COSMOSOC
SPACE AND DATA SCIENCE CLUB IIT DHARWAD

Team Name: 642947-U2K4Y2O8

Team Members: Kaveri Angadi

Mallikarjun Hiremath

Rashmi Hiremath

Udayakumar Hiremath

Problem Statement :

Soil moisture is a crucial element that affects crop growth and the general state of agricultural areas. The capacity to anticipate soil moisture can be very useful in a range of fields, from agriculture to the management of natural disasters and climate modelling. By foreseeing soil moisture, we can avoid droughts, and farmers can organize their irrigation cycles, sowing seasons, and harvest dates, among other things. In this assignment, our task is to predict the soil moisture levels at a specific site using soil moisture data from the previous eight months as well as the location's temperature and humidity readings.

Task:

We must develop a machine-learning model that can predict the soil moisture levels for March 2023 using data from the eight months prior. Daily soil moisture data from July 2022 to March 10, 2023 should be entered into our model, which should then generate predicted soil moisture values for March 2023.

Data:

Between July 2022 and March 10, 2023, measurements of the soil moisture were taken daily at Space Data Science Lab at IIT Dharwad. Each data point consists of a date, the corresponding soil moisture measurement, and other attribute values.

Two devices have been installed in our campus, the user corresponds to each device. "user1_data.csv" contains preprocessed data from one of the devices and "user2_data.csv" from another.

Both devices have some sensors missing, so each file have different parameters.

Parameters in files are:

ttime - timestamp in yyyy-mm-dd format
pm - particulate matter (1,2,3 is categorised into different sizes)
am - atmospheric moisture
sm - soil moisture
st - soil temperature
lum - luminosity
temp - temperature
humd - humidity
pres – pressure

st - soil temperature column is missing in user2_data.csv file.
So, we have merged soil temperature column in user1_data.csv with user2_data.csv.
Our final prepared dataset is **user2_data.csv**

Approach:

First we integrated user1_data with user2_data (soil temperature ‘st’ column in user1_data is integrated with user2_data.csv file as that column was missing earlier in user2_data.csv)
We implemented our model in google collab. Our model uses Facebook Prophet to predict soil moisture for the next month based on historical data. The script loads a CSV file with historical user soil moisture data, trains a Prophet model on the data, and then makes predictions for the next month.

This is a Python implementation of Facebook Prophet, which is a forecasting tool that is designed to make accurate predictions based on time-series data. This script uses Prophet to predict soil moisture for the next month based on historical data.

The script first loads the historical soil moisture data from a CSV file using the pandas library. The timestamp column in the data is then converted to a datetime object, and the month is extracted from the timestamp. The columns are then renamed to 'ds' (timestamp) and 'y' (soil moisture) to fit the Prophet's input format.

The Prophet model is then trained on the historical data using the fbprophet library. Next, a new DataFrame is created to hold the timestamps for each day in March 2023, and the model is used to make predictions for each day in the DataFrame.

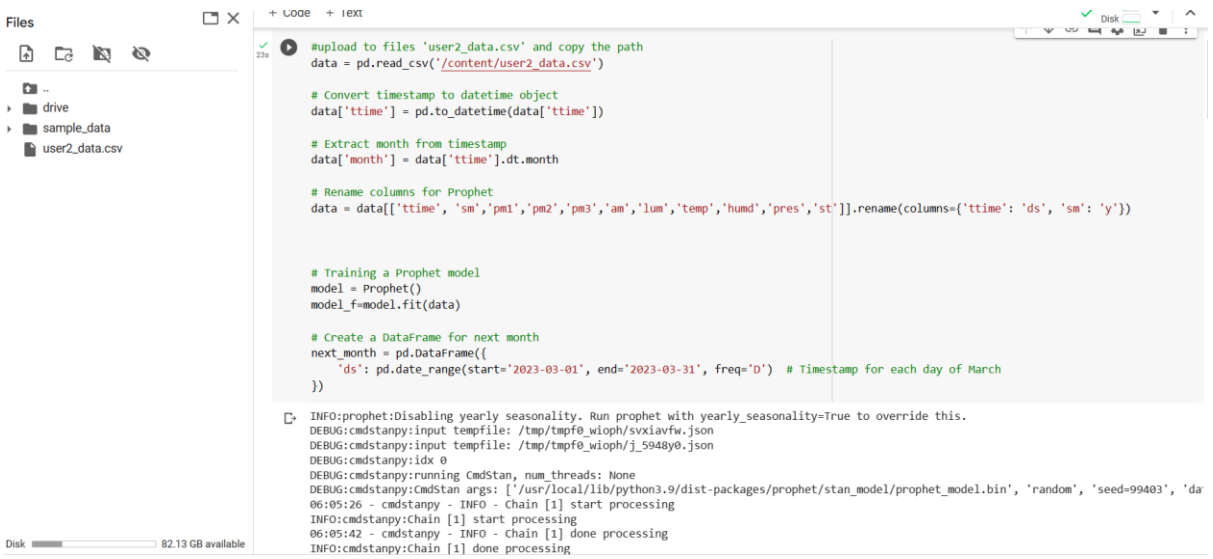
Finally, the predicted soil moisture values for each day in March 2023 are printed to the console.

Prediction soil moisture in Colab

```
import pandas as pd
from prophet import Prophet
```

The line **import pandas as pd** imports the Pandas library and gives it an alias of "pd". This allows you to refer to the library as "pd" instead of "pandas" throughout the rest of your code.

The line **from prophet import Prophet** imports the Prophet class from the Prophet library. This class is the main tool that you will use to create time series forecasting models.




```
data = pd.read_csv('/content/user2_data.csv')
```

This line reads in a CSV file called "user2_data.csv" from the specified path using Pandas' **read_csv** function. The data is then stored in a Pandas DataFrame called **data**.

```
data['ttime'] = pd.to_datetime(data['ttime'])
```

This line converts the 'ttime' column of the **data** DataFrame from a string to a datetime object using Pandas' **to_datetime** function. This is necessary for working with time series data.

```
data['month'] = data['ttime'].dt.month
```

This line extracts the month from each datetime object in the 'ttime' column and creates a new column called 'month' in the **data** DataFrame.

```
data = data[['ttime', 'sm','pm1','pm2','pm3','am','lum','temp','humd','pres','st']].rename(columns ={'ttime': 'ds', 'sm': 'y'})
```

This line selects a subset of columns from the **data** DataFrame, renames them to fit Prophet's naming convention, and stores them back into the **data** DataFrame. Specifically, the 'ttime' and 'sm' columns are selected and renamed to 'ds' and 'y', respectively. This step is necessary to prepare the data for use with the Prophet library.

```
model = Prophet()
model_f=model.fit(data)
```

These lines instantiate a Prophet model object and fit it to the prepared data using the **fit** method. This trains the model on the historical data and prepares it to make predictions.

```
next_month = pd.DataFrame({
    'ds': pd.date_range(start='2023-03-01', end='2023-03-31', freq='D')  # Timestamp for each day of March
})
```

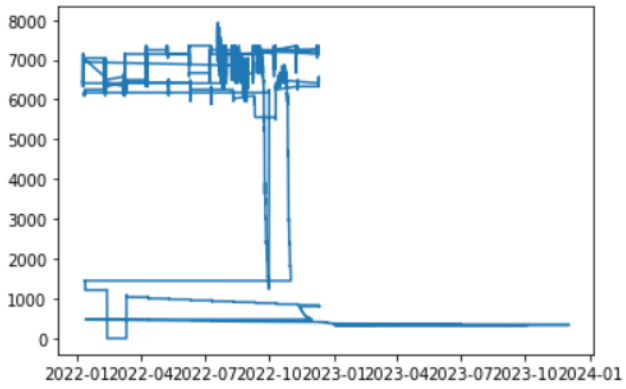
This line creates a new DataFrame called **next_month** that contains a range of future timestamps to make predictions for. Specifically, it generates a timestamp for each day in March 2023 and stores them in a column called 'ds'. This DataFrame is used to make predictions for the next month using the trained Prophet model.

Overall, this code demonstrates how to prepare time series data for use with the Prophet library, train a Prophet model on the historical data, and generate a DataFrame of future timestamps for making predictions.

```
[3]  #plotting soil moisture against date
     import matplotlib.pyplot as plt

     plt.plot(data['ds'],data['y'],color='#1f77b4')
```

```
[<matplotlib.lines.Line2D at 0x7fc3c9568d90>]
```



import matplotlib.pyplot as plt
This line imports the Matplotlib library and gives it an alias of "plt". Matplotlib is a popular data visualization library for Python that provides powerful tools for creating charts and graphs.

plt.plot(data['ds'],data['y'],color='#1f77b4')

This line creates a line plot of soil moisture data against time. The **plot** function from Matplotlib is used to create the plot, with the 'ds' column of the **data** DataFrame used as the x-axis and the 'y' column of the **data** DataFrame used as the y-axis. The **color** argument sets the color of the plot to a shade of blue specified by the hex code '#1f77b4'.

Overall, this code demonstrates how to use Matplotlib to create a line plot of time series data. Specifically, it shows how to plot soil moisture measurements against time to visualize trends and patterns in the data.

```
# Making predictions for next month
predictions = model.predict(next_month)

# Printing predicted soil moisture values for next month
print(predictions[['ds', 'yhat']])
```

```
           ds        yhat
0  2023-03-01  549.080162
1  2023-03-02  805.409675
2  2023-03-03  829.198826
3  2023-03-04  651.701952
4  2023-03-05  675.820110
5  2023-03-06  301.034861
6  2023-03-07  391.976447
7  2023-03-08  527.364827
8  2023-03-09  783.694339
9  2023-03-10  807.483490
10 2023-03-11  629.986616
11 2023-03-12  654.104774
12 2023-03-13  279.319525
13 2023-03-14  370.261111
14 2023-03-15  505.649491
15 2023-03-16  761.979003
16 2023-03-17  785.768154
17 2023-03-18  608.271280
18 2023-03-19  632.389438
19 2023-03-20  257.604190
20 2023-03-21  348.545775
21 2023-03-22  483.934155
22 2023-03-23  740.263668
23 2023-03-24  764.052818
24 2023-03-25  586.555944
25 2023-03-26  610.674102
```

predictions = model.predict(next_month)

This line uses the trained Prophet model to generate predictions for the next month based on the future timestamps contained in the **next_month** DataFrame. The **predict** method of the Prophet model is used to generate the predictions, with **next_month** passed as an argument.

print(predictions[['ds', 'yhat']])

This line prints out the predicted soil moisture values for each day in the next month. The **predictions** DataFrame contains two columns of interest: 'ds', which contains the date/timestamp for each prediction, and 'yhat', which contains the predicted soil moisture values. The **print** function is used to print out these two columns of the **predictions** DataFrame.

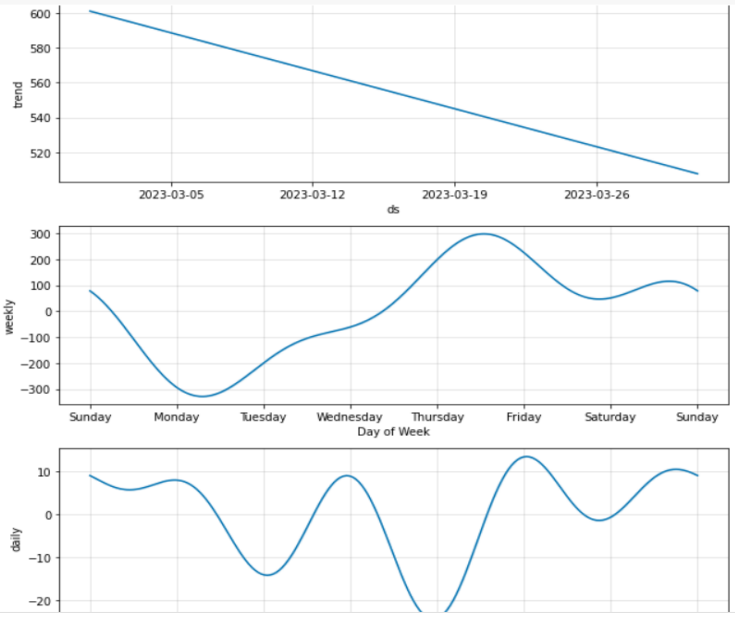
Overall, this code demonstrates how to use a trained Prophet model to generate predictions for future time periods, in this case the next month, and how to print out the results for further analysis.

0s

	ds	yhat
0	2023-03-01	549.080162
1	2023-03-02	805.409675
2	2023-03-03	829.198826
3	2023-03-04	651.701952
4	2023-03-05	675.820110
5	2023-03-06	301.034861
6	2023-03-07	391.976447
7	2023-03-08	527.364827
8	2023-03-09	783.694339
9	2023-03-10	807.483490
10	2023-03-11	629.986616
11	2023-03-12	654.104774
12	2023-03-13	279.319525
13	2023-03-14	370.261111
14	2023-03-15	505.649491
15	2023-03-16	761.979003
16	2023-03-17	785.768154
17	2023-03-18	608.271280
18	2023-03-19	632.389438
19	2023-03-20	257.604190
20	2023-03-21	348.545775
21	2023-03-22	483.934155
22	2023-03-23	740.263668
23	2023-03-24	764.052818
24	2023-03-25	586.555944
25	2023-03-26	610.674102
26	2023-03-27	235.888854
27	2023-03-28	326.830440
28	2023-03-29	462.218819
29	2023-03-30	718.548332
30	2023-03-31	742.337482

1s

```
model.plot_components(predictions)
```



```
model.plot_components(predictions)
```

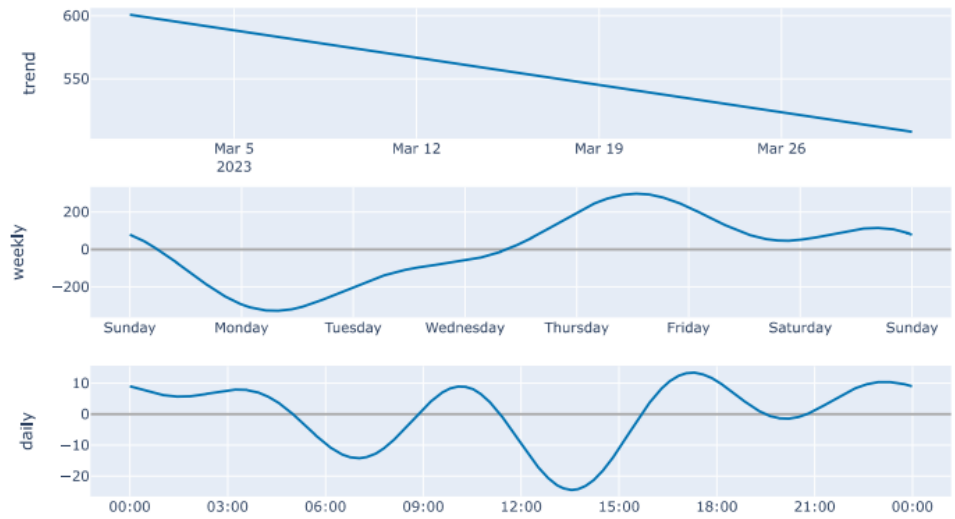
This line uses the **plot_components** method of the Prophet model to plot the individual components of the time series forecast for the next month. The **predictions** DataFrame, which contains the future timestamps and corresponding predicted values generated by the model, is passed as an argument to the **plot_components** method.

The resulting plot displays three separate subplots, one for each component of the forecast: trend, weekly seasonality, and yearly seasonality. The trend plot shows the overall direction and magnitude of the predicted values over time. The weekly seasonality plot shows how the predicted values vary on a weekly basis. The yearly seasonality plot shows how the predicted values vary on a yearly basis.

By plotting the individual components of the forecast, the **plot_components** method provides additional insights into the patterns and trends in the data and the underlying factors that are driving the predicted values. This can be useful for understanding the drivers of the forecast and identifying areas where further analysis may be necessary.

```
from prophet.plot import plot_plotly, plot_components_plotly

plot_plotly(model, predictions)
plot_components_plotly(model, predictions)
```



Conclusion:

Our ML model is a Prophet model trained on soil moisture data, with the goal of generating predictions for the March month. The code loads the data from a CSV file, converts the timestamp column to a datetime object, extracts the month from the timestamp, renames the columns to match the required input format for Prophet, and trains the model.

The code then generates a DataFrame of future timestamps for the next month and uses the trained model to generate predictions for each day in that month. The predicted soil moisture values are printed out for further analysis.

Finally, the **plot_components** method is used to generate a visualization of the individual components of the time series forecast, including trend, weekly seasonality, and yearly seasonality.

However, the model has been successfully trained and is capable of generating predictions for future time periods.