time-series-forecasting

February 3, 2024

[1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

from statsmodels.tsa.api import ExponentialSmoothing, SimpleExpSmoothing,Holt
from sklearn.linear_model import LinearRegression

import warnings
warnings.filterwarnings('ignore')
```

[2]:
```
df = pd.read_csv('gold_monthly_csv.csv')
```

[3]:
```
df.head()
```

[3]:

	Date	Price
0	1950-01	34.73
1	1950-02	34.73
2	1950-03	34.73
3	1950-04	34.73
4	1950-05	34.73

[4]:
```
df.shape
```

[4]: (847, 2)

1 EDA

[5]:
```
print(f"Date range of gold prices available from -{df.loc[:,'Date'][0]} to {df.
↪loc[:,'Date'][len(df)-1]}")
```

```
Date range of gold prices available from -1950-01 to 2020-07
```

[6]:
```
date = pd.date_range(start = "1/1/1950", end="8/1/2020", freq = "M")
date
```

[6]:
```
DatetimeIndex(['1950-01-31', '1950-02-28', '1950-03-31', '1950-04-30',
               '1950-05-31', '1950-06-30', '1950-07-31', '1950-08-31',
               '1950-09-30', '1950-10-31',
               …
               '2019-10-31', '2019-11-30', '2019-12-31', '2020-01-31',
               '2020-02-29', '2020-03-31', '2020-04-30', '2020-05-31',
               '2020-06-30', '2020-07-31'],
              dtype='datetime64[ns]', length=847, freq='M')
```

[7]:
```
df['month'] = date
df.drop('Date', axis=1, inplace=True)
df = df.set_index('month')
df.head()
```

[7]:

month	Price
1950-01-31	34.73
1950-02-28	34.73
1950-03-31	34.73
1950-04-30	34.73
1950-05-31	34.73

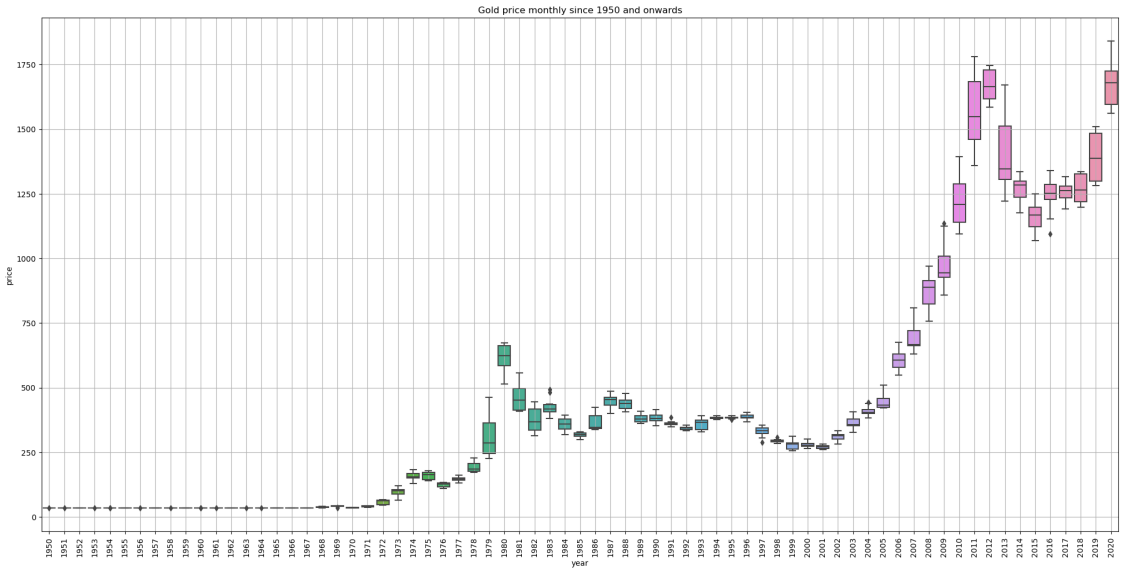
[8]:
```
df.plot(figsize=(15,6))
plt.title('Gold prices monthly since 1950 and onwards')
plt.xlabel('months')
plt.ylabel('price')
plt.grid()
plt.show()
```

[9]:
```
round(df.describe())
```

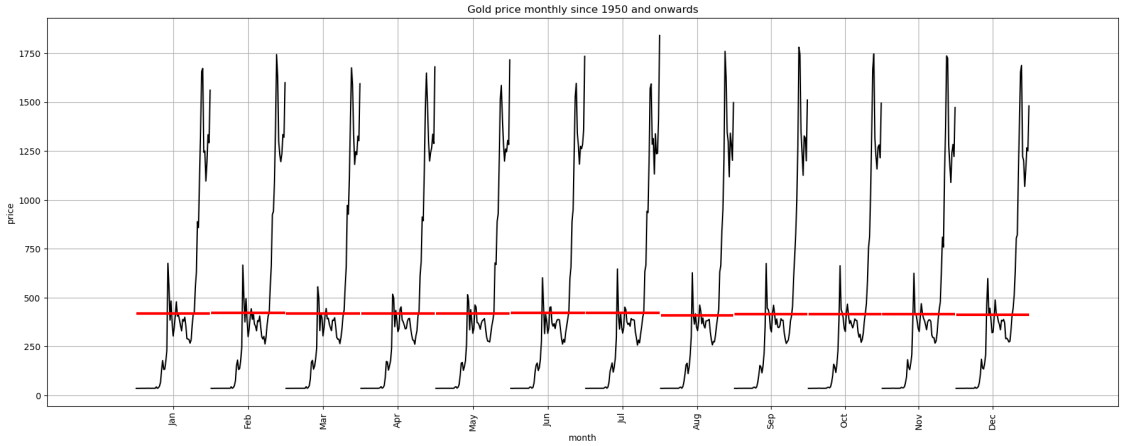
[9]:

	Price
count	847.0
mean	417.0
std	454.0
min	34.0
25%	35.0
50%	320.0
75%	447.0
max	1841.0

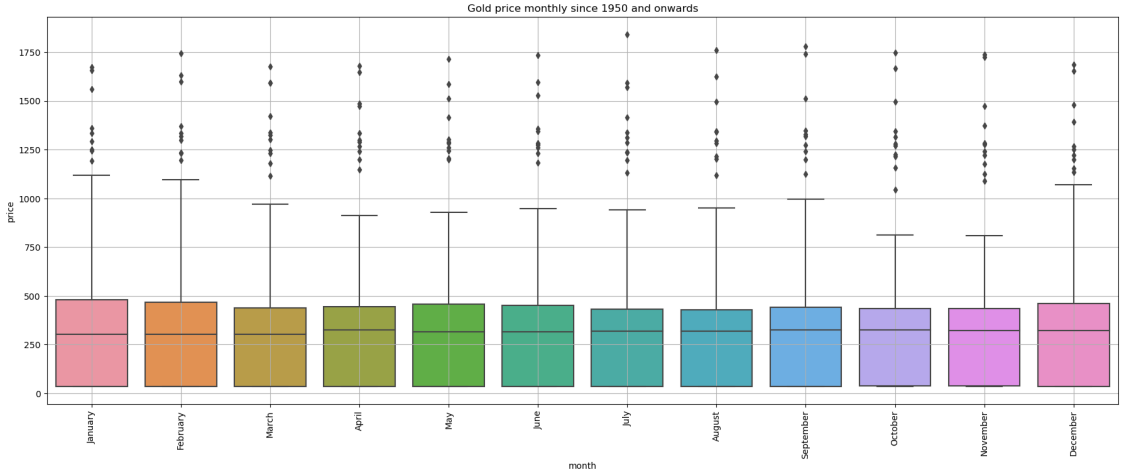
[10]:
```
_, ax = plt.subplots(figsize=(25,12))
sns.boxplot(x = df.index.year, y = df.values[:,0], ax=ax)
plt.title('Gold price monthly since 1950 and onwards')
plt.xlabel('year')
plt.ylabel('price')
plt.xticks(rotation=90)
plt.grid()
```



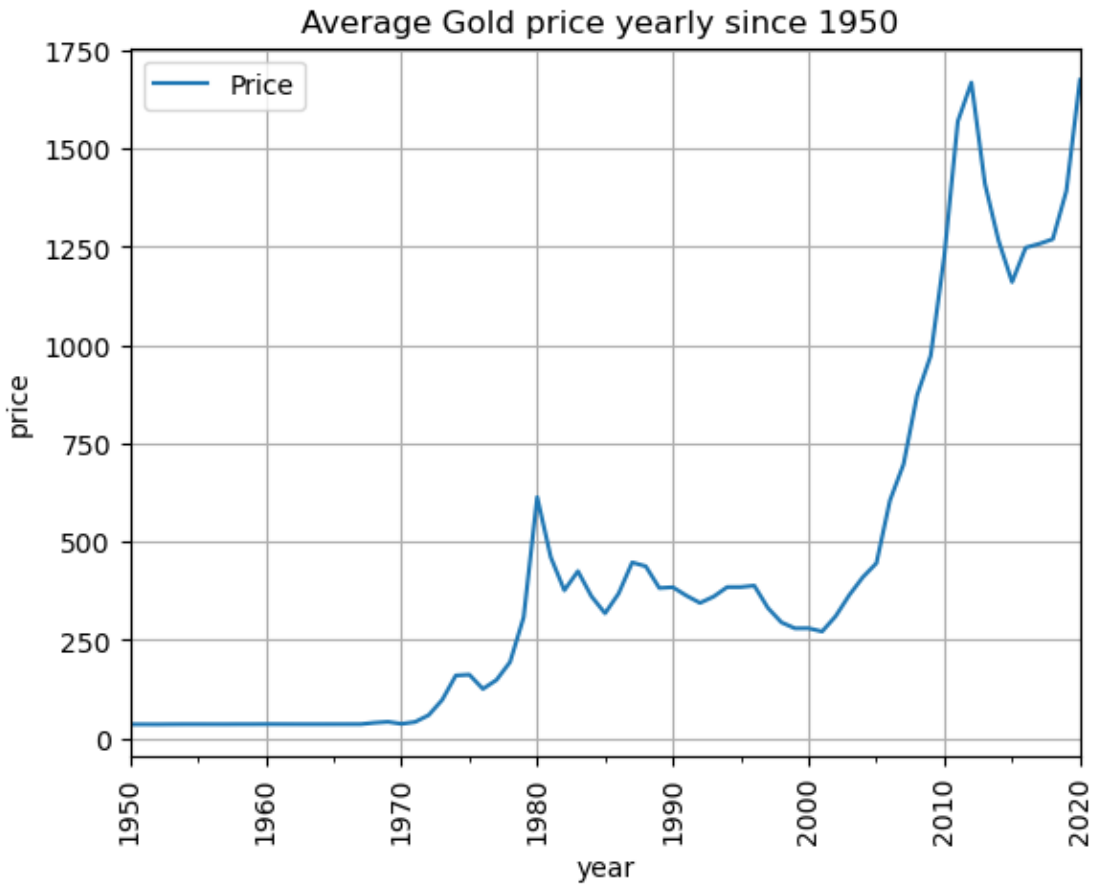
[11]:
```
from statsmodels.graphics.tsaplots import month_plot

fig, ax = plt.subplots(figsize=(22,8))
month_plot(df,ylabel='Gold price', ax=ax)
plt.title('Gold price monthly since 1950 and onwards')
plt.xlabel('month')
plt.ylabel('price')
plt.xticks(rotation=90)
plt.grid()
```



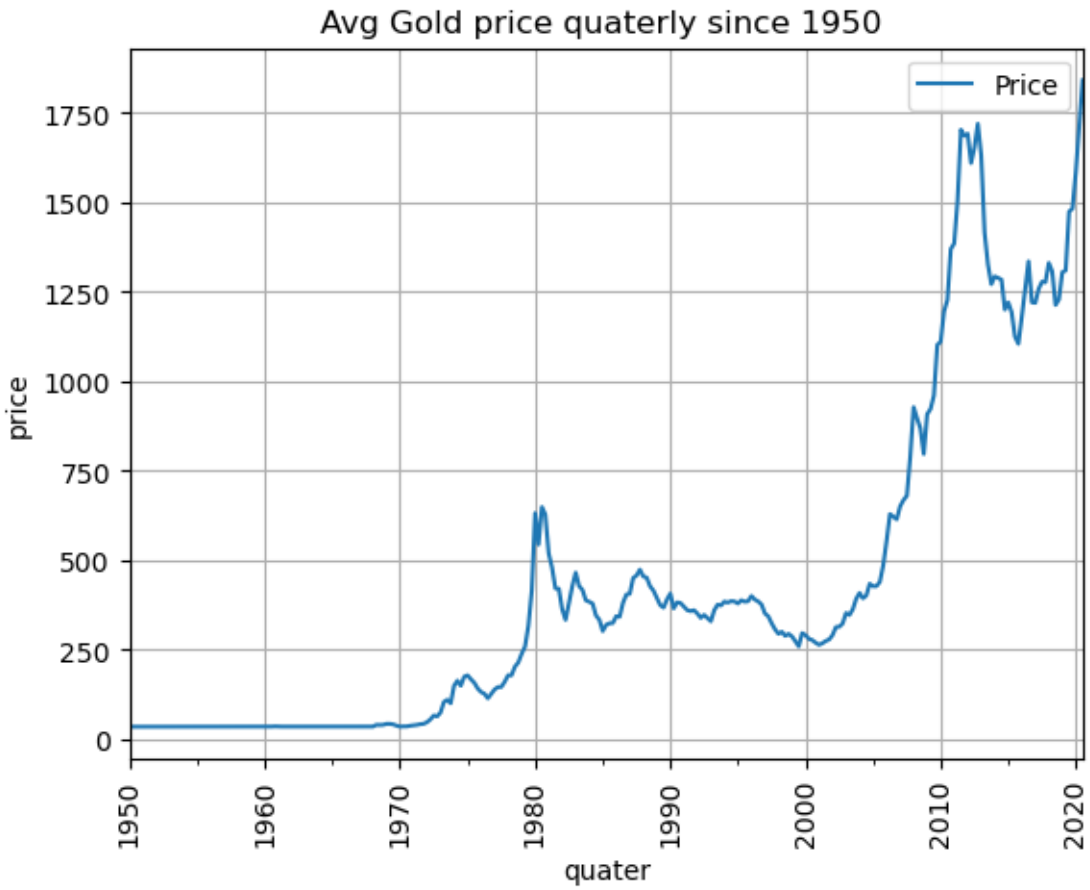
[12]:
```
_, ax = plt.subplots(figsize = (22,8))
sns.boxplot(x = df.index.month_name(), y = df.values[:,0], ax=ax)
plt.title('Gold price monthly since 1950 and onwards')
plt.xlabel('month')
plt.ylabel('price')
plt.xticks(rotation=90)
plt.grid()
```



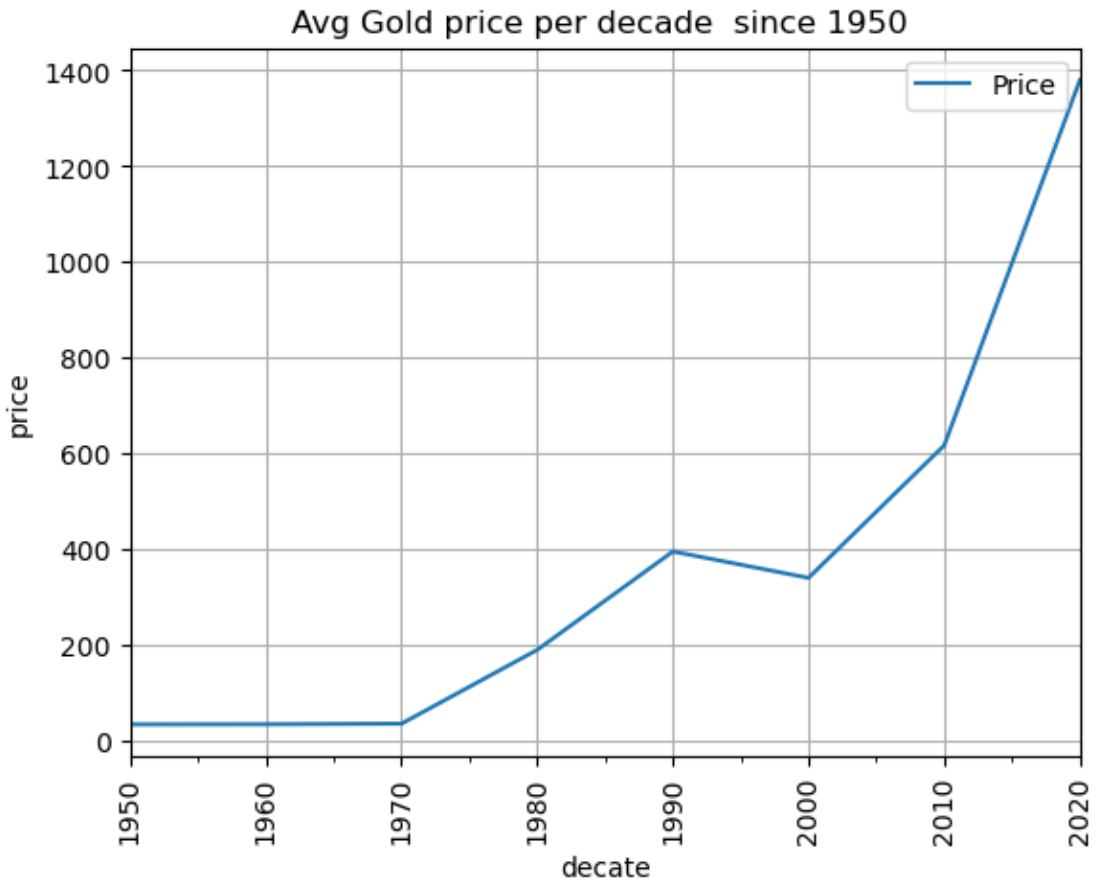
[13]:
```
df_yearly_sum = df.resample('A').mean()# A = Annual
df_yearly_sum.plot();
plt.title('Average Gold price yearly since 1950 ')
plt.xlabel('year')
plt.ylabel('price')
plt.xticks(rotation=90)
plt.grid()
```



[14]:
```
df_quaterly_sum = df.resample('Q').mean() # Q = Quaterly
df_quaterly_sum.plot()
plt.title('Avg Gold price quaterly since 1950 ')
plt.xlabel('quater')
plt.ylabel('price')
plt.xticks(rotation=90)
plt.grid()
```



[15]:
```
df_decade_sum = df.resample('10Y').mean() #10Y = 10 year
df_decade_sum.plot()
plt.title('Avg Gold price per decade  since 1950 ')
plt.xlabel('decate')
plt.ylabel('price')
plt.xticks(rotation=90)
plt.grid()
```



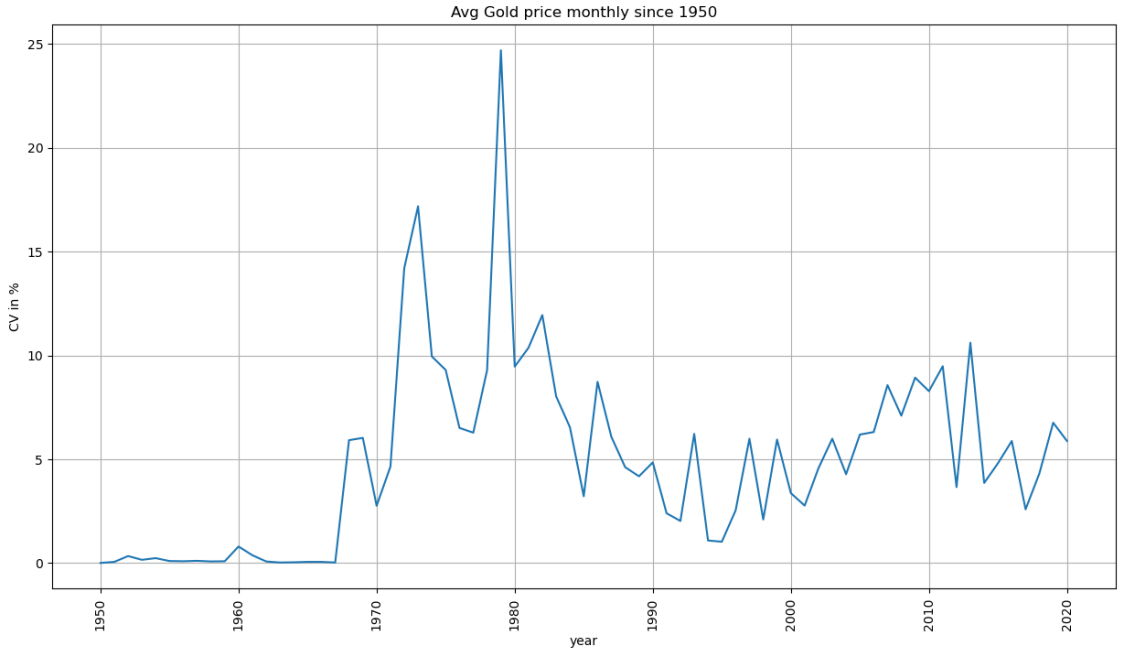
[16]:
```
df_1 = df.groupby(df.index.year).mean().rename(columns={'Price':'Mean'})
df_1 = df_1.merge(df.groupby(df.index.year).std().rename(columns={'Price':
 ↪'Std'}), left_index = True,
                  right_index = True)
df_1['Cov_pct'] = ((df_1['Std']/df_1['Mean'])*100).round(2)
df_1.head()
```

[16]:

month	Mean	Std	Cov_pct
1950	34.729167	0.002887	0.01
1951	34.717500	0.020057	0.06
1952	34.628333	0.117538	0.34
1953	34.879167	0.056481	0.16
1954	35.020000	0.082792	0.24

[17]:
```
fig, ax = plt.subplots(figsize=(15,8))
df_1['Cov_pct'].plot()
plt.title('Avg Gold price monthly since 1950')
plt.xlabel('year')
```

```
plt.ylabel('CV in %')
plt.xticks(rotation=90)
plt.grid()
```



[18]:
```
train = df[df.index.year <=2015]
test = df[df.index.year > 2015]
```

[19]:
```
print(train.shape)
print(test.shape)
```

```
(792, 1)
(55, 1)
```

[20]:
```
train['Price'].plot(figsize = (13,5), fontsize=15)
test['Price'].plot(figsize = (13,5), fontsize=15)
plt.grid()
plt.legend(['Training data','Test data'])
plt.show()
```

[21]:
```
train_time = [i+1 for i in range(len(train))]
test_time = [i+len(train)+1 for i in range(len(test))]
len(train_time), len(test_time)
```

[21]: (792, 55)

[22]:
```
lr_train = train.copy()
lr_test = test.copy()
```

[23]:
```
lr_train['time'] = train_time
lr_test['time'] = test_time
```

[24]:
```
lr= LinearRegression()
lr.fit(lr_train[['time']],lr_train['Price'].values)
```

[24]: LinearRegression()

[25]:
```
test_prediction_model1 = lr.predict(lr_test[['time']])
lr_test['forecast'] = test_prediction_model1
plt.figure(figsize=(14,6))
plt.plot(train['Price'], label = 'train')
plt.plot(test['Price'], label = 'test')
plt.plot(lr_test['forecast'], label = 'reg on time_test data')
plt.legend(loc = 'best')
plt.grid()
```

[26]:
```
def mape(actual, pred):
    return round((np.mean(abs(actual - pred)/ actual))*100,2)
```

[27]:
```
mape_model1_test = mape(test['Price'].values, test_prediction_model1)
print('MAPE is %3.3f'%(mape_model1_test),'%')
```

```
MAPE is 29.760 %
```

[28]:
```
result = pd.DataFrame({'Test Mape (%)':[mape_model1_test]},␣
 ↪index=['RegressionOnTime'])
result
```

[28]:

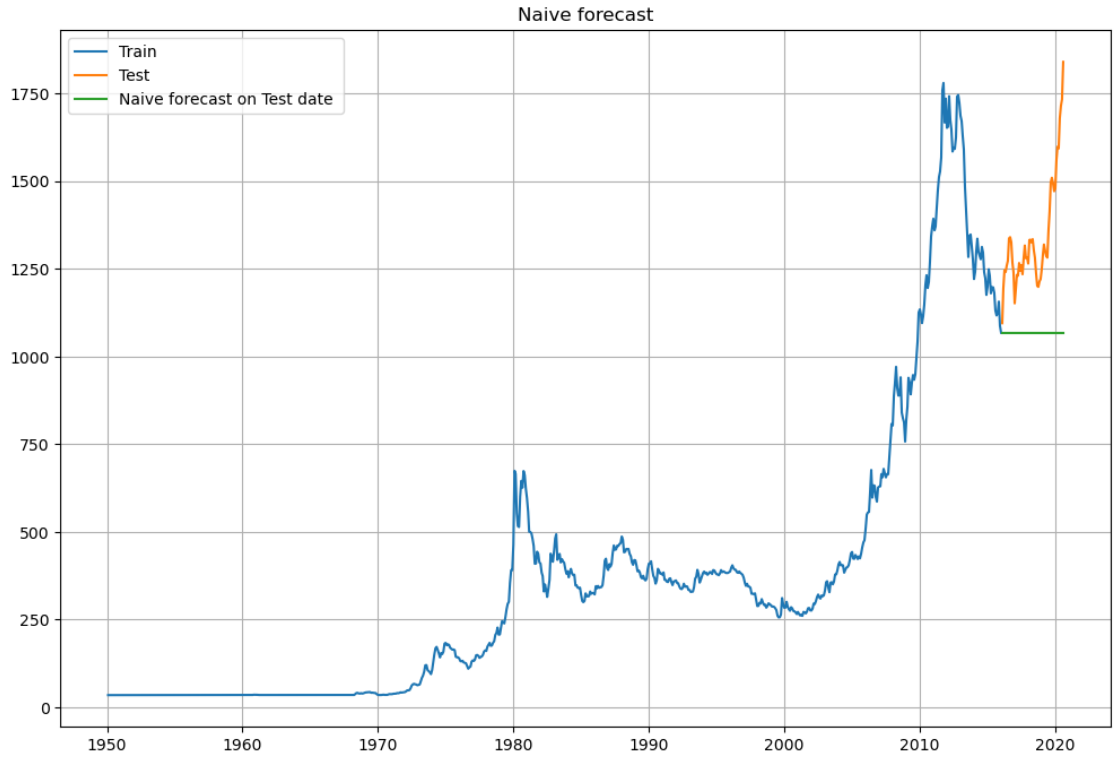
	Test Mape (%)
RegressionOnTime	29.76

[29]:
```
naive_train = train.copy()
naive_test = test.copy()
```

[30]:
```
naive_test['naive'] = np.asarray(train['Price'])[len(np.
 ↪asarray(train['Price']))-1]
naive_test['naive'].head()
```

[30]:
```
month
2016-01-31    1068.317
2016-02-29    1068.317
2016-03-31    1068.317
2016-04-30    1068.317
2016-05-31    1068.317
Name: naive, dtype: float64
```

[31]:
```
plt.figure(figsize=(12,8))
plt.plot(naive_train['Price'], label='Train')
plt.plot(test['Price'], label='Test')
plt.plot(naive_test['naive'], label='Naive forecast on Test date ')
plt.legend(loc='best')
plt.title('Naive forecast')
plt.grid()
plt.show()
```



[32]:
```
mape_model2_test = mape(test['Price'].values, naive_test['naive'].values)
print('For naive forecast on the test data, mape is %3.
  ↪3f'%(mape_model2_test),'%')
```

```
For naive forecast on the test data, mape is 19.380 %
```

[33]:
```
resultdf2 = pd.DataFrame({"Test mape (%)" : [mape_model2_test]},␣
  ↪index=['NaiveModel'])
result = pd.concat([result,resultdf2])
result
```

[33]:

	Test Mape (%)	Test mape (%)
RegressionOnTime	29.76	NaN
NaiveModel	NaN	19.38

[ ]:

[34]:
```
final_model = ExponentialSmoothing(df, trend='additive',
                                   seasonal='additive').fit(smoothing_level=0.4,␣
 ↪smoothing_trend=0.3,
                                                            smoothing_seasonal=0.
 ↪6)
```

[35]:
```
mape_fial_model = mape(df['Price'].values, final_model.fittedvalues)
print('MAPE: ', mape_fial_model)
```

```
MAPE:  17.24
```

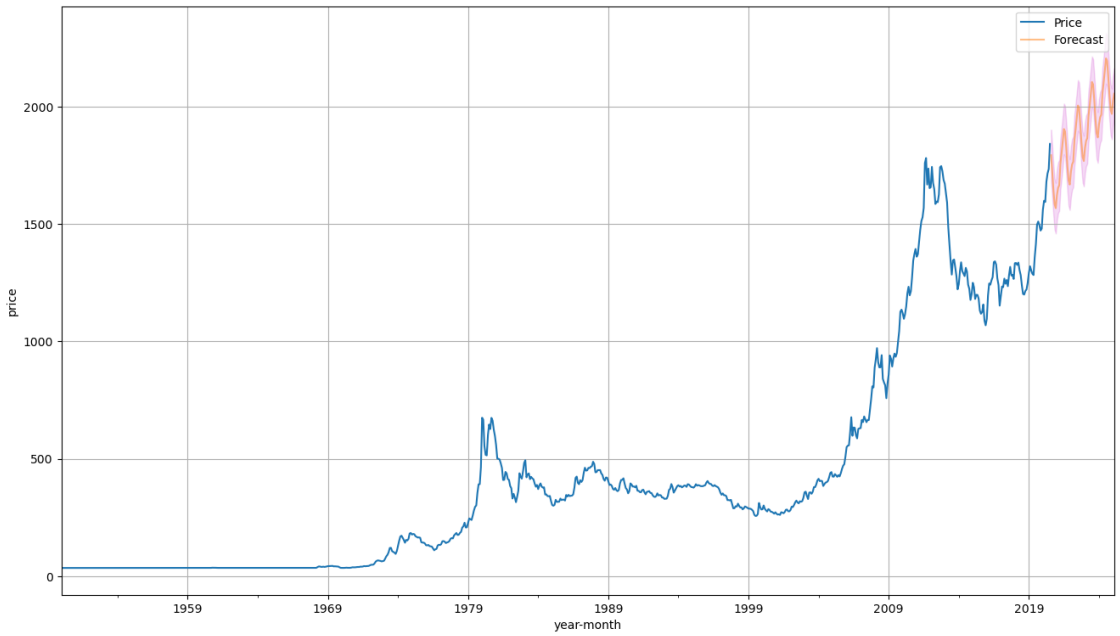
[36]:
```
prediction = final_model.forecast(steps=len(test))
```

[37]:
```
pred_df = pd.DataFrame({'lower_CI':prediction - 1.96*np.std(final_model.
 ↪resid,ddof=1),
                        'prediction': prediction,
                        'upper_CI': prediction + 1.96*np.std(final_model.
 ↪resid,ddof=1)})
pred_df.head()
```

[37]:

	lower_CI	prediction	upper_CI
2020-08-31	1684.720065	1792.871037	1901.022009
2020-09-30	1615.306077	1723.457050	1831.608022
2020-10-31	1538.567922	1646.718895	1754.869867
2020-11-30	1476.758600	1584.909572	1693.060545
2020-12-31	1459.327290	1567.478262	1675.629235

[38]:
```
axis = df.plot(label='Actual', figsize=(16,9))
pred_df['prediction'].plot(ax = axis, label='Forecast', alpha=0.5)
axis.fill_between(pred_df.index, pred_df['lower_CI'],pred_df['upper_CI'], color␣
 ↪= 'm', alpha= .15)
axis.set_xlabel('year-month')
axis.set_ylabel('price')
plt.legend(loc='best')
plt.grid()
plt.show()
```


Price
Forecast
2000
1500
1000
500
0
price
1959
1969
1979
1989
1999
2009
2019
year-month

[ ]:

[ ]: