Time series forecasting using Prophet library

```
#NVIDIA Stock predictions


#Importing the important libraries

import pandas as pd

import numpy as np

import plotly.express as px

!pip install prophet

import matplotlib.pyplot as plt

!pip install scikit-learn


#Loading and prepapring the dataset

df = pd.read_csv(r"C:\Users\vaibh\Downloads\Time Series Analysis - NVIDIA Stocks\NVDA Historical Data.csv")

df.head()

df.info()

df.describe()

df['Date'] = pd.to_datetime(df['Date'])


dfnew = df[["Date", "Price"]]


#Doing the train-test split

train_size = int(len(dfnew) * 0.8)

train = dfnew[:train_size]  # first 80%
```

```
test = dfnew[train_size:]   # last 20%

print("Train shape:", train.shape)
print("Test shape:", test.shape)

df.columns = df.columns.str.strip().str.lower()
# Now 'Date' becomes 'date', 'Price' becomes 'price'

#Plotting the graph initially
fig = px.line(df, x='date', y='price', title='NVIDIA Stock Price Over Time')
import plotly.io as pio
pio.renderers.default = 'browser'
fig.show()

train_df = train[["Date", "Price"]].copy()
train_df.rename(columns={"Date": "ds", "Price": "y"}, inplace=True)


#Calling prophet to make predictions
from prophet import Prophet
model = Prophet(daily_seasonality=True)
model.fit(train_df)

#Making predictions for next 51 days
future = model.make_future_dataframe(periods=73,freq="D")
forecast = model.predict(future)
```

```
#Plotting actual vs predicted graph
fig = px.line(df, x='date', y='price', title='Actual vs Predicted')
fig.add_scatter(x=forecast['ds'], y=forecast['yhat'], mode='lines', name='Predicted')

fig.show()

#Again calling prophet
from prophet import Prophet

model = Prophet(
   seasonality_mode="multiplicative",
   daily_seasonality=True,
   weekly_seasonality=True,
   changepoint_prior_scale=0.1
)
model.fit(train_df)

#Making predictions for next 51 days
future = model.make_future_dataframe(periods=73,freq="D")
forecast = model.predict(future)
#Checking the general forecast with upper and lower limits
model.plot(forecast)
#Checking the components
model.plot_components(forecast)
```

```
#Doing some Time Series Accuracy tests
from sklearn.metrics import mean_absolute_error


# Step 1: Prepare actual
actual = df[["date", "price"]].copy()
actual.rename(columns={"date": "ds", "price": "y"}, inplace=True)
actual.set_index('ds', inplace=True)


# Step 2: Prepare forecast
forecast.set_index('ds', inplace=True)


# Step 3: Keep only overlapping dates
common_dates = actual.index.intersection(forecast.index)


# Step 4: Align both series
actual_y = actual.loc[common_dates]['y']
predicted_y = forecast.loc[common_dates]['yhat']


# Step 5: Calculate MAE
mae = mean_absolute_error(actual_y, predicted_y)
print(f'MAE: {mae:.2f}')


avg_price = df['price'].mean()
print(f"Average Price: {avg_price:.2f}")
```