forecasting-checkpoint-checkpoint

May 24, 2024

1 Prashant Priyadarshi

1.1 Temperature Forecasting

1.2 Installing the package

[1]:

```
!pip install pmdarima
```

```
Requirement already satisfied: pmdarima in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (2.0.3)
Requirement already satisfied: joblib>=0.11 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pmdarima) (1.3.1)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in
c:\users\prashant priyadarshi\appdata\local\programs\python\python310\lib\site-
packages (from pmdarima) (3.0.0)
Requirement already satisfied: numpy>=1.21.2 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pmdarima) (1.24.2)
Requirement already satisfied: pandas>=0.19 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pmdarima) (2.0.0)
Requirement already satisfied: scikit-learn>=0.22 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pmdarima) (1.3.0)
Requirement already satisfied: scipy>=1.3.2 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pmdarima) (1.11.1)
Requirement already satisfied: statsmodels>=0.13.2 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pmdarima) (0.14.0)
Requirement already satisfied: urllib3 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pmdarima) (2.0.3)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pmdarima) (58.1.0)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pandas>=0.19->pmdarima) (2021.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
pandas>=0.19->pmdarima) (2023.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
scikit-learn>=0.22->pmdarima) (3.2.0)
Requirement already satisfied: patsy>=0.5.2 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
statsmodels>=0.13.2->pmdarima) (0.5.3)
Requirement already satisfied: packaging>=21.3 in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
statsmodels>=0.13.2->pmdarima) (23.0)
Requirement already satisfied: six in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
patsy>=0.5.2->statsmodels>=0.13.2->pmdarima) (1.16.0)
```

1.3 importing the libraries

[2]:

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

[3]:

```
df = pd.read_csv("MaunaLoaDailyTemps.csv",index_col='DATE',parse_dates=True)
df=df.dropna()
print('Shape of data',df.shape)
df.head()
```

```
Shape of data (1821, 5)
```

[3]:

```
            MinTemp  MaxTemp  AvgTemp  Sunrise  Sunset
DATE
2014-01-01     33.0     46.0     40.0      657    1756
2014-01-02     35.0     50.0     43.0      657    1756
2014-01-03     36.0     45.0     41.0      657    1757
2014-01-04     32.0     41.0     37.0      658    1757
2014-01-05     24.0     38.0     31.0      658    1758
```

[4]:

```
df.columns
```

[4]:

```
Index(['MinTemp', 'MaxTemp', 'AvgTemp', 'Sunrise', 'Sunset'], dtype='object')
```

[6]:

```
#tail in dataset
df.tail()
```

[6]:

```
            MinTemp  MaxTemp  AvgTemp  Sunrise  Sunset
DATE
2018-12-26     35.0     45.0     40.0      654    1752
2018-12-27     33.0     44.0     39.0      655    1752
2018-12-28     33.0     47.0     40.0      655    1753
2018-12-29     36.0     47.0     42.0      655    1753
2018-12-30     39.0     52.0     46.0      656    1754
```

[7]:

```
#getting the information of dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1821 entries, 2014-01-01 to 2018-12-30
Data columns (total 5 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   MinTemp  1821 non-null   float64
 1   MaxTemp  1821 non-null   float64
 2   AvgTemp  1821 non-null   float64
 3   Sunrise  1821 non-null   int64
 4   Sunset   1821 non-null   int64
dtypes: float64(3), int64(2)
memory usage: 85.4 KB
```

[8]:

```
# finding the mean,standard deviation,total count,minimum value,maximum value…
df.describe()
```

[8]:

```
           MinTemp      MaxTemp      AvgTemp      Sunrise       Sunset
count  1821.000000  1821.000000  1821.000000  1821.000000  1821.000000
mean     38.637013    54.515102    46.818781   607.108731  1823.003844
std       3.798284     5.013654     4.143192    40.815966    49.576486
min      22.000000    36.000000    31.000000   543.000000  1742.000000
25%      36.000000    52.000000    44.000000   557.000000  1802.000000
50%      39.000000    55.000000    47.000000   614.000000  1831.000000
75%      41.000000    58.000000    50.000000   640.000000  1851.000000
max      49.000000    67.000000    57.000000   700.000000  1905.000000
```

[9]:

```
# finding the nulls
print(df.isnull().sum())
```

```
MinTemp    0
MaxTemp    0
AvgTemp    0
Sunrise    0
Sunset     0
dtype: int64
```

[10]:

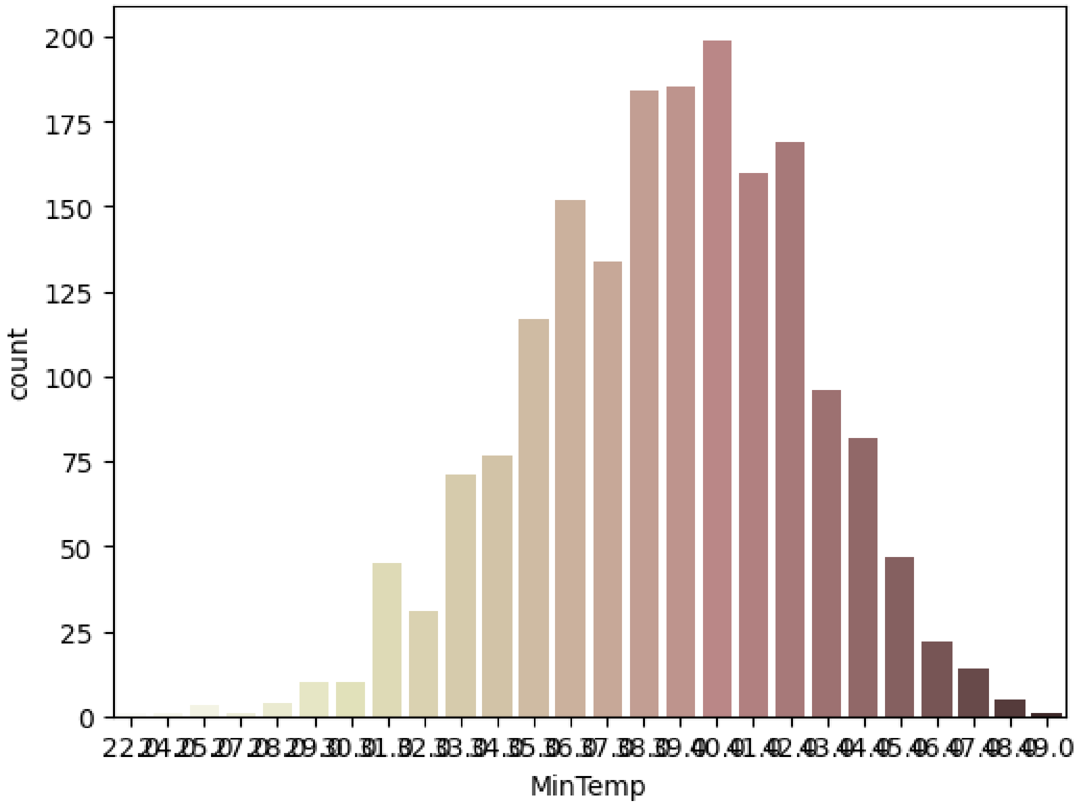
```
# importing the libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
print(df.value_counts())
```

```
MinTemp  MaxTemp  AvgTemp  Sunrise  Sunset
42.0     60.0     51.0     545      1904      3
36.0     54.0     45.0     548      1849      2
39.0     52.0     46.0     606      1840      2
38.0     53.0     46.0     622      1750      2
40.0     58.0     49.0     622      1750      2
                                             ..
37.0     55.0     46.0     608      1831      1
                           549      1849      1
                           545      1853      1
                           544      1902      1
49.0     64.0     57.0     658      1813      1
Name: count, Length: 1775, dtype: int64
```

1.4 Plotting the Data

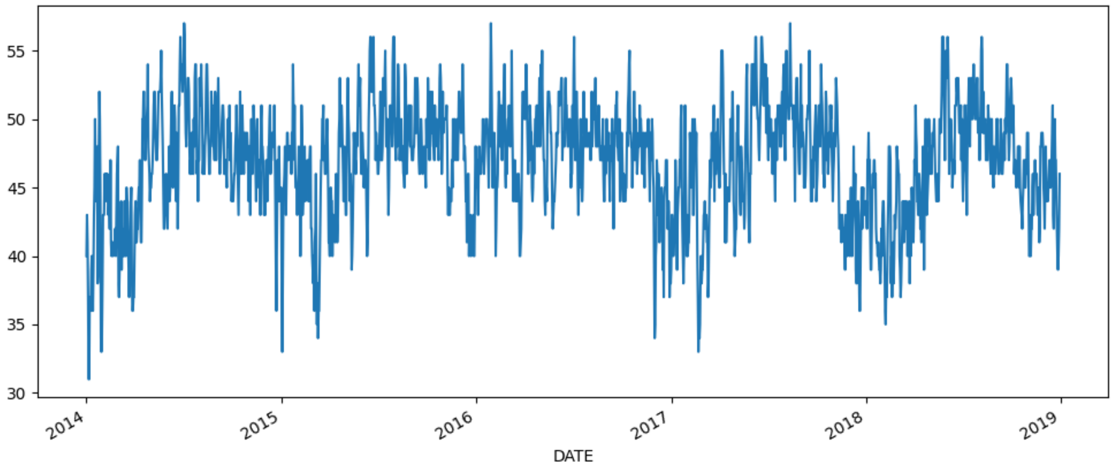
[11]:

```
sns.countplot(x="MinTemp", data=df, palette="pink_r")
plt.show()
```



```
[12]: df['AvgTemp'].plot(figsize=(12,5))
```

```
[12]: <Axes: xlabel='DATE'>
```



[13]:

```
# Checking for Stationarity
from statsmodels.tsa.stattools import adfuller

def adf_test(dataset):
  dftest = adfuller(dataset, autolag = 'AIC')
  print("1. ADF : ",dftest[0])
  print("2. P-Value : ", dftest[1])
  print("3. Num Of Lags : ", dftest[2])
  print("4. Num Of Observations Used For ADF Regression and Critical Values␣
 ↪Calculation :", dftest[3])
  print("5. Critical Values :")
  for key, val in dftest[4].items():
      print("\t",key, ": ", val)
```

[14]:

```
adf_test(df['AvgTemp'])
```

```
1. ADF :  -6.554680125068776
2. P-Value :  8.675937480199750e-09
3. Num Of Lags :  12
4. Num Of Observations Used For ADF Regression and Critical Values Calculation :
1808
5. Critical Values :
         1% :  -3.433972018026501
         5% :  -2.8631399192826676
         10% :  -2.5676217442756872
```

1.4.1 Figuring out order of ARIMA model

[15]:

```
from pmdarima import auto_arima
# Ignore harmless warnings
import warnings
warnings.filterwarnings("ignore")
```

[16]:

```
stepwise_fit = auto_arima(df['AvgTemp'],
                          suppress_warnings=True)

stepwise_fit.summary()
```

[16]:

Dep. Variable:	y	**No. Observations:**	1821
Model:	SARIMAX(1, 0, 5)	**Log Likelihood**	-4139.805
Date:	Sat, 22 Jul 2023	**AIC**	8295.611
Time:	11:44:34	**BIC**	8339.668
Sample:	0 - 1821	**HQIC**	8311.864
Covariance Type:	opg		

	coef	std err	z	P> |z|	[0.025	0.975]
intercept	1.2929	0.378	3.421	0.001	0.552	2.034
ar.L1	0.9721	0.008	119.279	0.000	0.956	0.988
ma.L1	-0.1219	0.024	-5.162	0.000	-0.168	-0.076
ma.L2	-0.2192	0.024	-9.126	0.000	-0.266	-0.172
ma.L3	-0.2048	0.024	-8.622	0.000	-0.251	-0.158
ma.L4	-0.1354	0.023	-5.992	0.000	-0.180	-0.091
ma.L5	-0.0487	0.024	-2.014	0.044	-0.096	-0.001
sigma2	5.4442	0.169	32.265	0.000	5.113	5.775

Ljung-Box (L1) (Q):	0.01	**Jarque-Bera (JB):**	20.99
Prob(Q):	0.93	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.81	**Skew:**	-0.17
Prob(H) (two-sided):	0.01	**Kurtosis:**	3.39

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

[17]:

```
pip install train
```

```
Requirement already satisfied: train in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (0.0.5)
Requirement already satisfied: numpy in c:\users\prashant
priyadarshi\appdata\local\programs\python\python310\lib\site-packages (from
train) (1.24.2)
Note: you may need to restart the kernel to use updated packages.
```

[20]:

```
from statsmodels.tsa.arima_model import ARIMA
```

1.4.2 Splitting the dataset into Training and Testing

[21]:

```
print(df.shape)
train=df.iloc[:-30]
test=df.iloc[-30:]
print(train.shape,test.shape)
print(test.iloc[0],test.iloc[-1])
```

```
(1821, 5)
(1791, 5) (30, 5)
MinTemp      36.0
MaxTemp      52.0
AvgTemp      44.0
Sunrise     640.0
Sunset     1743.0
Name: 2018-12-01 00:00:00, dtype: float64 MinTemp      39.0
MaxTemp      52.0
AvgTemp      46.0
Sunrise     656.0
```

```
Sunset      1754.0
Name: 2018-12-30 00:00:00, dtype: float64
```

1.4.3 Trainig the Model

[23]:

```
from statsmodels.tsa.arima.model import ARIMA
model=ARIMA(train['AvgTemp'],order=(1,0,5))
model=model.fit()
model.summary()
```

[23]:

Dep. Variable:	AvgTemp	**No. Observations:**	1791
Model:	ARIMA(1, 0, 5)	**Log Likelihood**	-4070.198
Date:	Sat, 22 Jul 2023	**AIC**	8156.395
Time:	11:54:26	**BIC**	8200.320
Sample:	0 - 1791	**HQIC**	8172.614
Covariance Type:	opg		

	coef	**std err**	**z**	**P> |z|**	**[0.025**	**0.975]**
const	46.5856	0.758	61.454	0.000	45.100	48.071
ar.L1	0.9856	0.005	188.230	0.000	0.975	0.996
ma.L1	-0.1412	0.023	-6.124	0.000	-0.186	-0.096
ma.L2	-0.2268	0.024	-9.635	0.000	-0.273	-0.181
ma.L3	-0.2168	0.023	-9.251	0.000	-0.263	-0.171
ma.L4	-0.1479	0.023	-6.491	0.000	-0.193	-0.103
ma.L5	-0.0595	0.024	-2.438	0.015	-0.107	-0.012
sigma2	5.5093	0.174	31.624	0.000	5.168	5.851

Ljung-Box (L1) (Q):	0.00	**Jarque-Bera (JB):**	14.88
Prob(Q):	0.97	**Prob(JB):**	0.00
Heteroskedasticity (H):	0.82	**Skew:**	-0.15
Prob(H) (two-sided):	0.01	**Kurtosis:**	3.33

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

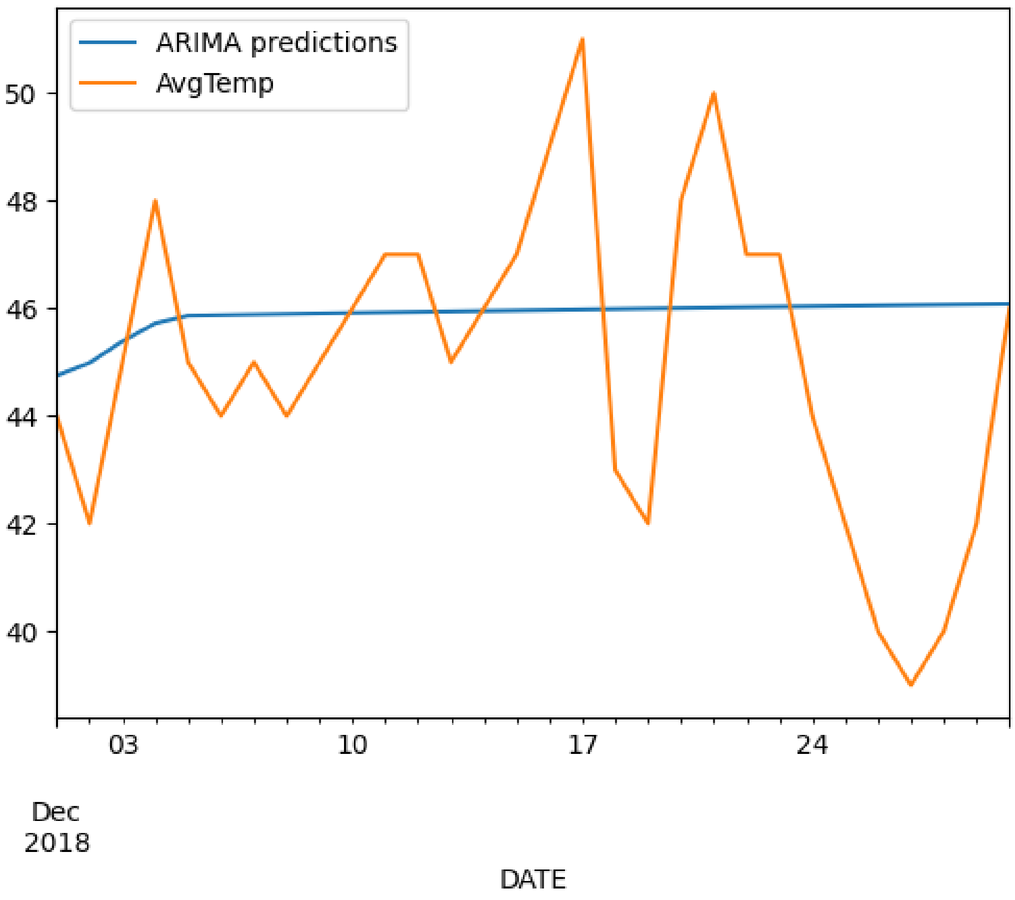
1.4.4 Making Prediction on Test Set

[25]:

```
start=len(train)
end=len(train)+len(test)-1
#if the predicted values dont have date values as index, you will have to␣
  ↪uncomment the following two commented lines to plot a graph
index_future_dates=pd.date_range(start='2018-12-01',end='2018-12-30')
ions')pred=model.predict(start=start,end=end,typ='levels').rename('ARIMA predict
pred.index=index_future_dates
pred.plot(legend=True)
test['AvgTemp'].plot(legend=True)
```

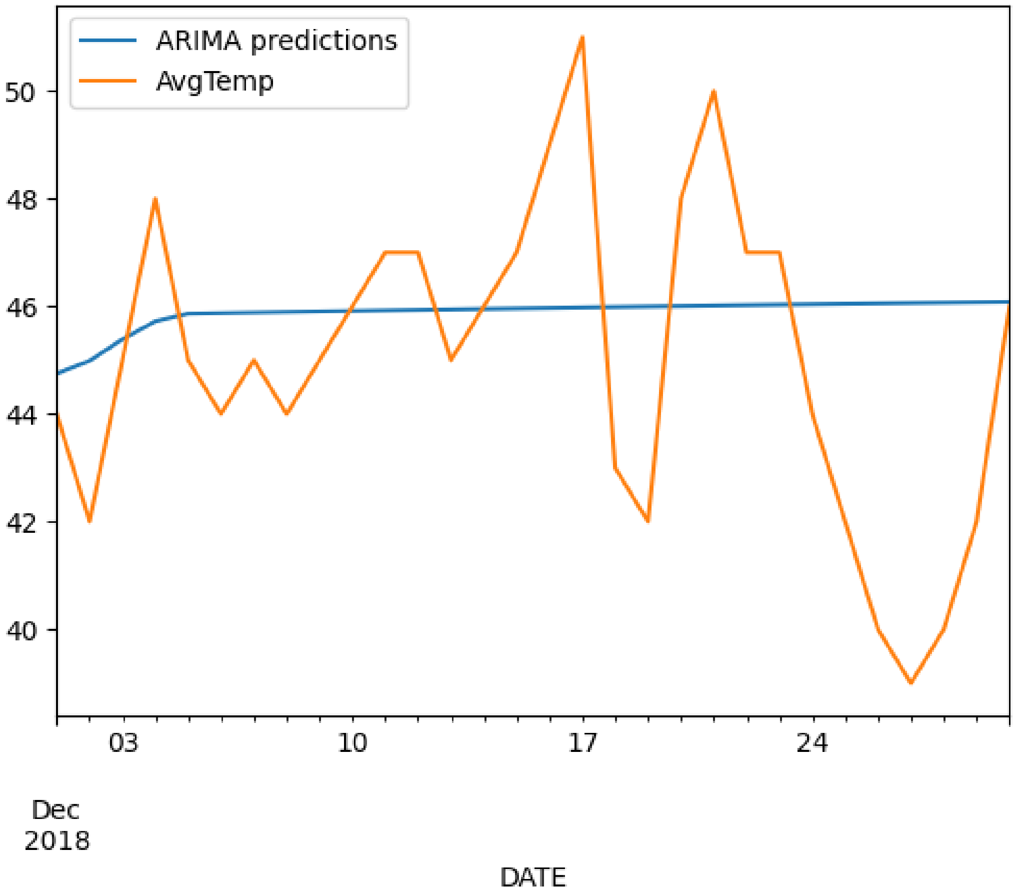
[25]:

```
<Axes: xlabel='DATE'>
```



```
[26]: pred.plot(legend='ARIMA Predictions')
      test['AvgTemp'].plot(legend=True)
```

```
[26]: <Axes: xlabel='DATE'>
```



```
[27]: test['AvgTemp'].mean()
```

```
[27]: 45.0
```

```
[28]: from sklearn.metrics import mean_squared_error
      from math import sqrt
      rmse=sqrt(mean_squared_error(pred,test['AvgTemp']))
      print(rmse)
```

```
3.000463708767501
```

```
[29]: model2=ARIMA(df['AvgTemp'],order=(1,0,5))
      model2=model2.fit()
      df.tail()
```

```
[29]:             MinTemp  MaxTemp  AvgTemp  Sunrise  Sunset
      DATE
      2018-12-26     35.0     45.0     40.0      654    1752
      2018-12-27     33.0     44.0     39.0      655    1752
```

```
 2018-12-28      33.0      47.0      40.0       655     1753
 2018-12-29      36.0      47.0      42.0       655     1753
 2018-12-30      39.0      52.0      46.0       656     1754
```

1.4.5 For Future Dates

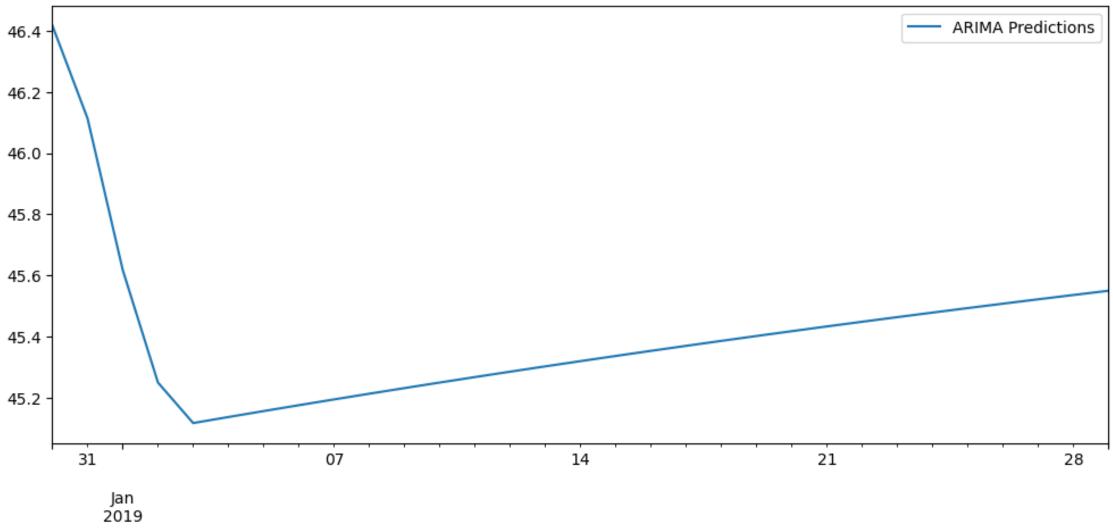
[30]:

```
index_future_dates=pd.date_range(start='2018-12-30',end='2019-01-29')
#print(index_future_dates)
pred=model2.predict(start=len(df),end=len(df)+30,typ='levels').rename('ARIMA␣
 ↪Predictions')
#print(comp_pred)
pred.index=index_future_dates
print(pred)
```

```
2018-12-30    46.418166
2018-12-31    46.113912
2019-01-01    45.617874
2019-01-02    45.249566
2019-01-03    45.116915
2019-01-04    45.136665
2019-01-05    45.156139
2019-01-06    45.175341
2019-01-07    45.194274
2019-01-08    45.212941
2019-01-09    45.231348
2019-01-10    45.249497
2019-01-11    45.267392
2019-01-12    45.285037
2019-01-13    45.302435
2019-01-14    45.319590
2019-01-15    45.336504
2019-01-16    45.353182
2019-01-17    45.369626
2019-01-18    45.385841
2019-01-19    45.401828
2019-01-20    45.417592
2019-01-21    45.433135
2019-01-22    45.448461
2019-01-23    45.463572
2019-01-24    45.478472
2019-01-25    45.493163
2019-01-26    45.507649
2019-01-27    45.521932
2019-01-28    45.536015
2019-01-29    45.549902
Freq: D, Name: ARIMA Predictions, dtype: float64
```

[31]:
```
pred.plot(figsize=(12,5),legend=True)
```

[31]:
```
<Axes: >
```



[ ]:
```
```