AI WORKFLOW IN PRODUCTION : Capstone Projects

Author : Pelani Malange Date: 09 February 2020

Part 1: Assimilate the business scenario and articulate testable hypotheses.

Business question to be answered

Can business managers predict the revenue for following month based on previous , frequency and value purchased . For instance , given a start date for prediction and country , can business get forecast for the following month.

Data Ingestion

Find ideal data to address the business opportunity and provide clarification on the rationale for needing specific data.

Requirement for Prediction:

1. Need to know country , patterns of buying products in the country (last purchased item, quantity, date, cost)
2. Follow business instructions train on 10 countries with most revenue
3. Find a feature for revenue inorder to predict sales from previos quantity sold and unit price

Create a python script to extract relevant data from multiple data sources, automating the process of data ingestion.

1. The script will fetch json based files and consolidate into one panda dataframe
2. The script will filter data based on top 10 countries that produced revenue
3. The data will be saved in /data folder for exploration and analysis

Then script has two main outputs

1. "!python3 ./runtime/cs_data_ingestor.py -c train" to load base training data , saved as top10countries.csv
2. "!python3 ./runtime/cs_data_ingestor.py -c update" to load updated data , saved as updtop10countries.csv

In [2]:

```
%%writefile ./runtime/cs_data_ingestor.py
# The line above create a file called "cs_data_ingestor.py" in the  runtime  worki
ng
# directory and write the the reste of the cell in this file.

import json
import os
import sys
import getopt
import re
import shutil
import time
import pickle
from collections import defaultdict
from datetime import datetime
import numpy as np
import pandas as pd
from IPython.display import Image
import matplotlib.pyplot as plt
## For plotting
import matplotlib.pyplot as plt## For outliers detection

data_dir = ""
DATA_DIR1 = os.path.join(".","data","cs-train")
DATA_DIR2 = os.path.join(".","data")
DATA_DIRP = os.path.join(".","data","cs-production")
IMAGE_DIR = os.path.join(".","images")


plt.style.use('seaborn')

SMALL_SIZE = 12
MEDIUM_SIZE = 14
LARGE_SIZE = 16

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
```

```
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=LARGE_SIZE)   # fontsize of the figure title

#code adopted from solution guidance
def fetch_data(data_dir):
    """
    load all json formatted files into a dataframe
    """

    ## input testing
    if not os.path.isdir(data_dir):
        raise Exception("specified data dir does not exist")
    if not len(os.listdir(data_dir)) > 0:
        raise Exception("specified data dir does not contain any files")

    file_list = [os.path.join(data_dir,f) for f in os.listdir(data_dir) if re.sear
ch("\.json",f)]
    correct_columns = ['country', 'customer_id', 'day', 'invoice', 'month',
                       'price', 'stream_id', 'times_viewed', 'year']

    ## read data into a temp structure
    all_months = {}
    for file_name in file_list:
        df = pd.read_json(file_name)
        all_months[os.path.split(file_name)[-1]] = df

    ## ensure the data are formatted with correct columns
    for f,df in all_months.items():
        cols = set(df.columns.tolist())
        if 'StreamID' in cols:
            df.rename(columns={'StreamID':'stream_id'},inplace=True)
        if 'TimesViewed' in cols:
            df.rename(columns={'TimesViewed':'times_viewed'},inplace=True)
        if 'total_price' in cols:
            df.rename(columns={'total_price':'price'},inplace=True)
```

```
        cols = df.columns.tolist()
        if sorted(cols) != correct_columns:
            raise Exception("columns name could not be matched to correct cols")

    ## concat all of the data
    df = pd.concat(list(all_months.values()),sort=True)
    years,months,days = df['year'].values,df['month'].values,df['day'].values
    dates = ["{}-{}-{}".format(years[i],str(months[i]).zfill(2),str(days[i]).zfill
(2)) for i in range(df.shape[0])]
    df['invoice_date'] = np.array(dates,dtype='datetime64[D]')
    df['invoice'] = [re.sub("\D+","",i) for i in df['invoice'].values]
    df["revenue"] = df["times_viewed"]*df["price"]

    ## sort by date and reset the index
    df.sort_values(by='invoice_date',inplace=True)
    df.reset_index(drop=True,inplace=True)

    #export file


    return(df)


def filter_data(df):
    ## find the top ten countries (wrt revenue)

    print("\n Imported data  with the following attrbutes \n")
    df.info()
    columns_to_show = ["revenue"]
    df_agg= df.groupby(['country'])[columns_to_show].sum().round(3).sort_values([
'revenue'],ascending=False)
    #df.sort_values([''])
    #top10 = df_agg.sort_values(['price'],ascending=False).groupby('country').head
(10)
    print("\nTop 10 countries to use \n{}".format("-"*15))

    top10 = df_agg.head(n=10)
```

```
    print(top10.index.unique())
    print("Filtering data based on top 10 countries to train model")
    df_top10 = df[df.country.isin(top10.index.unique())]
    print(df_top10.head(n=10))
    return df_top10



def update_target(target_file,df_clean, overwrite=False):
    """
    update line by line in case data are large
    """

    if overwrite or not os.path.exists(target_file):
        df_clean.to_csv(target_file, index=False)
    else:
        df_clean.to_csv(target_file, mode='a', header=False, index=False)


def create_plot(df):
    fig = plt.figure(figsize=(14,6))
    ax1 = fig.add_subplot(121)
    ax2 = fig.add_subplot(122)

    table1 = pd.pivot_table(df, index='country', columns='year', values='price',ag
gfunc='mean').round(3)
    table1.plot(kind='bar', ax=ax1)
    ax1.set_ylabel("Average price");

    table2 = pd.pivot_table(df, index='country', columns='year', values="revenue",
aggfunc='sum').round(3)
    table2.plot(kind='bar', ax=ax2)
    ax2.set_ylabel("Total revenue viewership");

    ## adjust the axis to accomadate the legend
    ax1.set_ylim((0,9.3))
    ax2.set_ylim((0,1.3))
    image_path = os.path.join(IMAGE_DIR,"revenue.png")
```

```
    plt.savefig(image_path,bbox_inches='tight',pad_inches = 0,dpi=200)
    print("{} created.".format(image_path))



if __name__ == "__main__":



    ## collect args
    arg_string = "%s -c update "%sys.argv[0]
    try:
        optlist, args = getopt.getopt(sys.argv[1:],'c:')
    except getopt.GetoptError:
        print(getopt.GetoptError)
        raise Exception(arg_string)

    ## handle args
    #streams_file = None

    #db_file = None
    mode_exec = None
    for o, a in optlist:
        if o == '-c':
            mode_exec = a

    if mode_exec == "train":
        data_dir  = DATA_DIR1
    else:
        data_dir = DATA_DIRP


    df_raw =fetch_data(data_dir)
    print("\n Data information after import\n{}".format("-"*15))
    print(df_raw.info())
    print(df_raw.describe())
```

```
    print("\n Number of of days \n {}",df_raw["invoice_date"].max() - df_raw["invo
ice_date"].min())
    print("\ndf_raw before cleaning \n{}".format("-"*15))
    print(df_raw.isnull().sum(axis = 0))

    print("\n Data after cleaning")
    columns = ['country', 'day','month', 'price',  'times_viewed','year','invoice_
date','revenue']
    df_analysis= df_raw[columns]
    print("\n df_analysis \n")
    create_plot(df_analysis)
    print ("\nFinal data for analysis\n{}".format("-"*15))
    print (df_analysis.info())

    print("Data with 10 countries for analysis \n{}",df_analysis.head(n=10))

    #update_target(os.path.join(data_dir2,'customer-data.csv'),df_analysis,overwri
te=True)
    #print('\nCreated  file customer-data.csv')



    #filter training data based on top 10 countries

    df10 = filter_data(df_analysis)
    #save the data for  later use in  model training and testing
    if mode_exec =="train":
        update_target(os.path.join(DATA_DIR2,'top10countries-data.csv'),df10,overw
rite=True)
        print('\n Created  top10country-data.csv  for training in ./data folder \n
' )
    else:
        update_target(os.path.join(DATA_DIR2,'updtop10countries-data.csv'),df10,ov
erwrite=True)
        print('\n  Created  uptop10countries-data.csv to simulate production data
 in ./data folder \n')
```

In [3]:
```
!python3 ./runtime/cs_data_ingestor.py -c train
```

```
 Data information after import
---------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 815011 entries, 0 to 815010
Data columns (total 11 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   country       815011 non-null  object
 1   customer_id   625249 non-null  float64
 2   day           815011 non-null  int64
 3   invoice       815011 non-null  object
 4   month         815011 non-null  int64
 5   price         815011 non-null  float64
 6   stream_id     815011 non-null  object
 7   times_viewed  815011 non-null  int64
 8   year          815011 non-null  int64
 9   invoice_date  815011 non-null  datetime64[ns]
 10  revenue       815011 non-null  float64
dtypes: datetime64[ns](1), float64(3), int64(4), object(3)
memory usage: 68.4+ MB
None
         customer_id            day  ...           year        revenue
count  625249.000000  815011.000000  ...  815011.000000  815011.000000
mean    15333.415068      15.064819  ...    2018.247654      14.291172
std      1698.360788       8.788845  ...       0.545261     197.463439
min     12346.000000       1.000000  ...    2017.000000  -53594.360000
25%     13956.000000       7.000000  ...    2018.000000       3.360000
50%     15279.000000      15.000000  ...    2018.000000       8.290000
75%     16813.000000      23.000000  ...    2019.000000      16.130000
max     18287.000000      31.000000  ...    2019.000000   50222.180000

[8 rows x 7 columns]

 Number of of days
{} 610 days 00:00:00
```

```
{} 610 days 00:00:00

df_raw before cleaning
---------------
country              0
customer_id     189762
day                  0
invoice              0
month                0
price                0
stream_id            0
times_viewed         0
year                 0
invoice_date         0
revenue              0
dtype: int64

 Data after cleaning

 df_analysis

./images/revenue.png created.

Final data for analysis
---------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 815011 entries, 0 to 815010
Data columns (total 8 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   country       815011 non-null  object
 1   day           815011 non-null  int64
 2   month         815011 non-null  int64
 3   price         815011 non-null  float64
 4   times_viewed  815011 non-null  int64
 5   year          815011 non-null  int64
 6   invoice_date  815011 non-null  datetime64[ns]
 7   revenue       815011 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(4), object(1)
```

```
memory usage: 49.7+ MB
None
Data with 10 countries for analysis
{}           country  day  month  price  times_viewed  year invoice_date  reve
nue
0  United Kingdom   28     11   5.95             1  2017   2017-11-28     5.95
1  United Kingdom   28     11   6.75            12  2017   2017-11-28    81.00
2  United Kingdom   28     11   2.10            21  2017   2017-11-28    44.10
3  United Kingdom   28     11   1.25             5  2017   2017-11-28     6.25
4  United Kingdom   28     11   1.65            17  2017   2017-11-28    28.05
5  United Kingdom   28     11   1.25            14  2017   2017-11-28    17.50
6  United Kingdom   28     11   5.95            10  2017   2017-11-28    59.50
7  United Kingdom   28     11   2.55            12  2017   2017-11-28    30.60
8  United Kingdom   28     11   3.75            12  2017   2017-11-28    45.00
9  United Kingdom   28     11   1.65            18  2017   2017-11-28    29.70

 Imported data  with the following attrbutes

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 815011 entries, 0 to 815010
Data columns (total 8 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   country       815011 non-null  object
 1   day           815011 non-null  int64
 2   month         815011 non-null  int64
 3   price         815011 non-null  float64
 4   times_viewed  815011 non-null  int64
 5   year          815011 non-null  int64
 6   invoice_date  815011 non-null  datetime64[ns]
 7   revenue       815011 non-null  float64
dtypes: datetime64[ns](1), float64(2), int64(4), object(1)
memory usage: 49.7+ MB

Top 10 countries to use
---------------
Index(['United Kingdom', 'EIRE', 'Germany', 'France', 'Norway', 'Netherlands',
       'Spain', 'Switzerland', 'Portugal', 'Belgium'],
```

```
      dtype='object', name='country')
Filtering data based on top 10 countries to train model
```

	country	day	month	price	times_viewed	year	invoice_date	revenue
0	United Kingdom	28	11	5.95	1	2017	2017-11-28	5.95
1	United Kingdom	28	11	6.75	12	2017	2017-11-28	81.00
2	United Kingdom	28	11	2.10	21	2017	2017-11-28	44.10
3	United Kingdom	28	11	1.25	5	2017	2017-11-28	6.25
4	United Kingdom	28	11	1.65	17	2017	2017-11-28	28.05
5	United Kingdom	28	11	1.25	14	2017	2017-11-28	17.50
6	United Kingdom	28	11	5.95	10	2017	2017-11-28	59.50
7	United Kingdom	28	11	2.55	12	2017	2017-11-28	30.60
8	United Kingdom	28	11	3.75	12	2017	2017-11-28	45.00
9	United Kingdom	28	11	1.65	18	2017	2017-11-28	29.70

```
 Created  top10country-data.csv  for training in ./data folder

./runtime/cs_data_ingestor.py:217: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df10["invoice_date"] = df10["invoice_date"].astype(str)
```

Investigate the relationship between the relevant data, the target and the business metric.

1. Trend Analysis

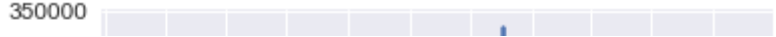
The intention is to visualiza how the data behaves over a time period . It is from this judgement that we can decide additional tests to try to explore the data.

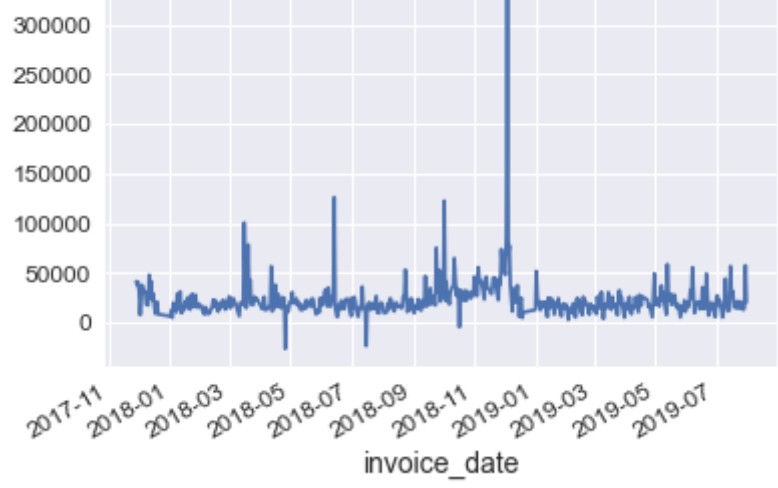
In [4]:

```
#%% create a new time series trend analysis of the data

#%%writefile timeseries_eda.py
import os
import re
import numpy as np
import pandas as pd
from IPython.display import Image
import matplotlib.pyplot as plt
## For plotting
import matplotlib.pyplot as plt## For outliers detection
from sklearn import preprocessing, svm## For stationarity test and decomposition
import statsmodels.tsa.api as smt
import statsmodels.api as sm

plt.style.use('seaborn')
%matplotlib inline

SMALL_SIZE = 12
MEDIUM_SIZE = 14
LARGE_SIZE = 16

## specify the directory you saved the data and images in
DATA_DIR = os.path.join(".","data")
IMAGE_DIR = os.path.join(".","images")

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=LARGE_SIZE)   # fontsize of the figure title

def save_plot(image_name):
    image_path = os.path.join(IMAGE_DIR,image_name)
```

```
    plt.savefig(image_path,bbox_inches='tight',pad_inches = 0,dpi=200)
    print("{} created.".format(image_path))

def fetch_data():
    df = pd.read_csv(os.path.join(DATA_DIR, "top10countries-data.csv"),index_col=0
, parse_dates=['invoice_date'])
    print("df: {} x {}".format(df.shape[0], df.shape[1]))
    ## check the first few rows
    print("\n  Check first 4 rows\n")
    print(df.head(n=4))

    return df

df = fetch_data()
## format datetime column
df["invoice_date"] = pd.to_datetime(df['invoice_date'], format='%d.%m.%Y')## creat
e time series
ts = df.groupby("invoice_date")["revenue"].sum().rename("sales")
ts.head()
#ts.tail()
ts.plot()
plt.figure(figsize=(15,5))
save_plot("raw_ts.png")
```

```
df: 801773 x 7

  Check first 4 rows

                day  month  price  times_viewed  year invoice_date  revenue
country
United Kingdom   28     11   5.95             1  2017   2017-11-28     5.95
United Kingdom   28     11   6.75            12  2017   2017-11-28    81.00
United Kingdom   28     11   2.10            21  2017   2017-11-28    44.10
United Kingdom   28     11   1.25             5  2017   2017-11-28     6.25
./images/raw_ts.png created.
```

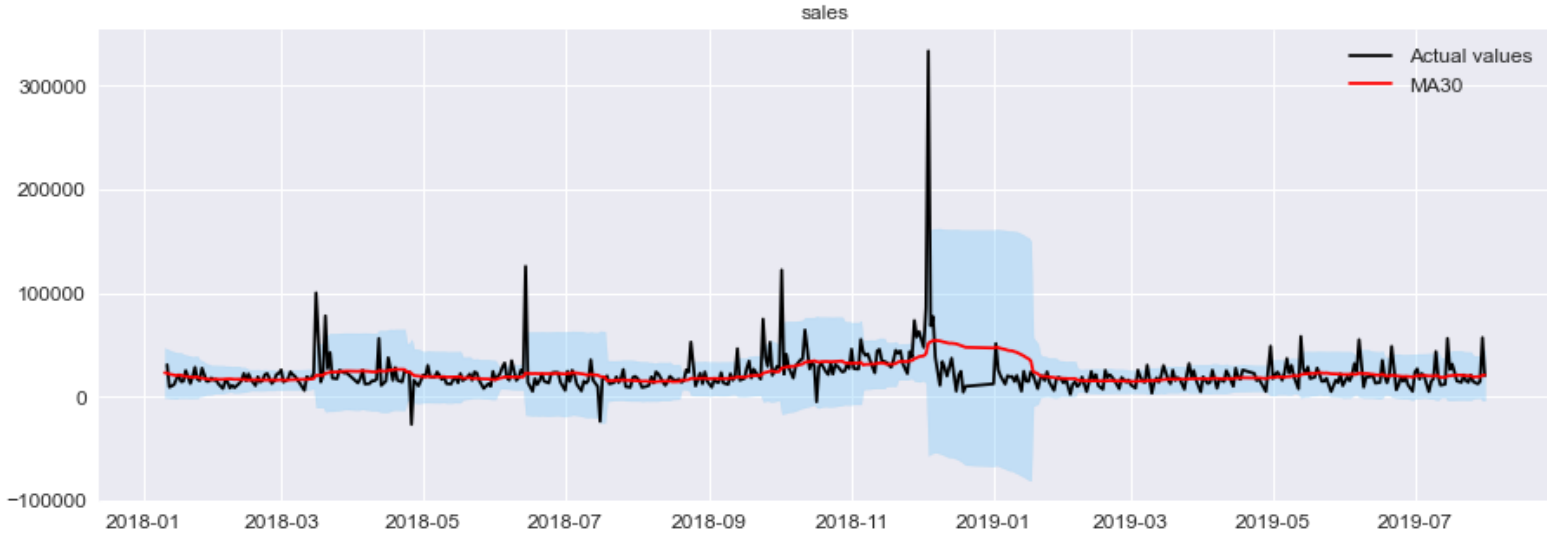





```
<Figure size 1080x360 with 0 Axes>
```

1. The spike in 2018 March, May, June, July , Oct, Dec seems to be an outlier that can affect the machine learnig algorithm, we need to identify algorithm that responds well to outliers .
2. More tests need to be done to pick up any trends

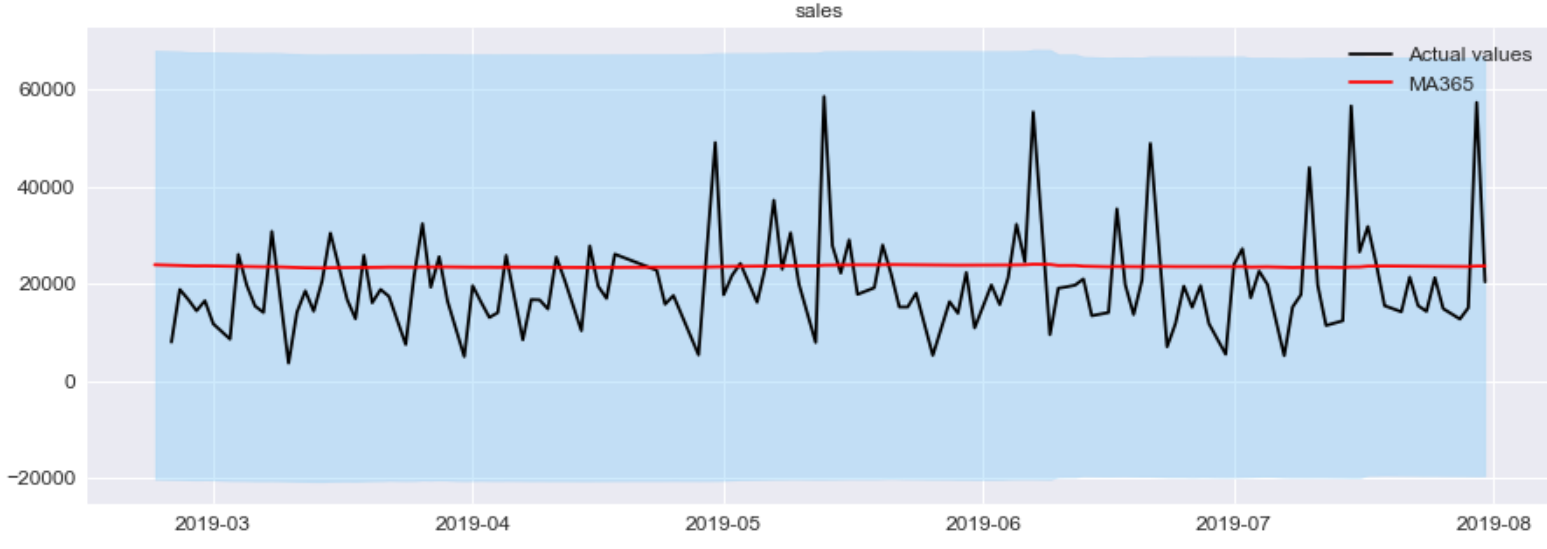
In [6]:
```
plot_ts(ts,window=30)
```




```
./images/rollingavg30days.png created.

<Figure size 432x288 with 0 Axes>
```

Plot of rolling window 30 days suggest theres are no trends in the data . Lets try to add the rolling windows to 365 days would suggest otherwise

In [35]:
```
plot_ts(ts,window=365)
save_plot("tsaverage365days.png")
```




```
./images/rollingavg30days.png created.
./images/tsaverage365days.png created.

<Figure size 432x288 with 0 Axes>
```

Rolling window across the year shows that the price remains fairly flat.

Part 2 Model Building and Selection

Tasks

1. State the different modeling approaches that you will compare to address the business opportunity.
2. Iterate on your suite of possible models by modifying data transformations, pipeline architectures, hyperparameters and other relevant factors.
3. Re-train your model on all of the data using the selected approach and prepare it for deployment. Articulate findings in a summary report.

Trend analysis Modeling approach to use

In prediction, there are several libraries such ARIMA , Prophet and other techniques that can be used . In our case, we shall explore forecasting depending on the exploratory phase and choose a model to use. In this case ARIMA python library will be explored as it has inbuilt models to compare against without need to import additional libraries .

1. We shall analyse the data for conformity to time-series
2. We shall test models and select we shall choose the best model to adopt going forward.

In [5]:

```
#%% create a new time series trend analysis of the data

#%%writefile timeseries_eda.py
import os
import re
import numpy as np
import pandas as pd
from IPython.display import Image
import matplotlib.pyplot as plt
## For plotting
import matplotlib.pyplot as plt## For outliers detection
from sklearn import preprocessing, svm## For stationarity test and decomposition
import statsmodels.tsa.api as smt
import statsmodels.api as sm

plt.style.use('seaborn')
%matplotlib inline

SMALL_SIZE = 12
MEDIUM_SIZE = 14
LARGE_SIZE = 16

## specify the directory you saved the data and images in
DATA_DIR = os.path.join(".","data")
IMAGE_DIR = os.path.join(".","images")

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=LARGE_SIZE)   # fontsize of the figure title

#function to save images
def save_plot(image_name):
    image_path = os.path.join(IMAGE_DIR,image_name)
```

```
    plt.savefig(image_path,bbox_inches='tight',pad_inches = 0,dpi=200)
    print("{} created.".format(image_path))

def fetch_data():
    df = pd.read_csv(os.path.join(DATA_DIR, "top10countries-data.csv"),index_col=0
, parse_dates=['invoice_date'])
    print("df size : {} x {}".format(df.shape[0], df.shape[1]))

    return df

## format datetime column
df = fetch_data()
df["invoice_date"] = pd.to_datetime(df['invoice_date'], format='%d.%m.%Y')## creat
e time series
ts = df.groupby("invoice_date")["revenue"].sum().rename("Sales")
#data does not have daily sales so for the model, it will be analysed based on  mo
nthly averages
y_ts = ts.resample('MS').mean()

#ts.tail()
```

```
df size : 801773 x 7
```

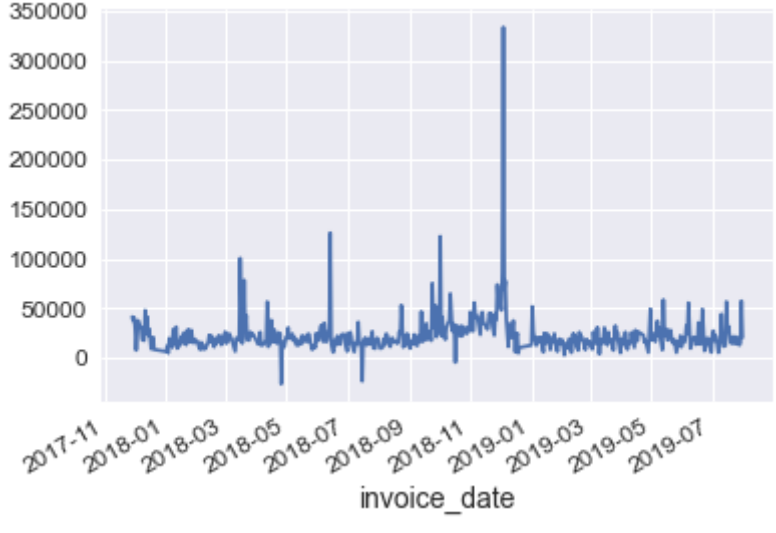
In [9]:

```
#%% create a new time series trend analysis of the data

#%%writefile timeseries_eda.py
import os
import re
import numpy as np
import pandas as pd
from IPython.display import Image
import matplotlib.pyplot as plt
## For plotting
import matplotlib.pyplot as plt## For outliers detection
from sklearn import preprocessing, svm## For stationarity test and decomposition
import statsmodels.tsa.api as smt
import statsmodels.api as sm

plt.style.use('seaborn')
%matplotlib inline

SMALL_SIZE = 12
MEDIUM_SIZE = 14
LARGE_SIZE = 16

## specify the directory you saved the data and images in
DATA_DIR = os.path.join(".","data")
IMAGE_DIR = os.path.join(".","images")

plt.rc('font', size=SMALL_SIZE)          # controls default text sizes
plt.rc('axes', titlesize=SMALL_SIZE)     # fontsize of the axes title
plt.rc('axes', labelsize=MEDIUM_SIZE)    # fontsize of the x and y labels
plt.rc('xtick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('ytick', labelsize=SMALL_SIZE)    # fontsize of the tick labels
plt.rc('legend', fontsize=SMALL_SIZE)    # legend fontsize
plt.rc('figure', titlesize=LARGE_SIZE)   # fontsize of the figure title

#function to save images
def save_plot(image_name):
    image_path = os.path.join(IMAGE_DIR,image_name)
```

In [6]:

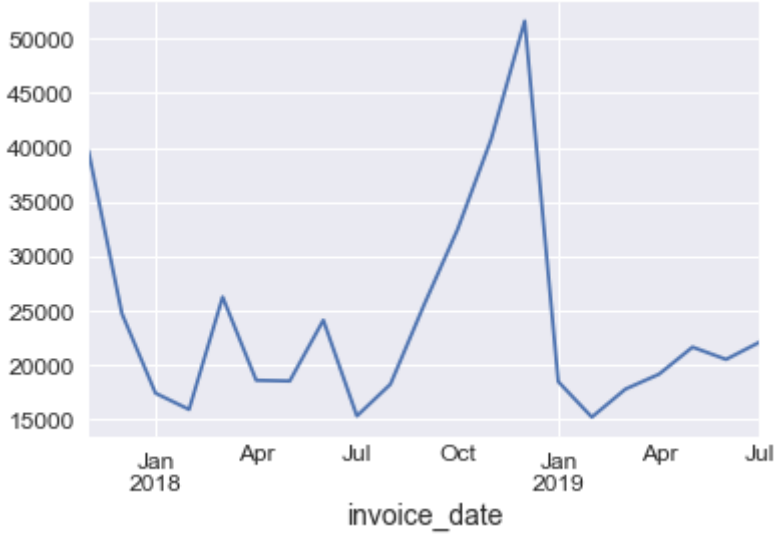
```
print("timeseries  with monthly average revenues".format( y_ts.shape[0]))
y_ts.head()


print("\n Show daily time series trend\n")
ts.plot()
plt.show()
save_plot("dailyts.png")

print("\n Show Monthly time series trend \n")
y_ts.plot()
plt.show()
save_plot("monthly.png")
```

```
timeseries  with monthly average revenues

 Show daily time series trend
```



```
./images/dailyts.png created.
```

```
 Show Monthly time series trend
```




```
./images/monthly.png created.
```

```
<Figure size 432x288 with 0 Axes>
```

Quarterly trend shows sales declining in Jan 2018 and picking up from mid Jul 2018.

As typical of time series data, the data points after simple feature reduction is rduced to 21points (21 months). This will require algorithm that is advanced to forecast based on the moving averages .

The spike in Dec 2018 seems to be an outlier that can affect the machine learnig algorithm, we need to identify algorithm that responds well to outliers . More tests need to be done to pick up any trends

Test for hyper parameters using auto arima in the times series

Auto_arima() uses a stepwise approach to search multiple combinations of p,d,q parameters and chooses the best MODEL that has the least AIC. AIC stands for Akaike Information Criterion, which estimates the relative amount of information lost by a given model. A model with lower AIC value wins! So lets get exploring

One of the requirements for ARIMA is that the time series should be stationary. A stationary series is one where the properties do not change over time. There are several methods to check the stationarity of a series. The one you'll use in this guide is the Augmented Dickey-Fuller test. Test for stationary before proceeding with tests.

In [10]:
```
from statsmodels.tsa.stattools import adfuller
print("p-value:", adfuller(y_ts)[1])
```

```
p-value: 0.029955476715040872
```

P-value is below 0.05 so it fits for ARIMA tests using average revenue feature , with auto-arima, the best hyperparameters will be predicted

Model 1: Exploration

We shall use the y_ts data to test for AR model

In [18]:
```
from statsmodels.tsa.arima_model import ARIMA
import pmdarima as pm


model = pm.auto_arima(y_ts, start_p=1, start_q=1,
                      test='adf',       # use adftest to find optimal 'd'
                      max_p=3, max_q=3, # maximum p and q
                      m=1,              # frequency of series
                      d=None,           # let model determine 'd'
                      seasonal=False,   # No Seasonality, not enough data to cross
24 months period
                      start_P=0,
                      D=0,
                      trace=True,
                      error_action='ignore',
                      suppress_warnings=True,
                      stepwise=True)
```

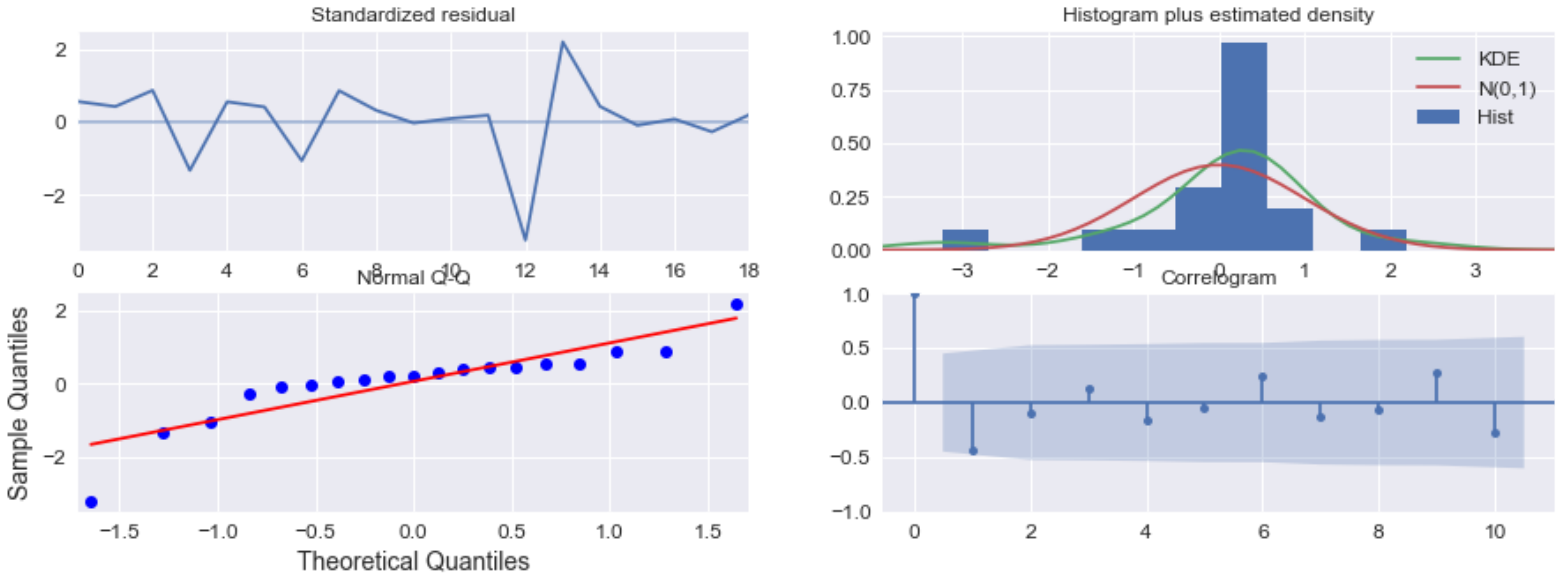
```
Performing stepwise search to minimize aic
 ARIMA(1,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.19 sec
 ARIMA(0,2,0)(0,0,0)[0] intercept   : AIC=421.525, Time=0.01 sec
 ARIMA(1,2,0)(0,0,0)[0] intercept   : AIC=420.264, Time=0.02 sec
 ARIMA(0,2,1)(0,0,0)[0] intercept   : AIC=inf, Time=0.09 sec
 ARIMA(0,2,0)(0,0,0)[0]             : AIC=419.694, Time=0.01 sec

Best model:  ARIMA(0,2,0)(0,0,0)[0]
Total fit time: 0.334 seconds
```

Model 1 Summary

In [12]:
```
print(model.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                   21
Model:               SARIMAX(0, 2, 0)   Log Likelihood                -208.847
Date:                Tue, 09 Feb 2021   AIC                            419.694
Time:                        10:51:51   BIC                            420.638
Sample:                             0   HQIC                           419.854
                                  - 21
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2      1.859e+08   3.48e+07      5.343      0.000    1.18e+08    2.54e+08
===================================================================================
Ljung-Box (L1) (Q):                   4.10   Jarque-Bera (JB):
13.77
Prob(Q):                              0.04   Prob(JB):
0.00
Heteroskedasticity (H):               1.47   Skew:
-1.31
Prob(H) (two-sided):                  0.65   Kurtosis:
6.25
===================================================================================

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

In [13]:
```
model.plot_diagnostics(figsize=(15,5))
plt.show()
save_plot("arimaplot_diag")
```




```
./images/arimaplot_diag created.

<Figure size 432x288 with 0 Axes>
```

Model fit analysis and justification for ARIMA

```
1.Standard residual revolves around mean 0
2. Histogram suggests the values center on mean zero and follows normal distribut
ion
3. There is no much deviation from the main red line
4. The residual errors are not auto correlated, no further tests required .
```

y_ts values seem a good fit done using Auto Arima . So we proceed with forecasting and optimisation.

Conclusion Model 1 analysis : AR based model analysis

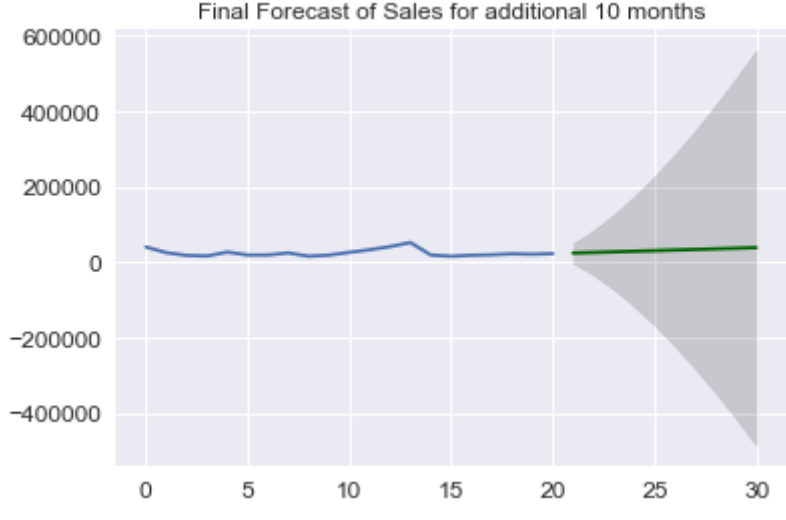
ARIMA time series use AIC ,BIC value for assessment . In the model summary above, the Best hyperparameters (p,d,q) for our 1st model are ARIMA(0,2,0)(0,0,0)[0] , giving us AIC value of 419.694. But we shall further optimise the model to have lower AIC value

We shall not stop, lets explore how we can optimise the model better and aim for better AIC values

Forecasting with Model 1 :

ARIMA (0,2,0)(0,0,0)[0]

In [20]:
```
# Plot
plt.plot(y_ts.values)
plt.plot(fc_series, color='darkgreen')
plt.fill_between(lower_series.index, lower_series, upper_series, color='k', alpha
=.15)
plt.title("Final Forecast of Sales for additional 10 months")
plt.show()
save_plot("arimaplot_ns.png")
```




```
./images/arimaplot_ns.png created.

<Figure size 432x288 with 0 Axes>
```

Using plots , we shall test for differention Plot seasonal differentiation if any

In [21]:

```
# Plot
fig, axes = plt.subplots(2, 1, figsize=(10,5), dpi=100, sharex=True)

# Usual Differencing
axes[0].plot(y_ts[:], label='Original Series')
axes[0].plot(y_ts[:].diff(1), label='Usual Differencing')
axes[0].set_title('Usual Differencing')
axes[0].legend(loc='upper left', fontsize=10)


# Seasinal Dei
axes[1].plot(y_ts[:], label='Original Series')
axes[1].plot(y_ts[:].diff(12), label='Seasonal Differencing', color='green')
axes[1].set_title('Seasonal Differencing')
plt.legend(loc='upper left', fontsize=10)
plt.suptitle(' Sales', fontsize=16)
plt.show()
save_plot("seasonal_diff")
```

Model 2: Explore ARIMA with Seasonality

Although the model fit well , we shall optimise but introducing seasonal factors in from Moving average (MA). Please refer to ARIMA formual for moving average part . It is not part of this exercise.

In [22]:

```
import pmdarima as pm
from statsmodels.tsa.arima_model import ARIMA

# Seasonal - fit stepwise auto-ARIMA

#pm.auto_arima
smodel = pm.auto_arima(y_ts, start_p=1, start_q=1,
                       test='adf',
                       max_p=3, max_q=3, m=12,
                       start_P=0, seasonal=True,
                       d=None, D=1, trace=True,
                       error_action='ignore',
                       suppress_warnings=True,
                       stepwise=True)



model_fit = smodel.fit(y_ts,disp=0)
```

```
Performing stepwise search to minimize aic
 ARIMA(1,2,1)(0,1,1)[12]             : AIC=inf, Time=0.18 sec
 ARIMA(0,2,0)(0,1,0)[12]             : AIC=163.441, Time=0.01 sec
 ARIMA(1,2,0)(1,1,0)[12]             : AIC=157.073, Time=0.08 sec
 ARIMA(0,2,1)(0,1,1)[12]             : AIC=inf, Time=0.13 sec
 ARIMA(1,2,0)(0,1,0)[12]             : AIC=155.075, Time=0.03 sec
 ARIMA(1,2,0)(0,1,1)[12]             : AIC=157.074, Time=0.07 sec
 ARIMA(1,2,0)(1,1,1)[12]             : AIC=inf, Time=0.38 sec
 ARIMA(2,2,0)(0,1,0)[12]             : AIC=154.754, Time=0.04 sec
 ARIMA(2,2,0)(1,1,0)[12]             : AIC=156.725, Time=0.08 sec
 ARIMA(2,2,0)(0,1,1)[12]             : AIC=156.725, Time=0.06 sec
 ARIMA(2,2,0)(1,1,1)[12]             : AIC=158.718, Time=0.35 sec
 ARIMA(3,2,0)(0,1,0)[12]             : AIC=156.335, Time=0.04 sec
 ARIMA(2,2,1)(0,1,0)[12]             : AIC=156.218, Time=0.10 sec
 ARIMA(1,2,1)(0,1,0)[12]             : AIC=inf, Time=0.07 sec
 ARIMA(3,2,1)(0,1,0)[12]             : AIC=157.908, Time=0.46 sec
 ARIMA(2,2,0)(0,1,0)[12] intercept   : AIC=157.961, Time=0.18 sec
```

```
 ARIMA(2,2,0)(0,1,0)[12] intercept   : AIC 137.961, Time 0.18 sec

Best model:  ARIMA(2,2,0)(0,1,0)[12]
Total fit time: 2.319 seconds
```

Model 2: Summary

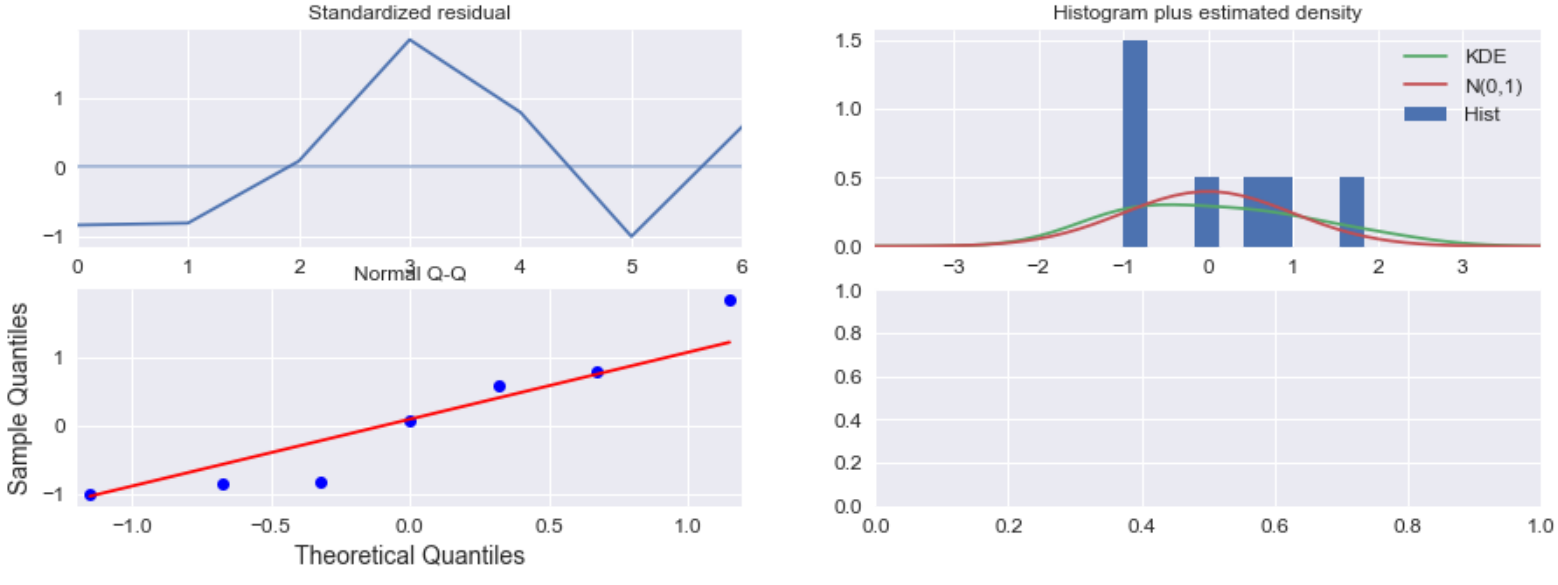
In [23]:
```
print(model_fit.summary())
```

```
                                     SARIMAX Results
==========================================================================================
Dep. Variable:                                   y   No. Observations:
21
Model:             SARIMAX(2, 2, 0)x(0, 1, 0, 12)   Log Likelihood
-74.377
Date:                            Thu, 04 Feb 2021   AIC
154.754
Time:                                    15:17:29   BIC
154.592
Sample:                                         0   HQIC
152.749
                                              - 21
Covariance Type:                              opg
==============================================================================
=
                 coef    std err          z      P>|z|      [0.025      0.97
5]
------------------------------------------------------------------------------
-
ar.L1         -0.9786      0.277     -3.528      0.000      -1.522      -0.43
5
ar.L2         -0.3231      0.249     -1.295      0.195      -0.812       0.16
6
sigma2      1.021e+08   1.26e-09   8.07e+16      0.000    1.02e+08    1.02e+0
8
==========================================================================================
======
Ljung-Box (L1) (Q):                   0.13   Jarque-Bera (JB):
0.55
Prob(Q):                              0.72   Prob(JB):
0.76
Heteroskedasticity (H):               0.99   Skew:
```

0.45
Prob(H) (two-sided): 1.00 Kurtosis:
1.97
==========
======

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (comple
x-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.0
6e+34. Standard errors may be unstable.

In [29]:
```
#Plotting diagnotics will bring a shape related error which we shall suppress for
 now.
#The most import aspect to check how MOdel 2 fits across
try :
    model_fit.plot_diagnostics(figsize=(15,5))
    plt.show()
    save_plot("sarimaplot_diag")
except ValueError:
    pass
```



Model fit analysis and justification for ARIMA

Conclusion: Model 2 analysis : AR +MA

ARIMA time series use AIC ,BIC value for assessment . In the model summary above, the Best hyperparameters (p,d,q) for our 1st model are ARIMA(0,2,0)(0,0,0)[0] , giving us AIC value of 154.754, 2.5 lower than model 1.

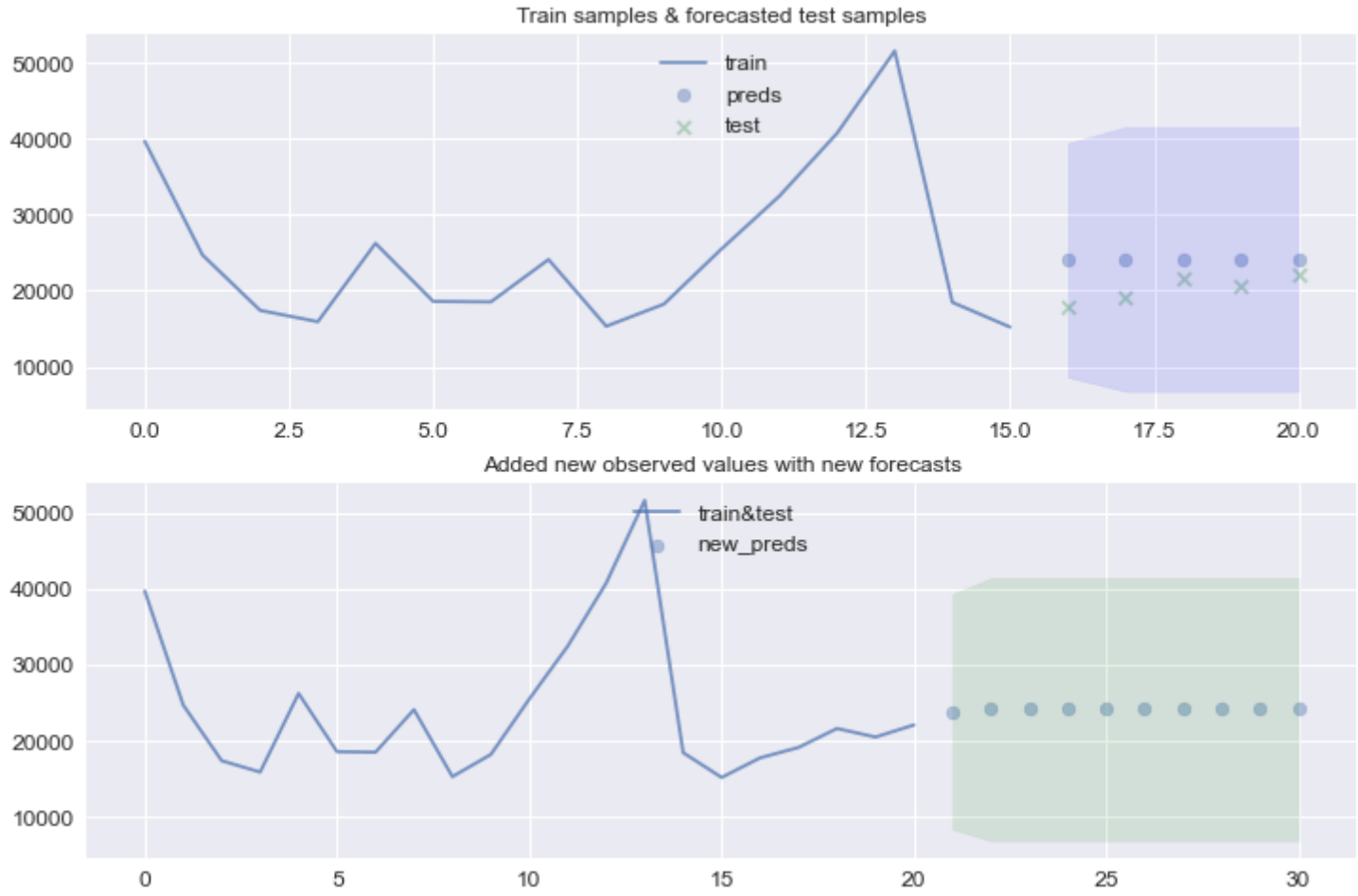
So model 2 is performing better than 1 and will be adopted and saved for forecasting

Forecasting using Model 2

In [30]:

```
from pmdarima import model_selection
train, test = model_selection.train_test_split(y_ts, train_size=0.8)

# #############################################################################
# Fit with some validation (cv) samples
arima = pm.auto_arima(train, start_p=1, start_q=1, d=0, max_p=5, max_q=5,
                      out_of_sample_size=10, suppress_warnings=True,
                      stepwise=True, error_action='ignore')

# Now plot the results and the forecast for the test set
preds, conf_int = arima.predict(n_periods=test.shape[0],return_conf_int=True)

fig, axes = plt.subplots(2, 1, figsize=(12, 8))
x_axis = np.arange(train.shape[0] + preds.shape[0])
axes[0].plot(x_axis[:train.shape[0]], train, alpha=0.75,label='train')
axes[0].scatter(x_axis[train.shape[0]:], preds, alpha=0.4, marker='o',label='pred
s')
axes[0].scatter(x_axis[train.shape[0]:], test, alpha=0.4, marker='x',label='test')
axes[0].fill_between(x_axis[-preds.shape[0]:], conf_int[:, 0], conf_int[:, 1],alph
a=0.1, color='b')
axes[0].legend(loc='upper center')

# fill the section where we "held out" samples in our model fit

axes[0].set_title("Train samples & forecasted test samples")

# Now add the actual samples to the model and create NEW forecasts
arima.update(test)
new_preds, new_conf_int = arima.predict(n_periods=10, return_conf_int=True)
new_x_axis = np.arange(y_ts.shape[0] + 10)

axes[1].plot(new_x_axis[:y_ts.shape[0]], y_ts, alpha=0.75,label='train&test')
axes[1].scatter(new_x_axis[y_ts.shape[0]:], new_preds, alpha=0.4, marker='o', labe
l='new_preds')
axes[1].fill_between(new_x_axis[-new_preds.shape[0]:],
                     new_conf_int[:, 0],
```

```
                    new_conf_int[:, 1],
                    alpha=0.1, color='g')
axes[1].set_title("Added new observed values with new forecasts")
axes[1].legend(loc='upper center')
plt.show()
save_plot("pmdarima_analysis.png")
```




```
./images/pmdarima_analysis.png created.

<Figure size 432x288 with 0 Axes>
```

Result of comparison of auto arima with or without seasonal trend

From graph above , its shows forecasting cannot be accurate but the confidence interval, it helps to predict values within a certain risk threshold.

AIC value of 154.74 is lower than normal arima the 494 above . We shall use auto arima with seasonal trends SARIMA. Although we have ar.L2 p value of 0.195 which is higer than 0.05 and rendering the X feature insignificant, we shall proceed with the suggested hyperparameters

In [121]:

```
# Forecast
n_periods = 3
fitted, confint = smodel.predict(n_periods=n_periods, return_conf_int=True)
index_of_fc = pd.date_range(y_ts.index[-1], periods = n_periods, freq='MS')

# make series for plotting purpose
fitted_series = pd.Series(fitted, index=index_of_fc)
lower_series = pd.Series(confint[:, 0], index=index_of_fc)
upper_series = pd.Series(confint[:, 1], index=index_of_fc)

# Plot
plt.figure(figsize=(12, 8))
plt.plot(y_ts, label ="actual")
plt.plot(fitted_series, color='darkgreen', label="fitted")
plt.fill_between(lower_series.index, lower_series, upper_series,color='k', alpha=.
15)

plt.title("SARIMA - Final Forecast  of  3 months sales starting from 01/08/2019")
plt.legend(loc ="upper center")
plt.show()
save_plot("arimasfc.png")
```

Part 2 conclusion

The data shall be modeled using monthly average inorder to account for some missing daily values , this doesnt offer much impact on ARIMA.

Auto arima and inclusion of seasonal trend shows better model performace and against none seasonal.

Scripts will be developed to model based on second SARIMA model

Seasonal index

SARIMAX with exogenous features depend on seaesonal index requires seasonal decompose , with current data limited to 21 months, it does not fulfill the requirements to decompose. So this will be skipped until we have more data

Part 3: Modeling, testing , deployment into production

In this section we shall turn our EDA into model development for deployment on docker containers so there might be a bit of repeition in the code .

The main script is app.py which will be running on http://0.0.0.0:8082 (http://0.0.0.0:8082). The flask app be able to

1. Model train: train the base model
2. Predict, given date and country to forecast and return a value
3. Model update : update the model with new data and check any model drifts

Test scripts are also available to for model train, prediction, logging

Create scripts

1. logging
2. python scripts for modeling offline
3. Script for docker deployment
4. Script for app predict, train , update and monitoring

The scripts are available as part of CS_PKJM notebook

1. Overwriting ./runtime/logger.py
2. Overwriting ./runtime/model.py
3. Overwriting ./runtime/app.py

Logger script

In [32]:

```
%%writefile ./runtime/logger.py
"""
module with functions to enable logging
"""

import time,os,re,csv,sys,uuid,joblib
from datetime import date

if not os.path.exists(os.path.join(".","logs")):
    os.mkdir("logs")

def update_train_log(data_shape, eval_test, runtime, MODEL_VERSION, MODEL_VERSION_
NOTE, test=False):
    """
    update train log file
    """

    ## name the logfile using something that cycles with date (day, month, year)
    today = date.today()
    if test:
        logfile = os.path.join("logs", "train-test.log")
    else:
        logfile = os.path.join("logs", "train-{}-{}.log".format(today.year, today.
month))

    ## write the data to a csv file
    header = ['unique_id','timestamp','x_shape','eval_test','model_version',
              'model_version_note','runtime']
    write_header = False
    if not os.path.exists(logfile):
        write_header = True
    with open(logfile, 'a') as csvfile:
        writer = csv.writer(csvfile, delimiter=',')
        if write_header:
            writer.writerow(header)

        to_write = map(str, [uuid.uuid4(), time.time(), data_shape, eval_test,
```

App.py script:

This is the main app for model training, forecasting , retrain. for sake of presentation, I have skipped the fragments as it is long but you can find script under ./runtime/app.py

Create deployment scripts

This part will do the following

1. Create docker file
2. Create requirements file
3. Create Test scripts
4. Test the scripts
5. Train the model
6. Update the model

In [34]:

```
%%writefile ./runtime/Dockerfile

# Use an official Python runtime as a parent image
FROM python:3.7.5-stretch

MAINTAINER Pelani Malange "pmalange@za.ibm.com"

RUN apt-get update && apt-get install -y \
python3-dev \
build-essential

# Set the working directory to /app
WORKDIR /app

## Copy the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip3 install --upgrade pip
RUN pip3 install --no-cache-dir -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python3", "app.py"]
```

```
Overwriting ./runtime/Dockerfile
```

In [35]:

```
%%writefile ./runtime/requirements.txt
#create a requirements file for docker

cython
numpy
flask
flask_json
pandas
pmdarima
```

```
Overwriting ./runtime/requirements.txt
```

Sample information on docker running

Build docker file

Removing intermediate container 4957071b2c24 ---> 794aa2a93a9e Step 10/10 : CMD ["python3", "app.py"] ---> Running in 1ede8ae6d7a8 Removing intermediate container 1ede8ae6d7a8 ---> 8eda8b2fa639 Successfully built 8eda8b2fa639 Successfully tagged cs-forecast-app:latest pelanimac:runtime pelani$

- Environment: production WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
- Debug mode: on
- Running on http://0.0.0.0:8082/ (http://0.0.0.0:8082/) (Press CTRL+C to quit)
- Restarting with stat
- Debugger is active!
- Debugger PIN: 576-887-112

Testing

Unit test: API

In [38]:

```
%%writefile ./runtime/unittests/ApiTests.py


#Create API test script

import sys
import os
import unittest
import requests
import re
from ast import literal_eval
import numpy as np

port = 8082

try:
    requests.post('http://0.0.0.0:{}/predict'.format(port))
    server_available = True
except:
    server_available = False

## test class for the main window function
class ApiTest(unittest.TestCase):
    """
    test the essential functionality
    """

    @unittest.skipUnless(server_available, "local server is not running")
    def test_01_train(self):
        """
        test the train functionality
        """

        fileup= open("./data/test.txt","rb")
        r = requests.post("http://0.0.0.0:{}/train".format(port),files ={"file":fi
leup})
```

```
        fileup.close()
        #train_complete = re.sub("\W+", "", r.text)
        train_complete = re.sub("\W+", "", r.text)
        self.assertEqual(train_complete, 'country0allmape003rmse002')
        '''
         response  = r.text
        eval_test = dict({"country": {"0": "all"}, "mape": {"0": "0.3"}, "rmse":
 {"0": "0.2"}})
        self.assertEqual(response, eval_test)
        '''

    @unittest.skipUnless(server_available, "local server is not running")
    def test_02_predict_empty(self):
        """
        ensure appropriate failure types
        """

        ## provide no data at all
        r = requests.post('http://0.0.0.0:{}/predict'.format(port))
        self.assertEqual(literal_eval(r.text), [])

        ## provide improperly formatted data
        #r = requests.post('http://0.0.0.0:{}/predict'.format(port), json={"ke
y":"value"})
        #self.assertEqual(literal_eval(r.text),[])


    @unittest.skipUnless(server_available,"local server is not running")
    def test_03_predict(self):
        """
        test the predict functionality
        """

        query_data = {"country":"United Kingdom", "date" :"01/08/2019"}

        query_type = 'dict'
        request_json = {'query':query_data, 'type':query_type}
```

```
        r = requests.post('http://0.0.0.0:{}/predict'.format(port), json=request_j
son)
        response = literal_eval(r.text)

        self.assertEqual(response,{'Predrevenue': 23607.119, 'status': 200})



    @unittest.skipUnless(server_available, "local server is not running")
    def test_04_logs(self):
        """
        test the log functionality
        """

        file_name = 'train-test.log'
        request_json = {'file':'train-test.log'}
        r = requests.get('http://0.0.0.0:{}/logs/{}'.format(port, file_name))

        with open(file_name, 'wb') as f:
            f.write(r.content)

        self.assertTrue(os.path.exists(file_name))

        if os.path.exists(file_name):
            os.remove(file_name)


### Run the tests
if __name__ == '__main__':
    unittest.main()
```

```
Overwriting ./runtime/unittests/ApiTests.py
```

In [39]:
```
#test the APIs
!python3 ./runtime/unittests/ApiTests.py
```

```
....
----------------------------------------------------------------------
Ran 4 tests in 0.263s

OK
```

Unit test : Model

In [40]:

```
%%writefile ./runtime/unittests/ModelTests.py

"""
model tests
"""

import sys, os
import unittest
sys.path.insert(1, os.path.join('..', os.getcwd()))

## import model specific functions and variables
from model import *


class ModelTest(unittest.TestCase):
    """
    test the essential functionality
    """

    def test_01_train(self):
        """
        test the train functionality
        """

        ## train the model
        model_train(test=True)
        self.assertTrue(os.path.exists(os.path.join("models", "sales-arima-0_1.job
lib")))

    def test_02_load(self):
        """
        test the train functionality
        """

        ## train the model
        model = model_load(test=True)
```

```
        self.assertTrue('predict' in dir(model))
        self.assertTrue('fit' in dir(model))


### Run the tests
if __name__ == '__main__':
    unittest.main()
```

```
Overwriting ./runtime/unittests/ModelTests.py
```

Unit test: Logging

In [39]:

```
%%writefile ./runtime/unittests/LoggerTests.py
"""
model tests
"""

import os, sys
import csv
import unittest
from ast import literal_eval
import pandas as pd
sys.path.insert(1, os.path.join('..', os.getcwd()))

## import model specific functions and variables
from logger import update_train_log, update_predict_log



class LoggerTest(unittest.TestCase):
    """
    test the essential functionality
    """

    def test_01_train(self):
        """
        ensure log file is created
        """

        log_file = os.path.join("logs", "train-test.log")
        if os.path.exists(log_file):
            os.remove(log_file)

        ## update the log
        data_shape = (100,10)
        eval_test = {'rmse':0.5}
        runtime = "00:00:01"
        model_version = 0.1
        model_version_note = "test model"
```

```
        update_train_log(data_shape,eval_test, runtime,model_version, model_versio
n_note, test=True)

        self.assertTrue(os.path.exists(log_file))

    def test_02_train(self):
        """
        ensure that content can be retrieved from log file
        """

        log_file = os.path.join("logs", "train-test.log")

        ## update the log
        data_shape = (100,10)
        eval_test = {'rmse':0.5}
        runtime = "00:00:01"
        model_version = 0.1
        model_version_note = "test model"

        update_train_log(data_shape,eval_test, runtime,model_version, model_versio
n_note, test=True)

        df = pd.read_csv(log_file)
        logged_eval_test = [literal_eval(i) for i in df['eval_test'].copy()][-1]
        self.assertEqual(eval_test, logged_eval_test)


    def test_03_predict(self):
        """
        ensure log file is created
        """

        log_file = os.path.join("logs","predict-test.log")
        if os.path.exists(log_file):
            os.remove(log_file)

        ## update the log
```

```
        y_pred = [0]
        y_proba = [0.6, 0.4]
        runtime = "00:00:02"
        model_version = 0.1
        query = ['united_states', 24, 'aavail_basic', 8]

        update_predict_log(y_pred, y_proba, query, runtime,
                           model_version, test=True)

        self.assertTrue(os.path.exists(log_file))


    def test_04_predict(self):
        """
        ensure that content can be retrieved from log file
        """

        log_file = os.path.join("logs","predict-test.log")

        ## update the log
        y_pred = [0]
        y_proba = [0.6, 0.4]
        runtime = "00:00:02"
        model_version = 0.1
        query = {"country":"United Kindgom","date":"01/08/2019"}

        update_predict_log(y_pred, y_proba, query, runtime,
                           model_version, test=True)

        df = pd.read_csv(log_file)
        logged_y_pred = [literal_eval(i) for i in df['y_pred'].copy()][-1]
        self.assertEqual(y_pred,logged_y_pred)


### Run the tests
if __name__ == '__main__':
    unittest.main()
```

Run all tests at once

In [45]:
```
%%writefile ./runtime/run_all_tests

import sys
import unittest

from unittests import *
unittest.main()
```

```
Overwriting ./runtime/run_all_tests
```

In [44]:
```
!python3  ./runtime/run_all_tests
```

```
None
........Performing stepwise search to minimize aic
 ARIMA(1,2,1)(0,1,1)[12]             : AIC=inf, Time=0.38 sec
 ARIMA(0,2,0)(0,1,0)[12]             : AIC=163.441, Time=0.03 sec
 ARIMA(1,2,0)(1,1,0)[12]             : AIC=157.073, Time=0.17 sec
 ARIMA(0,2,1)(0,1,1)[12]             : AIC=inf, Time=0.51 sec
 ARIMA(1,2,0)(0,1,0)[12]             : AIC=155.075, Time=0.05 sec
 ARIMA(1,2,0)(0,1,1)[12]             : AIC=157.074, Time=0.73 sec
 ARIMA(1,2,0)(1,1,1)[12]             : AIC=inf, Time=0.96 sec
 ARIMA(2,2,0)(0,1,0)[12]             : AIC=154.754, Time=0.13 sec
 ARIMA(2,2,0)(1,1,0)[12]             : AIC=156.725, Time=0.18 sec
 ARIMA(2,2,0)(0,1,1)[12]             : AIC=156.725, Time=0.22 sec
 ARIMA(2,2,0)(1,1,1)[12]             : AIC=158.718, Time=0.50 sec
 ARIMA(3,2,0)(0,1,0)[12]             : AIC=156.335, Time=0.16 sec
 ARIMA(2,2,1)(0,1,0)[12]             : AIC=156.218, Time=0.42 sec
 ARIMA(1,2,1)(0,1,0)[12]             : AIC=inf, Time=0.19 sec
 ARIMA(3,2,1)(0,1,0)[12]             : AIC=157.908, Time=0.32 sec
 ARIMA(2,2,0)(0,1,0)[12] intercept   : AIC=157.961, Time=0.21 sec

Best model:  ARIMA(2,2,0)(0,1,0)[12]
Total fit time: 5.157 seconds
.... loading test version of model
.
----------------------------------------------------------------------
Ran 10 tests in 7.228s

OK
```

Systems Admin user : checking log files

In [43]:

```
port =8082
file_name = 'train-test.log'
request_json = {'file':'train-test.log'}
r = requests.get('http://0.0.0.0:{}/logs/{}'.format(port, file_name))
print(r.text)



#with open(file_name, 'wb') as f:
#    f.write(r.content)
```

```
unique_id,timestamp,x_shape,eval_test,model_version,model_version_note,runtime
8a79577f-168f-4a51-8c8a-79bfcb3b84f9,1612197491.6594791,21,"{'country': 'Unite
d Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:00
b614ad3d-2eea-4499-bbdd-bae541b81cd2,1612197493.465964,21,"{'country': 'Franc
e', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:02
49d22a2c-7c07-46e4-8fcd-1e210c515c70,1612197494.979776,21,"{'country': 'Belgiu
m', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:04
37686fa2-da68-4ffe-b7c7-151a20b6c198,1612197497.1419232,21,"{'country': 'EIR
E', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:06
18f3ca8d-a386-43b2-84a7-2f7f9eb21944,1612197498.790972,21,"{'country': 'German
y', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:08
bd213df5-dc66-4148-ad76-56de8966eef8,1612197500.574254,21,"{'country': 'Portug
al', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:09
c17f9d06-f988-4495-9c8a-dff45e4882ad,1612197503.544734,21,"{'country': 'Nether
lands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:12
0708770e-e0ca-4a25-9814-8fe682a0ce06,1612197505.699559,21,"{'country': 'Spai
n', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:14
d832b8c3-4a22-4084-a6f7-61229304be28,1612200178.1979892,21,"{'country': 'Unite
d Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:02
3024d47d-17e0-4541-8ae9-ed097a4284f2,1612200178.310789,21,"{'country': 'Franc
e', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:02
4843206d-4653-4474-b20f-86b4867851a1,1612200178.4099789,21,"{'country': 'Belgi
um', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:02
6707f1d1-0e08-427d-8d9c-4d78c366d9e4,1612200178.513358,21,"{'country': 'EIRE',
'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:02
```

```
fb8caeac-3153-4b6e-8a5f-adc23497e90b,1612200178.6158779,21,"{'country': 'Germa
ny', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:02
af40a115-5aa1-4ef7-8b8e-f6f1171b4703,1612200178.7146938,21,"{'country': 'Portu
gal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:02
008139e2-05ad-4f8f-ae4a-43e4b39c0355,1612200178.841161,21,"{'country': 'Nether
lands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:02
f7e8e5ea-45db-4412-bbcf-b892de1d7e09,1612200178.952953,21,"{'country': 'Spai
n', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:02
1d0da3c6-b6f3-4c8b-aac5-ddbca5ca888a,1612200179.0555708,20,"{'country': 'Norwa
y', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:03
7f02c596-dfdf-40e0-ae95-310a10aa5249,1612200179.165059,20,"{'country': 'Switze
rland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:03
910d7beb-b91b-4f5d-94ad-1f95fb97ad39,1612200271.7611809,21,"{'country': 'Unite
d Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
ba2b410a-cda5-48f0-88a5-dd458f51773c,1612200271.874722,21,"{'country': 'Franc
e', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
4a5f2c86-7599-4920-814d-72c8ceab37e5,1612200271.9820108,21,"{'country': 'Belgi
um', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
93ca8ac2-2bd5-46b7-b6ca-41b05241a146,1612200272.096315,21,"{'country': 'EIRE',
'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
17939d62-821e-4a67-9f15-295e9c6ddf7a,1612200272.225479,21,"{'country': 'German
y', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01
e937f354-255c-444e-aea7-45027757d86d,1612200272.337064,21,"{'country': 'Portug
al', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:01
d10d65a5-53d6-45dd-9fb1-facc85791ba8,1612200272.442785,21,"{'country': 'Nether
lands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:01
8f5e3240-88da-4df1-aec9-6e36288fa4c4,1612200272.549472,21,"{'country': 'Spai
n', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:02
f00235bb-41a2-4137-be36-2789d6a22c35,1612200272.663116,20,"{'country': 'Norwa
y', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:02
4dd963b7-df5f-4172-968d-96f43a5374a4,1612200272.766829,20,"{'country': 'Switze
rland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
def4ad9c-c213-4557-88d7-d3a9d79e3bcb,1612202215.088038,21,"{'country': 'United
Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
24d2ec5e-1ddd-48e0-b144-afdc64a9e33c,1612202215.1999118,21,"{'country': 'Franc
e', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
0942a24e-cf9c-46c7-94c9-648a75ed2dd1,1612202215.306164,21,"{'country': 'Belgiu
m', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
```

a573cdb8-eb0b-484a-a73a-f345b671b067,1612202215.413891,21,"{'country': 'EIRE',
'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
97d1f7d4-5f03-41dd-8cf1-7a4562903b71,1612202215.521256,21,"{'country': 'German
y', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01
26ed7cdd-ef34-4e02-b2a6-9855e0543755,1612202215.6264262,21,"{'country': 'Portu
gal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:01
817a9806-d102-47be-9742-8a6d1697153f,1612202215.738503,21,"{'country': 'Nether
lands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:01
8c23524b-a7db-4fc7-8357-e44d561109ae,1612202215.8462532,21,"{'country': 'Spai
n', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:01
fc134a0b-e3e2-440c-b2b1-ca2aedb20bc3,1612202215.950031,20,"{'country': 'Norwa
y', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:01
b8ff4264-6420-4fd4-b48a-ff9d4f0b6f14,1612202216.0577438,20,"{'country': 'Switz
erland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
091d992d-bc8d-4a99-9d64-1802b36b501a,1612251140.635807,21,"{'country': 'United
Kingdom', 'rmse': 4805.55, 'mape': 19, 'AIC': <function ARIMA.aic at 0x7f96e18
d6550>}",0.1,auto_arima,000:00:01
61e500f6-0a48-406e-896c-57f57683d51f,1612251140.751391,21,"{'country': 'Franc
e', 'rmse': 230.66, 'mape': 26, 'AIC': <function ARIMA.aic at 0x7f96e18d6f70
>}",0.1,auto_arima,000:00:01
4d7885d5-04ed-4d59-aee6-c1c87a5ee6f4,1612251140.8609369,21,"{'country': 'Belgi
um', 'rmse': 156.52, 'mape': 48, 'AIC': <function ARIMA.aic at 0x7f96e18d6820
>}",0.1,auto_arima,000:00:01
8b94e4d5-c76c-40b2-aabb-1ccebafb78fb,1612251140.9720352,21,"{'country': 'EIR
E', 'rmse': 442.61, 'mape': 24, 'AIC': <function ARIMA.aic at 0x7f970491f280
>}",0.1,auto_arima,000:00:01
8b6f2c29-f83e-4072-b752-3e4218f0232c,1612251141.092972,21,"{'country': 'German
y', 'rmse': 385.81, 'mape': 36, 'AIC': <function ARIMA.aic at 0x7f96e18d6550
>}",0.1,auto_arima,000:00:01
a2ba143e-0b4e-4c2a-817f-4fd7ae61896b,1612251141.198559,21,"{'country': 'Portug
al', 'rmse': 2895.75, 'mape': 68, 'AIC': <function ARIMA.aic at 0x7f96e18d6f70
>}",0.1,auto_arima,000:00:01
cdf077c3-1919-4a2b-834c-29206340199d,1612251141.300041,21,"{'country': 'Nether
lands', 'rmse': 358.91, 'mape': 958, 'AIC': <function ARIMA.aic at 0x7f9704909
f70>}",0.1,auto_arima,000:00:01
e4564a19-d606-47c1-a52e-fd9d7b87d0e1,1612251141.402768,21,"{'country': 'Spai
n', 'rmse': 233.64, 'mape': 51, 'AIC': <function ARIMA.aic at 0x7f9704909f70
>}",0.1,auto_arima,000:00:02

d65f1d92-6a1c-4e00-94fe-491838cf8021,1612251141.516053,20,"{'country': 'Norwa
y', 'rmse': 1344.39, 'mape': 100, 'AIC': <function ARIMA.aic at 0x7f9704909f70
>}",0.1,auto_arima,000:00:02
b70bd316-8759-4d31-9378-baf7cdf26310,1612251141.62627,20,"{'country': 'Switzer
land', 'rmse': 471.57, 'mape': 44, 'AIC': <function ARIMA.aic at 0x7f9704909f7
0>}",0.1,auto_arima,000:00:02
c10587a1-d4a3-4822-aa0b-4d840555309a,1612251240.845609,21,"{'country': 'United
Kingdom', 'rmse': 4805.55, 'mape': 19, 'AIC': <function ARIMA.aic at 0x7fdc120
1fdc0>}",0.1,auto_arima,000:00:01
11656122-e504-485c-ac17-7b1c62b08158,1612251240.95678,21,"{'country': 'Franc
e', 'rmse': 230.66, 'mape': 26, 'AIC': <function ARIMA.aic at 0x7fdbef1da3a0
>}",0.1,auto_arima,000:00:01
3d46449b-d426-4154-b9ea-e4e083938d36,1612251241.0539021,21,"{'country': 'Belgi
um', 'rmse': 156.52, 'mape': 48, 'AIC': <function ARIMA.aic at 0x7fdc1201faf0
>}",0.1,auto_arima,000:00:01
3655c4ca-3c10-441e-b0be-3f9ffc2f50fa,1612251241.152826,21,"{'country': 'EIRE',
'rmse': 442.61, 'mape': 24, 'AIC': <function ARIMA.aic at 0x7fdc1201fdc0>}",0.
1,auto_arima,000:00:01
d571ec0f-6fc8-44bf-b745-171105d4c0d7,1612251241.262628,21,"{'country': 'German
y', 'rmse': 385.81, 'mape': 36, 'AIC': <function ARIMA.aic at 0x7fdc1201faf0
>}",0.1,auto_arima,000:00:01
7fec97e8-6c94-4680-869c-8478e758204c,1612251241.373149,21,"{'country': 'Portug
al', 'rmse': 2895.75, 'mape': 68, 'AIC': <function ARIMA.aic at 0x7fdbef1dad30
>}",0.1,auto_arima,000:00:01
56712564-333c-4214-8a17-a8677b63c54e,1612251241.489654,21,"{'country': 'Nether
lands', 'rmse': 358.91, 'mape': 958, 'AIC': <function ARIMA.aic at 0x7fdbef1da
e50>}",0.1,auto_arima,000:00:02
567a41e8-14d6-4394-94a7-48856f53b8ec,1612251241.605136,21,"{'country': 'Spai
n', 'rmse': 233.64, 'mape': 51, 'AIC': <function ARIMA.aic at 0x7fdc1201f3a0
>}",0.1,auto_arima,000:00:02
586f5fc6-70f0-4779-97fa-327af9b9e6ab,1612251241.713297,20,"{'country': 'Norwa
y', 'rmse': 1344.39, 'mape': 100, 'AIC': <function ARIMA.aic at 0x7fdc1201f4c0
>}",0.1,auto_arima,000:00:02
5b671ff4-4070-48d7-91fd-82499add0cfb,1612251241.818978,20,"{'country': 'Switze
rland', 'rmse': 471.57, 'mape': 44, 'AIC': <function ARIMA.aic at 0x7fdbef1da2
80>}",0.1,auto_arima,000:00:02
21cdeaf5-b338-42c3-8e1d-d37576375477,1612251418.050495,21,"{'country': 'United
Kingdom', 'rmse': 4805.55, 'mape': 19, 'AIC': <function ARIMA.aic at 0x7fbad29

09f70>}",0.1,auto_arima,000:00:01
037d40dc-fe9f-4526-a884-99d3efbc0a0d,1612251418.168885,21,"{'country': 'Franc
e', 'rmse': 230.66, 'mape': 26, 'AIC': <function ARIMA.aic at 0x7fbaaf9d4820
>}",0.1,auto_arima,000:00:01
96433023-5672-4cbe-8249-43a6d16f115c,1612251418.274867,21,"{'country': 'Belgiu
m', 'rmse': 156.52, 'mape': 48, 'AIC': <function ARIMA.aic at 0x7fbaaf9d4550
>}",0.1,auto_arima,000:00:01
62b5faf7-0625-470b-8a73-4af4b33c8a07,1612251418.386466,21,"{'country': 'EIRE',
'rmse': 442.61, 'mape': 24, 'AIC': <function ARIMA.aic at 0x7fbad291fc10>}",0.
1,auto_arima,000:00:01
73f95aa6-31e6-4063-a5ee-402507d6100c,1612251418.4963799,21,"{'country': 'Germa
ny', 'rmse': 385.81, 'mape': 36, 'AIC': <function ARIMA.aic at 0x7fbad291fdc0
>}",0.1,auto_arima,000:00:01
96b9592d-d36e-4c9b-9fe0-7b6c9b253bcd,1612251418.612003,21,"{'country': 'Portug
al', 'rmse': 2895.75, 'mape': 68, 'AIC': <function ARIMA.aic at 0x7fbaaf9d43a0
>}",0.1,auto_arima,000:00:01
9374cc70-ded8-44c9-9b18-b489f2628c63,1612251418.725589,21,"{'country': 'Nether
lands', 'rmse': 358.91, 'mape': 958, 'AIC': <function ARIMA.aic at 0x7fbad2909
f70>}",0.1,auto_arima,000:00:01
c529cfc0-7e63-48c2-8814-c5dca50f2f75,1612251418.843699,21,"{'country': 'Spai
n', 'rmse': 233.64, 'mape': 51, 'AIC': <function ARIMA.aic at 0x7fbaaf9d4550
>}",0.1,auto_arima,000:00:01
2af119be-c163-4d7a-824a-45e6ea5bcd94,1612251418.9539661,20,"{'country': 'Norwa
y', 'rmse': 1344.39, 'mape': 100, 'AIC': <function ARIMA.aic at 0x7fbaaf9d4820
>}",0.1,auto_arima,000:00:02
86fd591c-11e2-4275-a58d-68be3fa06fcc,1612251419.061545,20,"{'country': 'Switze
rland', 'rmse': 471.57, 'mape': 44, 'AIC': <function ARIMA.aic at 0x7fbad291f0
40>}",0.1,auto_arima,000:00:02
de0744b5-8809-413b-90b6-5fa97b83011a,1612290756.2929122,21,"{'country': 'Unite
d Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
3c4dfe67-0a32-409a-80b6-ef16f7098f00,1612290756.41638,21,"{'country': 'Franc
e', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
b238b743-2858-40b4-8286-710620a83390,1612290756.522076,21,"{'country': 'Belgiu
m', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
877dee15-1fb2-4546-a485-b7a81eaadf6d,1612290756.6227121,21,"{'country': 'EIR
E', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
9bc6762a-95b7-4213-9b64-81c9215dccfa,1612290756.725698,21,"{'country': 'German
y', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01

```
f7c6a1a5-eaf2-422a-9727-4483c475b829,1612290756.848114,21,"{'country': 'Portug
al', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:01
80b56f33-a216-4207-a02a-bc5d19f077b3,1612290756.948029,21,"{'country': 'Nether
lands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:01
3627b17a-060a-4b42-9969-c86775559586,1612290757.0501099,21,"{'country': 'Spai
n', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:02
ac8d1107-ae48-4b0e-bd1f-15b6f3340721,1612290757.1549199,20,"{'country': 'Norwa
y', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:02
29c2589c-870a-4195-8aee-e3ce681e9049,1612290757.2547998,20,"{'country': 'Switz
erland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
6e5e02d5-65a5-464c-85f0-e69a18f90cd5,1612290899.771384,21,"{'country': 'United
Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:03
a0b8e492-e096-4eb3-a400-92f7d5d62365,1612290899.9746828,21,"{'country': 'Franc
e', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:03
5dd35862-5678-4c4f-9630-1fc224d0f309,1612290900.117698,21,"{'country': 'Belgiu
m', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:03
5916fd01-0e1e-40e9-8d85-f9879dce6f95,1612290900.2344708,21,"{'country': 'EIR
E', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:03
c0c22d2c-ef59-4dba-9c5f-5021cfb6f1c3,1612290900.342806,21,"{'country': 'German
y', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:03
4f82f798-7581-4c84-ade9-caac444851c5,1612290900.547949,21,"{'country': 'Portug
al', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:03
9bdd5b3e-8f61-4e7f-bf4e-09bfadb0f3e1,1612290900.739151,21,"{'country': 'Nether
lands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:04
c0d1880e-55b4-40f9-8a90-77b2b777825f,1612290900.923937,21,"{'country': 'Spai
n', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:04
9b565d52-3298-4fc3-a7fc-819d0ab2f95a,1612290901.154864,20,"{'country': 'Norwa
y', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:04
f322a30f-581f-40e2-8c39-28dc94dfbadd,1612290901.3263779,20,"{'country': 'Switz
erland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:04
30bb951c-c873-4c45-a712-ad5b3de96e46,1612336552.51092,21 x  1,"{'country': <bu
ilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:01
7ff37570-29de-4ce0-aa6a-654c9fd1cb85,1612337698.401654,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:00
865f41b8-be5f-4d77-a44c-94714f98ce48,1612337777.717977,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:01
2b7d3ff2-5056-4b53-9873-4cdfe066c9e8,1612338022.373685,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:01
```

```
fc49b468-502c-48fe-bd9a-635dd50a9b39,1612338196.71887,21 x  1,"{'country': <bu
ilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:01
3605cff3-f56a-45e5-96c2-2921a511c522,1612338233.031222,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:00
981c0d8e-fc43-4192-86ba-1cad2ff69d9c,1612338301.022485,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:00
edb7cb79-6642-4a60-9b91-53ebe9030517,1612339363.2721071,21 x  1,"{'country': <
built-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:0
1
eb1885a9-b2a8-4861-b7aa-7b664665d271,1612339520.686462,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:01
e0e52a74-0b81-430c-938c-83c635432204,1612339866.670476,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:01
fdf5c5e6-092d-41ba-8e5f-7335257c4236,1612339892.691184,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:01
f4109d1d-4615-4d91-80f3-bec49d7bc074,1612339919.4260259,21 x  1,"{'country': <
built-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:0
1
071d30c9-1e9c-4d2c-ae9a-5f92ed056797,1612339979.366573,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:01
3fbad7a9-8ec4-45eb-87f7-e6ae28fd1659,1612340069.8497732,21 x  1,"{'country': <
built-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:0
1
0d05292b-d4b1-4975-afa2-cdaccf081b4b,1612340082.462212,21 x  1,"{'country': <b
uilt-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:01
96e74847-7729-4571-a108-d411a23fa486,1612340478.8528051,21 x  1,"{'country': <
built-in function all>, 'rmse': '0.2', 'mape': '0.3'}",0.1,auto_arima,000:00:0
1
36894b81-58ec-4f58-bd9b-aeeeac320866,1612342844.681252,21 x 1,"{'country': 'Un
ited Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
1c26e981-87c8-4b6f-8da2-2754e2191830,1612342844.8197608,21 x 1,"{'country': 'F
rance', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
c90512f4-9800-4380-83d5-1f92f7e36d09,1612342844.940618,21 x 1,"{'country': 'Be
lgium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
0836db12-f4de-482c-b941-7da187483c88,1612342845.064142,21 x 1,"{'country': 'EI
RE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
9f86891c-8642-4a3b-8e71-66251d0ec7a1,1612342845.1754398,21 x 1,"{'country': 'G
ermany', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01
```

a602c6d7-e312-441e-9bf0-592404db0122,1612342845.2900858,21 x 1,"{'country': 'P
ortugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:01
e9026201-424a-4ab8-9819-132c087453e0,1612342845.390271,21 x 1,"{'country': 'Ne
therlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:01
d9078ebc-22e2-4f8f-912c-6152644f1fc8,1612342845.490313,21 x 1,"{'country': 'Sp
ain', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:01
fb918f91-e673-496f-8de2-3c1c2d83f181,1612342845.594667,21 x 1,"{'country': 'No
rway', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:02
7abfb697-2302-4fbf-9211-74dd2a518c42,1612342845.693654,21 x 1,"{'country': 'Sw
itzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
5e6cb3ef-6d78-47fd-b7a8-5b813e39819e,1612342895.941867,21 x 1,"{'country': 'Un
ited Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
82d7977f-4704-418c-82a7-aceef02c3b82,1612342896.1208699,21 x 1,"{'country': 'F
rance', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
f8f2d160-411c-4c63-9dc5-af5258e26861,1612342896.3053272,21 x 1,"{'country': 'B
elgium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
f0848904-610a-4225-8b46-e504164e8519,1612342896.419956,21 x 1,"{'country': 'EI
RE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
ceb7c991-e98d-4d4c-a41b-6d7a3ba4d606,1612342896.5217018,21 x 1,"{'country': 'G
ermany', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01
20546e65-72ed-4357-856a-996ff2996f00,1612342896.624565,21 x 1,"{'country': 'Po
rtugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:01
781bdca3-03e3-4a16-bced-f9bf5b35898b,1612342896.723975,21 x 1,"{'country': 'Ne
therlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:02
8e472058-e38b-47b7-86b0-b373bbbbb20a,1612342896.823669,21 x 1,"{'country': 'Sp
ain', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:02
fe31af08-02b1-4630-9727-48ee7b3fa5f5,1612342896.926684,21 x 1,"{'country': 'No
rway', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:02
594d9a9d-ca07-4c05-b1be-761374b4f37d,1612342897.027988,21 x 1,"{'country': 'Sw
itzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
2efebe3c-ed33-4a61-be27-d7f17663c788,1612342989.497254,21 x 1,"{'country': 'Un
ited Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
150e7d66-9bbb-4826-bc45-2012e5b92612,1612342989.698549,21 x 1,"{'country': 'Fr
ance', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
d1a57d19-fe83-414f-ba5f-279031abdb94,1612342989.825849,21 x 1,"{'country': 'Be
lgium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
83f8e676-ec6a-4fec-9332-8d383f44b2af,1612342989.936358,21 x 1,"{'country': 'EI
RE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01

4ca85eb1-e5a9-4b78-9d25-6cd9a495f8e2,1612342990.05188,21 x 1,"{'country': 'Ger
many', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:02
1e992ed9-bfeb-4af8-afda-a88a5d680249,1612342990.1789062,21 x 1,"{'country': 'P
ortugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:02
5ebade29-c178-4344-a018-140610d8df5b,1612342990.298375,21 x 1,"{'country': 'Ne
therlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:02
c3df148f-0e91-47fc-a17d-dcd68620a47e,1612342990.39903,21 x 1,"{'country': 'Spa
in', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:02
48ea2bc0-e094-4479-a219-d54c3d24f173,1612342990.5020428,21 x 1,"{'country': 'N
orway', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:02
e9c424a1-1cdf-4bb0-89ad-1e34993de101,1612342990.604149,21 x 1,"{'country': 'Sw
itzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
20608664-e3f0-489f-a659-1f96752e7351,1612343025.177172,21 x 1,"{'country': 'Un
ited Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:00
5f1d897c-0037-492b-ab0b-94df97931b59,1612343025.287445,21 x 1,"{'country': 'Fr
ance', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
a18d2215-a2bd-4430-ac04-ec869c5c0082,1612343025.40655,21 x 1,"{'country': 'Bel
gium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
5e0d1f54-b1d0-49c4-852c-2172ed05e03e,1612343025.515789,21 x 1,"{'country': 'EI
RE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
04084284-c982-4112-a810-eec0141953ee,1612343025.6211798,21 x 1,"{'country': 'G
ermany', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01
9d82a270-82ac-4934-9045-ebace2dfbe74,1612343025.736264,21 x 1,"{'country': 'Po
rtugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:01
e97b8534-30a6-4819-9fcd-ffeee9c10066,1612343025.8463428,21 x 1,"{'country': 'N
etherlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:01
7fe90998-caed-455c-a4a1-975a4c64f544,1612343025.9520352,21 x 1,"{'country': 'S
pain', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:01
1d86852c-6e91-414c-b230-4b440be1db60,1612343026.067028,21 x 1,"{'country': 'No
rway', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:01
21c64dca-7554-45a6-8f1d-13128b312e4e,1612343026.1857998,21 x 1,"{'country': 'S
witzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:01
404acebb-27e7-4627-b2c4-f903cf71fd51,1612343049.284055,21 x 1,"{'country': 'Un
ited Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
a0e6632e-162b-482f-9b89-7cd58e4cb011,1612343049.466311,21 x 1,"{'country': 'Fr
ance', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
24334969-5013-4803-b0e1-f905cd1c1f47,1612343049.62133,21 x 1,"{'country': 'Bel
gium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01

dfca0d44-1974-48b5-8167-bb435bff0a22,1612343049.725479,21 x 1,"{'country': 'EIRE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
3eb70da1-c09e-4f20-9382-630049cab242,1612343049.831743,21 x 1,"{'country': 'Germany', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:02
ba172282-4474-449d-901a-d5b0e5adce10,1612343049.9430442,21 x 1,"{'country': 'Portugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:02
29b6c06b-bccd-4990-912a-8250a6a98b5e,1612343050.059465,21 x 1,"{'country': 'Netherlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:02
ee3507c3-6c8c-4c51-9a14-469192c52e00,1612343050.1733508,21 x 1,"{'country': 'Spain', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:02
e94ec7d5-0a40-403c-9234-7a46b6dd6cf0,1612343050.293213,21 x 1,"{'country': 'Norway', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:02
0d1f4d14-f3c7-48ad-bead-10f8d506d1bb,1612343050.413613,21 x 1,"{'country': 'Switzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
2d2579df-a2fb-4166-9934-515d40559aa6,1612343082.391484,21 x 1,"{'country': 'United Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
cfc276ef-fa08-4889-9851-102e1e6ab9c0,1612343082.503701,21 x 1,"{'country': 'France', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
18a78b4f-bf38-41b7-86c8-66d698af0aee,1612343082.620537,21 x 1,"{'country': 'Belgium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
fd66af57-4b17-4fd2-bf6c-620082bbcbfc,1612343082.735552,21 x 1,"{'country': 'EIRE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
a6d21b95-f7bc-42af-8e94-fd03505529f2,1612343082.97365,21 x 1,"{'country': 'Germany', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01
ca748c0c-d9d5-4cef-b55b-60573a449932,1612343083.103916,21 x 1,"{'country': 'Portugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:01
787c4732-d2c9-4da3-bd3b-ef6bf913739f,1612343083.238034,21 x 1,"{'country': 'Netherlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:01
ce8ca6be-01cb-4b5d-8e75-144501c7632d,1612343083.355113,21 x 1,"{'country': 'Spain', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:02
7493b050-6297-459a-8523-90814e710659,1612343083.4827251,21 x 1,"{'country': 'Norway', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:02
6215d532-480a-47c5-bfb9-edb7fc0e5fc3,1612343083.593151,21 x 1,"{'country': 'Switzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
ae7a5419-6f4d-4eee-91d3-4774c1c2ec4b,1612343093.547681,21 x 1,"{'country': 'United Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
c209b3d7-a2da-46f5-b7c6-efce72552a76,1612343093.660616,21 x 1,"{'country': 'France', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01

8d23ceda-e0bd-48f2-80a4-60e5b1d11fcd,1612343093.786195,21 x 1,"{'country': 'Be
lgium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
2c384fcd-aeae-42a7-ac78-9914c4d941db,1612343093.903702,21 x 1,"{'country': 'EI
RE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
237feacc-5498-48da-8dd1-b4538c399e05,1612343094.009823,21 x 1,"{'country': 'Ge
rmany', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01
aaa183a6-76fc-4a82-8915-b3a27fbb4905,1612343094.11181,21 x 1,"{'country': 'Por
tugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:01
c1ffe1f8-344b-4cc7-937a-95ce2f493adf,1612343094.214445,21 x 1,"{'country': 'Ne
therlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:01
b23e9edd-2224-497b-9e66-c671e1b8cb2e,1612343094.326433,21 x 1,"{'country': 'Sp
ain', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:01
d2e1c9c1-af3a-4b06-92db-838c069f3b5c,1612343094.43462,21 x 1,"{'country': 'Nor
way', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:01
80d537b6-0c5a-45f5-b003-dc8771f28882,1612343094.5349991,21 x 1,"{'country': 'S
witzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
d3ce7d8d-1d90-49fe-bebd-32860801d57d,1612343144.6771462,21 x 1,"{'country': 'U
nited Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
3ed2bfe7-2453-41b8-ad09-402179a84e5a,1612343144.816585,21 x 1,"{'country': 'Fr
ance', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
c51454a8-2a89-4d4f-a8a4-a384f49dc003,1612343144.926584,21 x 1,"{'country': 'Be
lgium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
c8305e76-1a75-450c-92eb-fb610e7e632d,1612343145.0391748,21 x 1,"{'country': 'E
IRE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
8f8fda85-ca41-41f2-b1a8-86f7e00585aa,1612343145.142408,21 x 1,"{'country': 'Ge
rmany', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01
4111a7e2-d3a0-4417-8b97-dd2fb370465c,1612343145.259265,21 x 1,"{'country': 'Po
rtugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:01
cd977c7b-19fe-4479-b1ed-e3d9e2472879,1612343145.365335,21 x 1,"{'country': 'Ne
therlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:01
d7610a2d-7e22-44d4-80ba-a626116583e0,1612343145.467755,21 x 1,"{'country': 'Sp
ain', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:01
b1aef43e-2cad-4dea-b0c5-2ecbcbaad79a,1612343145.574002,21 x 1,"{'country': 'No
rway', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:02
7ce1e562-5e46-4032-8a16-0cf9dc9e4ff6,1612343145.682577,21 x 1,"{'country': 'Sw
itzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
6c1cbc63-7b63-49d5-9a94-f19ad06dc4fe,1612346354.660695,5,"{'country': 'United
Kingdom', 'rmse': 35593.61, 'mape': 70, 'model': 'retrain'}",0.1,auto_arima,00

0:00:00
c16af613-5955-4e4a-9649-ae929175cfbd,1612346354.771971,5,"{'country': 'Spain',
'rmse': 180.49, 'mape': 106, 'model': 'retrain'}",0.1,auto_arima,000:00:00
3e36501b-a068-42d2-af99-23ec862feaf2,1612346354.872101,5,"{'country': 'Franc
e', 'rmse': 118.94, 'mape': 17, 'model': 'retrain'}",0.1,auto_arima,000:00:00
036fca6d-0008-4a1e-b5d7-794a85e4e95a,1612346354.9724019,5,"{'country': 'German
y', 'rmse': 86.45, 'mape': 10, 'model': 'retrain'}",0.1,auto_arima,000:00:00
0def3c04-4d49-44fd-aff7-af89a8efbc8f,1612346355.076393,5,"{'country': 'EIRE',
'rmse': 1336.56, 'mape': 101, 'model': 'retrain'}",0.1,auto_arima,000:00:00
7119a868-58f1-480b-820b-ac1a7ccd1ca9,1612346355.189901,5,"{'country': 'Belgiu
m', 'rmse': 313.41, 'mape': 57, 'model': 'retrain'}",0.1,auto_arima,000:00:00
c1a3f5cd-3491-4f88-afbc-7de16e29d628,1612346355.293869,5,"{'country': 'Netherl
ands', 'rmse': 634.96, 'mape': 54, 'model': 'retrain'}",0.1,auto_arima,000:00:
00
f6de91a4-23a1-40f3-9c21-cf567a22cf52,1612346355.401623,5,"{'country': 'Norwa
y', 'rmse': 362.24, 'mape': 57, 'model': 'retrain'}",0.1,auto_arima,000:00:00
7af638e5-82c8-4c0d-8f6d-1776b31ee139,1612346355.522371,4,"{'country': 'Switzer
land', 'rmse': 334.58, 'mape': 30, 'model': 'retrain'}",0.1,auto_arima,000:00:
00
8735593b-0769-4f40-9de3-61cd19816b0e,1612346842.061891,5,"{'country': 'United
Kingdom', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,00
0:00:00
ca4ef5e3-8946-4c39-bc26-8bf5715cffe9,1612346842.242357,5,"{'country': 'Spain',
'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:00
cbb39876-0f77-4e3e-b54f-263bc2cec6a5,1612346842.378633,5,"{'country': 'Franc
e', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:0
0
e7e9ec29-d84a-421b-9b9b-d4c07c429852,1612346842.494345,5,"{'country': 'German
y', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:0
0
f9b4705e-0c4c-43a9-8e54-e350f0cd2303,1612346842.618728,5,"{'country': 'EIRE',
'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:00
8840c6ce-42cc-4da2-a08d-483bb652c263,1612346842.734169,5,"{'country': 'Belgiu
m', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:0
0
989cf399-d38d-4cc1-80a4-5b4ba4c67252,1612346842.9073339,5,"{'country': 'Nether
lands', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:
00:01

f5ab16c8-4ba6-4784-a16f-48fae445b14c,1612346843.044727,5,"{'country': 'Norwa
y', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:0
1
45a25edb-b284-453a-a860-aeb482107db9,1612346843.177227,4,"{'country': 'Switzer
land', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:0
0:01
7de961dc-9aff-454e-8dc0-07f7dfa5ab60,1612347232.438503,5,"{'country': 'United
Kingdom', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,00
0:00:00
ed5da32c-6332-4387-801b-589a34ec0a43,1612347232.583404,5,"{'country': 'Spain',
'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:00
c2a21d8b-fb0b-4288-b4d5-9dec116ad210,1612347232.694782,5,"{'country': 'Franc
e', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:0
0
abef6897-fcd1-4ed2-9472-df9917358caf,1612347232.798209,5,"{'country': 'German
y', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:0
0
5ee49995-ef73-483d-8537-836d2ad33795,1612347232.930468,5,"{'country': 'EIRE',
'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:00
81f31a7f-eb2f-4a49-a2fd-12c2b1eca088,1612347233.063612,5,"{'country': 'Belgiu
m', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:0
0
1058a264-91e6-4623-80f9-e69d824e64f2,1612347233.170519,5,"{'country': 'Netherl
ands', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:0
0:00
bd9c1d99-0654-46c5-86ed-63787d42674c,1612347233.279348,5,"{'country': 'Norwa
y', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:0
1
27ad75c1-2399-4872-aef4-eb6095f9b44f,1612349730.084003,21 x 1,"{'country': 'Un
ited Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:01
2008723f-a44e-4841-b047-221e9b44cd20,1612349730.213283,21 x 1,"{'country': 'Fr
ance', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:01
fe59e77e-7ec2-4cb2-a075-b36bff20d670,1612349730.3556879,21 x 1,"{'country': 'B
elgium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:01
e255b92c-4dfa-422e-8b59-9b6332b26d10,1612349730.485142,21 x 1,"{'country': 'EI
RE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:01
9d9a9012-e62a-42fe-9605-e2840bde582e,1612349730.6294641,21 x 1,"{'country': 'G
ermany', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:01

c6dee326-6e51-45be-8a28-b82bdbd2f506,1612349730.749404,21 x 1,"{'country': 'Portugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:02
883fc88c-4563-45de-a652-080ff8def55b,1612349730.879137,21 x 1,"{'country': 'Netherlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:02
6ac6e7a3-3e50-4848-a394-d03b2e315f6c,1612349731.0272121,21 x 1,"{'country': 'Spain', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:02
7d209e91-6a46-4626-a7c2-16133907d28c,1612349731.151126,21 x 1,"{'country': 'Norway', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:02
139c6748-d73a-407b-96a4-49899f223333,1612349731.264637,21 x 1,"{'country': 'Switzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:02
f3f894bf-7d18-4f41-81c6-a615e96512ce,1612352773.70977,21 x 1,"{'country': 'United Kingdom', 'rmse': 4805.55, 'mape': 19}",0.1,auto_arima,000:00:02
cd2c0e74-2004-4009-8234-27766b13bb3f,1612352773.874091,21 x 1,"{'country': 'France', 'rmse': 230.66, 'mape': 26}",0.1,auto_arima,000:00:02
5b27ae98-dbfe-48d4-8bda-7645d7748954,1612352774.083839,21 x 1,"{'country': 'Belgium', 'rmse': 156.52, 'mape': 48}",0.1,auto_arima,000:00:02
f3d6fcc4-d663-4868-85a3-f2fd471b27d3,1612352774.237306,21 x 1,"{'country': 'EIRE', 'rmse': 442.61, 'mape': 24}",0.1,auto_arima,000:00:02
8822ccbd-3607-426f-8318-f4ca09fd1d02,1612352774.3669279,21 x 1,"{'country': 'Germany', 'rmse': 385.81, 'mape': 36}",0.1,auto_arima,000:00:02
4158e533-7c4e-494f-8afe-cc00686edcb9,1612352774.5095072,21 x 1,"{'country': 'Portugal', 'rmse': 2895.75, 'mape': 68}",0.1,auto_arima,000:00:02
e6756543-19fa-4eb3-9067-b160d3b17fa3,1612352774.65222,21 x 1,"{'country': 'Netherlands', 'rmse': 358.91, 'mape': 958}",0.1,auto_arima,000:00:03
b181c303-922d-47ac-822d-bec12cd3c7e1,1612352774.799382,21 x 1,"{'country': 'Spain', 'rmse': 233.64, 'mape': 51}",0.1,auto_arima,000:00:03
bf0b825a-92e7-4a17-acb7-773c6899e87c,1612352774.943995,21 x 1,"{'country': 'Norway', 'rmse': 1344.39, 'mape': 100}",0.1,auto_arima,000:00:03
40e1924e-09ed-4490-809e-ef426e6041dd,1612352775.0941372,21 x 1,"{'country': 'Switzerland', 'rmse': 471.57, 'mape': 44}",0.1,auto_arima,000:00:03
78eae381-68c6-40c6-b546-5596850a971b,1612358167.614645,5,"{'country': 'United Kingdom', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:00
4219ff4e-90f4-4648-afda-889fa3759926,1612358167.794119,5,"{'country': 'Spain', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:00
0cd83c93-256b-465d-b576-f463f85c6ee1,1612358167.91309,5,"{'country': 'France', 'rmse': 'NAN', 'mape': 'NAN', 'model': 'retrain'}",0.1,auto_arima,000:00:00
b2f1943b-307c-4c0d-9e56-4f970291b350,1612358168.026426,5,"{'country': 'German

Business user simulation

Ideally, a front edn will be developed for business user to query the flask application. For the sake of this exercise, I shall simulate using api calls from python

1. The business user will submit a country and date to pick up forecast for next month
2. Additional forecasts are saved in log files and can be used in future to project for more than one month

Forecast

Given date 01/08/2019 and country as United Kingdom

In [45]:

```
import requests
from ast import literal_eval

'''
Part A: forecast using existing models
The script is expected to forecast for average revenue given a specific start dat
e.
test value = 01/08/2019
country = United Kingdom
'''

## data needs to be in dict format for JSON
query_data = dict({'date' :'01/08/2019','country':'united kingdom'})
query_type = 'dict'
request_json = {'query':query_data, 'type':query_type, 'mode':'test'}

## test the Docker API
port = 8082
#r = requests.post('http://0.0.0.0:{}/predict/'.format(port),json=query)
r = requests.get('http://0.0.0.0:{}/predict'.format(port),json=request_json)
#r = requests.get('http://0.0.0.0:{}/predict'.format(port))

response = literal_eval(r.text)

print(response)
```

```
{'Predrevenue': 23607.119, 'status': 200}
```

Test training with new data against the base model

In this section , we shall simulate training the forecast with new data.

Prerequisites.

```
Files will be in json format
Run the data ingestion and eda modules to create the top 10 countries data
Submit the file using flask as data.txt but in json format
ARIMA model will train on the data and report for each country MAPE, RMSE
The log files will have both train and predict logs
```

In [46]:

```
port = 8082

fileup= open("./data/top10countries-data.csv","rb")
response = requests.post("http://0.0.0.0:{}/train".format(port),files ={"file":fil
eup})
print (response.text)
```


```
{
  "country": {
    "0": "United Kingdom",
    "1": "France",
    "2": "Belgium",
    "3": "EIRE",
    "4": "Germany",
    "5": "Portugal",
    "6": "Netherlands",
    "7": "Spain",
    "8": "Norway",
    "9": "Switzerland"
  },
  "mape": {
    "0": 19,
    "1": 26,
    "2": 48,
    "3": 24,
    "4": 36,
    "5": 68,
    "6": 958,
    "7": 51,
    "8": 100,
    "9": 44
  },
  "rmse": {
    "0": 4805.55,
    "1": 230.66,
    "2": 156.52,
    "3": 442.61,
```

Model monitoring for drift

Using the data available in /data/cs_production , we shall update arima model 2 with additional data, predict the results and compare the output MAPE,RSME values against each base country model.

Normally graphs are use, but in my case, I shall use data structures such as dictionaries

In [47]:
```
#test with initial data , revisited
port = 8082

fileup= open("./data/top10countries-data.csv","rb")
response = requests.post("http://0.0.0.0:{}/train".format(port),files ={"file":fil
eup})
print(response.text)
```


```
{
  "country": {
    "0": "United Kingdom",
    "1": "France",
    "2": "Belgium",
    "3": "EIRE",
    "4": "Germany",
    "5": "Portugal",
    "6": "Netherlands",
    "7": "Spain",
    "8": "Norway",
    "9": "Switzerland"
  },
  "mape": {
    "0": 19,
    "1": 26,
    "2": 48,
    "3": 24,
    "4": 36,
    "5": 68,
    "6": 958,
    "7": 51,
    "8": 100,
    "9": 44
  },
  "rmse": {
    "0": 4805.55,
    "1": 230.66,
    "2": 156.52
```

```
      "2": 130.32,
      "3": 442.61,
      "4": 385.81,
      "5": 2895.75,
      "6": 358.91,
      "7": 233.64,
      "8": 1344.39,
      "9": 471.57
    }
  }
```

In [48]:
```
#test with initial data , revisited
port = 8082

fileup= open("./data/updtop10countries-data.csv","rb")
response = requests.post("http://0.0.0.0:{}/update".format(port),files ={"file":fi
leup})
print(response.text)
```


```
{
  "country": {
    "0": "United Kingdom",
    "1": "Spain",
    "2": "France",
    "3": "Germany",
    "4": "EIRE",
    "5": "Belgium",
    "6": "Netherlands",
    "7": "Norway",
    "8": "Switzerland"
  },
  "mape": {
    "0": 38,
    "1": 131,
    "2": 32,
    "3": 50,
    "4": 21,
    "5": 64,
    "6": 36,
    "7": 100,
    "8": 87
  },
  "rmse": {
    "0": 18453.19,
    "1": 424.42,
    "2": 551.04,
    "3": 526.57,
    "4": 235.34,
```

```
      "4": 253.34,
      "5": 344.82,
      "6": 341.2,
      "7": 826.75,
      "8": 380.8
    }
  }
```

In [49]:

```
import requests
from ast import literal_eval

'''
Part A: forecast using existing models
The script is expected to forecast for average revenue given a specific start dat
e.
test value = 01/08/2019
country = United Kingdom
before updating the model the results were
{'Predrevenue': 24301.823, 'status': 200}
'''

## data needs to be in dict format for JSON
query_data = dict({'date' :'01/08/2019','country':'united kingdom'})
query_type = 'dict'
request_json = {'query':query_data, 'type':query_type, 'mode':'prod'}

## test the Docker API
port = 8082
#r = requests.post('http://0.0.0.0:{}/predict/'.format(port),json=query)
r = requests.get('http://0.0.0.0:{}/predict'.format(port),json=request_json)
#r = requests.get('http://0.0.0.0:{}/predict'.format(port))

response = literal_eval(r.text)

print(response)
```

```
{'Predrevenue': 24301.823, 'status': 200}
```

Conclusion Model Monitoring for drift

As seen from the data , rmse for the countries have increased with additional data. For instance, RMSE at base model for United kinddom was 4805.3 which has increased to 18453.19.

In this case , I would go into the data again and see how i can optimise my model 2. Perhaps include exogenous variables which i was not able to generate due to my data points being limited to 21 and the model for generation requires a cycle of at least 24 data points

For that track, I will skip for now

In [ ]: