**CODE EXPLANATION**

* Pandas is a python software library which is used for manipulation and analysis of data and also used to read files such as csv, excel etc. a into data frame
* Matplotlib is a python plotting library.

Matplotlib rcParams

The rcParams of matplotlib helps to change the default **runtime configuration (rc)** of matplotlib.

It changes the default stylesheet of every plot element you create.

The code started by importing the necessary libraries such as matplotlib and pandas.

Then we assigned headers to the various read csv files

And then converted Year, Doy and Hour of the converted file to correct date-time format

Lastly the read data was used to plot the graph and saved the graph as a png file.

# importing required libraries

import numpy as np

import pandas as pd

import glob

import os

import matplotlib

import matplotlib.pyplot as plt

import datetime

import fnmatch

from matplotlib import dates

import matplotlib

font = {'family' : 'DejaVu Sans',

'weight' : 'bold',

'size' : 24}

matplotlib.rc('font', \*\*font)

matplotlib.rcParams["savefig.bbox"] ="tight"

matplotlib.rcParams["figure.titlesize"] =24

matplotlib.rcParams["figure.titleweight"] ="bold"

matplotlib.rc('text', usetex=True)

matplotlib.rcParams['text.latex.preamble'] = [r'\boldmath']

matplotlib.rcParams['axes.linewidth'] = 2

matplotlib.rcParams['xtick.major.width'] = 3

matplotlib.rcParams['axes.linewidth'] = 2

matplotlib.rcParams['xtick.major.width'] = 3

matplotlib.rcParams['ytick.major.width'] = 4

matplotlib.rcParams['ytick.right'] = True

matplotlib.rcParams["ytick.major.size"] = 9

matplotlib.rcParams.update({'legend.fontsize':22})

matplotlib.rcParams.update(matplotlib.rcParamsDefault)

low\_res\_header = ['YEAR', 'DOY','Hour','Scalar\_B','Proton\_Density N/cm^3','Kp\_index',"Dst\_index, nT","F10.7\_index"]

# Reading Solargeomag\_2018\_low.txt as a csv file

gdf = pd.read\_csv(r"Solargeomag\_2018\_low.txt" , delim\_whitespace=True, skiprows=13, names =low\_res\_header, na\_values =navalues)

#Convert YEAR< DOY and HOUR to Datetime Correctly

index = pd.to\_datetime(gdf["YEAR"] \* 100000 + gdf["DOY"] \* 100 + gdf["Hour"],

)

gdf.set\_index(index, inplace =True)

gdf["2018-03":"2018-06"]

gdf["UT"]=gdf.index

g2\_header =['Year',

'Day',

'Hour',

'Minute',

'Field\_magnitude\_average','BZ,nT(GSM)',

'Speed,km/s',

'Proton Density',

'Electricfield,mV',

'SYM/H']

# Reading solargeomag\_2018\_high.txt" as a csv file.

gdf2= pd.read\_csv(r"solargeomag\_2018\_high.txt" , sep ='\s+',

names = g2\_header, na\_values =['9999', '99999.9', '\*9999\*.99',

'99999','999.99','99999', '99.99000'], skiprows=15,

parse\_dates= {"UT" : ["Year","Day","Hour","Minute"]}, keep\_date\_col=True )

gdf2["UT"] = pd.to\_datetime(gdf2.UT, format ="%Y %j %H %M")

gdf2.set\_index(gdf2["UT"], inplace=True, drop= True)

gdf2["Time"] = (gdf2['Hour'].astype(str) + '.' + gdf2['Minute'].astype(str))

gdf2["Time"] =pd.to\_numeric(gdf2["Time"])

gdf2.set\_index("UT", inplace =True,drop=True)

gdf = pd.read\_csv(r"solargeomag\_2018\_low.txt" , sep ='\s+', skiprows=13, names =low\_res\_header, na\_values =navalues)

index = pd.to\_datetime(gdf["YEAR"] \* 100000 + gdf["DOY"] \* 100 + gdf["Hour"], format="%Y%j%H") #Convert YEAR< DOY and HOUR to Datetime Correctly

gdf.set\_index(index, inplace =True)

gdf[["Minute", 'Field\_magnitude\_average',

'BZ,nT(GSM)', 'Speed,km/s', 'Proton Density', 'Electricfield,mV',

'SYM/H', 'Time']]= gdf2[["Minute", 'Field\_magnitude\_average',

'BZ,nT(GSM)', 'Speed,km/s', 'Proton Density', 'Electricfield,mV',

'SYM/H', 'Time']]

"Create subplots"

from matplotlib import dates

from matplotlib.ticker import (MultipleLocator, AutoMinorLocator)

mon1=1

mon2=12

day1=1

day2=31

year=2018

months = dates.MonthLocator() #The matplotlib.ticker.MultipleLocator class is used for setting a tick for every integer

# multiple of a base within the view interval

monthsFmt = dates.DateFormatter('%b') #Formatting to month form

h\_fmt = dates.DateFormatter('%H')##Formatting to hour form

days = dates.DayLocator(interval=10)

dfmt = dates.DateFormatter('%d') ##Formatting to day form

hours =dates.HourLocator(byhour=range(0,24,6))

mins = dates.MinuteLocator(byminute=None, interval=30)

fig, axs = plt.subplots(5, 1, figsize=(25, 20), facecolor='w', edgecolor='k')

axs[4].grid(b=True, which='major', axis='x',linewidth = 1.5, color='black')

axs[4].grid(b=True, which='major', axis='y',linestyle ='--',linewidth = 1, color='black')

axs[4].grid(b=True, which='minor', axis='both',linestyle ='--', linewidth = 1, color='black')

axs[3].grid(b=True, which='major', axis='x',linewidth = 1.5, color='black')

axs[3].grid(b=True, which='major', axis='y',linestyle ='--',linewidth = 1, color='black')

axs[3].grid(b=True, which='minor', axis='both',linestyle ='--', linewidth = 1, color='black')

axs[2].grid(b=True, which='major', axis='x',linewidth = 1.5, color='black')

axs[2].grid(b=True, which='major', axis='y',linestyle ='--',linewidth = 1, color='black')

axs[2].grid(b=True, which='minor', axis='both',linestyle ='--', linewidth = 1, color='black')

axs[1].grid(b=True, which='major', axis='x',linewidth = 1.5, color='black')

axs[1].grid(b=True, which='major', axis='y',linestyle ='--',linewidth = 1, color='black')

axs[1].grid(b=True, which='minor', axis='both',linestyle ='--', linewidth = 1, color='black')

axs[0].grid(b=True, which='major', axis='x',linewidth = 1.5, color='black')

axs[0].grid(b=True, which='major', axis='y',linestyle ='--',linewidth = 1, color='black')

axs[0].grid(b=True, which='minor', axis='both',linestyle ='--', linewidth = 1, color='black')

# This section of the code is responsible for setting x-axis label on each graph of respective index

axs[4].xaxis.set\_major\_locator(months)

axs[3].xaxis.set\_major\_locator(months)

axs[2].xaxis.set\_major\_locator(months)

axs[1].xaxis.set\_major\_locator(months)

axs[0].xaxis.set\_major\_locator(months)

# This section is used for setting label for the y-axis with font size of 26 and making the font bold.

axs[0].set\_ylabel("$Vsw(km/s)$", size=26, weight ='bold')

axs[1].set\_ylabel("$|B|(nT)$", size=26, weight ='bold')

axs[2].set\_ylabel("$Sym/H(nT)$", size=26, weight ='bold')

axs[3].set\_ylabel("$Kp$", size=26, weight ='bold')

axs[4].set\_ylabel("$F10.7(SFU)$", size=26, weight ='bold')

# This section of the code is setting range for the y-axis.It takes one argument with a python list of two integers.

axs[1].set\_ylim([5,20])

axs[2].set\_ylim([-100,0])

axs[3].set\_ylim([0,60])

axs[4].set\_ylim([60,80])

# This section of the code is for plot the graph. The bar is responsible for plotting bar graph and the plot method is for

# plotting a linear graph.

axs[0].plot(gdf['Speed,km/s'],lw=2,color='black')

axs[1].plot(gdf['Scalar\_B'],lw=2,color="orange")

axs[3].bar(gdf.index,gdf['Kp\_index'],lw=2,color="yellow")

axs[2].plot(gdf2['SYM/H'],lw=2,color="red")

axs[4].bar(gdf.index, gdf['F10.7\_index'],lw=2,width =0.1,color="blue")

fig.text(0.5, 0.04, 'Month', ha='center',size = 27 ,weight = 'bold')# This line of code is for labeling the x-axis 'Month'.

fig.align\_ylabels()

# Saving the graph as a png file(picture)

plt.savefig(str(year)+"imf.png", dpi=500)# this line of code is responsible for saving the graph as an image(png file).

plt.show()# this line of code is responsible for showing the plotted graph to the screen.