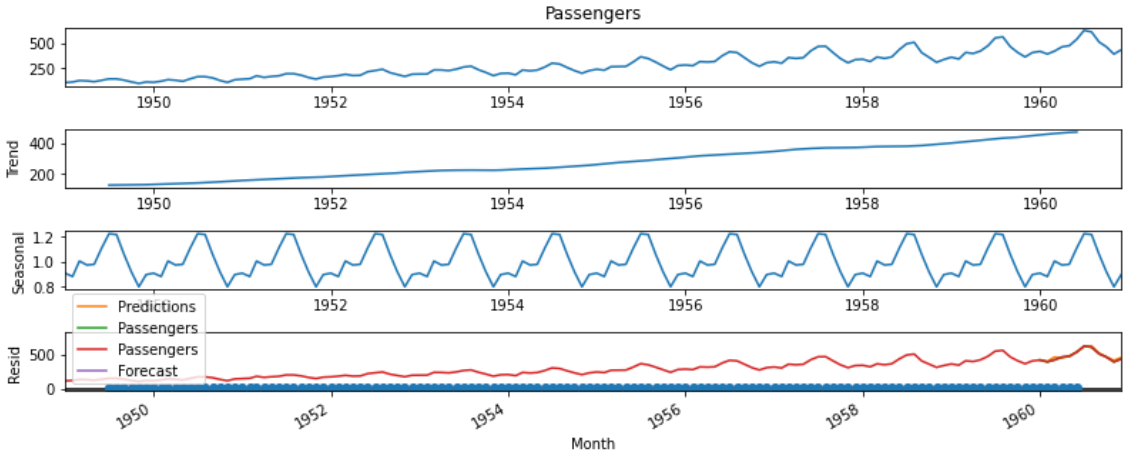
```
# Importing required libraries
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose

import warnings
warnings.filterwarnings('ignore')

# Read the AirPassengers dataset
airline = pd.read_csv('airline-passengers.csv',
                      index_col ='Month',
                      parse_dates = True)

# Print the first five rows of the dataset
airline.head()

# ETS Decomposition
result = seasonal_decompose(airline['Passengers'],
                            model ='multiplicative')

# ETS plot
result.plot()

# Split data into train / test sets
train = airline.iloc[:len(airline)-12]
test = airline.iloc[len(airline)-12:] # set one year(12 months) for testing

# Fit a SARIMAX(0, 1, 1)x(2, 1, 1, 12) on the training set
from statsmodels.tsa.statespace.sarimax import SARIMAX

model = SARIMAX(train['Passengers'],
                order = (0, 1, 1),
                seasonal_order =(2, 1, 1, 12))

result = model.fit()
result.summary()

start = len(train)
end = len(train) + len(test) - 1

# Predictions for one-year against the test set
predictions = result.predict(start, end,
                            typ = 'levels').rename("Predictions")

# plot predictions and actual values
predictions.plot(legend = True)
test['Passengers'].plot(legend = True)

# Load specific evaluation tools
```

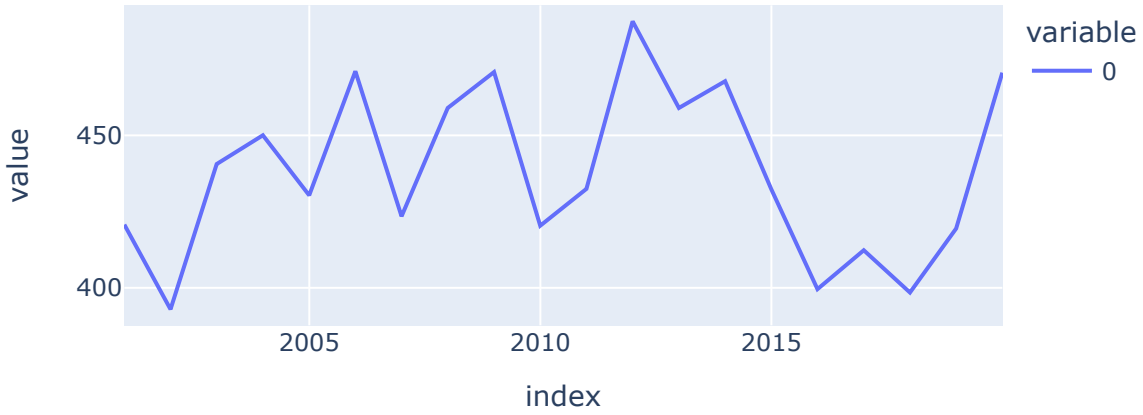
```
from sklearn.metrics import mean_squared_error
from statsmodels.tools.eval_measures import rmse

# Calculate root mean squared error
rmse(test["Passengers"], predictions)

# Calculate mean squared error
mean_squared_error(test["Passengers"], predictions)

# Train the model on the full dataset
model = model = SARIMAX(airline['Passengers'],
                        order = (0, 1, 1),
                        seasonal_order =(2, 1, 1, 12))
result = model.fit()

# Forecast for the next 3 years
forecast = result.predict(start = len(airline),
                          end = (len(airline)-1) + 3 * 12,
                          typ = 'levels').rename('Forecast')

# Plot the forecast values
airline['Passengers'].plot(figsize = (12, 5), legend = True)
forecast.plot(legend = True)
```

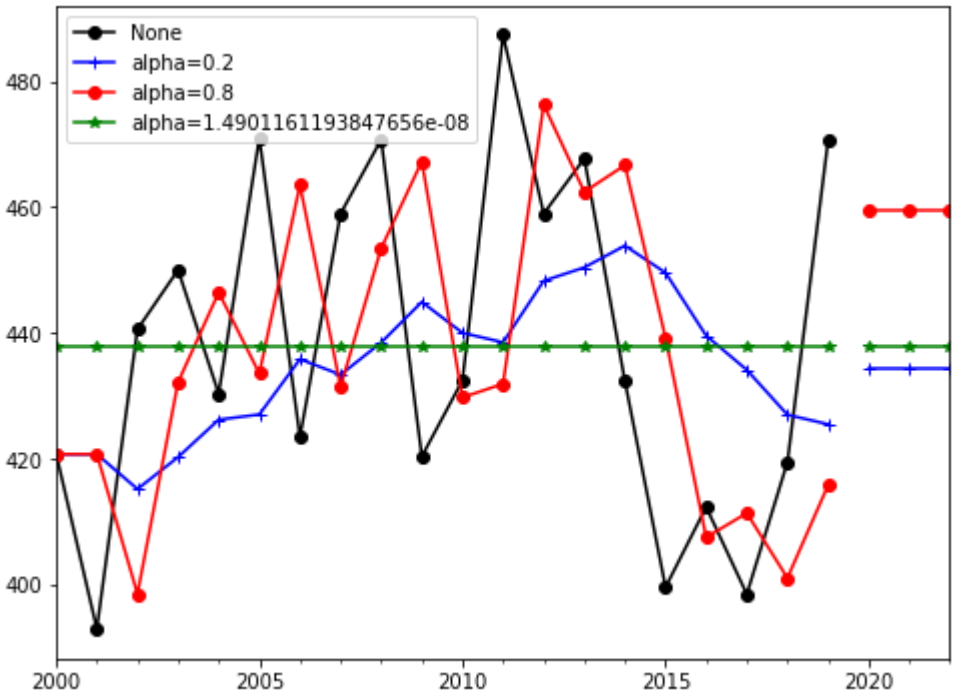
```
<Axes: xlabel='Month', ylabel='Resid'>
```




```
import pandas as pd
from statsmodels.tsa.api import SimpleExpSmoothing

df = [ 420.735,392.943, 440.593, 450.037, 430.345, 471.033, 423.456, 458.989, 470.767, 420.36
index= pd.date_range(start='2000', end='2020', freq='A')
data = pd.Series(df, index)
```

```
import plotly.express as px
fig = px.line(data,width=600, height=300)
fig.show()

#First Instance
ins1 = SimpleExpSmoothing(data).fit(smoothing_level=0.2,optimized=False)
ins_cast1 = ins1.forecast(3).rename('alpha=0.2')

#Second Instance
ins2 = SimpleExpSmoothing(data).fit(smoothing_level=0.8,optimized=False)
ins_cast2 = ins2.forecast(3).rename('alpha=0.8')

#Third Instance
ins3 = SimpleExpSmoothing(data).fit()
ins_cast3 = ins3.forecast(3).rename('alpha=%s'%ins3.model.params['smoothing_level'])

#After creating model we will visualize the plot
ax = data.plot(marker='o', color='black', figsize=(8,6), legend=True)

#Plot for alpha =0.2
ins_cast1.plot(marker='+', ax=ax, color='blue', legend=True)
ins1.fittedvalues.plot(marker='+', ax=ax, color='blue')

#Plot for alpha = 0.8
ins_cast2.plot(marker='o', ax=ax, color='red', legend=True)
ins2.fittedvalues.plot(marker='o', ax=ax, color='red')

#Plot for alpha=Optimized by statsmodel
ins_cast3.plot(marker='*', ax=ax, color='green', legend=True)
ins3.fittedvalues.plot(marker='*', ax=ax, color='green')

plt.show()
```





Colab paid products - Cancel contracts here