

Start coding or [generate](#) with AI.



Task: Choose a dataset with a time component and perform time series analysis

Embark on a time series analysis project using a dataset with a time component, specifically historical stock prices. The objective is to uncover patterns, trends, and insights from the temporal data, enabling a better understanding of stock price movements over time.

```
#import libraries

import pandas as pd
from prophet import Prophet
from matplotlib import pyplot
from matplotlib.pyplot import figure
from sklearn.metrics import mean_absolute_error
import plotly.express as px
import plotly.graph_objects as go

df=pd.read_csv('/content/long_data_.csv')
df.head()
```



	States	Regions	latitude	longitude	Dates	Usage	
0	Punjab	NR	31.519974	75.980003	02/01/2019 00:00:00	119.9	
1	Haryana	NR	28.450006	77.019991	02/01/2019 00:00:00	130.3	
2	Rajasthan	NR	26.449999	74.639981	02/01/2019 00:00:00	234.1	
3	Delhi	NR	28.669993	77.230004	02/01/2019 00:00:00	85.8	
4	UP	NR	27.599981	78.050006	02/01/2019 00:00:00	313.9	

```
df.shape

(16599, 6)
```

```
df['Dates'] = pd.to_datetime(df['Dates'])
```

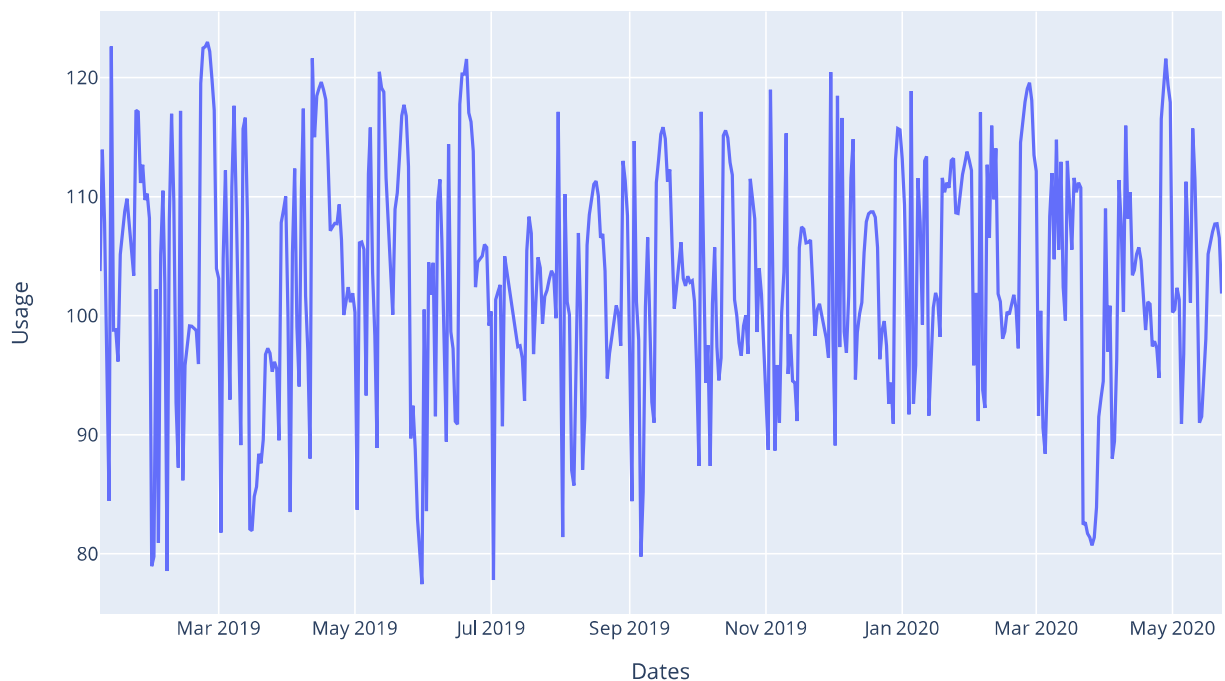
```
df = df.groupby(df['Dates'], as_index = False).mean()
df.head()
```

	Dates	latitude	longitude	Usage	
0	2019-01-07	23.17822	81.794533	103.736364	
1	2019-01-08	23.17822	81.794533	113.951515	
2	2019-01-09	23.17822	81.794533	107.836364	
3	2019-01-10	23.17822	81.794533	98.045455	
4	2019-01-11	23.17822	81.794533	84.463636	

```
df.shape

(498, 4)
```

```
df = df[['Dates','Usage']]
fig = px.line(df, x='Dates', y='Usage')
fig.show()
```



```
df.columns = ['ds','y']
df.head()
```

	ds	y	
0	2019-01-07	103.736364	
1	2019-01-08	113.951515	
2	2019-01-09	107.836364	
3	2019-01-10	98.045455	
4	2019-01-11	84.463636	

```
model=Prophet()
```

```
model.fit(df)
```



```
model.component_modes
```

```
{'additive': ['weekly',
'additive_terms',
'extra_regressors_additive',
'holidays'],
'multiplicative': ['multiplicative_terms', 'extra_regressors_multiplicative']}
```

```
future_dates = model.make_future_dataframe(periods=365,freq='d',include_history=True)
future_dates.shape
```

(863, 1)

future_dates.head()

	ds	
0	2019-01-07	
1	2019-01-08	
2	2019-01-09	
3	2019-01-10	
4	2019-01-11	

```
prediction=model.predict(future_dates)
prediction.head()
```

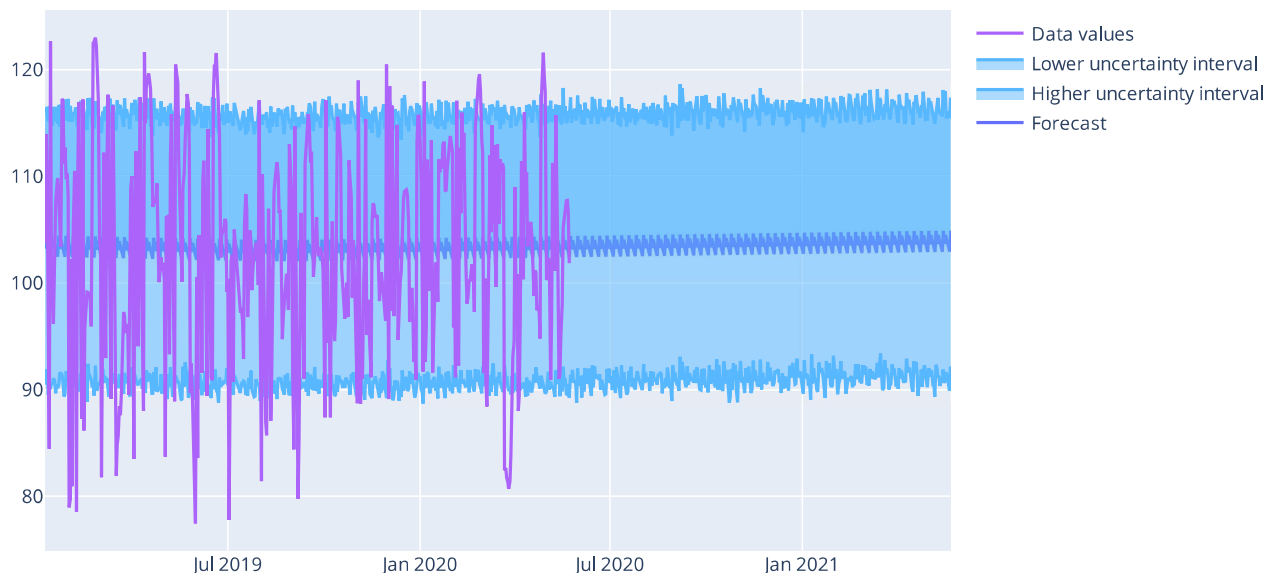
	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	ad
0	2019-01-07	103.380111	90.459822	116.431464	103.380111	103.380111	-0.184882	-0.184882	
1	2019-01-08	103.377977	91.897656	116.328942	103.377977	103.377977	0.650408	0.650408	
2	2019-01-09	103.375844	91.175910	115.699504	103.375844	103.375844	0.151419	0.151419	
3	2019-01-10	103.373710	89.499716	115.164709	103.373710	103.373710	-0.293240	-0.293240	
4	2019-01-11	103.371577	90.476944	116.565652	103.371577	103.371577	-0.470830	-0.470830	
	ds	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	ad
0	2019-01-07	103.380111	90.425593	115.456817	103.380111	103.380111	-0.184882	-0.184882	
1	2019-01-08	103.377977	90.969717	116.232757	103.377977	103.377977	0.650408	0.650408	
2	2019-01-09	103.375844	91.046321	116.504051	103.375844	103.375844	0.151419	0.151419	
3	2019-01-10	103.373710	89.957171	116.434679	103.373710	103.373710	-0.293240	-0.293240	
4	2019-01-11	103.371577	90.802180	114.206728	103.371577	103.371577	-0.470830	-0.470830	

```

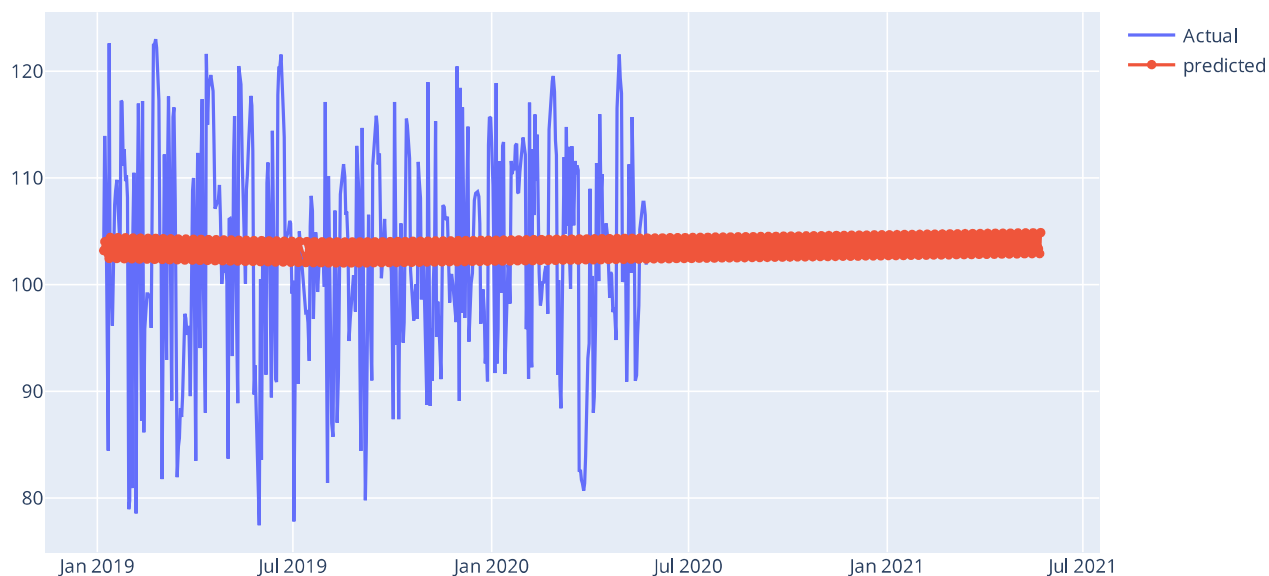
trace_open = go.Scatter(
    x = prediction["ds"],
    y = prediction["yhat"],
    mode = 'lines',
    name="Forecast"
)
trace_high = go.Scatter(
    x = prediction["ds"],
    y = prediction["yhat_upper"],
    mode = 'lines',
    fill = "tonexty",
    line = {"color": "#57b8ff"},
    name="Higher uncertainty interval"
)
trace_low = go.Scatter(
    x = prediction["ds"],
    y = prediction["yhat_lower"],
    mode = 'lines',
    fill = "tonexty",
    line = {"color": "#57b8ff"},
    name="Lower uncertainty interval"
)
trace_close = go.Scatter(
    x = df["ds"],
    y = df["y"],
    name="Data values"
)
#make list for all three scattle objects.
data = [trace_open,trace_high,trace_low,trace_close]
# Construct a new Layout object.
#title - It will display string as a title of graph
layout = go.Layout(title="Power consumption forecasting")

fig = go.Figure(data=data)
fig.show()

```



```
fig = go.Figure([go.Scatter(x=df['ds'], y=df['y'],mode='lines',
                           name='Actual')])
#You can add traces using an Express plot by using add_trace
fig.add_trace(go.Scatter(x=prediction['ds'], y=prediction['yhat'],
                        mode='lines+markers',
                        name='predicted'))
fig.show()
```



```
#Return a Numpy representation of the DataFrame.
```

```
y_true = df['y'].values
```

```
#Here we have specified [:498] because in y_true we have 498 data points so for comparing both series we need equal
```

```
y_pred = prediction['yhat'][:498].values
```

```
#Parameters:
```

```
#y_truearray-like of shape = (n_samples)
```

```
#Ground truth (correct) target values.
```

```
#y_predarray-like of shape = (n_samples)
```

```
#Estimated target values.
```

```
mae = mean_absolute_error(y_true, y_pred)
```

```
print('MAE: %.3f' % mae)
```

```
MAE: 7.910
```

```
#Optimizing the model for better forecasting
```

```
model1=Prophet(daily_seasonality=True).add_seasonality(name='yearly',period=365,fourier_order=70)
```

```
model1.fit(df)
```

```
model1.component_modes
```

```
{'additive': ['yearly',
              'weekly',
              'daily',
              'additive_terms',
              'extra_regressors_additive',
              'holidays'],
 'multiplicative': ['multiplicative_terms', 'extra_regressors_multiplicative']}
```

```
future_dates1=model1.make_future_dataframe(periods=365)
```

```
prediction1=model1.predict(future_dates1)
```

```
from sklearn.metrics import mean_absolute_error
y_true = df['y'].values
y_pred = prediction1['yhat'][:498].values
mae = mean_absolute_error(y_true, y_pred)
print('MAE: %.3f' % mae)
```

```
MAE: 5.600
```

```
import plotly.graph_objects as go
fig = go.Figure([go.Scatter(x=df['ds'], y=df['y'],mode='lines',
                           name='Actual')])

fig.add_trace(go.Scatter(x=prediction1['ds'], y=prediction1['yhat'],
                        mode='lines+markers',
                        name='predicted'))

fig.show()
```

