+05	420328.975693
18	1.323843e+06	340800.149471	5.563544e+05	303871.204313
19	2.830945e+06	626800.643279	1.096092e+06	541238.927464
20	1.336000e+06	266895.086865	4.345426e+05	265973.402755

	5 Years Upper	2Yr-ROI	3Yr-ROI	5Yr-ROI	10Yr-ROI	15Yr-ROI	20Yr-ROI	30Yr-ROI	\
0	1.092929e+06	0.13	0.19	0.31	0.63	0.94	1.25	1.88	
1	9.845291e+05	0.21	0.32	0.53	1.06	1.59	2.12	3.18	
2	1.119864e+06	0.18	0.28	0.46	0.91	1.37	1.83	2.74	
3	1.201577e+06	0.15	0.23	0.38	0.76	1.14	1.53	2.29	
4	1.230993e+06	0.14	0.21	0.35	0.70	1.04	1.39	2.08	
5	1.047340e+06	0.20	0.30	0.49	0.96	1.44	1.92	2.87	
6	9.836038e+05	0.17	0.25	0.41	0.83	1.24	1.65	2.47	
7	5.776172e+05	0.17	0.26	0.44	0.88	1.31	1.75	2.63	
8	9.468316e+05	0.14	0.21	0.34	0.68	1.02	1.36	2.04	
9	1.126147e+06	0.15	0.23	0.38	0.76	1.14	1.52	2.27	
10	7.574595e+05	0.26	0.39	0.65	1.29	1.93	2.58	3.87	
11	7.843711e+05	0.25	0.38	0.63	1.26	1.88	2.51	3.76	
12	1.067506e+06	0.15	0.23	0.39	0.78	1.17	1.57	2.36	
13	6.436930e+05	0.21	0.31	0.52	1.03	1.55	2.07	3.11	
14	8.478684e+05	0.34	0.51	0.84	1.68	2.52	3.35	5.03	
15	5.016246e+05	0.25	0.37	0.61	1.21	1.81	2.42	3.62	
16	8.247744e+05	0.23	0.34	0.56	1.12	1.68	2.24	3.37	
17	9.909868e+05	0.15	0.23	0.37	0.74	1.11	1.48	2.22	
18	7.226138e+05	0.19	0.28	0.47	0.93	1.39	1.86	2.79	
19	1.473142e+06	0.23	0.35	0.57	1.14	1.71	2.28	3.42	
20	5.813080e+05	0.31	0.46	0.77	1.53	2.29	3.05	4.57	

	3Yr-ROI-Lower	3Yr-ROI-Upper	5Yr-ROI-Lower	5Yr-ROI-Upper
0	-0.04	0.41	-0.13	0.76
1	0.03	0.61	-0.05	1.12
2	0.01	0.55	-0.06	0.98
3	-0.11	0.56	-0.26	1.03
4	-0.08	0.50	-0.24	0.94
5	-0.06	0.65	-0.24	1.22
6	-0.00	0.50	-0.07	0.90
7	-0.02	0.54	-0.13	1.01
8	-0.11	0.52	-0.30	0.98
9	-0.07	0.53	-0.22	0.98
10	0.12	0.65	0.11	1.18
11	0.10	0.66	0.08	1.18
12	-0.06	0.52	-0.18	0.95
13	0.03	0.58	-0.03	1.06
14	0.10	0.92	0.03	1.66
15	0.04	0.70	-0.03	1.24
16	0.01	0.67	-0.09	1.21
17	-0.06	0.51	-0.18	0.93
18	-0.03	0.59	-0.13	1.07
19	-0.02	0.71	-0.15	1.30
20	0.11	0.81	0.11	1.42

```
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/statsmodels/tsa/base/tsa_model.py:376: ValueWarning: No supported index is available. Prediction results will be given with an integer index beginning at `start`.
  warnings.warn('No supported index is available.'
```

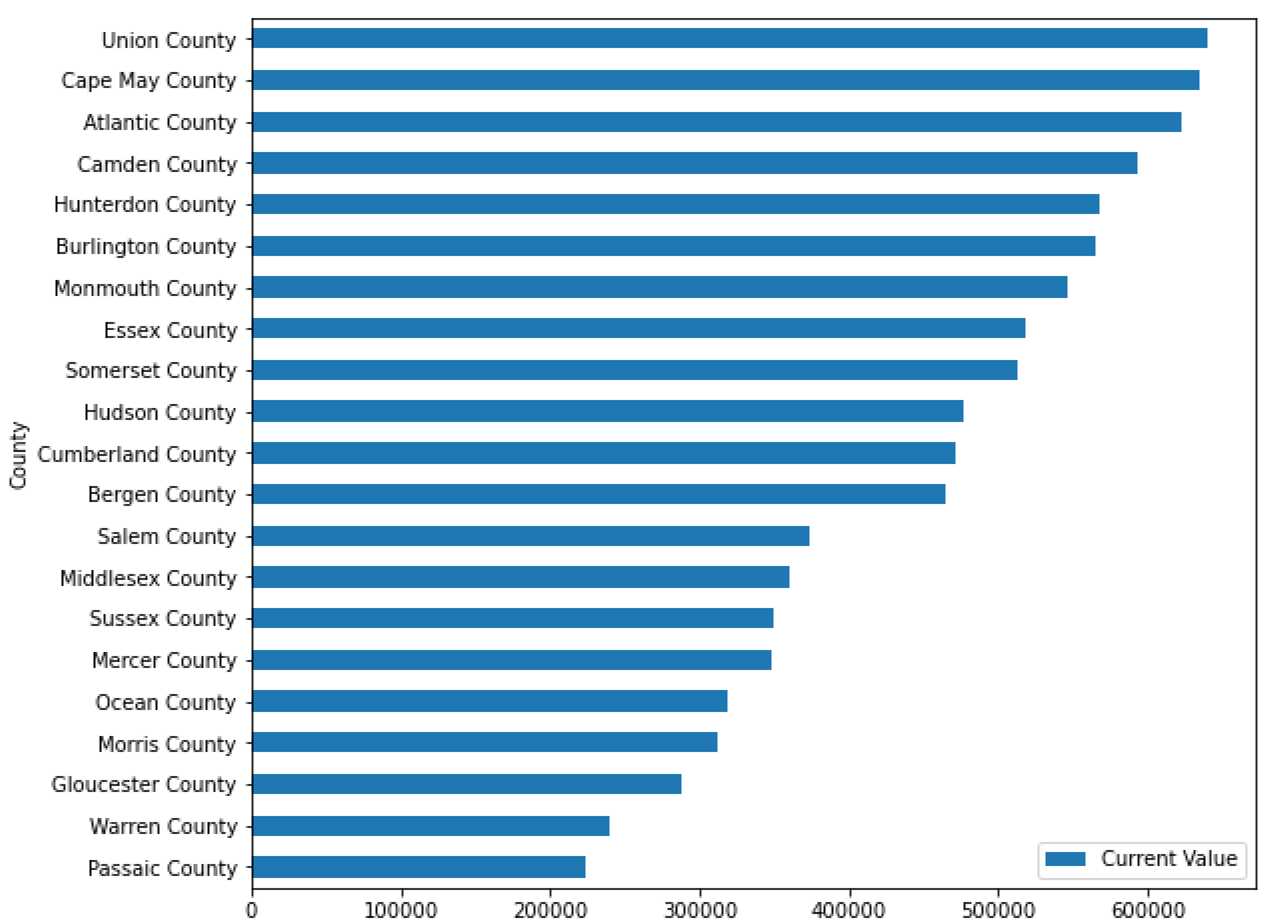
In [44]:

```
forecast.to_csv('/Users/nilanjana.chatterjee/DS670_UI/house_price_pred.csv')
```

In [45]:

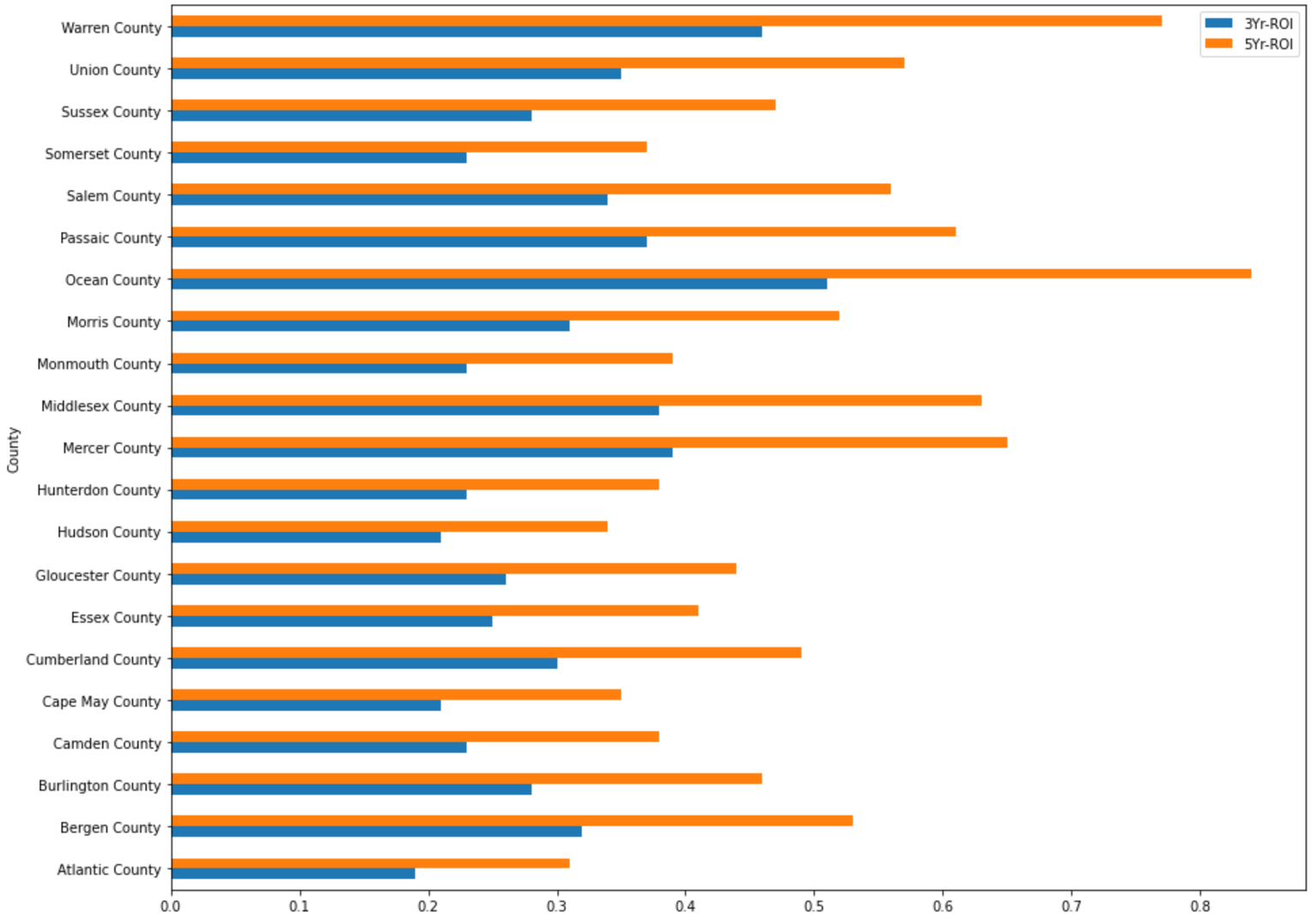
```
cur_df=forecast[['County','Current Value']].copy()
cur_df.set_index('County', inplace = True)
cur_df['Current Value'] = cur_df['Current Value'].astype(int)
cur_df = cur_df.sort_values('Current Value', ascending = True)
cur_df.plot.barh(figsize=(9,8))
```

Out[45]: <AxesSubplot:ylabel='County'>



In [46]:
```
roi_df=forecast[['County','3Yr-ROI', '5Yr-ROI']].copy()
roi_df.set_index('County', inplace = True)
roi_df['3Yr-ROI'] = roi_df['3Yr-ROI'].astype(float)
roi_df['5Yr-ROI'] = roi_df['5Yr-ROI'].astype(float)

roi_df.plot.barh(figsize=(15,12))
```

Out[46]: <AxesSubplot:ylabel='County'>



In [47]:
```
roi3_df=roi_df[['3Yr-ROI']].copy()
roi3_df = roi3_df.sort_values('3Yr-ROI', ascending = True)

roi3_df.plot.barh(figsize=(9,8))
```

Out[47]: <AxesSubplot:ylabel='County'>

In [48]:
```
roi5_df=roi_df[['5Yr-ROI']].copy()
roi5_df = roi5_df.sort_values('5Yr-ROI', ascending = True)

roi5_df.plot.barh(figsize=(9,8), color={"5Yr-ROI": "DarkOrange"})
```

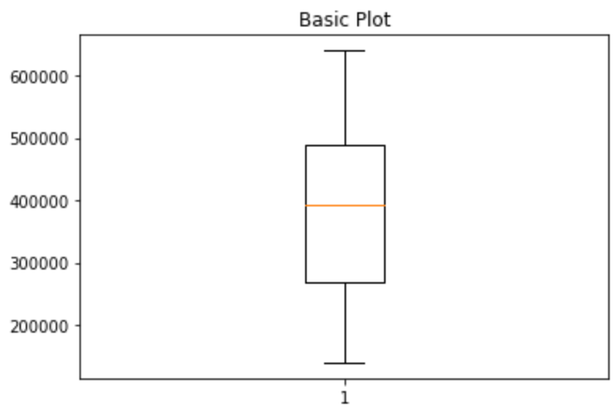
Out[48]: <AxesSubplot:ylabel='County'>

In [49]:
```
print(dfm['2020':]['MeanValue'].mean())

fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(dfm['2020':]['MeanValue'])
```

```
381918.4693877551
```

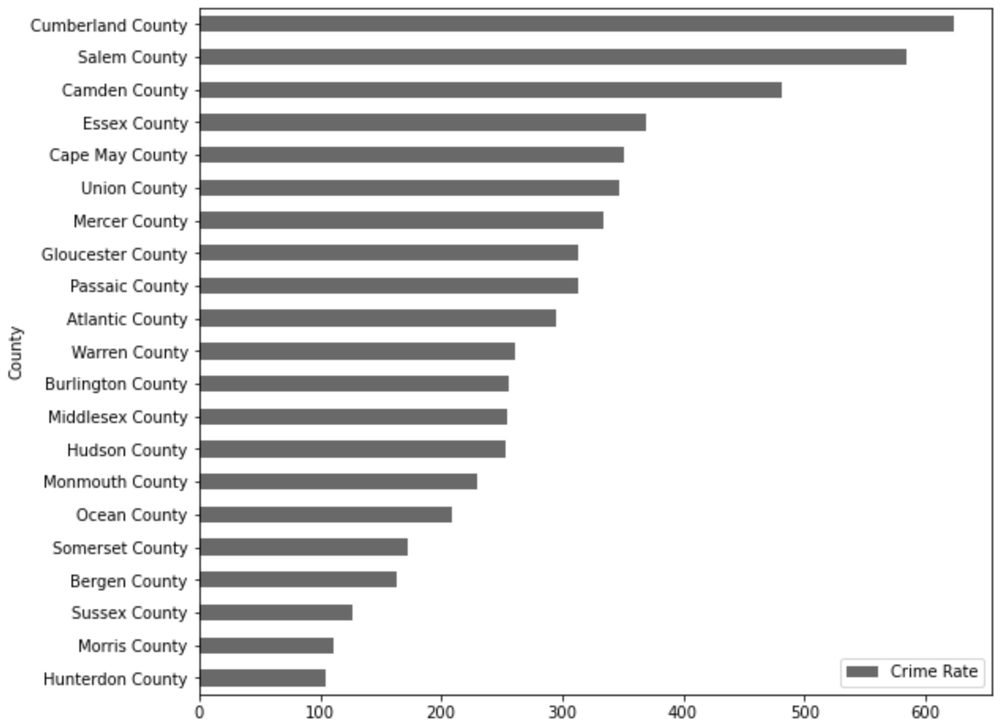
Out[49]:
```
Text(0.5, 1.0, 'Basic Plot')
```

Out[49]:
```
{'whiskers': [<matplotlib.lines.Line2D at 0x124e4ad60>,
  <matplotlib.lines.Line2D at 0x124e4ac10>],
 'caps': [<matplotlib.lines.Line2D at 0x152750e50>,
  <matplotlib.lines.Line2D at 0x152750610>],
 'boxes': [<matplotlib.lines.Line2D at 0x124e4a610>],
 'medians': [<matplotlib.lines.Line2D at 0x1527507f0>],
 'fliers': [<matplotlib.lines.Line2D at 0x152750a90>],
 'means': []}
```



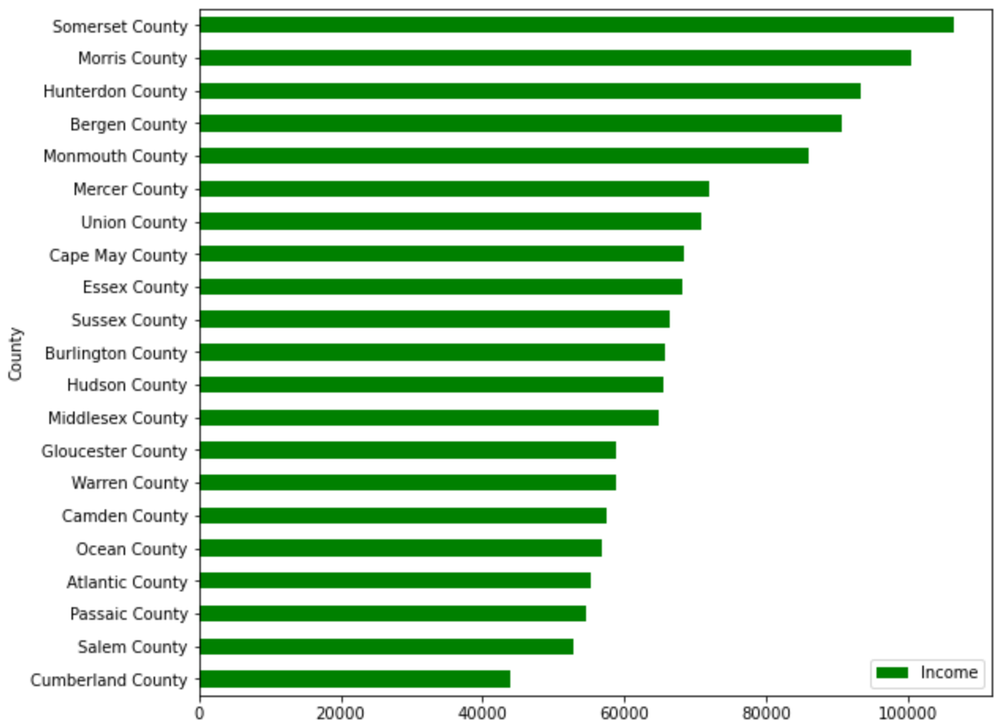
In [50]:
```
#Crime Rate
crime_df = pd.read_excel(r'Crime Rate.xlsx', sheet_name='Summary')
crime_df.set_index('County', inplace = True)
crime_df = crime_df.sort_values('Crime Rate', ascending = True)
crime_df.plot.barh(figsize=(9,8), color={"Crime Rate": "DimGray"})
```

Out[50]: <AxesSubplot:ylabel='County'>



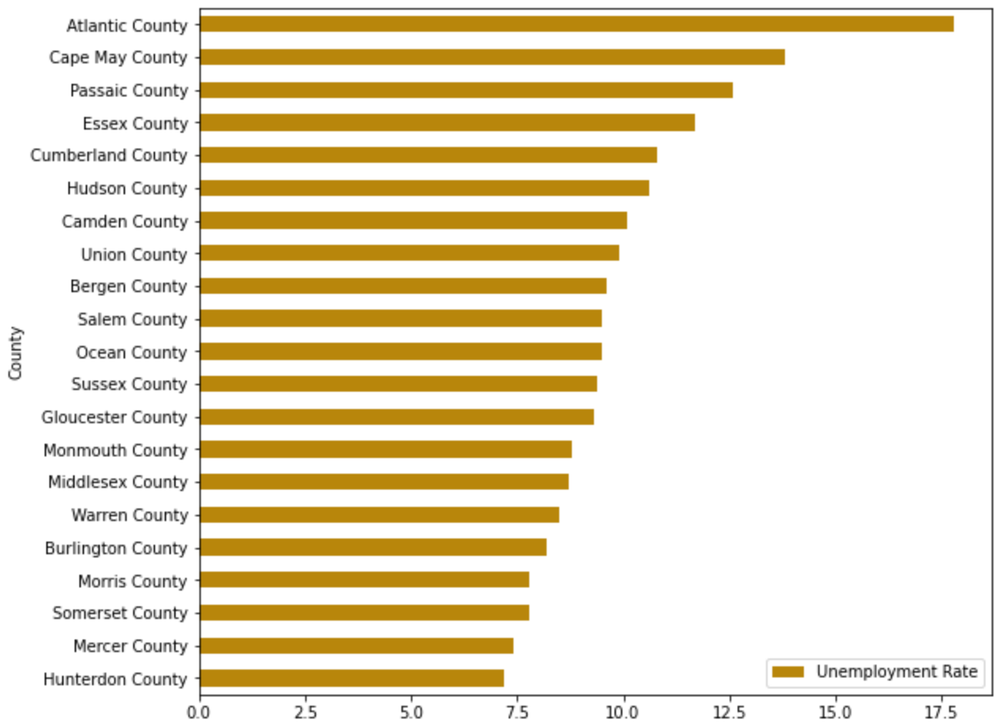
In [51]:
```
#Income
income_df = pd.read_excel(r'Income_Data.xlsx', sheet_name='Summary')
income_df.set_index('County', inplace = True)
income_df = income_df.sort_values('Income', ascending = True)
income_df.plot.barh(figsize=(9,8), color={"Income": "Green"})
```

Out[51]: <AxesSubplot:ylabel='County'>



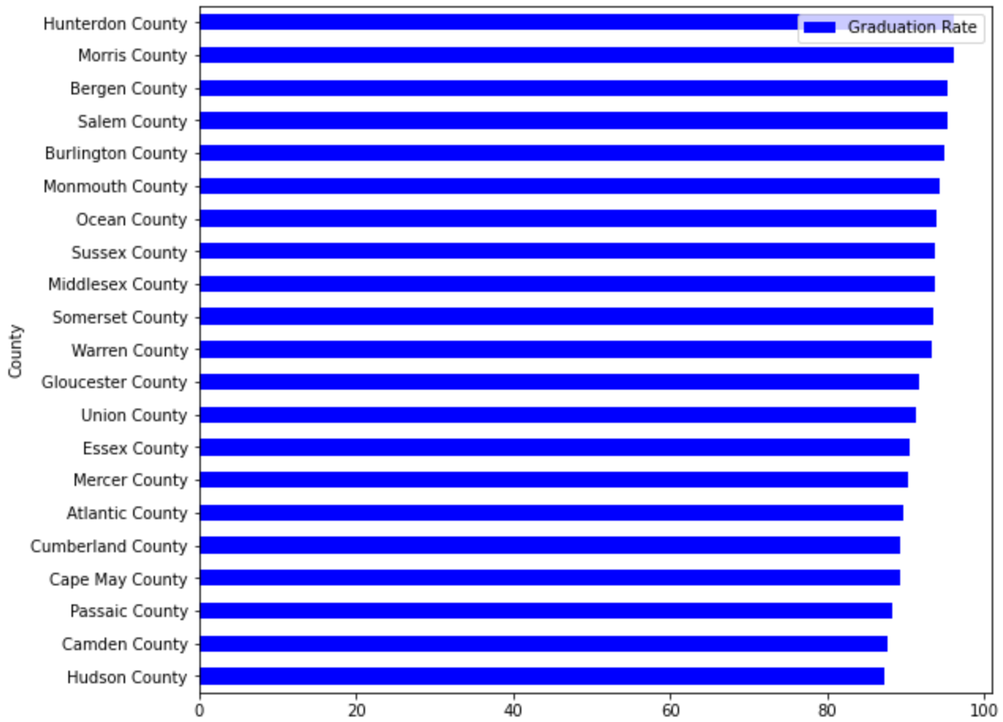
In [52]:
```
#Unemployment Rate
Unemployment_df = pd.read_excel(r'Unemployment_Data.xlsx', sheet_name='Summary')
Unemployment_df.set_index('County', inplace = True)
Unemployment_df = Unemployment_df.sort_values('Unemployment Rate', ascending = True)
Unemployment_df.plot.barh(figsize=(9,8), color={"Unemployment Rate": "DarkGoldenrod"})
```

Out[52]: <AxesSubplot:ylabel='County'>



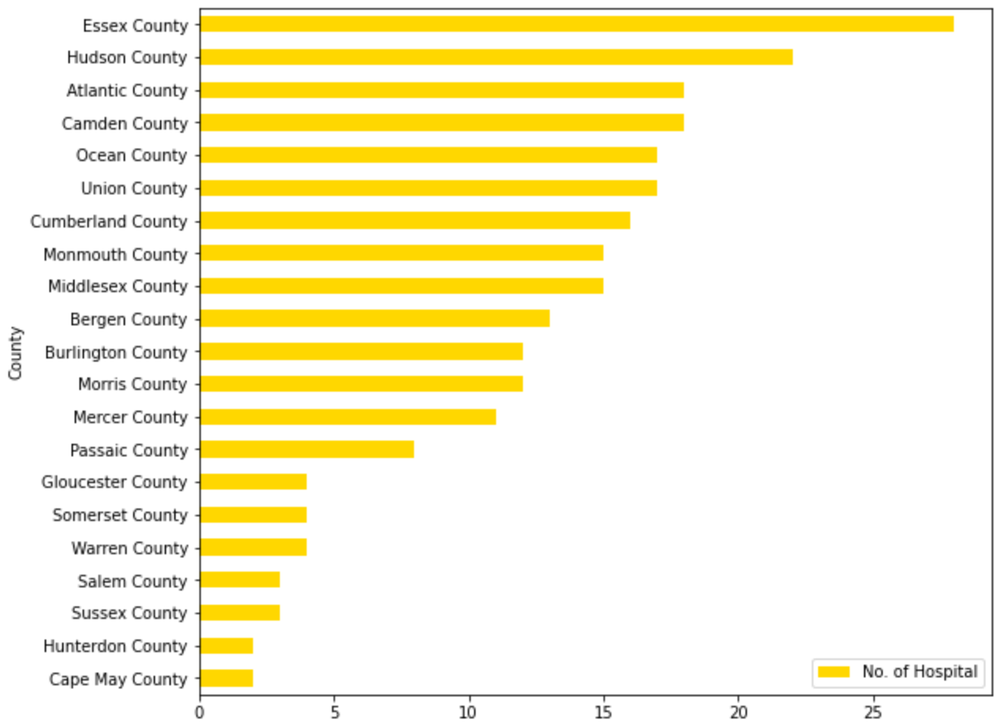
In [53]:
```
#School Graduation Rate
SchoolGraduationRate_df = pd.read_excel(r'SchoolGraduationRate.xlsx')
SchoolGraduationRate_df.set_index('County', inplace = True)
SchoolGraduationRate_df = SchoolGraduationRate_df.sort_values('Graduation Rate', ascending = True)
SchoolGraduationRate_df.plot.barh(figsize=(9,8), color={"Graduation Rate": "Blue"})
```

Out[53]: <AxesSubplot:ylabel='County'>



In [54]:
```
#Number of Hospitals
Hospital_df = pd.read_excel(r'Hospitals.xlsx')
Hospital_df.set_index('County', inplace = True)
Hospital_df = Hospital_df.sort_values('No. of Hospital', ascending = True)
Hospital_df.plot.barh(figsize=(9,8), color={"No. of Hospital": "Gold"})
```

Out[54]: <AxesSubplot:ylabel='County'>



In [55]:
```
#Combining all the parameters with the respective county
comb_df = crime_df.copy()
comb_df = comb_df.join(income_df , on='County')
comb_df = comb_df.join(Unemployment_df , on='County')
comb_df = comb_df.join(SchoolGraduationRate_df , on='County')
comb_df = comb_df.join(Hospital_df , on='County')

comb_df.head()
```

Out[55]:

County	Crime Rate	Income	Unemployment Rate	Graduation Rate	No. of Hospital
Hunterdon County	105.2	93279	7.2	96.133333	2
Morris County	110.7	100617	7.8	96.092000	12
Sussex County	126.7	66431	9.4	93.655556	3
Bergen County	163.4	90759	9.6	95.343750	13
Somerset County	171.8	106558	7.8	93.566667	4

In [56]:
```
#Median Price
medv = dfm['2020':'2020']
medv = pd.DataFrame(medv.groupby(["RegionName"])["MeanValue"].median())
medv.rename(columns={'MeanValue': 'MedianPrice'}, inplace=True)
medv.index.names = ['County']

medv.head()
```

Out[56]:

County	MedianPrice
Atlantic County	209624.5
Bergen County	507058.0
Burlington County	244567.0
Camden County	195833.5
Cape May County	417091.5

In [57]:
```
comb_df = comb_df.join(medv , on='County')
comb_df.head()
```

Out[57]:

County	Crime Rate	Income	Unemployment Rate	Graduation Rate	No. of Hospital	MedianPrice
Hunterdon County	105.2	93279	7.2	96.133333	2	394491.5
Morris County	110.7	100617	7.8	96.092000	12	449321.0
Sussex County	126.7	66431	9.4	93.655556	3	262761.0
Bergen County	163.4	90759	9.6	95.343750	13	507058.0
Somerset County	171.8	106558	7.8	93.566667	4	420302.0

In [58]:
```
print(comb_df.shape)
print("------------------")
print(comb_df.dtypes)
print("------------------")
print(comb_df.nunique())
print("------------------")
print(comb_df.isnull().sum())
```

```
(21, 6)
------------------
Crime Rate           float64
Income                 int64
Unemployment Rate    float64
Graduation Rate      float64
No. of Hospital        int64
MedianPrice          float64
dtype: object
------------------
Crime Rate           21
Income               21
Unemployment Rate    19
Graduation Rate      21
No. of Hospital      13
MedianPrice          21
dtype: int64
------------------
Crime Rate           0
Income               0
Unemployment Rate    0
Graduation Rate      0
No. of Hospital      0
MedianPrice          0
dtype: int64
```

In [59]:
```
comb_df.describe()
```

Out[59]:

	Crime Rate	Income	Unemployment Rate	Graduation Rate	No. of Hospital	MedianPrice
count	21.000000	21.000000	21.000000	21.000000	21.000000	21.000000
mean	293.233333	69417.333333	9.952381	92.160673	11.619048	329355.809524
std	139.428678	16728.201999	2.452676	2.853231	7.351709	114193.449061
min	105.200000	43844.000000	7.200000	87.318182	2.000000	146339.000000
25%	209.300000	57483.000000	8.500000	89.590000	4.000000	244567.000000
50%	261.100000	65654.000000	9.500000	93.333333	12.000000	352808.500000
75%	347.300000	71990.000000	10.600000	94.328125	17.000000	420302.000000
max	623.700000	106558.000000	17.800000	96.133333	28.000000	507058.000000

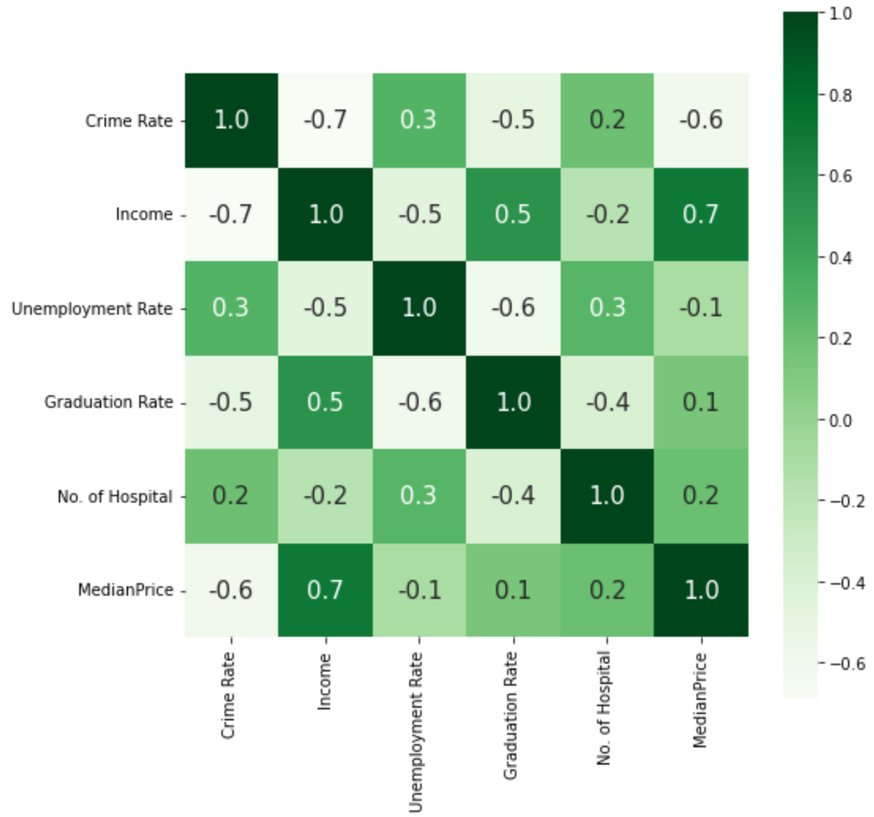
In [60]:
```
# Finding out the correlation between the features
corr = comb_df.corr()
corr.shape
```

Out[60]:
```
(6, 6)
```

In [61]:
```
plt.figure(figsize=(8,8))
sns.heatmap(corr, cbar=True, square= True, fmt='.1f', annot=True, annot_kws={'size':15}, cmap='Greens')
```

Out[61]: <Figure size 576x576 with 0 Axes>

Out[61]: <AxesSubplot:>



In [62]:
```
# Spliting target variable and independent variables
x = comb_df.drop(['MedianPrice'], axis = 1)
y = comb_df['MedianPrice']
```

In [63]:
```
# Splitting to training and testing data

from sklearn.model_selection import train_test_split
x_train, x_test,y_train, y_test = train_test_split(x,y, test_size = 0.3, random_state = 4)
```

In [64]:
```
# Import library for Linear Regression
from sklearn.linear_model import LinearRegression

# Create a Linear regressor
lm = LinearRegression()

# Train the model using the training sets
lm.fit(x_train, y_train)
```

Out[64]: LinearRegression()

In [65]:
```
#Converting the coefficient values to a dataframe
coeffcients = pd.DataFrame([x_train.columns,lm.coef_]).T
coeffcients = coeffcients.rename(columns={0: 'Attribute', 1: 'Coefficients'})
coeffcients
```

Out[65]:

	Attribute	Coefficients
0	Crime Rate	-262.993378
1	Income	4.068048
2	Unemployment Rate	-3911.659411
3	Graduation Rate	-22962.249812
4	No. of Hospital	4124.952704

In [66]:
```
# Model prediction on train data
y_pred = lm.predict(x_train)
```

In [67]:
```
from sklearn import metrics

# Model Evaluation
print('R^2:',metrics.r2_score(y_train, y_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(len(y_train)-x_train.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',metrics.mean_squared_error(y_train, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
R^2: 0.6005243999626726
Adjusted R^2: 0.350852149939343
MAE: 48247.54803735217
MSE: 3471429957.639339
RMSE: 58918.84212744968
```

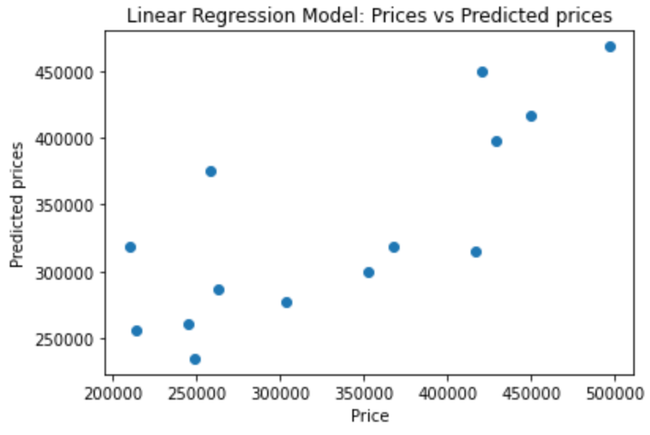
In [68]:
```
# Visualizing the differences between actual prices and predicted values
plt.scatter(y_train, y_pred)
plt.xlabel("Price")
plt.ylabel("Predicted prices")
plt.title("Linear Regression Model: Prices vs Predicted prices")
plt.show()
```

Out[68]: <matplotlib.collections.PathCollection at 0x1576ebe80>

Out[68]: Text(0.5, 0, 'Price')

Out[68]: Text(0, 0.5, 'Predicted prices')

Out[68]: Text(0.5, 1.0, 'Linear Regression Model: Prices vs Predicted prices')



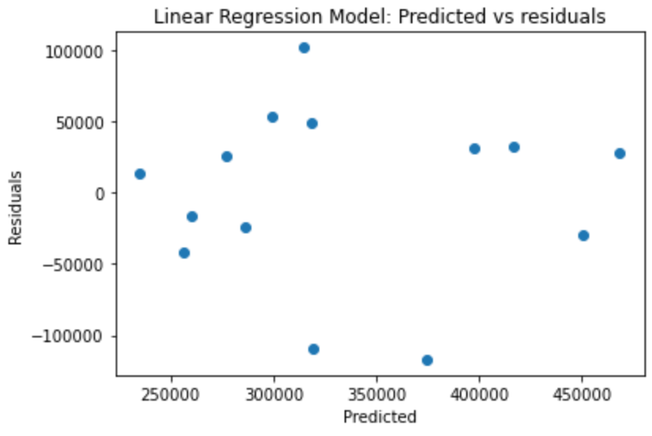
In [69]:
```
# Checking residuals
plt.scatter(y_pred,y_train-y_pred)
plt.title("Linear Regression Model: Predicted vs residuals")
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.show()
```

Out[69]: <matplotlib.collections.PathCollection at 0x157753be0>

Out[69]: Text(0.5, 1.0, 'Linear Regression Model: Predicted vs residuals')

Out[69]: Text(0.5, 0, 'Predicted')

Out[69]: Text(0, 0.5, 'Residuals')



In [70]:
```
# Checking Normality of errors
sns.distplot(y_train-y_pred)
plt.title("Histogram of Residuals")
plt.xlabel("Residuals")
plt.ylabel("Frequency")
plt.show()
```

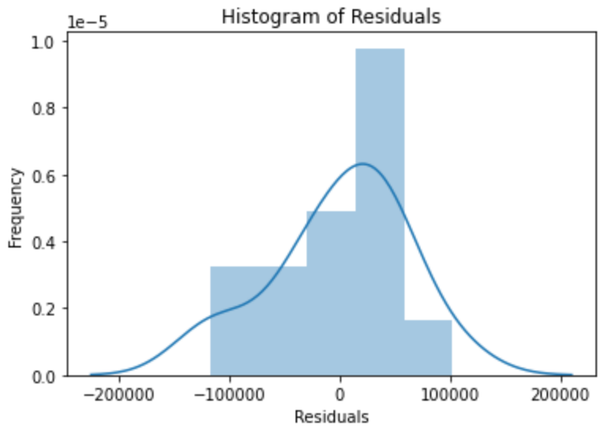
```
/Users/nilanjana.chatterjee/opt/anaconda3/lib/python3.9/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a fu
ture version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[70]: <AxesSubplot:xlabel='MedianPrice', ylabel='Density'>

Out[70]: Text(0.5, 1.0, 'Histogram of Residuals')

Out[70]: Text(0.5, 0, 'Residuals')

Out[70]: Text(0, 0.5, 'Frequency')



In [71]:
```
# Predicting Test data with the model
y_test_pred = lm.predict(x_test)
# Model Evaluation
acc_linreg = metrics.r2_score(y_test, y_test_pred)
print('R^2:', acc_linreg)
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_test, y_test_pred))*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_test, y_test_pred))
print('MSE:',metrics.mean_squared_error(y_test, y_test_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
R^2: 0.5629765619848929
Adjusted R^2: -1.6221406280906425
MAE: 83847.84374889592
MSE: 8633553263.055128
RMSE: 92916.9159144616
```

In [72]:
```
# Import Random Forest Regressor
from sklearn.ensemble import RandomForestRegressor

# Create a Random Forest Regressor
reg = RandomForestRegressor()

# Train the model using the training sets
reg.fit(x_train, y_train)
```

Out[72]:
```
RandomForestRegressor()
```

In [73]:
```
# Model prediction on train data
y_pred = reg.predict(x_train)
# Model Evaluation
print('R^2:',metrics.r2_score(y_train, y_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(len(y_train)-x_train.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',metrics.mean_squared_error(y_train, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
R^2: 0.8539075047392677
Adjusted R^2: 0.76259969520131
MAE: 32725.217142857142
MSE: 1269539027.1320713
RMSE: 35630.591170117725
```

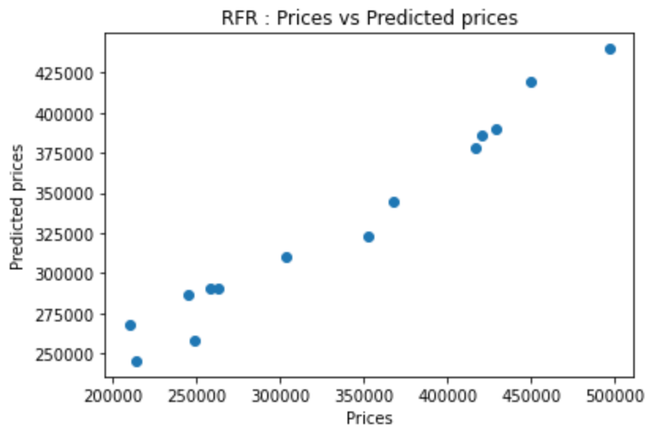
In [74]:
```
# Visualizing the differences between actual prices and predicted values
plt.scatter(y_train, y_pred)
plt.xlabel("Prices")
plt.ylabel("Predicted prices")
plt.title("RFR : Prices vs Predicted prices")
plt.show()
```

Out[74]:
```
<matplotlib.collections.PathCollection at 0x1597541c0>
```

Out[74]:
```
Text(0.5, 0, 'Prices')
```

Out[74]:
```
Text(0, 0.5, 'Predicted prices')
```

Out[74]:
```
Text(0.5, 1.0, 'RFR : Prices vs Predicted prices')
```



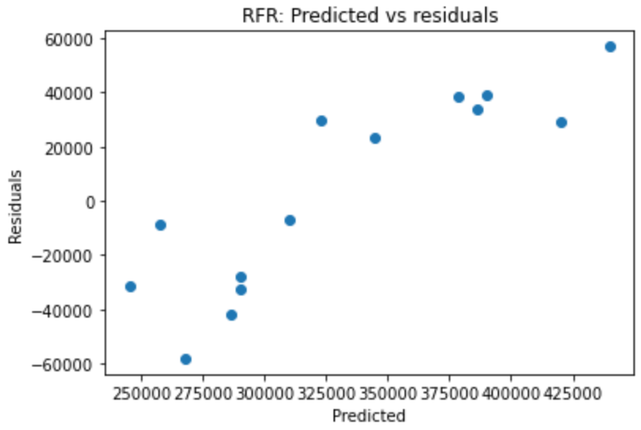
In [75]:
```
# Checking residuals
plt.scatter(y_pred,y_train-y_pred)
plt.title("RFR: Predicted vs residuals")
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.show()
```

Out[75]: <matplotlib.collections.PathCollection at 0x159e49730>

Out[75]: Text(0.5, 1.0, 'RFR: Predicted vs residuals')

Out[75]: Text(0.5, 0, 'Predicted')

Out[75]: Text(0, 0.5, 'Residuals')



In [76]:
```
# Predicting Test data with the model
y_test_pred = reg.predict(x_test)
# Model Evaluation
acc_rf = metrics.r2_score(y_test, y_test_pred)
print('R^2:', acc_rf)
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_test, y_test_pred))*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_test, y_test_pred))
print('MSE:',metrics.mean_squared_error(y_test, y_test_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
R^2: 0.24145783093733464
Adjusted R^2: -3.551253014375992
MAE: 108133.37071428572
MSE: 14985270008.903975
RMSE: 122414.33743195269
```

In [77]:
```
# Import XGBoost Regressor
from xgboost import XGBRegressor

#Create a XGBoost Regressor
xb = XGBRegressor()

# Train the model using the training sets
xb.fit(x_train, y_train)
```

Out[77]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
             gamma=0, gpu_id=-1, importance_type=None,
             interaction_constraints='', learning_rate=0.300000012,
             max_delta_step=0, max_depth=6, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=100, n_jobs=8,
             num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
             reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
             validate_parameters=1, verbosity=None)
```

In [78]:
```
# Model prediction on train data
y_pred = xb.predict(x_train)

# Model Evaluation
print('R^2:',metrics.r2_score(y_train, y_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(len(y_train)-x_train.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',metrics.mean_squared_error(y_train, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
R^2: 0.9999999999998415
Adjusted R^2: 0.9999999999997424
MAE: 0.030133928571428572
MSE: 0.0013776506696428572
RMSE: 0.0371167168489194
```

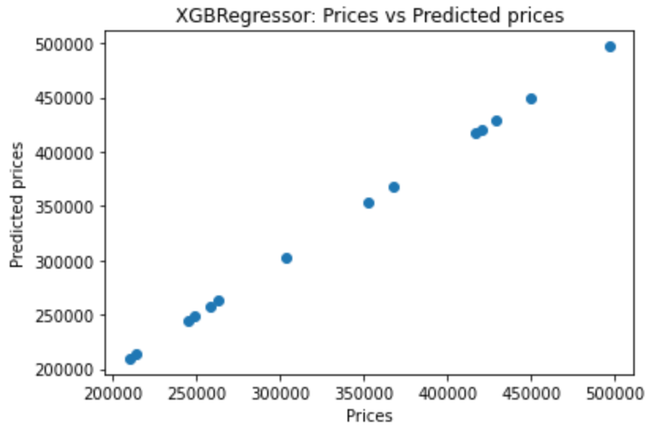
In [79]:
```
# Visualizing the differences between actual prices and predicted values
plt.scatter(y_train, y_pred)
plt.xlabel("Prices")
plt.ylabel("Predicted prices")
plt.title("XGBRegressor: Prices vs Predicted prices")
plt.show()
```

Out[79]: <matplotlib.collections.PathCollection at 0x15a087fa0>

Out[79]: Text(0.5, 0, 'Prices')

Out[79]: Text(0, 0.5, 'Predicted prices')

Out[79]: Text(0.5, 1.0, 'XGBRegressor: Prices vs Predicted prices')



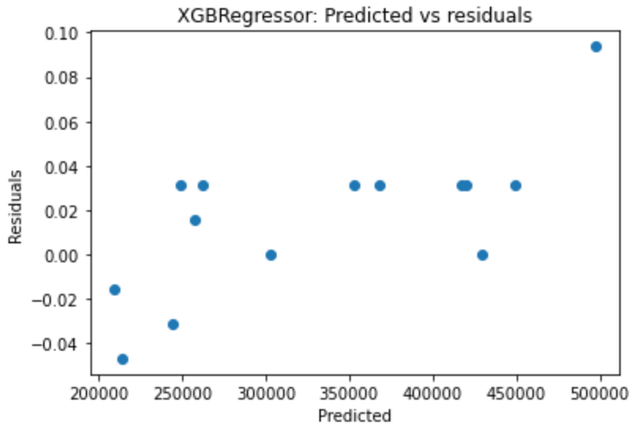
In [80]:
```
# Checking residuals
plt.scatter(y_pred,y_train-y_pred)
plt.title("XGBRegressor: Predicted vs residuals")
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.show()
```

Out[80]: <matplotlib.collections.PathCollection at 0x15a1dcfd0>

Out[80]: Text(0.5, 1.0, 'XGBRegressor: Predicted vs residuals')

Out[80]: Text(0.5, 0, 'Predicted')

Out[80]: Text(0, 0.5, 'Residuals')



In [81]:
```
#Predicting Test data with the model
y_test_pred = xb.predict(x_test)
# Model Evaluation
acc_xgb = metrics.r2_score(y_test, y_test_pred)
print('R^2:', acc_xgb)
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_test, y_test_pred))*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_test, y_test_pred))
print('MSE:',metrics.mean_squared_error(y_test, y_test_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
R^2: 0.24040408405305824
Adjusted R^2: -3.5575754956816503
MAE: 108738.52901785714
MSE: 15006087152.928328
RMSE: 122499.33531627152
```

In [82]:
```
# Creating scaled set to be used in model to improve our results
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
X_test = sc.transform(x_test)
```

In [83]:
```
# Import SVM Regressor
from sklearn import svm

# Create a SVM Regressor
sc_reg = svm.SVR()
# Train the model using the training sets
sc_reg.fit(x_train, y_train)
# Model prediction on train data
y_pred = sc_reg.predict(x_train)
```

Out[83]:
```
SVR()
```

In [84]:
```
# Model Evaluation
print('R^2:',metrics.r2_score(y_train, y_pred))
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_train, y_pred))*(len(y_train)-1)/(len(y_train)-x_train.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_train, y_pred))
print('MSE:',metrics.mean_squared_error(y_train, y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_train, y_pred)))
```

```
R^2: -0.0040551777336947925
Adjusted R^2: -0.631589663817254
MAE: 85175.35370112685
MSE: 8725206803.073704
RMSE: 93408.8154462613
```

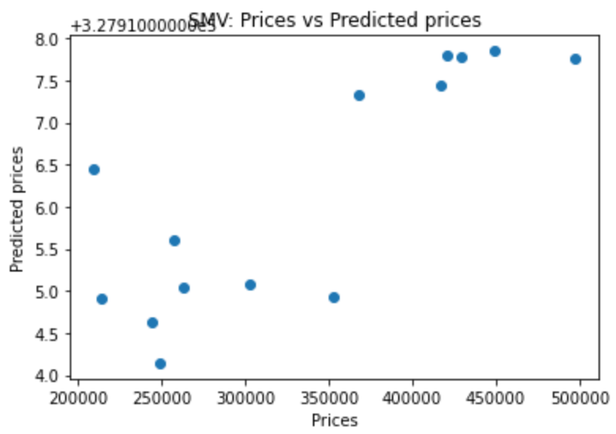
In [85]:
```
# Visualizing the differences between actual prices and predicted values
plt.scatter(y_train, y_pred)
plt.xlabel("Prices")
plt.ylabel("Predicted prices")
plt.title("SMV: Prices vs Predicted prices")
plt.show()
```

Out[85]:
```
<matplotlib.collections.PathCollection at 0x15a82eb50>
```

Out[85]:
```
Text(0.5, 0, 'Prices')
```

Out[85]:
```
Text(0, 0.5, 'Predicted prices')
```

Out[85]:
```
Text(0.5, 1.0, 'SMV: Prices vs Predicted prices')
```



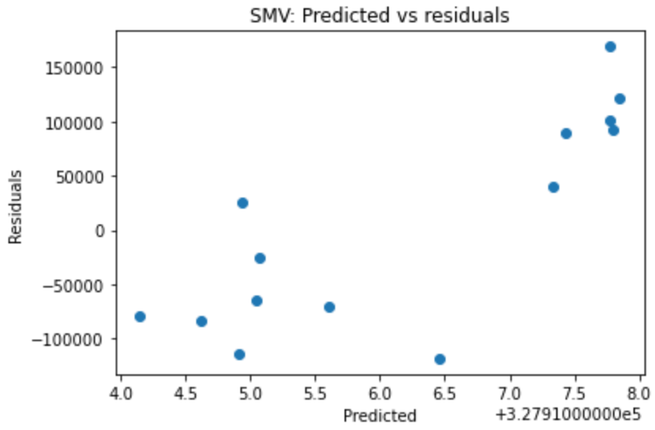
In [86]:
```
# Checking residuals
plt.scatter(y_pred,y_train-y_pred)
plt.title("SMV: Predicted vs residuals")
plt.xlabel("Predicted")
plt.ylabel("Residuals")
plt.show()
```

Out[86]: <matplotlib.collections.PathCollection at 0x15a272b20>

Out[86]: Text(0.5, 1.0, 'SMV: Predicted vs residuals')

Out[86]: Text(0.5, 0, 'Predicted')

Out[86]: Text(0, 0.5, 'Residuals')



In [87]:
```
# Predicting Test data with the model
y_test_pred = sc_reg.predict(X_test)
# Model Evaluation
acc_svm = metrics.r2_score(y_test, y_test_pred)
print('R^2:', acc_svm)
print('Adjusted R^2:',1 - (1-metrics.r2_score(y_test, y_test_pred))*(len(y_test)-1)/(len(y_test)-x_test.shape[1]-1))
print('MAE:',metrics.mean_absolute_error(y_test, y_test_pred))
print('MSE:',metrics.mean_squared_error(y_test, y_test_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test, y_test_pred)))
```

```
R^2: -0.002915824214640761
Adjusted R^2: -5.017494945287845
MAE: 132949.3456299203
MSE: 19812958375.973003
RMSE: 140758.5108473836
```

In [88]:
```
models = pd.DataFrame({
    'Model': ['Linear Regression', 'Random Forest', 'XGBoost', 'Support Vector Machines'],
    'R-squared Score': [acc_linreg*100, acc_rf*100, acc_xgb*100, acc_svm*100]})
models.sort_values(by='R-squared Score', ascending=False)
```

Out[88]:

	Model	R-squared Score
0	Linear Regression	56.297656
1	Random Forest	24.145783
2	XGBoost	24.040408
3	Support Vector Machines	-0.291582

In [89]:
```
import joblib

joblib.dump(lm, '/Users/nilanjana.chatterjee/DS670_UI/lm_house_price.pkl')
```

Out[89]:
```
['/Users/nilanjana.chatterjee/DS670_UI/lm_house_price.pkl']
```

In [90]:
```
comb_df.to_csv('/Users/nilanjana.chatterjee/DS670_UI/comb_df.csv')
```

In [ ]:

In [ ]: