DEVELOPMENT OF WEATHER ASSIMILATION SYSTEM

A CAPSTONE PROJECT REPORT

Submitted in partial fulfillment of the
requirement for the award of the
Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

by

ABHAY CHAUDHARY (19BCE7290)
HRIDESH SINGH (19BCE7102)
DHRITI RANJAN (19BCE7249)

Under the Guidance of

DR. ANUPAMA NAMBURU

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VIT-AP UNIVERSITY
AMARAVATI- 522237

DECEMBER 2022

CERTIFICATE

This is to certify that the Capstone Project work titled “**DEVELOPMENT OF WEATHER ASSIMILATION SYSTEM**” that is being submitted by **ABHAY CHAUDHARY (19BCE7290), HRIDESH SINGH (19BCE7102) and DHRITI RANJAN (19BCE7249)** are in partial fulfillment of the requirements for the award of Bachelor of Technology, is a record of bonafide work done under my guidance. The contents of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for award of any degree or diploma and the same is certified.

Dr. ANUPAMA NAMBURU
Guide

The thesis is satisfactory / unsatisfactory

Internal Examiner **External Examiner**

Approved by

PROGRAM CHAIR
B. Tech. CSE

DEAN
SCOPE

ACKNOWLEDGEMENTS

We would like to express our sincere gratitude to our guide, Dr. Anupama Namburu, for her invaluable guidance, support, and encouragement throughout the course of this project. We are also grateful to the Vellore Institute of Technology for providing the resources and support necessary to complete this project. We would also like to extend our sincere thanks to our Programme Chair Dr. Saroj Kumar Panigrahi for his guidance. We would like to extend our heartfelt gratitude to our parents, for their unconditional love, support, and encouragement throughout the course of this project.

We would also like to thank our colleagues for their insights and assistance during the development of this project. Their invaluable contributions have been a constant source of inspiration and have helped to make this project a success.

We are deeply grateful for the support of these individuals and organizations and would not have been able to complete this project without them.

ABSTRACT

Weather prediction systems using assimilation techniques are a valuable tool for forecasting the weather. These systems combine observations of the weather with computer models to produce accurate forecasts of the weather, particularly for weather events that are well-represented by the computer model. Assimilation techniques, such as three-dimensional variational (3D-Var) assimilation, four-dimensional variational (4D-Var) assimilation, and ensemble Kalman filter (EnKF) assimilation, allow the model to be continuously updated and improved as new observations become available, which can help to reduce the error in the model's forecast and improve the accuracy of the forecast. Weather prediction systems using assimilation techniques are used by a wide range of organizations, including government agencies, aviation companies, and energy utilities, to help them make informed decisions and plan for potential weather events. Despite their limitations, these systems are an important tool for improving our understanding of the weather and helping us to prepare for and respond to potential weather events. This research aims to assimilate the rainfall measurements estimated from multi-satellite (polar orbiting like GPM, CUBSAT and geostationary like INSAT-3D and 3DR) into our model using the advanced variational assimilation techniques to predict and differentiate rainfall measurements where ever necessary while comparing and using the trend between the past predictions(4C) and actual predictions to predict future rainfall estimates. Our proposed model uses a fusion of classic components and neural networks to produce highly accurate time series forecasts rapidly to minutely get the observations and give more optimized prediction we are able to achieve this by neural prophet model.

TABLE OF CONTENTS

List of Figures and Tables

CHAPTER 1

INTRODUCTION

A weather assimilation system is a computer system that combines data from a variety of sources, such as weather stations, satellite observations, and atmospheric models, to produce a detailed and accurate representation of the current state of the atmosphere. These systems are used to produce weather forecasts and to study the atmosphere and climate [1].

There are various assimilation techniques that can be used in a weather assimilation system, including three-dimensional variational (3D-Var) assimilation, four-dimensional variational (4D-Var) assimilation, and ensemble Kalman filter (EnKF) assimilation. These techniques differ in the way they incorporate the observations into the model and the level of detail they take into account when updating the model's forecast [2]. Weather assimilation systems typically use a technique called data assimilation, which combines data from observations with a mathematical model of the atmospheric processes to produce the best estimate of the current state of the atmosphere. This estimate is then used as the starting point for a forecast, which involves running the model forward in time to predict how the atmosphere will evolve over the coming days or weeks.

Weather assimilation systems are used by a wide range of organizations, including government agencies, aviation companies and energy utilities, to help them make informed decisions and plan for potential weather events [3-5]. These systems rely on a combination of advanced computer models and sophisticated assimilation techniques to produce accurate forecasts of the weather.

Weather assimilation systems can be used to produce forecasts for a wide range of applications, including aviation, agriculture, energy production and emergency management. They are an essential tool for understanding and predicting the weather, and are constantly being improved to increase their accuracy and efficiency.

Overall, weather assimilation systems play a crucial role in improving our understanding of the weather and helping us to prepare for and respond to potential weather events.

Rainfall prediction is useful for India in several ways:

- Improved agricultural planning: Accurate rainfall predictions can help farmers to plan their planting and harvesting activities, as well as make informed decisions about irrigation and other agricultural practices. This can help to improve crop yields and reduce waste.
- Enhanced public safety: Accurate rainfall predictions can help to alert the public to potential flood hazards, allowing them to take precautions to protect themselves and their property.
- Improved transportation: Accurate rainfall predictions can help transportation companies to plan for and respond to potential weather disruptions, such as storms and floods, reducing the risk of delays and cancellations.
- Enhanced energy production: Accurate rainfall predictions can help energy utilities to plan for and respond to changes in demand for energy due to weather conditions, improving the efficiency and reliability of energy production.
- Improved water management: Accurate rainfall predictions can help water management authorities to plan for and respond to changes in water availability, ensuring that there is sufficient water for agricultural, industrial, and domestic use.

Overall, accurate rainfall prediction is an important tool for helping individuals, businesses, and organizations in India to plan for and respond to potential weather events, improving public safety, efficiency, and economic productivity.

From the time of the first operational numerical weather prediction (NWP) models built the analysis was primarily made based on satellite images while the ability of numerical weather and climate models to predict is mostly based on correct model beginning circumstances initial conditions (ICs) and various other dependable variables [6-8]. Currently, weather data from Automated Weather Stations (AWS), terrestrial weather radar networks, and weather satellites are used to generate model ICs. A growing number of multidisciplinary studies are focusing on the technique of data incorporation, which combines meteorological readings with narrow forecasting to produce the beginning circumstances for NWP.

The actual system is really complicated. Before they can be used for assimilation, observations from a variety of earth observation and in-situ observation operating systems with its own set of error characteristics need to be thinned, corrected for skew, and subjected to intricate quality controls.

Brief (typically 1-6 h) forecasts, accompanied by their own immensely complicated stochastic and random errors, perpetuate measurements to the moment mostly in advanced data assimilation algorithms [9]. This fills in the gaps in which no observational data are available and gives the evaluation a climatically sound "noise field." Along with the attempt, the model provides the best approximation of the meteorological nation analysis—at a certain moment, modern meteorological data forecasting data assimilation algorithms incorporate a wide range of resources. Generic error metrics, the principles of meteorology, and knowledge from earlier predictions (a history and a first estimate) can all be used to complement estimations of meteorological parameters from (incomplete and imprecise) data and allow.

Regarding organizational linear regression methods must deliver the assessment (with a comparable number of levels of freedom) in a time window somewhere between 10 and 30 min, within just a few hours of a measurement being gathered [10]. They must assimilate billions of data. To produce thorough insight and future forecasts, new and upgraded NWP models, new analytical methods, and observation kinds are continuously developed. Yet, history over the previous few decades shows that these initiatives have only had a little influence on NWP skill, with average prediction gains of only a few percent every year (e.g., Simmons and Hollingsworth 2002).

1.1 Objectives

The following are the objectives of this project:

- Development of machine learning based satellite rain rate retrieval model using in-house prepared algorithms.
- Optimal short-range forecasting systems using neural prophet model setup to predict and differentiate between surface rainfall.

- Utilize commercially available tools, python-based algorithms, and state-of-the-art Artificial Intelligence methods for optimizing the rainfall prediction.

1.2 Background and Literature Survey

The development of weather prediction systems dates to the 19th century, when meteorologists began using basic physical principles, such as the conservation of energy and the laws of thermodynamics, to understand and predict the weather. In the early 20th century, weather prediction began to rely more heavily on the use of computer models, which allowed meteorologists to simulate the complex processes that govern the weather more accurately.

The use of assimilation techniques in weather prediction systems emerged in the 1970s, with the development of three-dimensional variational (3D-Var) assimilation. This technique allowed meteorologists to incorporate observations of the weather into computer models in a more systematic and accurate way, resulting in improved forecast accuracy [3].

Since then, various assimilation techniques have been developed and refined, including four-dimensional variational (4D-Var) assimilation and ensemble Kalman filter (EnKF) assimilation. These techniques have been widely adopted by weather prediction centers around the world and have played a crucial role in improving the accuracy of weather forecasts.

Today, weather prediction systems rely on a combination of advanced computer models and sophisticated assimilation techniques to produce accurate forecasts of the weather. These systems are used by a wide range of organizations, including government agencies, aviation companies, and energy utilities, to help them make informed decisions and plan for potential weather events [11].

The prediction skill of numerical weather and climate models is mainly dependent on accurate model's initial conditions (ICs). The generation of model ICs is tightly relay on weather observations from Automated Weather Stations (AWS), ground weather radar network, and weather satellites.

This research aims to assimilate the rainfall measurements estimated from multi-satellite (polar orbiting like GPM, CUBESAT and geostationary like INSAT-3D and 3DR) and rain-gauge blend product into the Weather Research and Forecasting (WRF) model using the advanced variational assimilation techniques to improve the model initial condition for prediction of severe convective systems over south-east coast of India [15].

The Weather Research and Forecasting model is a mesoscale numerical weather prediction system designed for both atmospheric research and operational forecasting applications.

Weather prediction systems use a combination of observations and computer models to forecast the weather. Assimilation techniques are used to incorporate observations into the computer models to improve the accuracy of the forecasts.

There are several different assimilation techniques that can be used, including:

1. Three-dimensional variational (3D-Var) assimilation: This technique minimizes the difference between the model's forecast and the observations by adjusting the model's initial conditions.
2. Four-dimensional variational (4D-Var) assimilation: This technique takes into account the evolution of the model's forecast over time and adjusts the model's initial conditions and forecast variables to minimize the difference between the model's forecast and the observations.
3. Ensemble Kalman filter (EnKF) assimilation: This technique uses a large ensemble of model forecasts to estimate the uncertainty in the model's forecast and adjust the forecast based on the observations.

The first part of the research proposal will be focused on developing a model to learn the trends in change in rainfall in mm during the past years. As there has been a drastic change in rainfall over the past years we can observe the trends and choose to use the data from the past decade or continue with the old data if it is consistent for better results [12-16]. We will also use these trends to make a relationship between the old data and data from the past decade to help our final prediction and increase accuracy.

The final part of this research proposal will be focused on assimilating the estimated surface rainfall rate from a hybrid machine learning system into the WRF model through the assimilation techniques to improve the prediction skill of short-term rainfall prediction at short-range time scales.

We are using prophet which is a procedure for forecasting time series data based on an additive model where nonlinear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data.

Here we are using neural prophet to determine the prediction as it is highly scalable, easy to use, and extensible, as it is built entirely in PyTorch and trained with standard deep learning methods. Neural Prophet introduces local context with support for auto-regression and lagged covariates [17-18].

Neural Prophet includes all the components from the original Prophet model: trend, seasonality, recurring events, and regressors. Further, Neural Prophet now also provides support for auto-regression and lagged covariates [19]. That's particularly relevant in the kinds of applications in which the near-term future depends on the current state of the system. The majority of time series forecasting exhibits those dynamics, evidenced in scenarios related to energy consumption, traffic patterns, air quality measures, and much more.

Neural Prophet improves on Prophet by addressing its key shortcomings: extensibility of the framework, missing local context for predictions and forecast accuracy.
Neural Prophet is highly scalable, easy to use, and extensible, as it is built entirely in PyTorch and trained with standard deep learning methods. Neural Prophet introduces local context with support for auto-regression and lagged covariates [20-22].

Neural Prophet improves forecast accuracy with a hybrid model, where some model components can be configured as neural networks.

CHAPTER 2

DEVELOPMENT OF WEATHER ASSIMILATION SYSTEM

In this chapter we will look at the procedures and techniques that are used to collect, analyze, and interpret data in order to test our hypothesis. It also describes the methods that were used in the study, including the research design, sample selection, data collection techniques, and data analysis methods. So, it basically describes the proposed system, working methodology, code snippets along with the description of the steps to be followed and other required details.

2.1 Proposed System

The following block diagram (figure) shows the architecture diagram of this project.

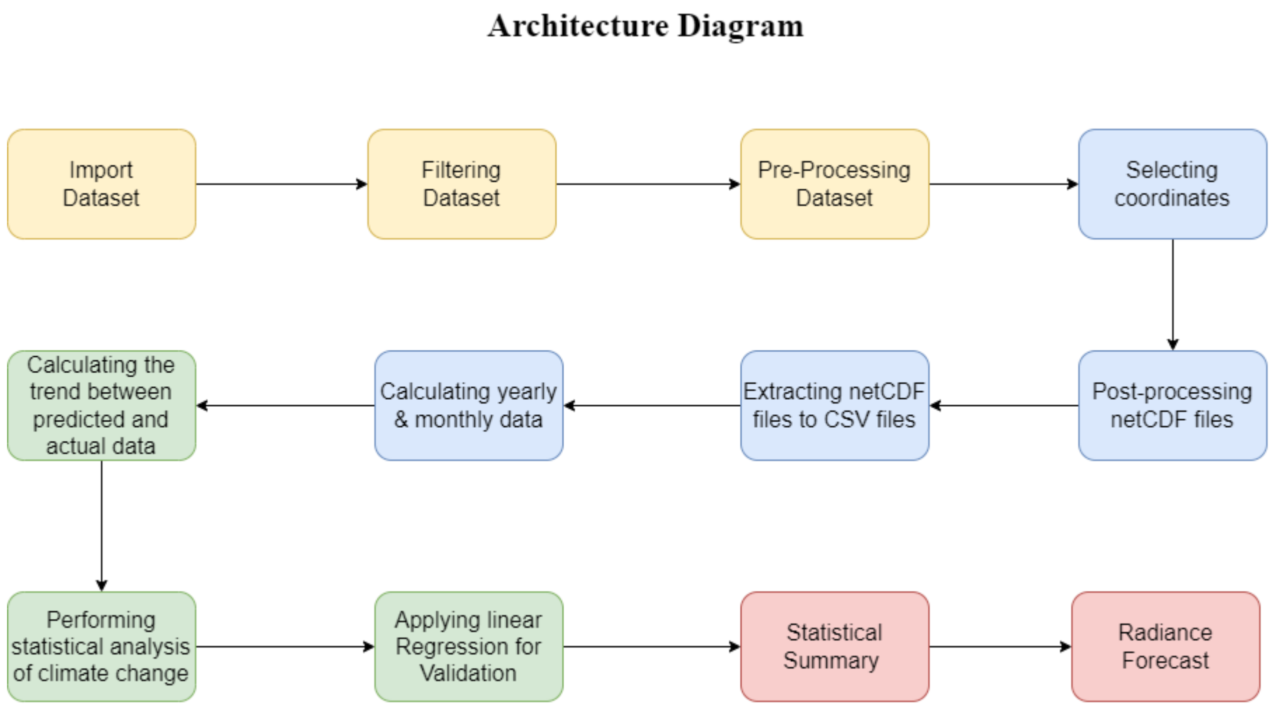

Architecture Diagram
Import Dataset
Filtering Dataset
Pre-Processing Dataset
Selecting coordinates
Post-processing netCDF files
Extracting netCDF files to CSV files
Calculating yearly & monthly data
Calculating the trend between predicted and actual data
Performing statistical analysis of climate change
Applying linear Regression for Validation
Statistical Summary
Radiance Forecast

Fig 1: Architecture Diagram

2.2 Working Methodology

The prediction skill of numerical weather and climate models is mainly dependent on accurate model's initial conditions (ICs). The generation of model ICs is tightly relay on weather observations from Automated Weather Stations (AWS), ground weather radar network, and weather satellites.

This research aims to assimilate the rainfall measurements estimated from multi-satellite (polar orbiting like GPM, CUBSAT and geostationary like INSAT-3D and 3DR) into our model using the advanced variational assimilation techniques to predict and differentiate rainfall measurements where ever necessary while comparing and using the trend between the past predictions(4C) and actual predictions to predict future rainfall estimates [23-26].

Our proposed model uses a fusion of classic components and neural networks to produce highly accurate time series forecasts rapidly to minutely get the observations and give more optimized predictions. We can achieve this by neural prophet model.

To develop a weather assimilation system, the following steps are typically followed:

1. Collect observations: Observations of the weather, such as temperature, humidity, and wind speed, are collected from a variety of sources, including weather stations, satellites, and aircraft.
2. Develop a computer model: A computer model is developed that simulates the physical processes that govern the weather, such as the movement of air and the transfer of heat and moisture.
3. Run the model: The model is run to generate a forecast of the weather.
4. Assimilate observations: The observations are incorporated into the model using one of the assimilation techniques mentioned above.

5. Produce the forecast: The updated model is used to produce a revised forecast of the weather.

6. Evaluate the forecast: The accuracy of the forecast is evaluated using various metrics, such as the root mean squared error and the mean absolute error.

7. Refine the model: If the forecast is not accurate enough, the model may be refined by adjusting the model's physical parameterizations or by incorporating additional observations.

2.3 About the Dataset

We introduce TerraClimate, a dataset of high spatial resolution (1/24°, 4-km) monthly climate and climatic water balance for all terrestrial surfaces worldwide from 1958 to 2020. In order to create a monthly dataset of precipitation, maximum and minimum temperatures, wind speed, vapour pressure, and solar radiation, TerraClimate uses climatically assisted interpolation, combining coarser resolution time varying (i.e., monthly) data from other sources with coarser resolution climatological normal from the WorldClim dataset. Additionally, TerraClimate generates monthly surface water balance statistics based on a water balance model that includes reference evapotranspiration, precipitation, temperature, and estimated plant extractable soil water capacity. When high spatial resolution, climate and climatic water balance data are needed for ecological and hydrological studies at the global scale, these data serve as crucial inputs. We used annual temperature, precipitation, reference evapotranspiration estimated from station data, annual runoff from streamflow gauges, and other variables to validate spatiotemporal elements of TerraClimate. Comparing TerraClimate records to coarser resolution gridded datasets, it was found that the overall mean absolute error had improved and the spatial realism had increased.

In order to analyse the reliability and trends for next projections, we have also chosen the 4C dataset, which is a prediction utilizing TerraClimate data. In order to understand the differences between utilizing the old and new data for our model, we have chosen three locations in south-east India and compared the results. We have also taken a few timeframes for the same locations.

These are the locations that attracted this project's attention.

Guntur - 16.3020, 80.4361

Nellore - 14.4402, 79.9869

Pondicherry - 11.9350, 79.8121

Years (1958-2020), (1990-2015)

(2000-2020), (2000-2015)

To properly compare the two timeframes and determine whether to use the most recent data from the last ten years (2000-2020) or to keep utilizing our broader dataset, we analyzed both yearly and monthly trends (1958-2020).

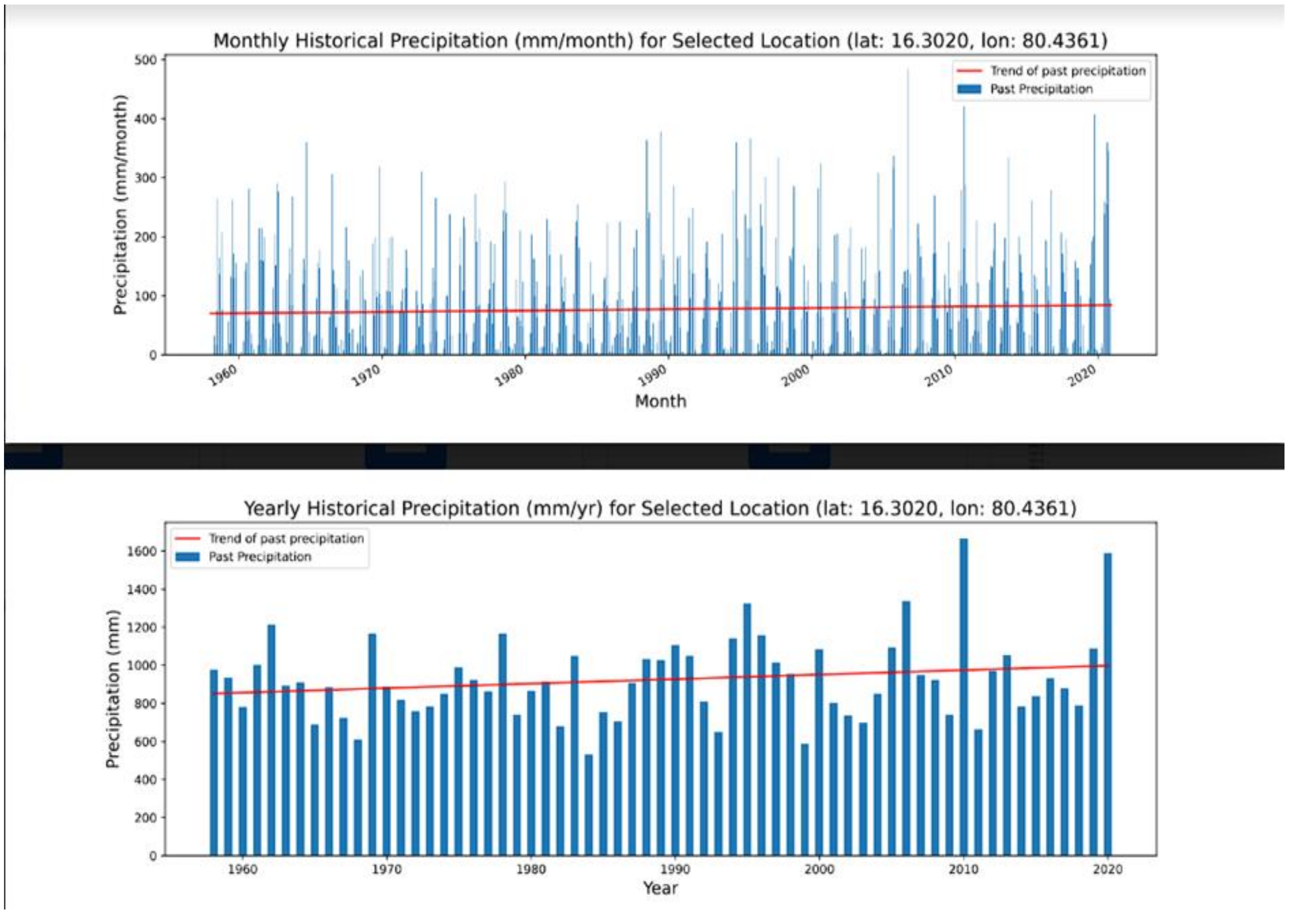


Fig 2: Trend in Guntur (1958-2020)

The increase in trend over the last decade or so will now be examined, together with the overall data, to see how much of a difference it makes.

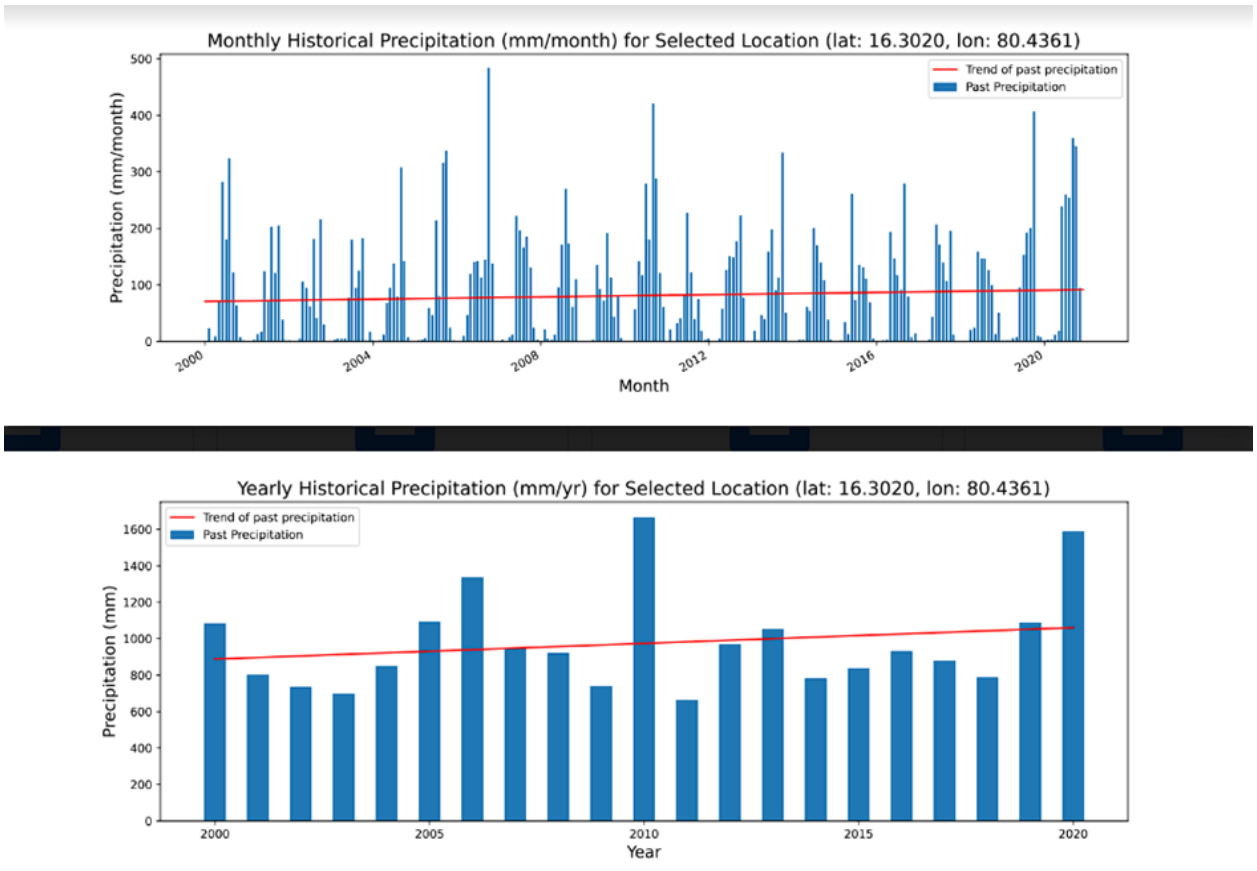


Fig 3: Trend in Guntur (2000-2020)

As we can see, the influence of global warming, volcano eruptions, and other natural and man-made tragedies has resulted in a dramatic rise in rainfall between 2000 and 2020. Since historical weather patterns and global conditions were very different from those of the present, it is preferable to base our projections on data from the previous decade and beyond. Doing so will prevent data from being too fitted. Therefore, using outdated data will only make it more difficult for us to foresee changes in the future.

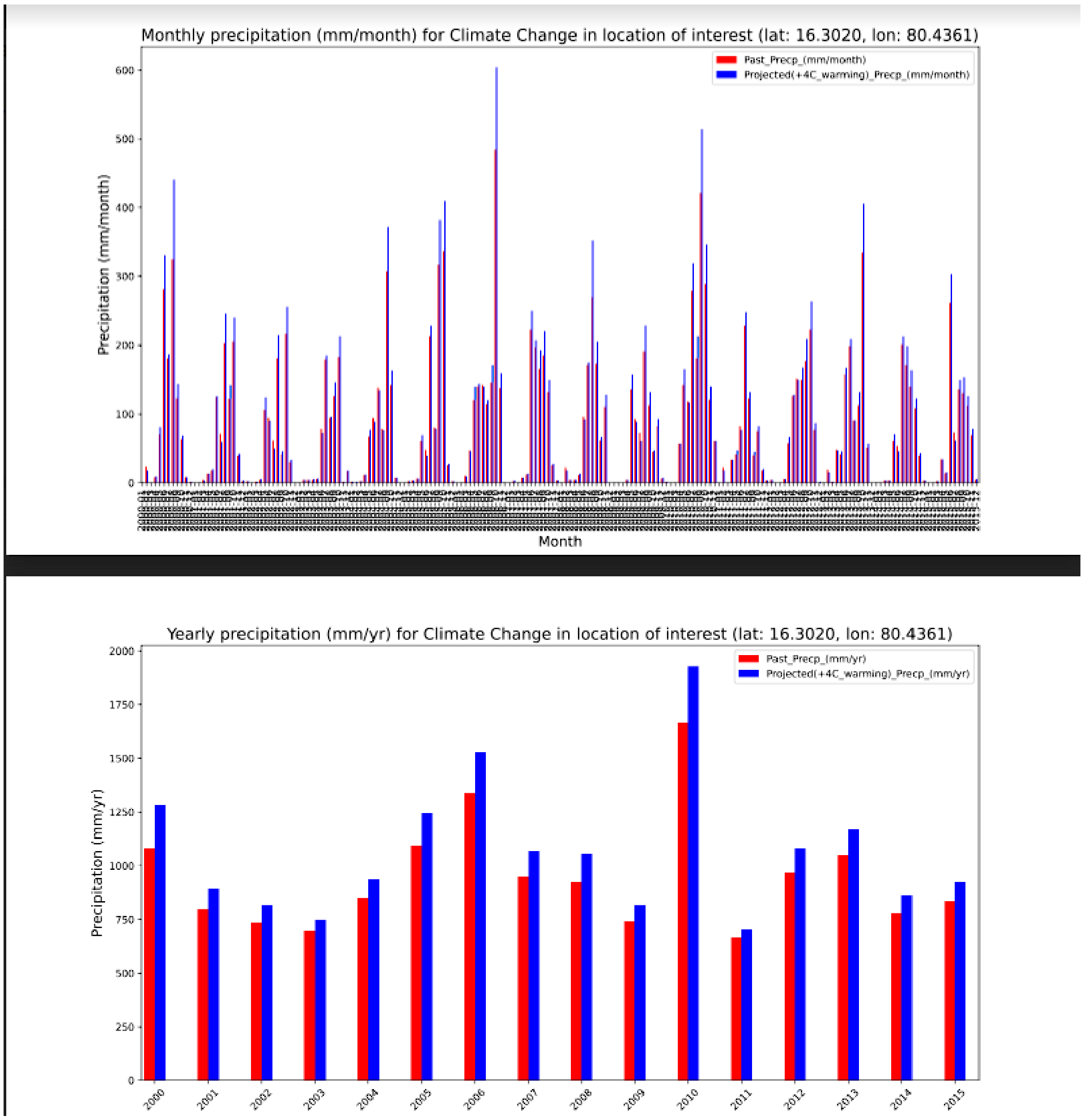
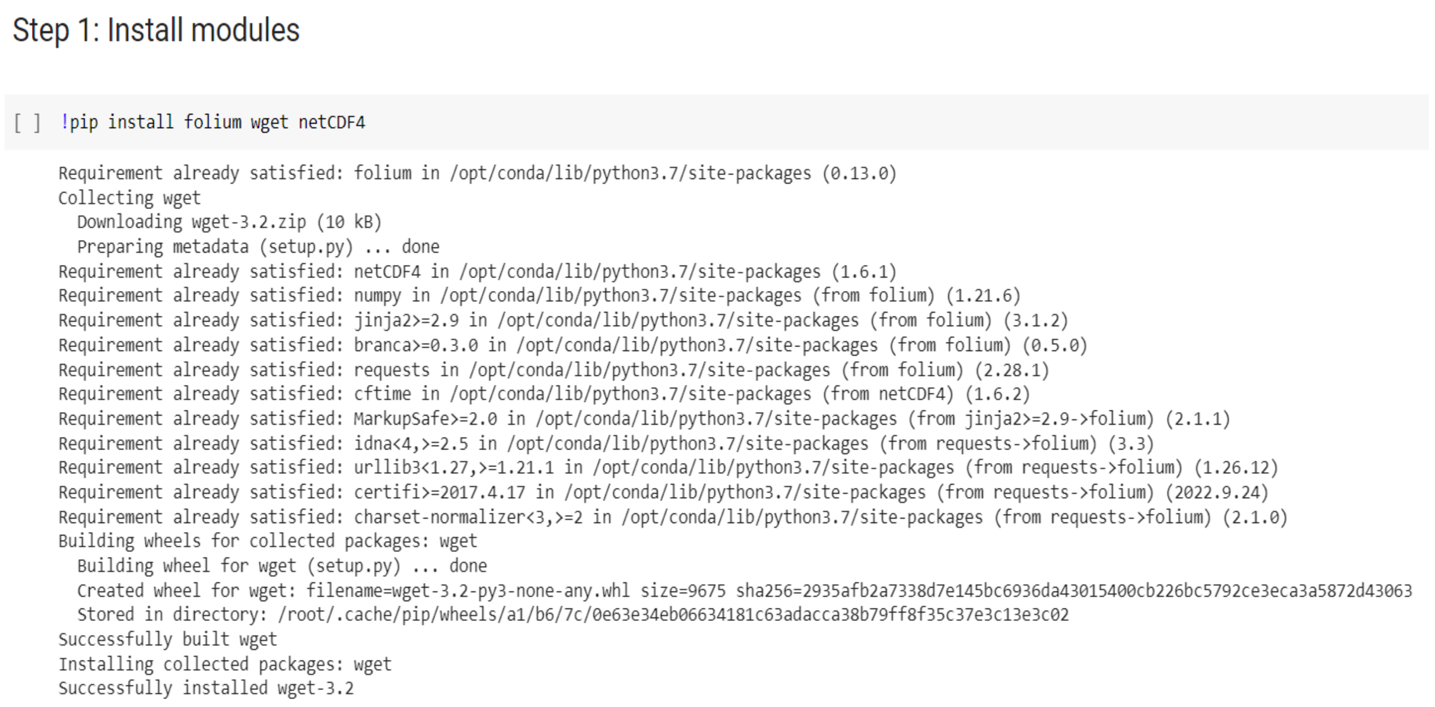


Fig 4: Difference from Predicted to Actual Rainfall (2000-2020)

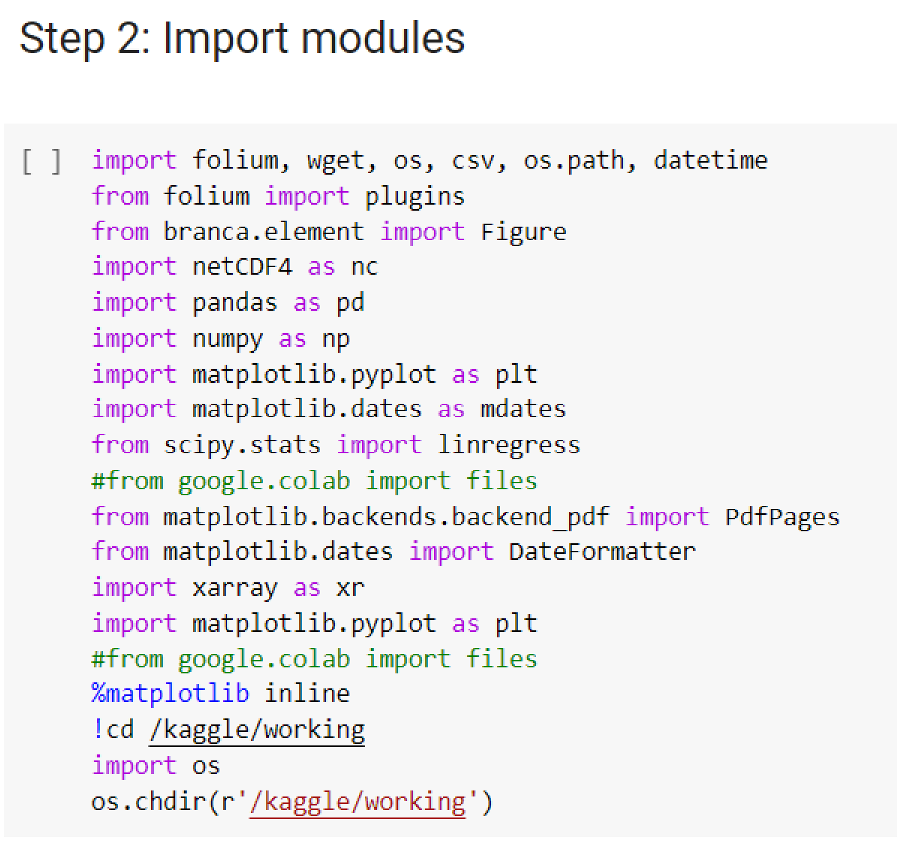
Similarly, we have processed and mapped data from (**Nellore - 14.4402, 79.9869)** and **(Pondicherry - 11.9350, 79.8121)** and arrived on similar conclusion.

2.3 Code Snippets



Step 1: Install modules

```
[ ] !pip install folium wget netCDF4

    Requirement already satisfied: folium in /opt/conda/lib/python3.7/site-packages (0.13.0)
    Collecting wget
      Downloading wget-3.2.zip (10 kB)
      Preparing metadata (setup.py) ... done
    Requirement already satisfied: netCDF4 in /opt/conda/lib/python3.7/site-packages (1.6.1)
    Requirement already satisfied: numpy in /opt/conda/lib/python3.7/site-packages (from folium) (1.21.6)
    Requirement already satisfied: jinja2>=2.9 in /opt/conda/lib/python3.7/site-packages (from folium) (3.1.2)
    Requirement already satisfied: branca>=0.3.0 in /opt/conda/lib/python3.7/site-packages (from folium) (0.5.0)
    Requirement already satisfied: requests in /opt/conda/lib/python3.7/site-packages (from folium) (2.28.1)
    Requirement already satisfied: cftime in /opt/conda/lib/python3.7/site-packages (from netCDF4) (1.6.2)
    Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.7/site-packages (from jinja2>=2.9->folium) (2.1.1)
    Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.7/site-packages (from requests->folium) (3.3)
    Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.7/site-packages (from requests->folium) (1.26.12)
    Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.7/site-packages (from requests->folium) (2022.9.24)
    Requirement already satisfied: charset-normalizer<3,>=2 in /opt/conda/lib/python3.7/site-packages (from requests->folium) (2.1.0)
    Building wheels for collected packages: wget
      Building wheel for wget (setup.py) ... done
      Created wheel for wget: filename=wget-3.2-py3-none-any.whl size=9675 sha256=2935afb2a7338d7e145bc6936da43015400cb226bc5792ce3eca3a5872d43063
      Stored in directory: /root/.cache/pip/wheels/a1/b6/7c/0e63e34eb06634181c63adacca38b79ff8f35c37e3c13e3c02
    Successfully built wget
    Installing collected packages: wget
    Successfully installed wget-3.2
```

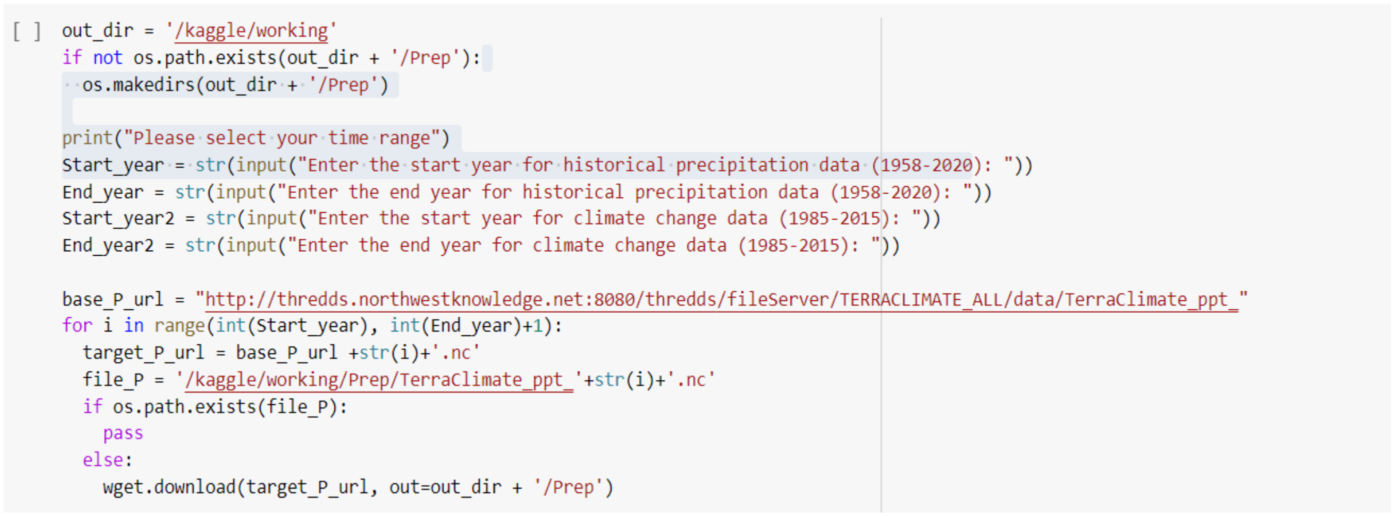


Step 2: Import modules

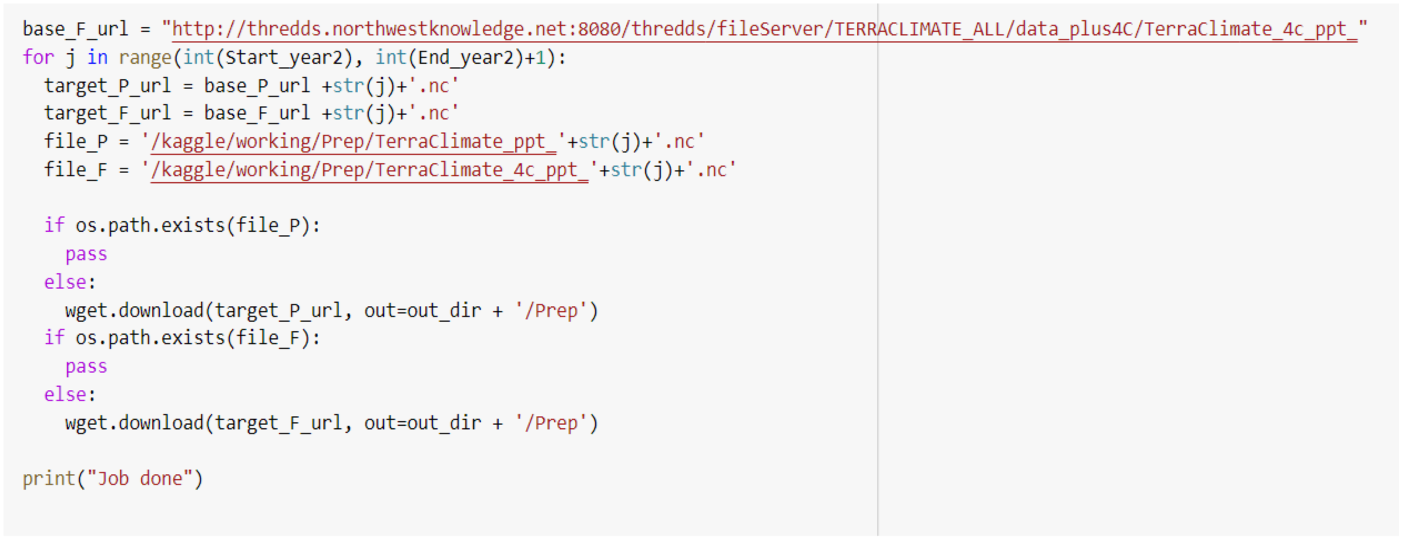
```
[ ] import folium, wget, os, csv, os.path, datetime
    from folium import plugins
    from branca.element import Figure
    import netCDF4 as nc
    import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import matplotlib.dates as mdates
    from scipy.stats import linregress
    #from google.colab import files
    from matplotlib.backends.backend_pdf import PdfPages
    from matplotlib.dates import DateFormatter
    import xarray as xr
    import matplotlib.pyplot as plt
    #from google.colab import files
    %matplotlib inline
    !cd /kaggle/working
    import os
    os.chdir(r'/kaggle/working')
```

Step 3: Download precipitation data

Nc files are a format of file for save climate data in multi-dimensions and the user can view each element of dimensions such as Lat, Lon, Level, Time, or... NetCDF (Network Common Data Form) is a set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data.



```
[ ] out_dir = '/kaggle/working'
    if not os.path.exists(out_dir + '/Prep'):
      os.makedirs(out_dir + '/Prep')

    print("Please select your time range")
    Start_year = str(input("Enter the start year for historical precipitation data (1958-2020): "))
    End_year = str(input("Enter the end year for historical precipitation data (1958-2020): "))
    Start_year2 = str(input("Enter the start year for climate change data (1985-2015): "))
    End_year2 = str(input("Enter the end year for climate change data (1985-2015): "))

    base_P_url = "http://thredds.northwestknowledge.net:8080/thredds/fileServer/TERRACLIMATE_ALL/data/TerraClimate_ppt_"
    for i in range(int(Start_year), int(End_year)+1):
      target_P_url = base_P_url +str(i)+'.nc'
      file_P = '/kaggle/working/Prep/TerraClimate_ppt_'+str(i)+'.nc'
      if os.path.exists(file_P):
        pass
      else:
        wget.download(target_P_url, out=out_dir + '/Prep')
```



```
base_F_url = "http://thredds.northwestknowledge.net:8080/thredds/fileServer/TERRACLIMATE_ALL/data_plus4C/TerraClimate_4c_ppt_"
for j in range(int(Start_year2), int(End_year2)+1):
  target_P_url = base_P_url +str(j)+'.nc'
  target_F_url = base_F_url +str(j)+'.nc'
  file_P = '/kaggle/working/Prep/TerraClimate_ppt_'+str(j)+'.nc'
  file_F = '/kaggle/working/Prep/TerraClimate_4c_ppt_'+str(j)+'.nc'

  if os.path.exists(file_P):
    pass
  else:
    wget.download(target_P_url, out=out_dir + '/Prep')
  if os.path.exists(file_F):
    pass
  else:
    wget.download(target_F_url, out=out_dir + '/Prep')

print("Job done")
```

```
Please select your time range
Enter the start year for historical precipitation data (1958-2020):  2000
Enter the end year for historical precipitation data (1958-2020):  2020
Enter the start year for climate change data (1985-2015):  2000
Enter the end year for climate change data (1985-2015):  2015
Job done
```

Step 4: Browse over an interactive global map and select your location of interest

Browse over the map, zoom-in to your location of interest, and left-click the mouse. The map will show the latitude and longtitude of your location.

```
[ ]  m = folium.Map(min_zoom=2, max_bounds=True, tiles = 'Stamen Terrain')
     m.add_child(folium.LatLngPopup())
     m.add_child(folium.LayerControl())
     m
```

Fig 5: World Map with their respective latitudes and longitudes

Note the latitude and longitude shown on the map and type them in the fields below.

```
[ ]  Lat = str(input("Please type latitude: "))
     Lon = str(input("Please type longitude: "))
```

```
Please type latitude:  23.000
Please type longitude:  78.000
```

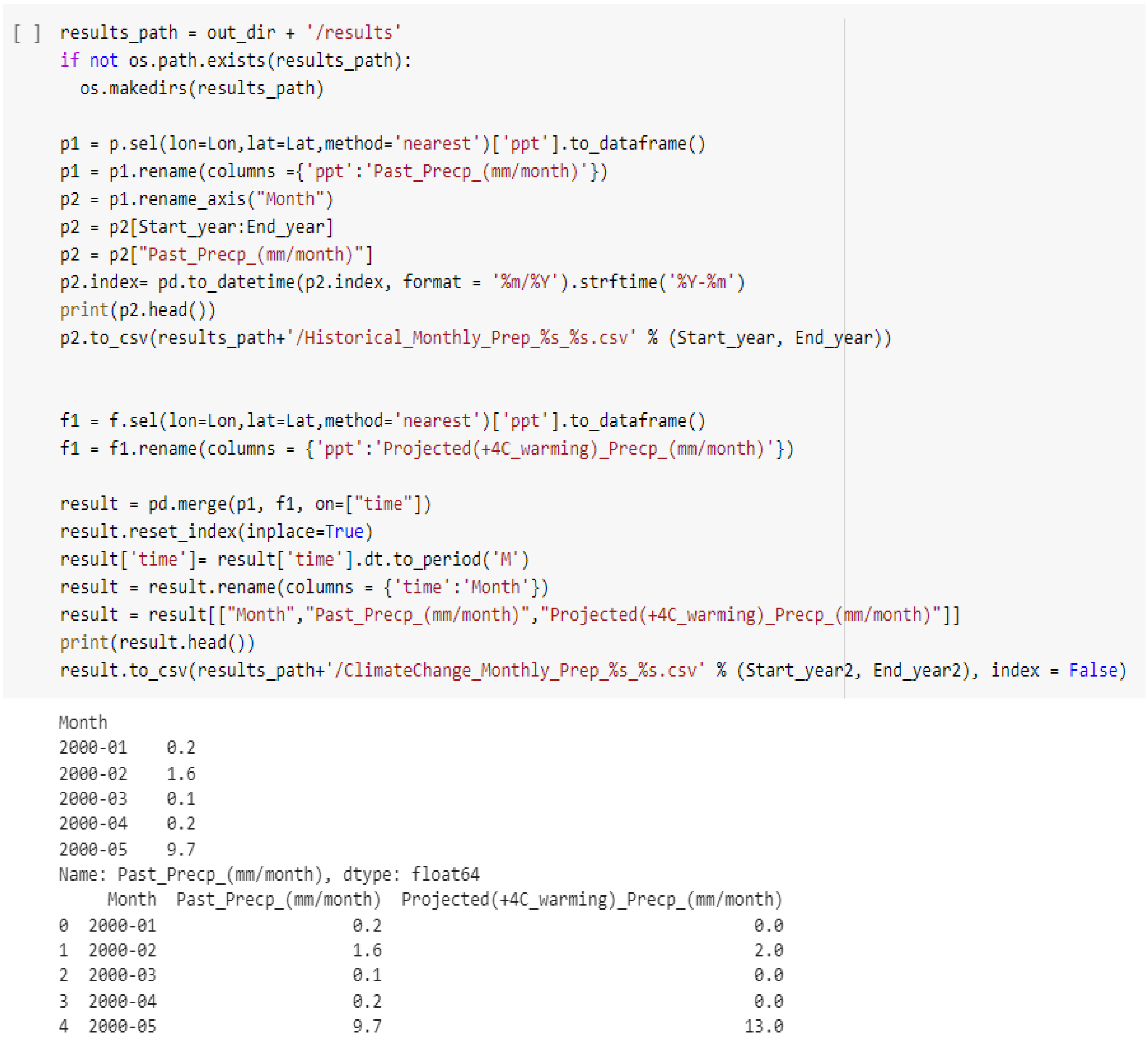
Step 5: Post-processing netCDF files

Read netCDF file and check the units of the datasets

```
[ ]  os.chdir("/kaggle/working/Prep")

     p = xr.open_mfdataset("TerraClimate_ppt_*.nc")
     f = xr.open_mfdataset("TerraClimate_4c_ppt_*.nc")

     f
```

```
xarray.Dataset
▸ Dimensions:          (lat: 4320, lon: 8640, time: 192, crs: 1)
▾ Coordinates:
    lat                (lat)                   float64  89.98 89.94 89.9 ... -89.94 -89.98
    lon                (lon)                   float64  -180.0 -179.9 ... 179.9 180.0
    time               (time)         datetime64[ns]  2000-01-01 ... 2015-12-01
    crs                (crs)                     int16  3
▾ Data variables:
    ppt                (time, lat, lon)        float32  dask.array<chunksize=(12, 4320, 8640), meta…
▸ Attributes:   (0)
```

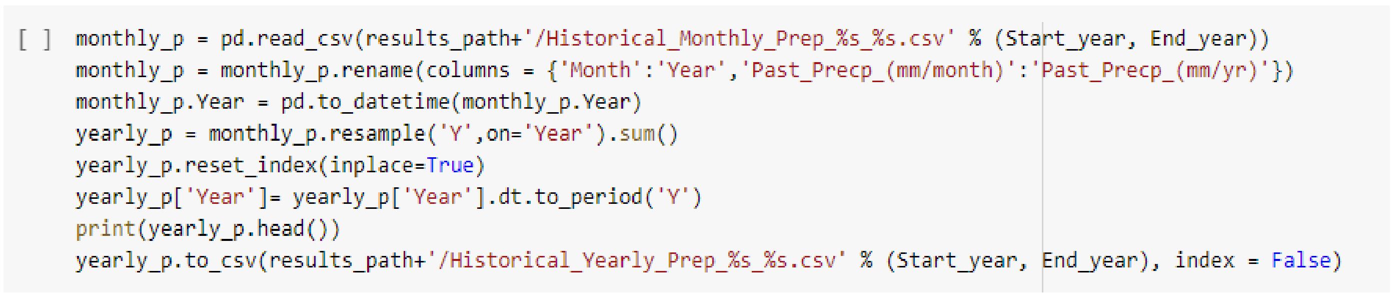
(i) Read and extract monthly precipitation data from netCDF, then save it as a CSV file



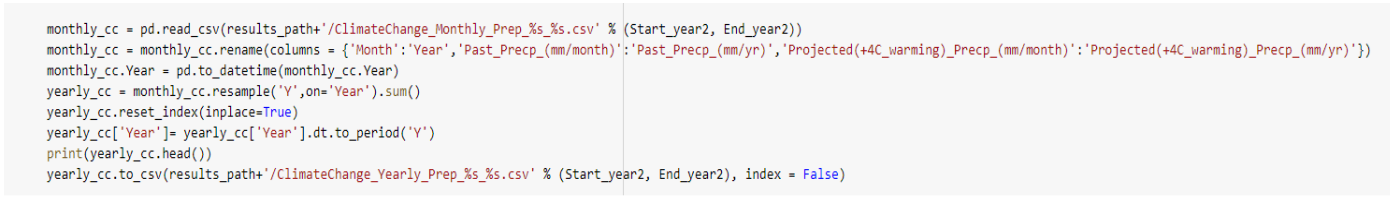
```
[ ] results_path = out_dir + '/results'
    if not os.path.exists(results_path):
      os.makedirs(results_path)

    p1 = p.sel(lon=Lon,lat=Lat,method='nearest')['ppt'].to_dataframe()
    p1 = p1.rename(columns ={'ppt':'Past_Precp_(mm/month)'})
    p2 = p1.rename_axis("Month")
    p2 = p2[Start_year:End_year]
    p2 = p2["Past_Precp_(mm/month)"]
    p2.index= pd.to_datetime(p2.index, format = '%m/%Y').strftime('%Y-%m')
    print(p2.head())
    p2.to_csv(results_path+'/Historical_Monthly_Prep_%s_%s.csv' % (Start_year, End_year))


    f1 = f.sel(lon=Lon,lat=Lat,method='nearest')['ppt'].to_dataframe()
    f1 = f1.rename(columns = {'ppt':'Projected(+4C_warming)_Precp_(mm/month)'})

    result = pd.merge(p1, f1, on=["time"])
    result.reset_index(inplace=True)
    result['time']= result['time'].dt.to_period('M')
    result = result.rename(columns = {'time':'Month'})
    result = result[["Month","Past_Precp_(mm/month)","Projected(+4C_warming)_Precp_(mm/month)"]]
    print(result.head())
    result.to_csv(results_path+'/ClimateChange_Monthly_Prep_%s_%s.csv' % (Start_year2, End_year2), index = False)
```

```
Month
2000-01    0.2
2000-02    1.6
2000-03    0.1
2000-04    0.2
2000-05    9.7
Name: Past_Precp_(mm/month), dtype: float64
     Month  Past_Precp_(mm/month)  Projected(+4C_warming)_Precp_(mm/month)
0  2000-01                    0.2                                      0.0
1  2000-02                    1.6                                      2.0
2  2000-03                    0.1                                      0.0
3  2000-04                    0.2                                      0.0
4  2000-05                    9.7                                     13.0
```

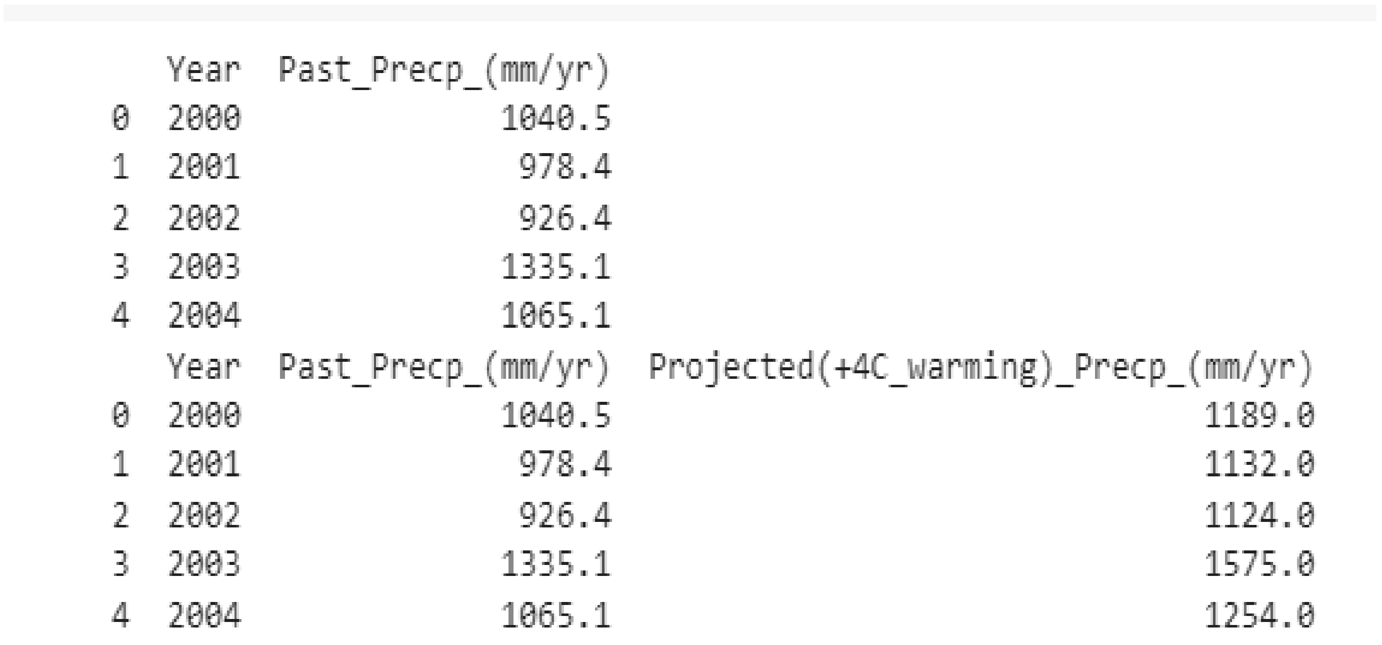
(ii) Calculate yearly total precipitation data, then save it as a CSV file



```
[ ] monthly_p = pd.read_csv(results_path+'/Historical_Monthly_Prep_%s_%s.csv' % (Start_year, End_year))
    monthly_p = monthly_p.rename(columns = {'Month':'Year','Past_Precp_(mm/month)':'Past_Precp_(mm/yr)'})
    monthly_p.Year = pd.to_datetime(monthly_p.Year)
    yearly_p = monthly_p.resample('Y',on='Year').sum()
    yearly_p.reset_index(inplace=True)
    yearly_p['Year']= yearly_p['Year'].dt.to_period('Y')
    print(yearly_p.head())
    yearly_p.to_csv(results_path+'/Historical_Yearly_Prep_%s_%s.csv' % (Start_year, End_year), index = False)
```



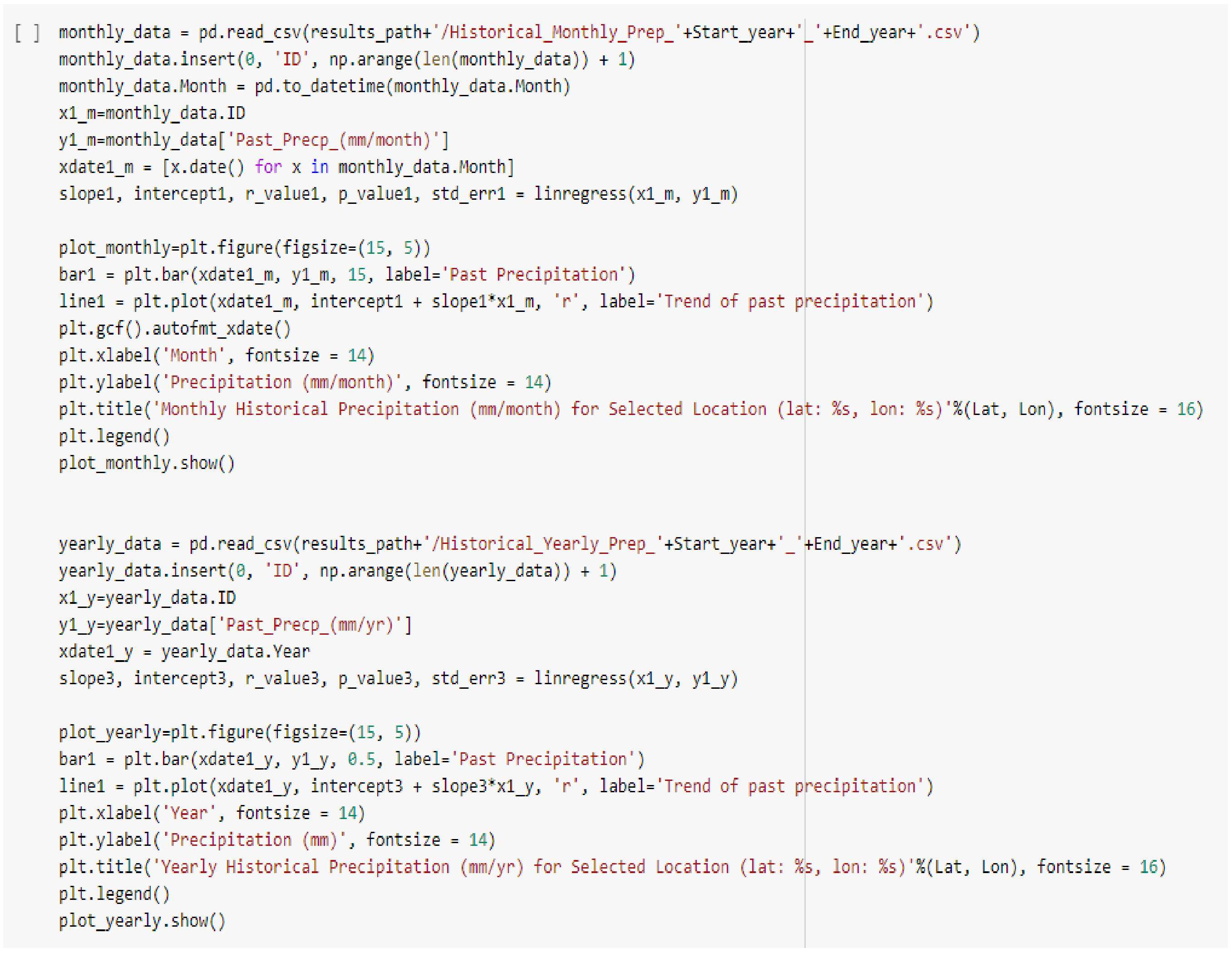
```
monthly_cc = pd.read_csv(results_path+'/ClimateChange_Monthly_Prep_%s_%s.csv' % (Start_year2, End_year2))
monthly_cc = monthly_cc.rename(columns = {'Month':'Year','Past_Precp_(mm/month)':'Past_Precp_(mm/yr)','Projected(+4C_warming)_Precp_(mm/month)':'Projected(+4C_warming)_Precp_(mm/yr)'})
monthly_cc.Year = pd.to_datetime(monthly_cc.Year)
yearly_cc = monthly_cc.resample('Y',on='Year').sum()
yearly_cc.reset_index(inplace=True)
yearly_cc['Year']= yearly_cc['Year'].dt.to_period('Y')
print(yearly_cc.head())
yearly_cc.to_csv(results_path+'/ClimateChange_Yearly_Prep_%s_%s.csv' % (Start_year2, End_year2), index = False)
```



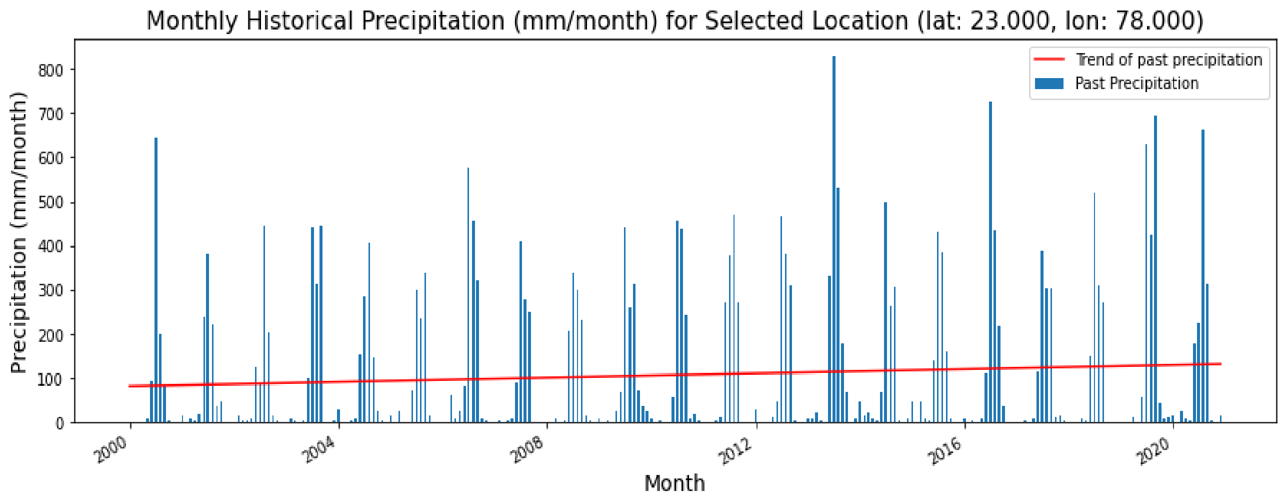
```
   Year  Past_Precp_(mm/yr)
0  2000              1040.5
1  2001               978.4
2  2002               926.4
3  2003              1335.1
4  2004              1065.1
   Year  Past_Precp_(mm/yr)  Projected(+4C_warming)_Precp_(mm/yr)
0  2000              1040.5                                1189.0
1  2001               978.4                                1132.0
2  2002               926.4                                1124.0
3  2003              1335.1                                1575.0
4  2004              1065.1                                1254.0
```

Step 6: Visualize how precipitation has been changing over the years in your location of interest

(i) Time-serie plot for historical precipitation over the year in location of interest



```python
monthly_data = pd.read_csv(results_path+'/Historical_Monthly_Prep_'+Start_year+'_'+End_year+'.csv')
monthly_data.insert(0, 'ID', np.arange(len(monthly_data)) + 1)
monthly_data.Month = pd.to_datetime(monthly_data.Month)
x1_m=monthly_data.ID
y1_m=monthly_data['Past_Precp_(mm/month)']
xdate1_m = [x.date() for x in monthly_data.Month]
slope1, intercept1, r_value1, p_value1, std_err1 = linregress(x1_m, y1_m)

plot_monthly=plt.figure(figsize=(15, 5))
bar1 = plt.bar(xdate1_m, y1_m, 15, label='Past Precipitation')
line1 = plt.plot(xdate1_m, intercept1 + slope1*x1_m, 'r', label='Trend of past precipitation')
plt.gcf().autofmt_xdate()
plt.xlabel('Month', fontsize = 14)
plt.ylabel('Precipitation (mm/month)', fontsize = 14)
plt.title('Monthly Historical Precipitation (mm/month) for Selected Location (lat: %s, lon: %s)'%(Lat, Lon), fontsize = 16)
plt.legend()
plot_monthly.show()


yearly_data = pd.read_csv(results_path+'/Historical_Yearly_Prep_'+Start_year+'_'+End_year+'.csv')
yearly_data.insert(0, 'ID', np.arange(len(yearly_data)) + 1)
x1_y=yearly_data.ID
y1_y=yearly_data['Past_Precp_(mm/yr)']
xdate1_y = yearly_data.Year
slope3, intercept3, r_value3, p_value3, std_err3 = linregress(x1_y, y1_y)

plot_yearly=plt.figure(figsize=(15, 5))
bar1 = plt.bar(xdate1_y, y1_y, 0.5, label='Past Precipitation')
line1 = plt.plot(xdate1_y, intercept3 + slope3*x1_y, 'r', label='Trend of past precipitation')
plt.xlabel('Year', fontsize = 14)
plt.ylabel('Precipitation (mm)', fontsize = 14)
plt.title('Yearly Historical Precipitation (mm/yr) for Selected Location (lat: %s, lon: %s)'%(Lat, Lon), fontsize = 16)
plt.legend()
plot_yearly.show()
```



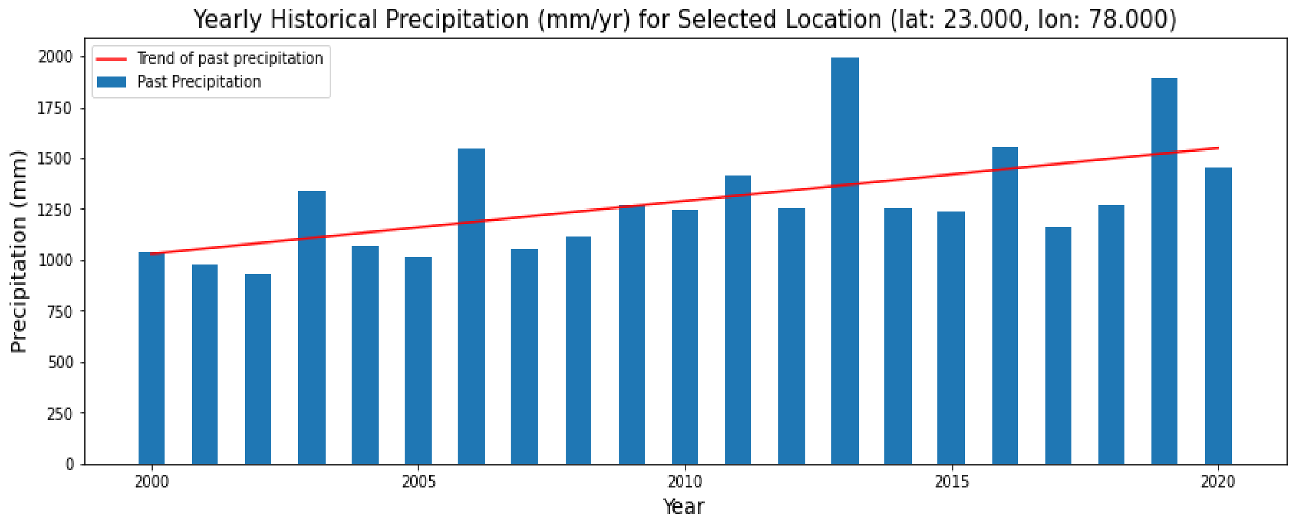
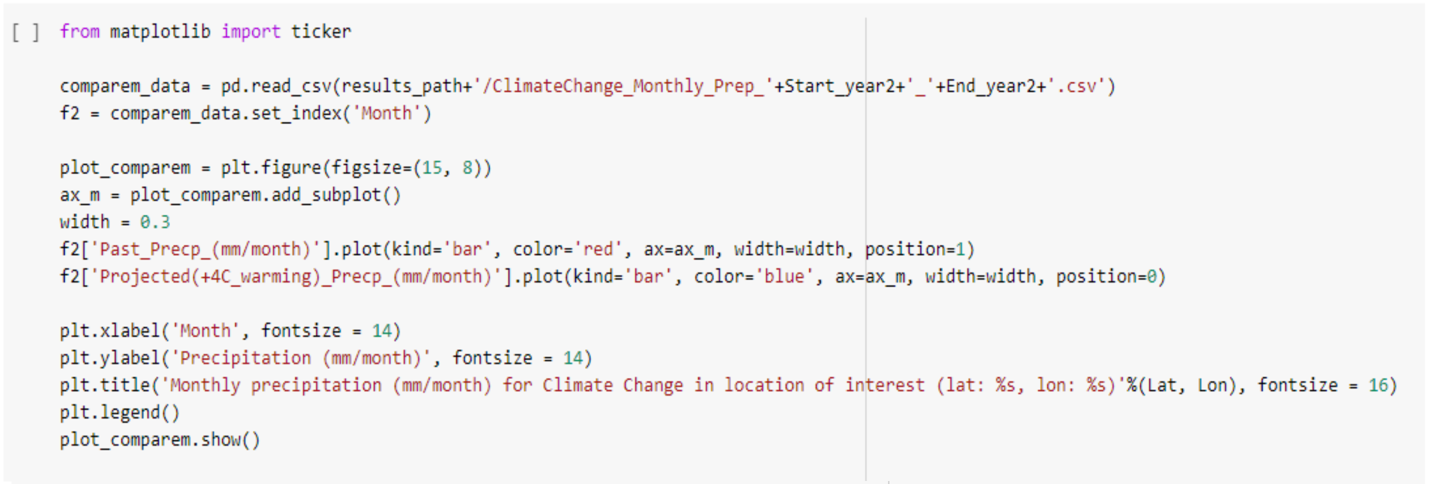
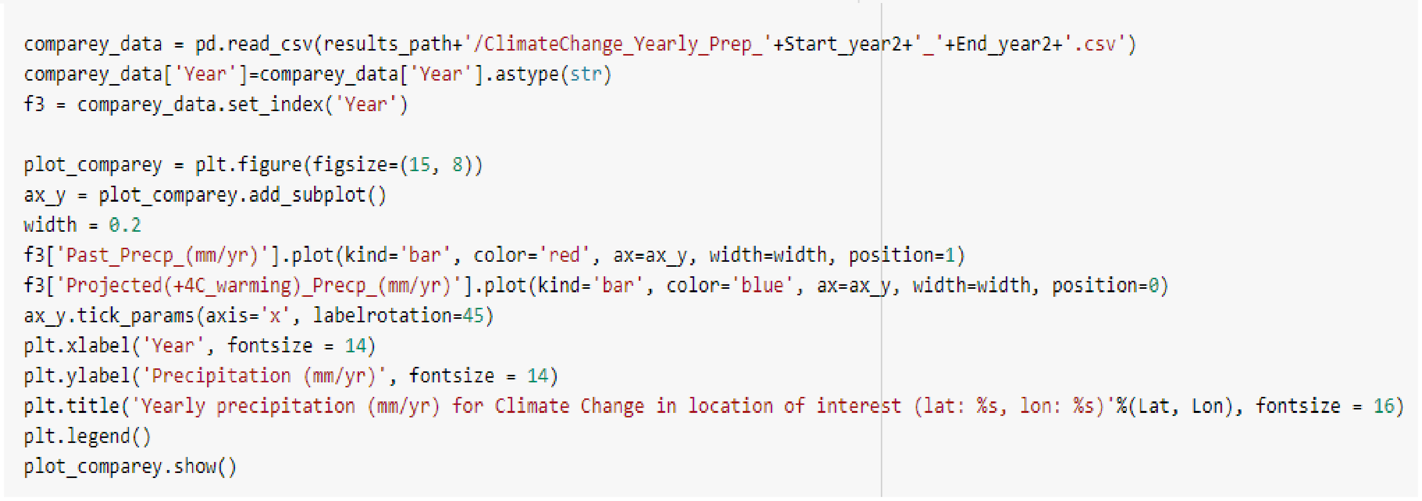


Fig 6: Comparison between trend of predicted vs Actual data

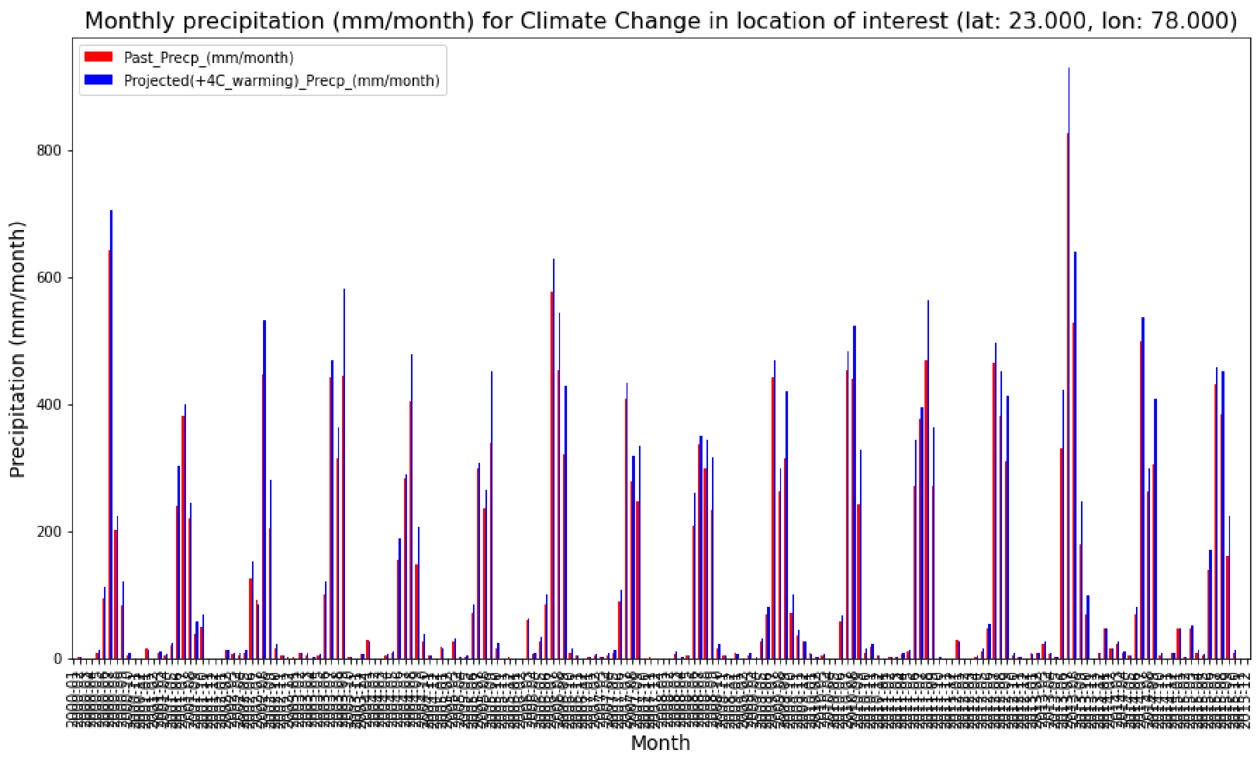
(ii) Comparison of historical and projected (+4C warming) preciptiation over the years in location of interest



```
[ ]  from matplotlib import ticker

     comparem_data = pd.read_csv(results_path+'/ClimateChange_Monthly_Prep_'+Start_year2+'_'+End_year2+'.csv')
     f2 = comparem_data.set_index('Month')

     plot_comparem = plt.figure(figsize=(15, 8))
     ax_m = plot_comparem.add_subplot()
     width = 0.3
     f2['Past_Precp_(mm/month)'].plot(kind='bar', color='red', ax=ax_m, width=width, position=1)
     f2['Projected(+4C_warming)_Precp_(mm/month)'].plot(kind='bar', color='blue', ax=ax_m, width=width, position=0)

     plt.xlabel('Month', fontsize = 14)
     plt.ylabel('Precipitation (mm/month)', fontsize = 14)
     plt.title('Monthly precipitation (mm/month) for Climate Change in location of interest (lat: %s, lon: %s)'%(Lat, Lon), fontsize = 16)
     plt.legend()
     plot_comparem.show()
```



```
comparey_data = pd.read_csv(results_path+'/ClimateChange_Yearly_Prep_'+Start_year2+'_'+End_year2+'.csv')
comparey_data['Year']=comparey_data['Year'].astype(str)
f3 = comparey_data.set_index('Year')

plot_comparey = plt.figure(figsize=(15, 8))
ax_y = plot_comparey.add_subplot()
width = 0.2
f3['Past_Precp_(mm/yr)'].plot(kind='bar', color='red', ax=ax_y, width=width, position=1)
f3['Projected(+4C_warming)_Precp_(mm/yr)'].plot(kind='bar', color='blue', ax=ax_y, width=width, position=0)
ax_y.tick_params(axis='x', labelrotation=45)
plt.xlabel('Year', fontsize = 14)
plt.ylabel('Precipitation (mm/yr)', fontsize = 14)
plt.title('Yearly precipitation (mm/yr) for Climate Change in location of interest (lat: %s, lon: %s)'%(Lat, Lon), fontsize = 16)
plt.legend()
plot_comparey.show()
```



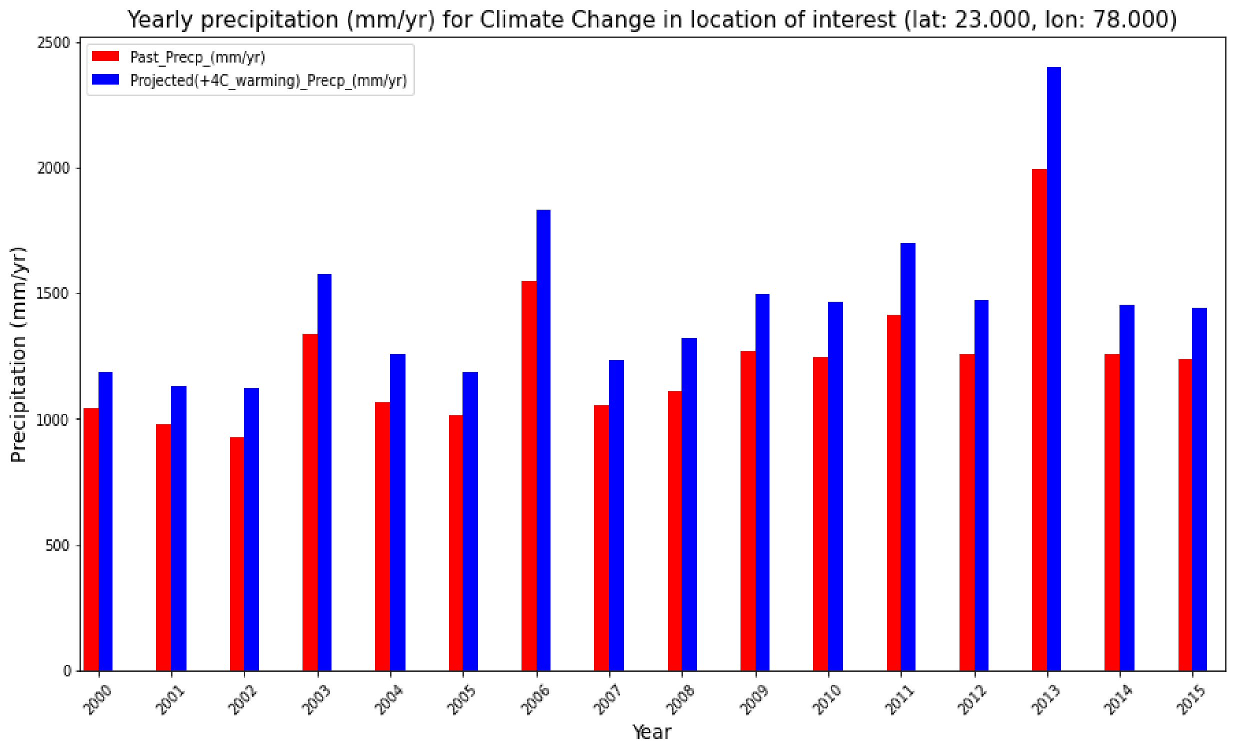
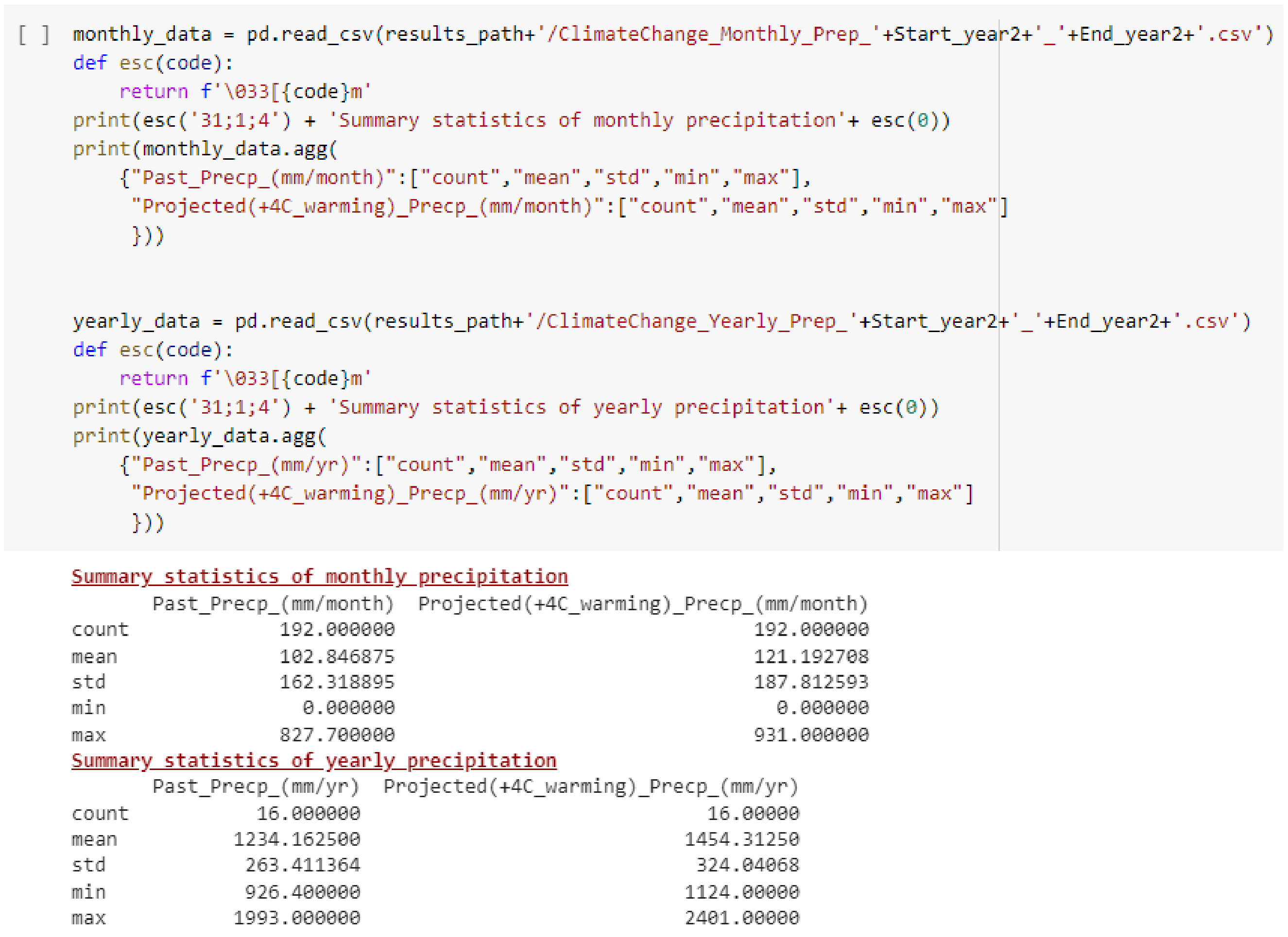


Fig 7: Comparison between predicted vs Actual data

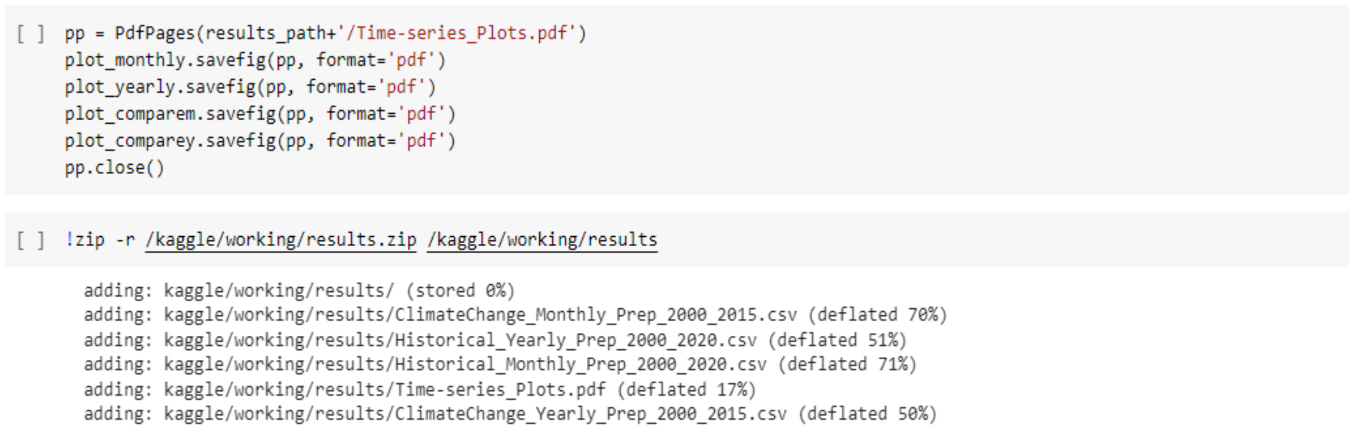
Step 7: Performing statistical analysis of climate change



```
[ ]  monthly_data = pd.read_csv(results_path+'/ClimateChange_Monthly_Prep_'+Start_year2+'_'+End_year2+'.csv')
     def esc(code):
         return f'\033[{code}m'
     print(esc('31;1;4') + 'Summary statistics of monthly precipitation'+ esc(0))
     print(monthly_data.agg(
         {"Past_Precp_(mm/month)":["count","mean","std","min","max"],
          "Projected(+4C_warming)_Precp_(mm/month)":["count","mean","std","min","max"]
          }))


     yearly_data = pd.read_csv(results_path+'/ClimateChange_Yearly_Prep_'+Start_year2+'_'+End_year2+'.csv')
     def esc(code):
         return f'\033[{code}m'
     print(esc('31;1;4') + 'Summary statistics of yearly precipitation'+ esc(0))
     print(yearly_data.agg(
         {"Past_Precp_(mm/yr)":["count","mean","std","min","max"],
          "Projected(+4C_warming)_Precp_(mm/yr)":["count","mean","std","min","max"]
          }))
```

```
Summary statistics of monthly precipitation
       Past_Precp_(mm/month)  Projected(+4C_warming)_Precp_(mm/month)
count             192.000000                               192.000000
mean              102.846875                               121.192708
std               162.318895                               187.812593
min                 0.000000                                 0.000000
max               827.700000                               931.000000
Summary statistics of yearly precipitation
       Past_Precp_(mm/yr)  Projected(+4C_warming)_Precp_(mm/yr)
count           16.000000                              16.00000
mean          1234.162500                            1454.31250
std            263.411364                             324.04068
min            926.400000                            1124.00000
max           1993.000000                            2401.00000
```

Step 8: Download four CSV files (monthly and yearly historical and projected precipitation data) and time-series plots to the local computer

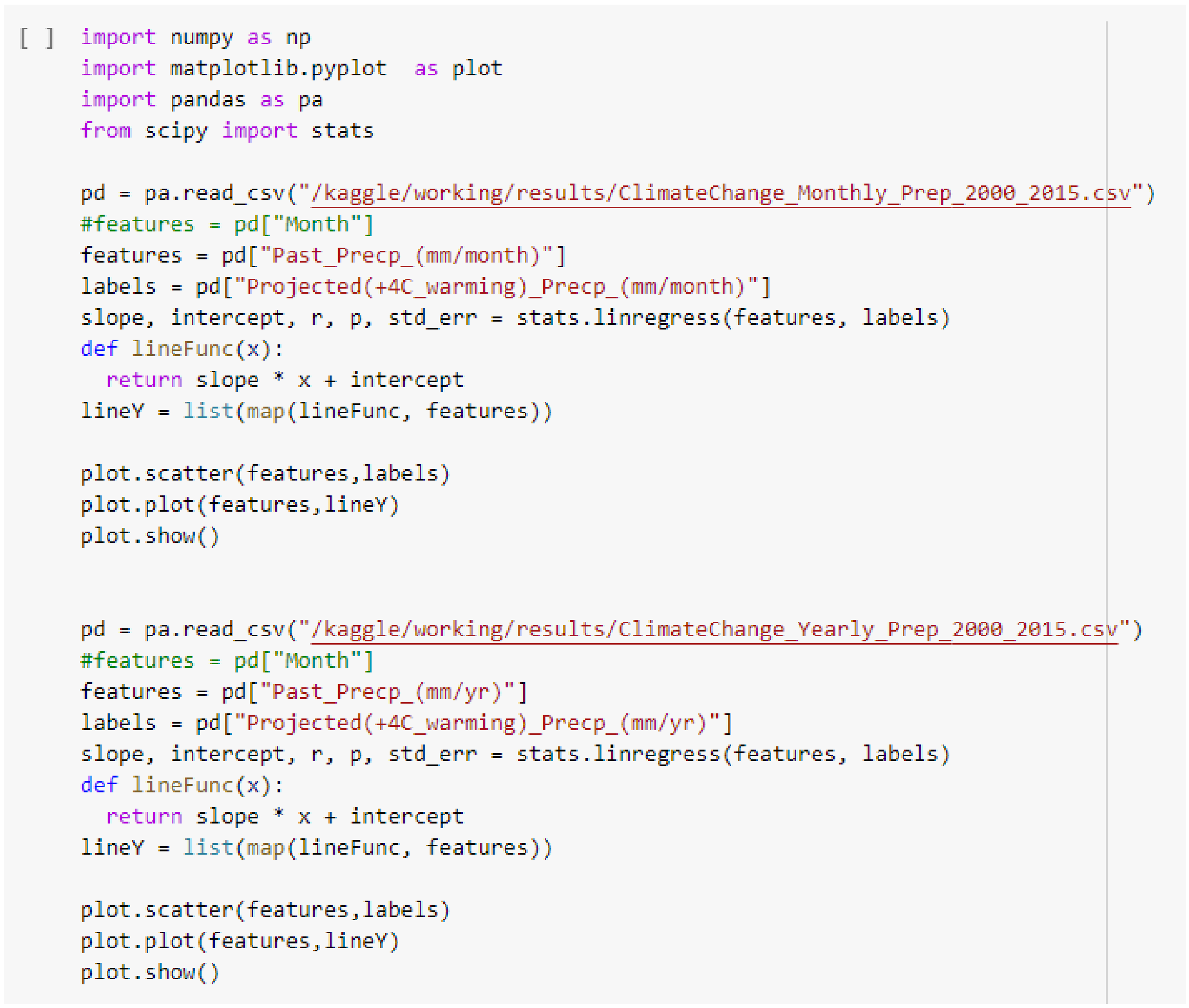


```
[ ]  pp = PdfPages(results_path+'/Time-series_Plots.pdf')
     plot_monthly.savefig(pp, format='pdf')
     plot_yearly.savefig(pp, format='pdf')
     plot_comparem.savefig(pp, format='pdf')
     plot_comparey.savefig(pp, format='pdf')
     pp.close()
```

```
[ ]  !zip -r /kaggle/working/results.zip /kaggle/working/results
```

```
  adding: kaggle/working/results/ (stored 0%)
  adding: kaggle/working/results/ClimateChange_Monthly_Prep_2000_2015.csv (deflated 70%)
  adding: kaggle/working/results/Historical_Yearly_Prep_2000_2020.csv (deflated 51%)
  adding: kaggle/working/results/Historical_Monthly_Prep_2000_2020.csv (deflated 71%)
  adding: kaggle/working/results/Time-series_Plots.pdf (deflated 17%)
  adding: kaggle/working/results/ClimateChange_Yearly_Prep_2000_2015.csv (deflated 50%)
```

Step 9: Applying Linear Regression to Validate the Projected Data



```
[ ]  import numpy as np
     import matplotlib.pyplot  as plot
     import pandas as pa
     from scipy import stats

     pd = pa.read_csv("/kaggle/working/results/ClimateChange_Monthly_Prep_2000_2015.csv")
     #features = pd["Month"]
     features = pd["Past_Precp_(mm/month)"]
     labels = pd["Projected(+4C_warming)_Precp_(mm/month)"]
     slope, intercept, r, p, std_err = stats.linregress(features, labels)
     def lineFunc(x):
       return slope * x + intercept
     lineY = list(map(lineFunc, features))

     plot.scatter(features,labels)
     plot.plot(features,lineY)
     plot.show()


     pd = pa.read_csv("/kaggle/working/results/ClimateChange_Yearly_Prep_2000_2015.csv")
     #features = pd["Month"]
     features = pd["Past_Precp_(mm/yr)"]
     labels = pd["Projected(+4C_warming)_Precp_(mm/yr)"]
     slope, intercept, r, p, std_err = stats.linregress(features, labels)
     def lineFunc(x):
       return slope * x + intercept
     lineY = list(map(lineFunc, features))

     plot.scatter(features,labels)
     plot.plot(features,lineY)
     plot.show()
```

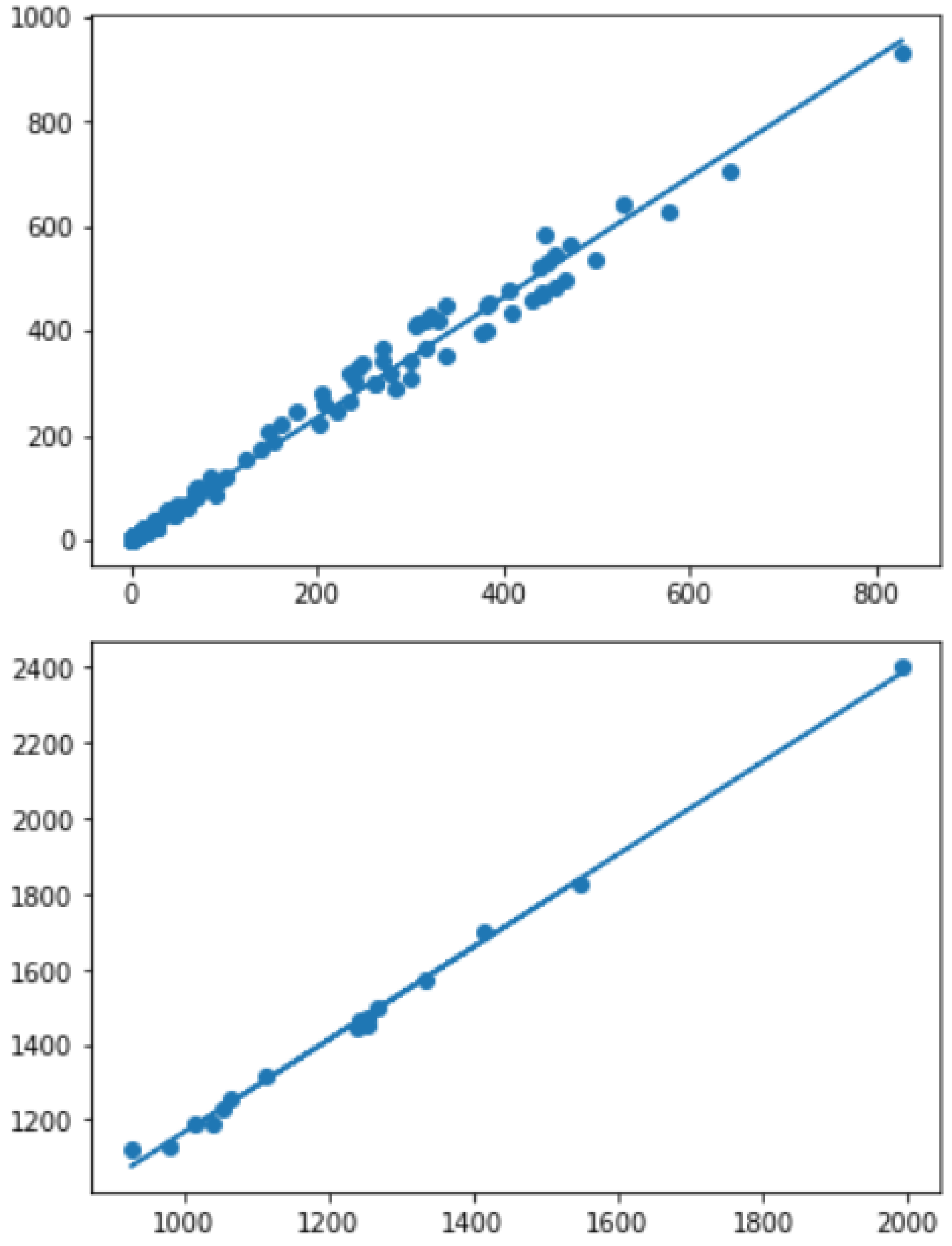
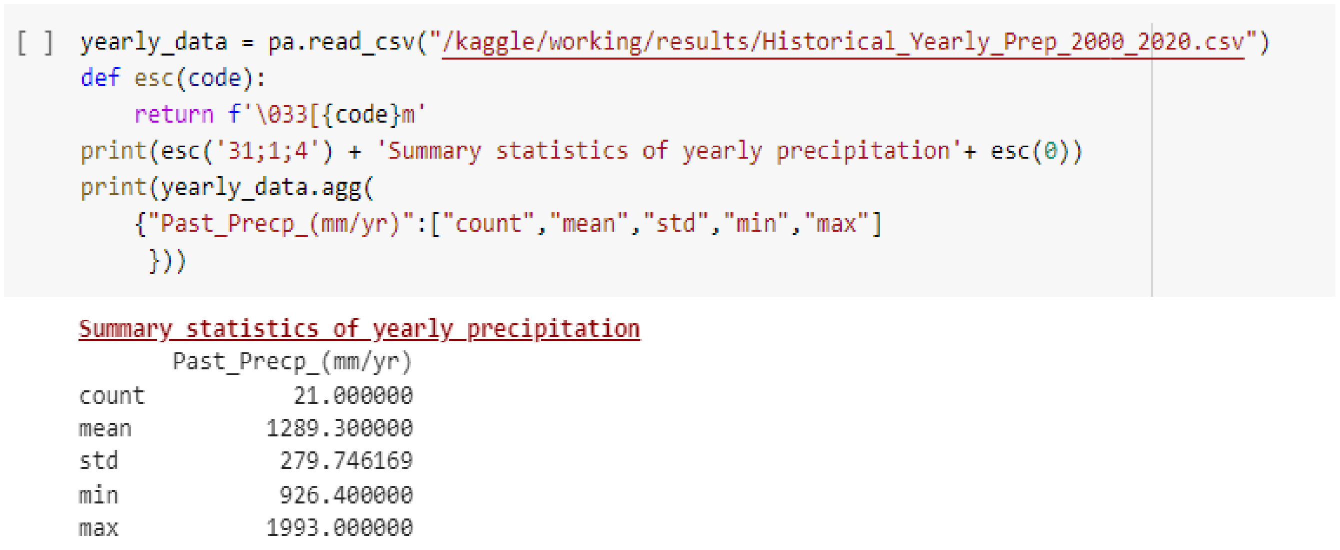


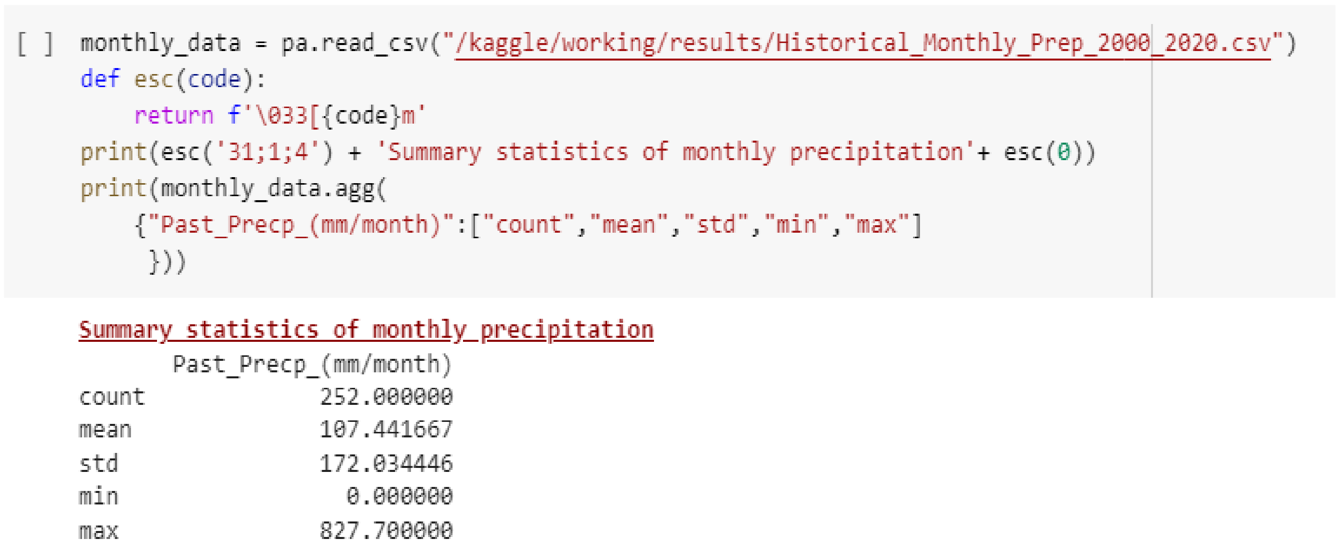
Fig 8: Linear Regression over Projected Data

Step 10: Statistical Summary



```
[ ] yearly_data = pa.read_csv("/kaggle/working/results/Historical_Yearly_Prep_2000_2020.csv")
    def esc(code):
        return f'\033[{code}m'
    print(esc('31;1;4') + 'Summary statistics of yearly precipitation'+ esc(0))
    print(yearly_data.agg(
        {"Past_Precp_(mm/yr)":["count","mean","std","min","max"]
         }))
```

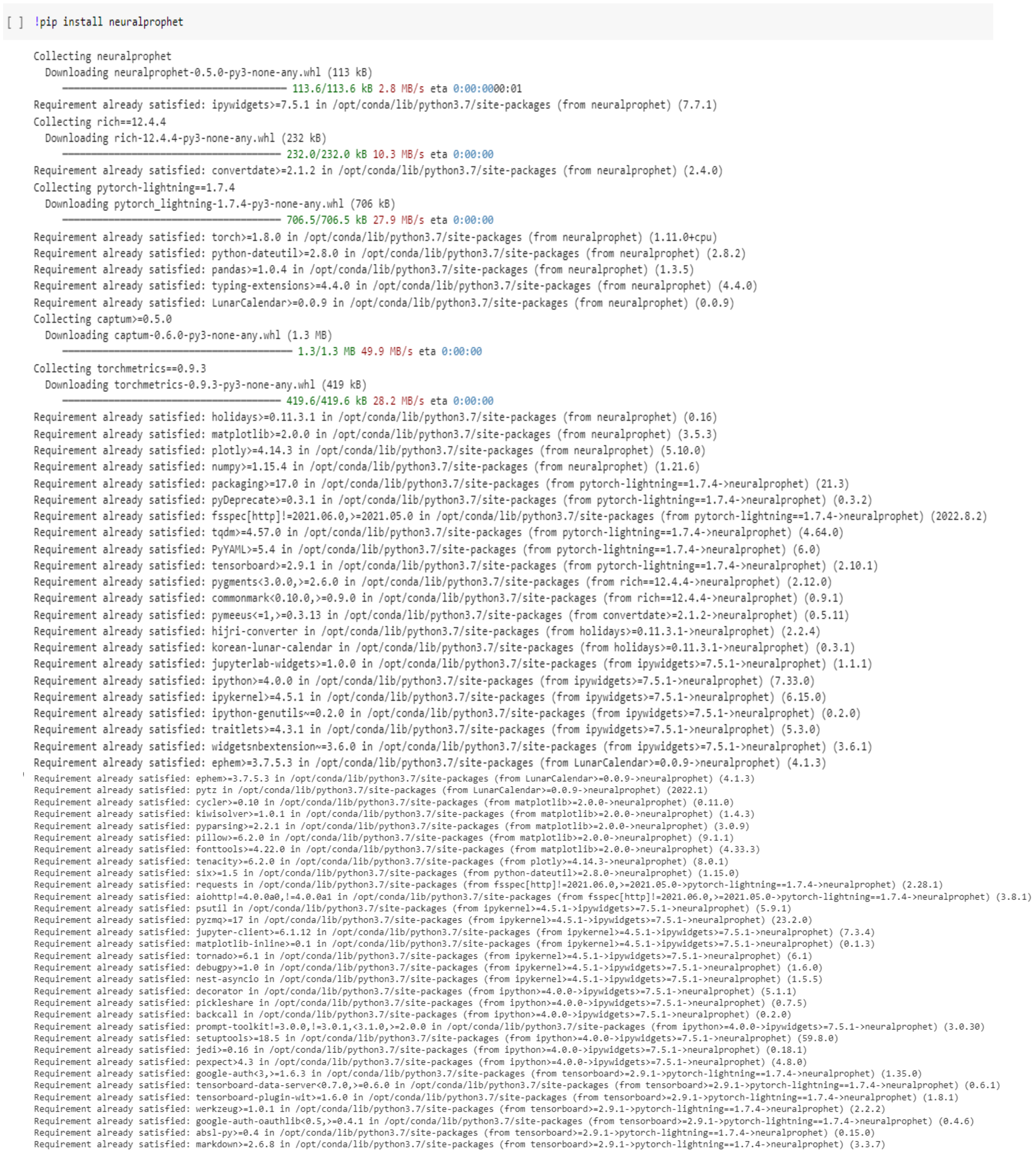
```
Summary statistics of yearly precipitation
       Past_Precp_(mm/yr)
count            21.000000
mean           1289.300000
std             279.746169
min             926.400000
max            1993.000000
```



```
[ ] monthly_data = pa.read_csv("/kaggle/working/results/Historical_Monthly_Prep_2000_2020.csv")
    def esc(code):
        return f'\033[{code}m'
    print(esc('31;1;4') + 'Summary statistics of monthly precipitation'+ esc(0))
    print(monthly_data.agg(
        {"Past_Precp_(mm/month)":["count","mean","std","min","max"]
         }))
```

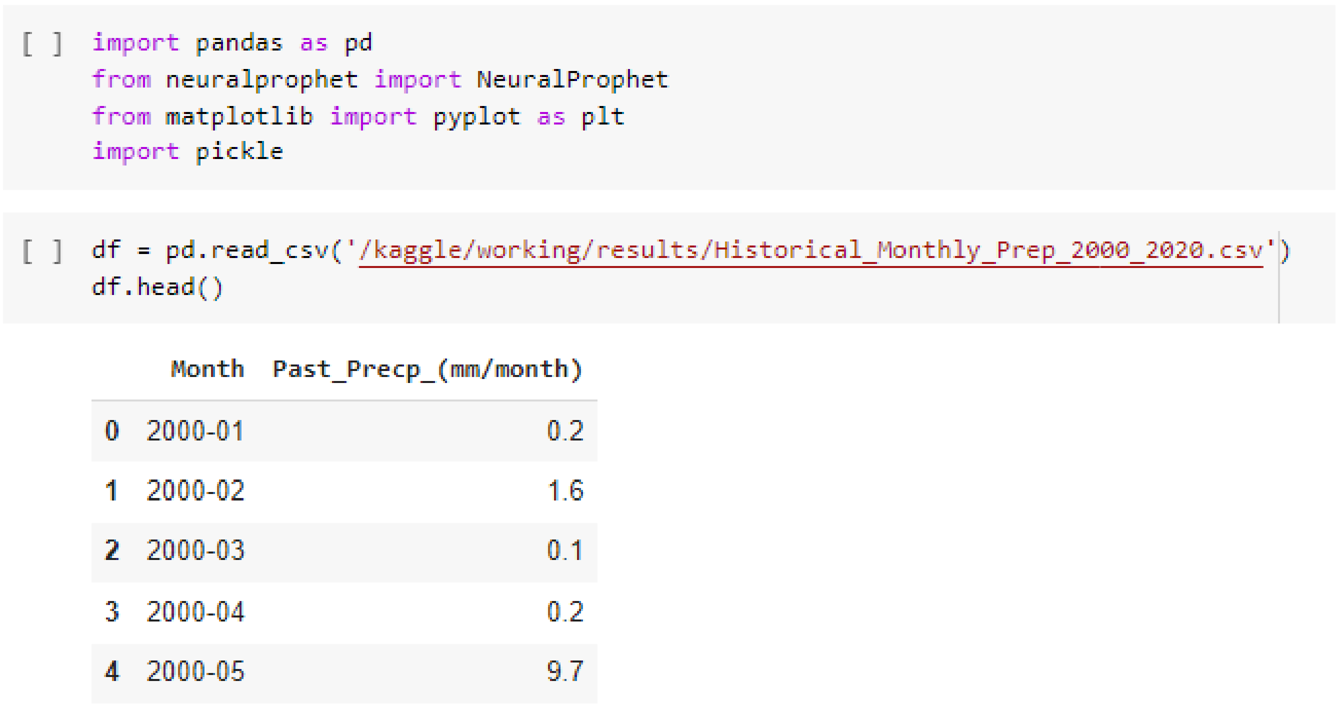
```
Summary statistics of monthly precipitation
       Past_Precp_(mm/month)
count               252.000000
mean                107.441667
std                 172.034446
min                   0.000000
max                 827.700000
```

Step 11: Radiance Forecast



```
[ ] !pip install neuralprophet
```

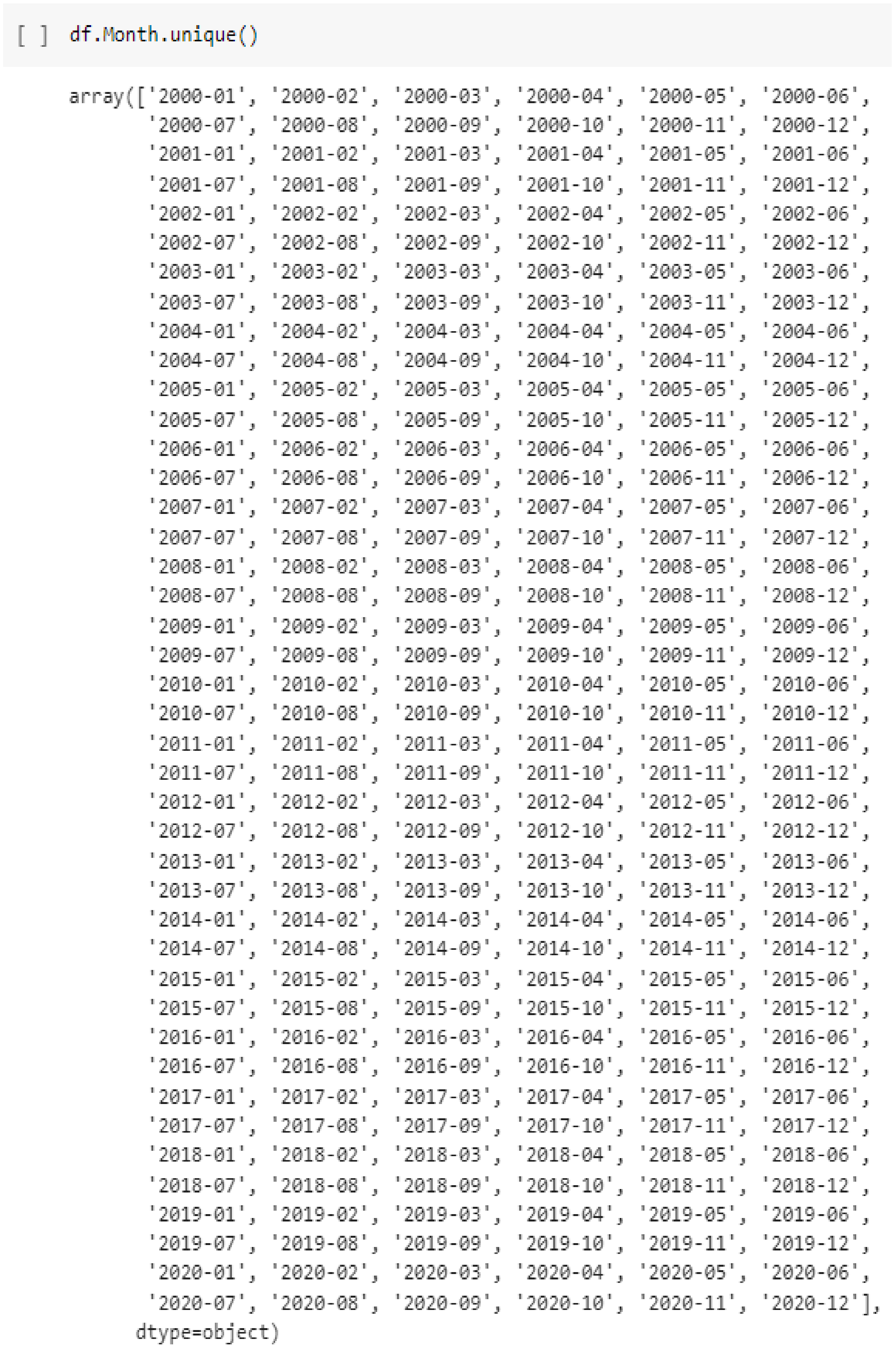
```
Collecting neuralprophet
  Downloading neuralprophet-0.5.0-py3-none-any.whl (113 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 113.6/113.6 kB 2.8 MB/s eta 0:00:0000:01
Requirement already satisfied: ipywidgets>=7.5.1 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (7.7.1)
Collecting rich==12.4.4
  Downloading rich-12.4.4-py3-none-any.whl (232 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 232.0/232.0 kB 10.3 MB/s eta 0:00:00
Requirement already satisfied: convertdate>=2.1.2 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (2.4.0)
Collecting pytorch-lightning==1.7.4
  Downloading pytorch_lightning-1.7.4-py3-none-any.whl (706 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 706.5/706.5 kB 27.9 MB/s eta 0:00:00
Requirement already satisfied: torch>=1.8.0 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (1.11.0+cpu)
Requirement already satisfied: python-dateutil>=2.8.0 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (2.8.2)
Requirement already satisfied: pandas>=1.0.4 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (1.3.5)
Requirement already satisfied: typing-extensions>=4.4.0 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (4.4.0)
Requirement already satisfied: LunarCalendar>=0.0.9 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (0.0.9)
Collecting captum>=0.5.0
  Downloading captum-0.6.0-py3-none-any.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 49.9 MB/s eta 0:00:00
Collecting torchmetrics==0.9.3
  Downloading torchmetrics-0.9.3-py3-none-any.whl (419 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 419.6/419.6 kB 28.2 MB/s eta 0:00:00
Requirement already satisfied: holidays>=0.11.3.1 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (0.16)
Requirement already satisfied: matplotlib>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (3.5.3)
Requirement already satisfied: plotly>=4.14.3 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (5.10.0)
Requirement already satisfied: numpy>=1.15.4 in /opt/conda/lib/python3.7/site-packages (from neuralprophet) (1.21.6)
Requirement already satisfied: packaging>=17.0 in /opt/conda/lib/python3.7/site-packages (from pytorch-lightning==1.7.4->neuralprophet) (21.3)
Requirement already satisfied: pyDeprecate>=0.3.1 in /opt/conda/lib/python3.7/site-packages (from pytorch-lightning==1.7.4->neuralprophet) (0.3.2)
Requirement already satisfied: fsspec[http]!=2021.06.0,>=2021.05.0 in /opt/conda/lib/python3.7/site-packages (from pytorch-lightning==1.7.4->neuralprophet) (2022.8.2)
Requirement already satisfied: tqdm>=4.57.0 in /opt/conda/lib/python3.7/site-packages (from pytorch-lightning==1.7.4->neuralprophet) (4.64.0)
Requirement already satisfied: PyYAML>=5.4 in /opt/conda/lib/python3.7/site-packages (from pytorch-lightning==1.7.4->neuralprophet) (6.0)
Requirement already satisfied: tensorboard>=2.9.1 in /opt/conda/lib/python3.7/site-packages (from pytorch-lightning==1.7.4->neuralprophet) (2.10.1)
Requirement already satisfied: pygments<3.0.0,>=2.6.0 in /opt/conda/lib/python3.7/site-packages (from rich==12.4.4->neuralprophet) (2.12.0)
Requirement already satisfied: commonmark<0.10.0,>=0.9.0 in /opt/conda/lib/python3.7/site-packages (from rich==12.4.4->neuralprophet) (0.9.1)
Requirement already satisfied: pymeeus<=1,>=0.3.13 in /opt/conda/lib/python3.7/site-packages (from convertdate>=2.1.2->neuralprophet) (0.5.11)
Requirement already satisfied: hijri-converter in /opt/conda/lib/python3.7/site-packages (from holidays>=0.11.3.1->neuralprophet) (2.2.4)
Requirement already satisfied: korean-lunar-calendar in /opt/conda/lib/python3.7/site-packages (from holidays>=0.11.3.1->neuralprophet) (0.3.1)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /opt/conda/lib/python3.7/site-packages (from ipywidgets>=7.5.1->neuralprophet) (1.1.1)
Requirement already satisfied: ipython>=4.0.0 in /opt/conda/lib/python3.7/site-packages (from ipywidgets>=7.5.1->neuralprophet) (7.33.0)
Requirement already satisfied: ipykernel>=4.5.1 in /opt/conda/lib/python3.7/site-packages (from ipywidgets>=7.5.1->neuralprophet) (6.15.0)
Requirement already satisfied: ipython-genutils~=0.2.0 in /opt/conda/lib/python3.7/site-packages (from ipywidgets>=7.5.1->neuralprophet) (0.2.0)
Requirement already satisfied: traitlets>=4.3.1 in /opt/conda/lib/python3.7/site-packages (from ipywidgets>=7.5.1->neuralprophet) (5.3.0)
Requirement already satisfied: widgetsnbextension~=3.6.0 in /opt/conda/lib/python3.7/site-packages (from ipywidgets>=7.5.1->neuralprophet) (3.6.1)
Requirement already satisfied: ephem>=3.7.5.3 in /opt/conda/lib/python3.7/site-packages (from LunarCalendar>=0.0.9->neuralprophet) (4.1.3)
Requirement already satisfied: ephem>=3.7.5.3 in /opt/conda/lib/python3.7/site-packages (from LunarCalendar>=0.0.9->neuralprophet) (4.1.3)
Requirement already satisfied: pytz in /opt/conda/lib/python3.7/site-packages (from LunarCalendar>=0.0.9->neuralprophet) (2022.1)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->neuralprophet) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->neuralprophet) (1.4.3)
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->neuralprophet) (3.0.9)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->neuralprophet) (9.1.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.7/site-packages (from matplotlib>=2.0.0->neuralprophet) (4.33.3)
Requirement already satisfied: tenacity>=6.2.0 in /opt/conda/lib/python3.7/site-packages (from plotly>=4.14.3->neuralprophet) (8.0.1)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.7/site-packages (from python-dateutil>=2.8.0->neuralprophet) (1.15.0)
Requirement already satisfied: requests in /opt/conda/lib/python3.7/site-packages (from fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning==1.7.4->neuralprophet) (2.28.1)
Requirement already satisfied: aiohttp!=4.0.0a0,!=4.0.0a1 in /opt/conda/lib/python3.7/site-packages (from fsspec[http]!=2021.06.0,>=2021.05.0->pytorch-lightning==1.7.4->neuralprophet) (3.8.1)
Requirement already satisfied: psutil in /opt/conda/lib/python3.7/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.5.1->neuralprophet) (5.9.1)
Requirement already satisfied: pyzmq>=17 in /opt/conda/lib/python3.7/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.5.1->neuralprophet) (23.2.0)
Requirement already satisfied: jupyter-client>=6.1.12 in /opt/conda/lib/python3.7/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.5.1->neuralprophet) (7.3.4)
Requirement already satisfied: matplotlib-inline>=0.1 in /opt/conda/lib/python3.7/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.5.1->neuralprophet) (0.1.3)
Requirement already satisfied: tornado>=6.1 in /opt/conda/lib/python3.7/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.5.1->neuralprophet) (6.1)
Requirement already satisfied: debugpy>=1.0 in /opt/conda/lib/python3.7/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.5.1->neuralprophet) (1.6.0)
Requirement already satisfied: nest-asyncio in /opt/conda/lib/python3.7/site-packages (from ipykernel>=4.5.1->ipywidgets>=7.5.1->neuralprophet) (1.5.5)
Requirement already satisfied: decorator in /opt/conda/lib/python3.7/site-packages (from ipython>=4.0.0->ipywidgets>=7.5.1->neuralprophet) (5.1.1)
Requirement already satisfied: pickleshare in /opt/conda/lib/python3.7/site-packages (from ipython>=4.0.0->ipywidgets>=7.5.1->neuralprophet) (0.7.5)
Requirement already satisfied: backcall in /opt/conda/lib/python3.7/site-packages (from ipython>=4.0.0->ipywidgets>=7.5.1->neuralprophet) (0.2.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in /opt/conda/lib/python3.7/site-packages (from ipython>=4.0.0->ipywidgets>=7.5.1->neuralprophet) (3.0.30)
Requirement already satisfied: setuptools>=18.5 in /opt/conda/lib/python3.7/site-packages (from ipython>=4.0.0->ipywidgets>=7.5.1->neuralprophet) (59.8.0)
Requirement already satisfied: jedi>=0.16 in /opt/conda/lib/python3.7/site-packages (from ipython>=4.0.0->ipywidgets>=7.5.1->neuralprophet) (0.18.1)
Requirement already satisfied: pexpect>4.3 in /opt/conda/lib/python3.7/site-packages (from ipython>=4.0.0->ipywidgets>=7.5.1->neuralprophet) (4.8.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /opt/conda/lib/python3.7/site-packages (from tensorboard>=2.9.1->pytorch-lightning==1.7.4->neuralprophet) (1.35.0)
Requirement already satisfied: tensorboard-data-server<0.7.0,>=0.6.0 in /opt/conda/lib/python3.7/site-packages (from tensorboard>=2.9.1->pytorch-lightning==1.7.4->neuralprophet) (0.6.1)
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /opt/conda/lib/python3.7/site-packages (from tensorboard>=2.9.1->pytorch-lightning==1.7.4->neuralprophet) (1.8.1)
Requirement already satisfied: werkzeug>=1.0.1 in /opt/conda/lib/python3.7/site-packages (from tensorboard>=2.9.1->pytorch-lightning==1.7.4->neuralprophet) (2.2.2)
Requirement already satisfied: google-auth-oauthlib<0.5,>=0.4.1 in /opt/conda/lib/python3.7/site-packages (from tensorboard>=2.9.1->pytorch-lightning==1.7.4->neuralprophet) (0.4.6)
Requirement already satisfied: absl-py>=0.4 in /opt/conda/lib/python3.7/site-packages (from tensorboard>=2.9.1->pytorch-lightning==1.7.4->neuralprophet) (0.15.0)
Requirement already satisfied: markdown>=2.6.8 in /opt/conda/lib/python3.7/site-packages (from tensorboard>=2.9.1->pytorch-lightning==1.7.4->neuralprophet) (3.3.7)
```



```
[ ]  import pandas as pd
     from neuralprophet import NeuralProphet
     from matplotlib import pyplot as plt
     import pickle
```

```
[ ]  df = pd.read_csv('/kaggle/working/results/Historical_Monthly_Prep_2000_2020.csv')
     df.head()
```

```
      Month    Past_Precp_(mm/month)
0   2000-01                      0.2
1   2000-02                      1.6
2   2000-03                      0.1
3   2000-04                      0.2
4   2000-05                      9.7
```



```
[ ]  df.Month.unique()
```

```
array(['2000-01', '2000-02', '2000-03', '2000-04', '2000-05', '2000-06',
       '2000-07', '2000-08', '2000-09', '2000-10', '2000-11', '2000-12',
       '2001-01', '2001-02', '2001-03', '2001-04', '2001-05', '2001-06',
       '2001-07', '2001-08', '2001-09', '2001-10', '2001-11', '2001-12',
       '2002-01', '2002-02', '2002-03', '2002-04', '2002-05', '2002-06',
       '2002-07', '2002-08', '2002-09', '2002-10', '2002-11', '2002-12',
       '2003-01', '2003-02', '2003-03', '2003-04', '2003-05', '2003-06',
       '2003-07', '2003-08', '2003-09', '2003-10', '2003-11', '2003-12',
       '2004-01', '2004-02', '2004-03', '2004-04', '2004-05', '2004-06',
       '2004-07', '2004-08', '2004-09', '2004-10', '2004-11', '2004-12',
       '2005-01', '2005-02', '2005-03', '2005-04', '2005-05', '2005-06',
       '2005-07', '2005-08', '2005-09', '2005-10', '2005-11', '2005-12',
       '2006-01', '2006-02', '2006-03', '2006-04', '2006-05', '2006-06',
       '2006-07', '2006-08', '2006-09', '2006-10', '2006-11', '2006-12',
       '2007-01', '2007-02', '2007-03', '2007-04', '2007-05', '2007-06',
       '2007-07', '2007-08', '2007-09', '2007-10', '2007-11', '2007-12',
       '2008-01', '2008-02', '2008-03', '2008-04', '2008-05', '2008-06',
       '2008-07', '2008-08', '2008-09', '2008-10', '2008-11', '2008-12',
       '2009-01', '2009-02', '2009-03', '2009-04', '2009-05', '2009-06',
       '2009-07', '2009-08', '2009-09', '2009-10', '2009-11', '2009-12',
       '2010-01', '2010-02', '2010-03', '2010-04', '2010-05', '2010-06',
       '2010-07', '2010-08', '2010-09', '2010-10', '2010-11', '2010-12',
       '2011-01', '2011-02', '2011-03', '2011-04', '2011-05', '2011-06',
       '2011-07', '2011-08', '2011-09', '2011-10', '2011-11', '2011-12',
       '2012-01', '2012-02', '2012-03', '2012-04', '2012-05', '2012-06',
       '2012-07', '2012-08', '2012-09', '2012-10', '2012-11', '2012-12',
       '2013-01', '2013-02', '2013-03', '2013-04', '2013-05', '2013-06',
       '2013-07', '2013-08', '2013-09', '2013-10', '2013-11', '2013-12',
       '2014-01', '2014-02', '2014-03', '2014-04', '2014-05', '2014-06',
       '2014-07', '2014-08', '2014-09', '2014-10', '2014-11', '2014-12',
       '2015-01', '2015-02', '2015-03', '2015-04', '2015-05', '2015-06',
       '2015-07', '2015-08', '2015-09', '2015-10', '2015-11', '2015-12',
       '2016-01', '2016-02', '2016-03', '2016-04', '2016-05', '2016-06',
       '2016-07', '2016-08', '2016-09', '2016-10', '2016-11', '2016-12',
       '2017-01', '2017-02', '2017-03', '2017-04', '2017-05', '2017-06',
       '2017-07', '2017-08', '2017-09', '2017-10', '2017-11', '2017-12',
       '2018-01', '2018-02', '2018-03', '2018-04', '2018-05', '2018-06',
       '2018-07', '2018-08', '2018-09', '2018-10', '2018-11', '2018-12',
       '2019-01', '2019-02', '2019-03', '2019-04', '2019-05', '2019-06',
       '2019-07', '2019-08', '2019-09', '2019-10', '2019-11', '2019-12',
       '2020-01', '2020-02', '2020-03', '2020-04', '2020-05', '2020-06',
       '2020-07', '2020-08', '2020-09', '2020-10', '2020-11', '2020-12'],
      dtype=object)
```

```
[ ]  df.columns

     Index(['Month', 'Past_Precp_(mm/month)'], dtype='object')
```

```
[ ]  melb = df[df['Past_Precp_(mm/month)']>50]
     #melb['Year'] = pd.to_time(melb['Year'])
     melb.head()
```

	Month	Past_Precp_(mm/month)
5	2000-06	93.7
6	2000-07	643.1
7	2000-08	201.3
8	2000-09	84.4
17	2001-06	241.0

```
[ ]  plt.plot(melb['Month'], melb['Past_Precp_(mm/month)'])
     plt.show()
```

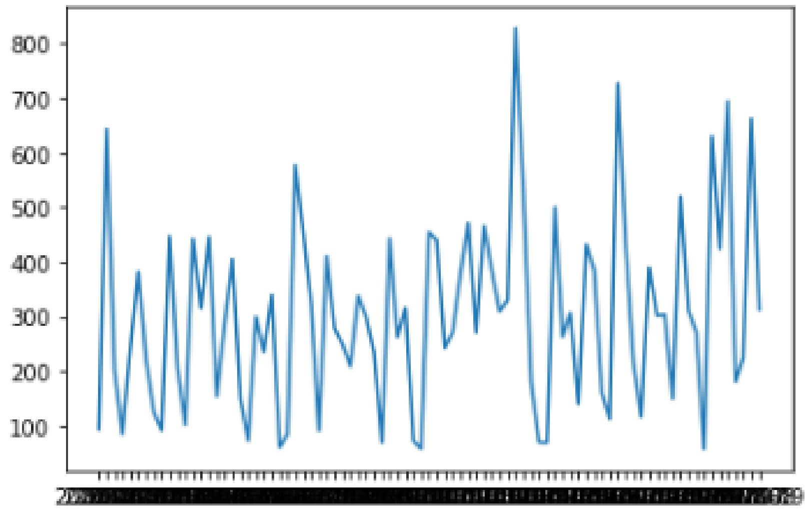
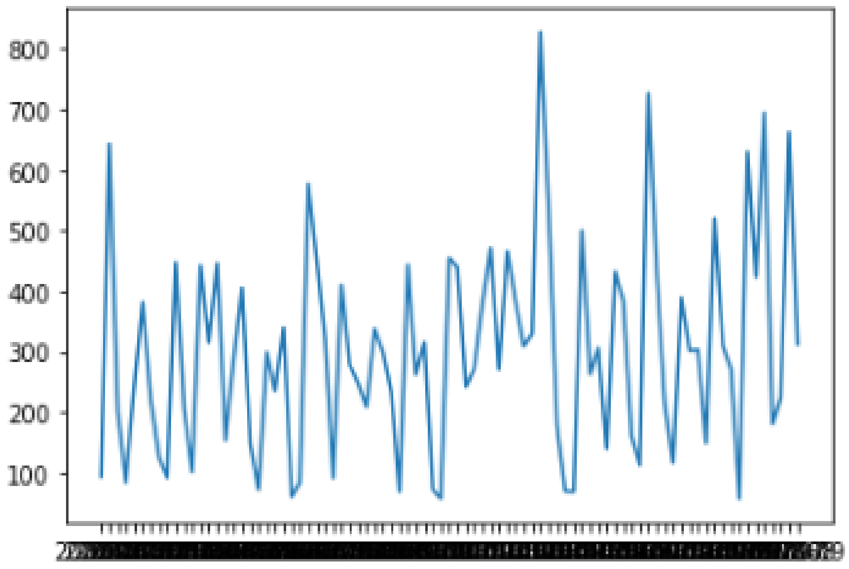


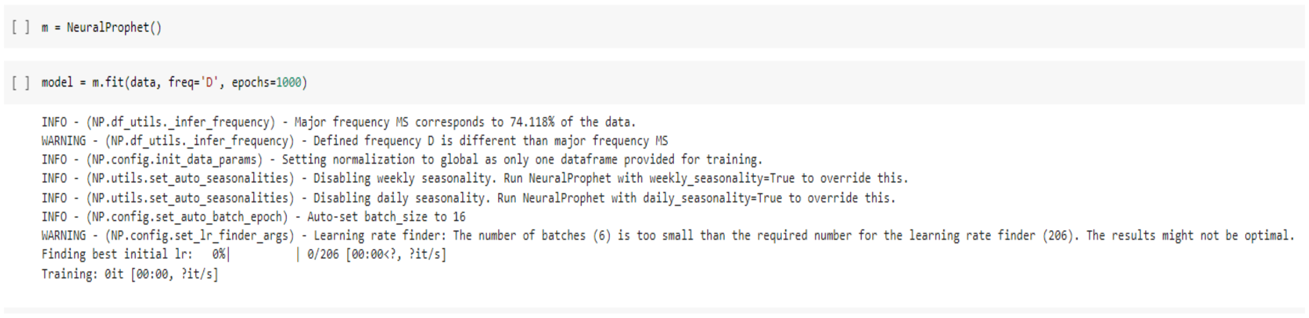
Fig 9: Monthly Precipitation

```
[ ]  #melb['Year'] = melb['Past_Precp_(mm/yr)'].apply(lambda x: x.Year)
     melb = melb[melb['Past_Precp_(mm/month)']>50]
     plt.plot(melb['Month'], melb['Past_Precp_(mm/month)'])
     plt.show()
```



```
[ ]  data = melb[['Month', 'Past_Precp_(mm/month)']]
     data.dropna(inplace=True)
     data.columns = ['ds', 'y']
     data.head()
```

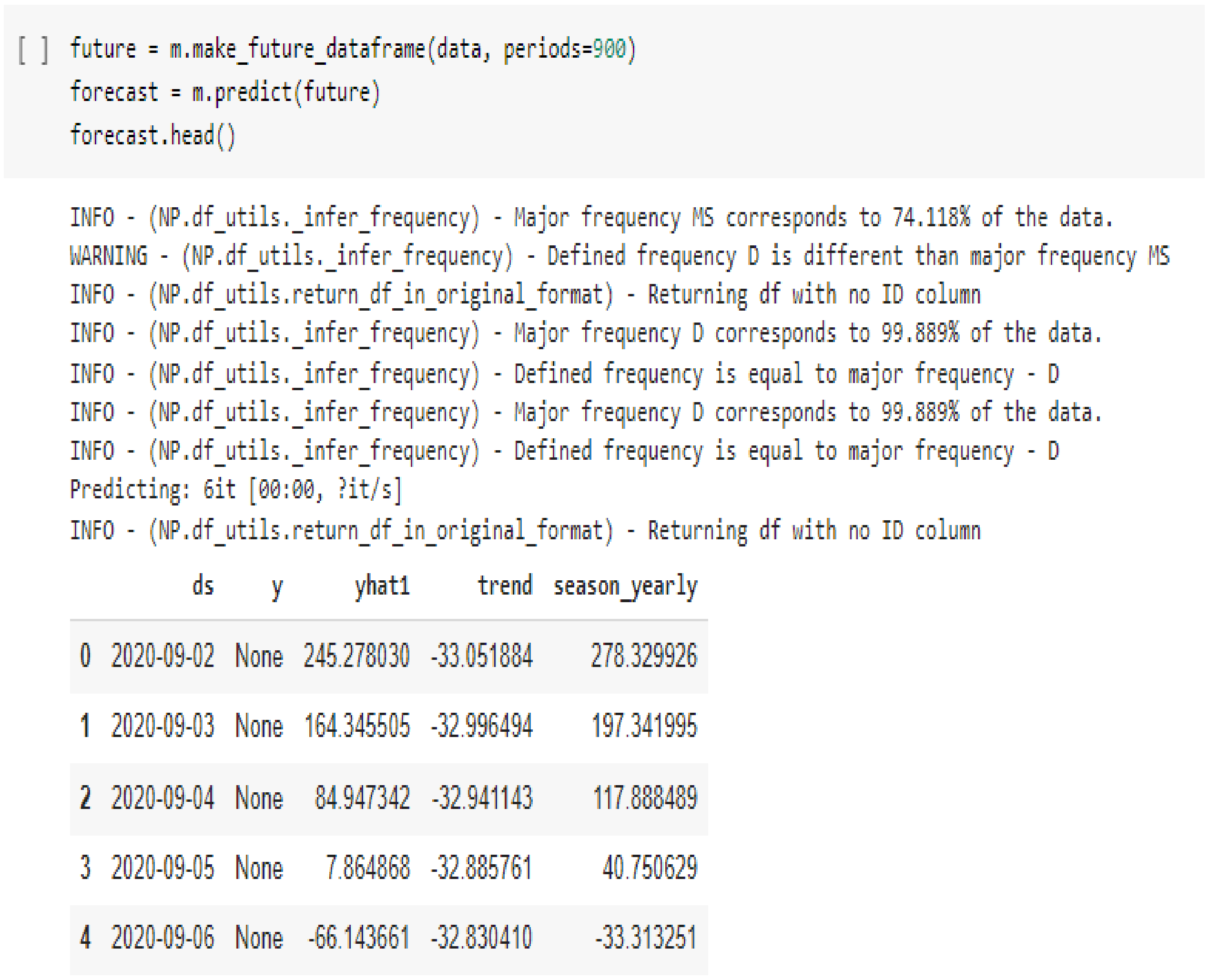
	ds	y
5	2000-06	93.7
6	2000-07	643.1
7	2000-08	201.3
8	2000-09	84.4
17	2001-06	241.0



```
[ ] m = NeuralProphet()
```

```
[ ] model = m.fit(data, freq='D', epochs=1000)
```

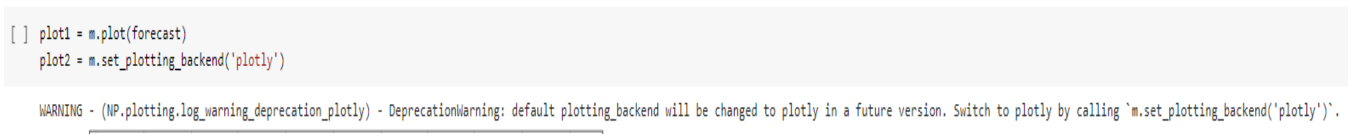
```
INFO - (NP.df_utils._infer_frequency) - Major frequency MS corresponds to 74.118% of the data.
WARNING - (NP.df_utils._infer_frequency) - Defined frequency D is different than major frequency MS
INFO - (NP.config.init_data_params) - Setting normalization to global as only one dataframe provided for training.
INFO - (NP.utils.set_auto_seasonalities) - Disabling weekly seasonality. Run NeuralProphet with weekly_seasonality=True to override this.
INFO - (NP.utils.set_auto_seasonalities) - Disabling daily seasonality. Run NeuralProphet with daily_seasonality=True to override this.
INFO - (NP.config.set_auto_batch_epoch) - Auto-set batch_size to 16
WARNING - (NP.config.set_lr_finder_args) - Learning rate finder: The number of batches (6) is too small than the required number for the learning rate finder (206). The results might not be optimal.
Finding best initial lr:   0%|          | 0/206 [00:00<?, ?it/s]
Training: 0it [00:00, ?it/s]
```



```
[ ] future = m.make_future_dataframe(data, periods=900)
    forecast = m.predict(future)
    forecast.head()
```

```
INFO - (NP.df_utils._infer_frequency) - Major frequency MS corresponds to 74.118% of the data.
WARNING - (NP.df_utils._infer_frequency) - Defined frequency D is different than major frequency MS
INFO - (NP.df_utils.return_df_in_original_format) - Returning df with no ID column
INFO - (NP.df_utils._infer_frequency) - Major frequency D corresponds to 99.889% of the data.
INFO - (NP.df_utils._infer_frequency) - Defined frequency is equal to major frequency - D
INFO - (NP.df_utils._infer_frequency) - Major frequency D corresponds to 99.889% of the data.
INFO - (NP.df_utils._infer_frequency) - Defined frequency is equal to major frequency - D
Predicting: 6it [00:00, ?it/s]
INFO - (NP.df_utils.return_df_in_original_format) - Returning df with no ID column
```

	ds	y	yhat1	trend	season_yearly
0	2020-09-02	None	245.278030	-33.051884	278.329926
1	2020-09-03	None	164.345505	-32.996494	197.341995
2	2020-09-04	None	84.947342	-32.941143	117.888489
3	2020-09-05	None	7.864868	-32.885761	40.750629
4	2020-09-06	None	-66.143661	-32.830410	-33.313251



```
[ ] plot1 = m.plot(forecast)
    plot2 = m.set_plotting_backend('plotly')
```

```
WARNING - (NP.plotting.log_warning_deprecation_plotly) - DeprecationWarning: default plotting_backend will be changed to plotly in a future version. Switch to plotly by calling `m.set_plotting_backend('plotly')`.
```

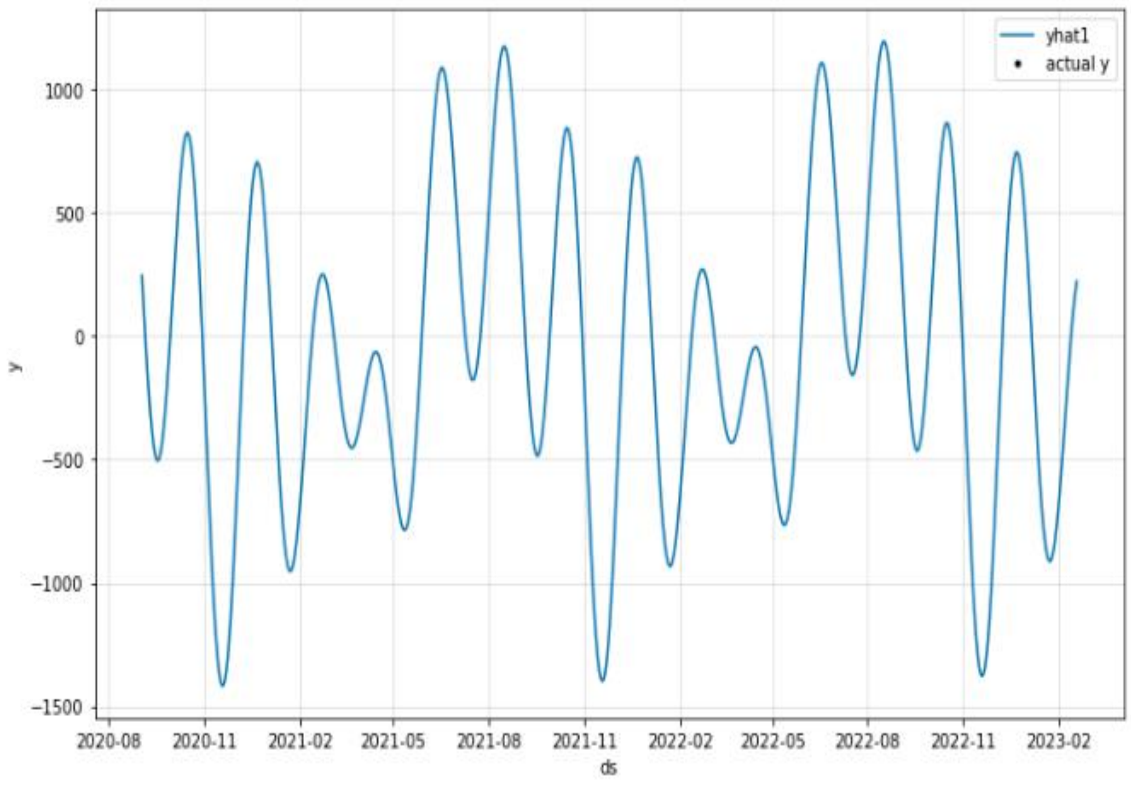
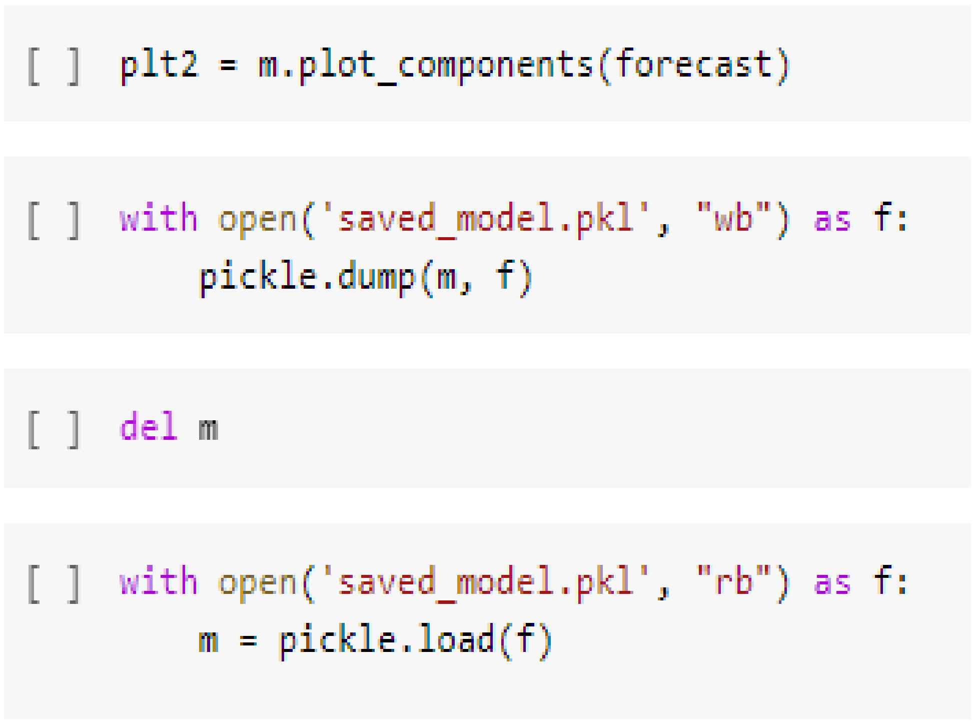


Fig 10: Difference in Prediction



```
[ ]  plt2 = m.plot_components(forecast)
```

```
[ ]  with open('saved_model.pkl', "wb") as f:
         pickle.dump(m, f)
```

```
[ ]  del m
```

```
[ ]  with open('saved_model.pkl', "rb") as f:
         m = pickle.load(f)
```

```
[ ]  future = m.make_future_dataframe(data, periods=900)

     INFO - (NP.df_utils._infer_frequency) - Major frequency MS corresponds to 74.118% of the data.
     WARNING - (NP.df_utils._infer_frequency) - Defined frequency D is different than major frequency MS
     INFO - (NP.df_utils.return_df_in_original_format) - Returning df with no ID column
```

```
[ ]  plot1 = m.plot(forecast)
```

```
[ ]  os.chdir("/kaggle/working/Prep")

     p = xr.open_mfdataset("/kaggle/working/Prep/TerraClimate_ppt_2015.nc")
     f = xr.open_mfdataset("/kaggle/working/Prep/TerraClimate_4c_ppt_2015.nc")

     f
```

```
xarray.Dataset
▸ Dimensions:        (lat: 4320, lon: 8640, time: 12, crs: 1)
▾ Coordinates:
     lat             (lat)                 float64  89.98 89.94 89.9 ... -89.94 -89.98
     lon             (lon)                 float64  -180.0 -179.9 ... 179.9 180.0
     time            (time)        datetime64[ns]  2015-01-01 ... 2015-12-01
     crs             (crs)                   int16  3
▾ Data variables:
     ppt             (time, lat, lon)      float32  dask.array<chunksize=(12, 4320, 8640), meta...
▸ Attributes:   (0)
```

```
[ ]  p_mon = p.groupby('time.month').sum()
     p_mon.ppt[0,:,:].plot(cmap='jet', vmax=300)

     <matplotlib.collections.QuadMesh at 0x7f12abdadbd0>
```

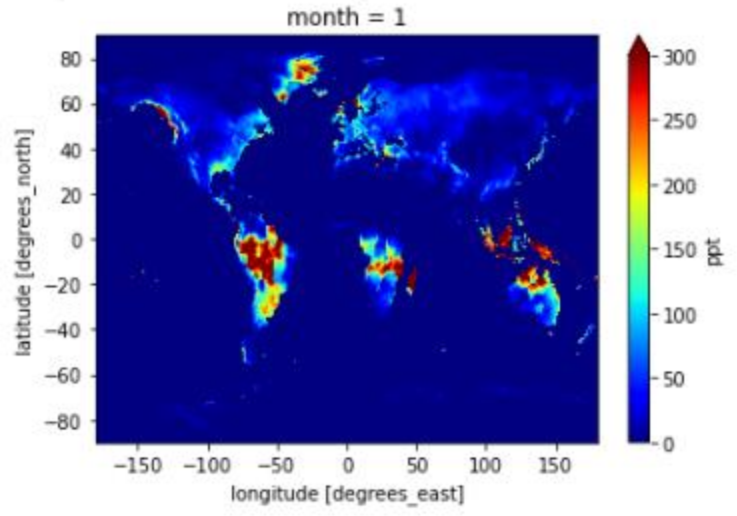


Fig 11: Radiance Map

2.4 Standards

Various standards used in this project are:

- **Automatic Weather Station (AWS)**

  An automatic weather station (AWS) is an automated version of the traditional weather station, either to save human labor or to enable measurements from remote areas. An AWS will typically consist of a weather-proof enclosure containing the data logger, rechargeable battery, telemetry (optional) and the meteorological sensors with an attached solar panel or wind turbine and mounted upon a mast. The specific configuration may vary due to the purpose of the system. The system may report in near real time via the Argos System, LoRa and the Global Telecommunications System, or save the data for later recovery.

- **Neural Prophet**

  Presented in a user-friendly Python package, Neural Prophet uses a fusion of classic components and neural networks to produce highly accurate time series forecasts quickly. Current Prophet users will find the package to be familiar in design.

  The framework provides automatic hyperparameter selection, making it a convenient and accessible tool for beginners. Advanced forecasting practitioners can incorporate domain knowledge and leverage deeper expertise with a superset of custom modules, model weight scarification, and global modeling capabilities.

2.5 System Details

This section describes the software and hardware details of the system:

2.5.1 Software Details

Kaggle and Google colab is used for the project.

i) Kaggle

Kaggle is a platform for data science and machine learning that provides a variety of resources and tools for data scientists, including datasets, competitions, and online courses. It is often used as a resource for practicing and learning data science skills, as well as for finding and sharing data science resources.

Kaggle is home to a large community of data scientists and machine learning practitioners who use the platform to collaborate on projects, share code and resources, and compete in machine learning competitions. The platform also hosts a variety of datasets, ranging from small and specific datasets to large and comprehensive datasets, which can be used for data analysis and machine learning tasks.

In addition to its core features, Kaggle also provides a range of tools and services for data scientists, including access to GPUs and TPUs for training machine learning models, as well as tools for working with data and visualizing results. It is a popular choice among data scientists and machine learning practitioners as it provides a convenient and easy-to-use platform for developing and running code, as well as for finding and working with data.

ii) Google Colab

Google Colab is a free online platform that allows users to run and execute code in a variety of programming languages, including Python, R, and TensorFlow. It provides a Jupyter notebook-style development environment that runs in the cloud, allowing users to write, run, and share code, as well as collaborate with other users in real-time.

Google Colab is designed for data science and machine learning tasks and provides access to a variety of resources, including GPUs and TPUs, which can be used to accelerate the training of machine learning models. It also integrates with a number of Google services, such as Google Drive and Google Sheets, making it easy to access and work with data stored in these platforms.

Google Colab is a popular choice among data scientists and machine learning practitioners as it provides a convenient and easy-to-use platform for developing and running code. It is particularly useful for prototyping and testing code, as well as for sharing code and results with others.

CHAPTER 3

RESULT & DISCUSSION

In this study, we evaluated the relative impacts of different rainfall patterns around the world during the past decade and how these trends affect our prediction.

Using this knowledge, we were able to incorporate a model which analyzes the changes in these trends to improve its accuracy. This enables us to precisely forecast and examine variations in the pattern of rainfall over a specific region anywhere in the world.

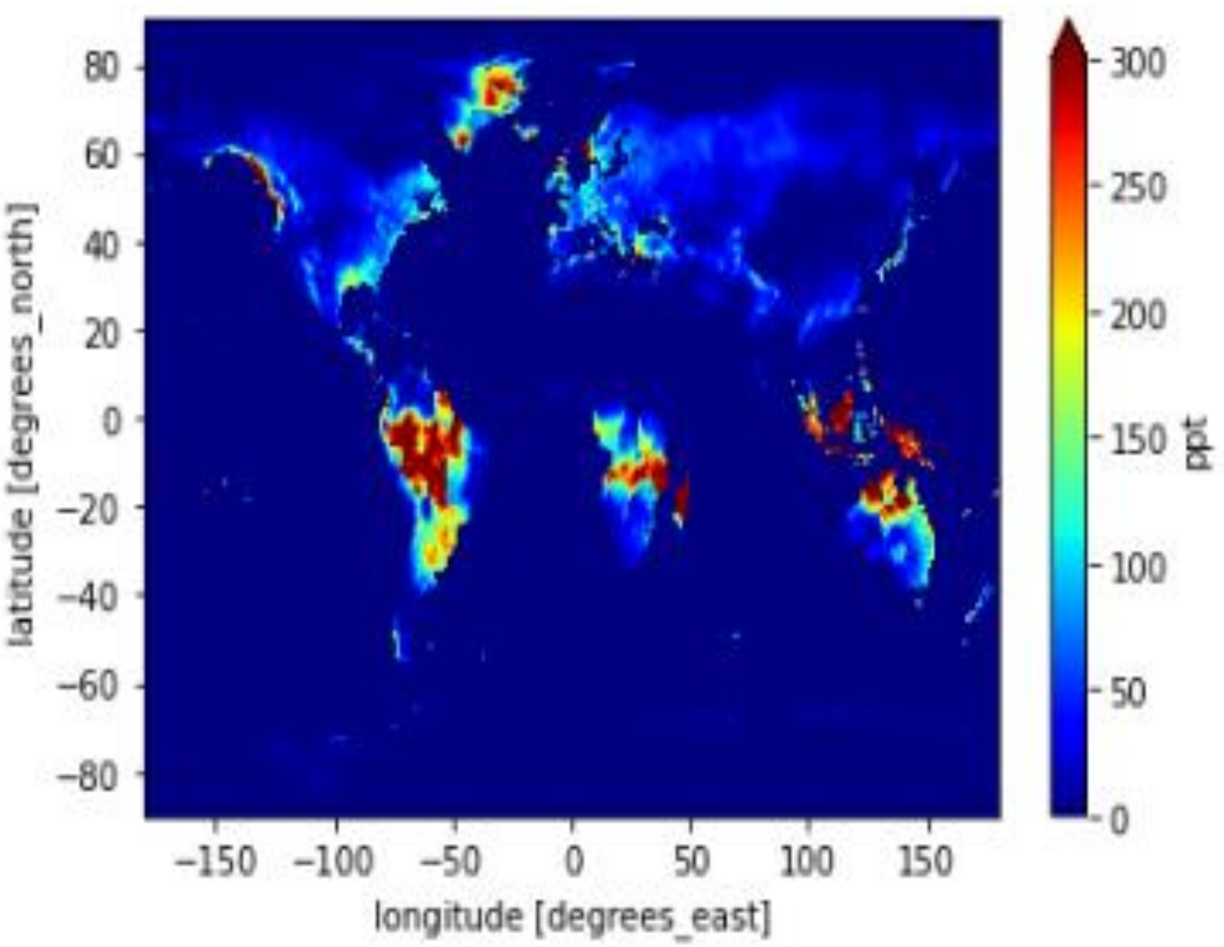
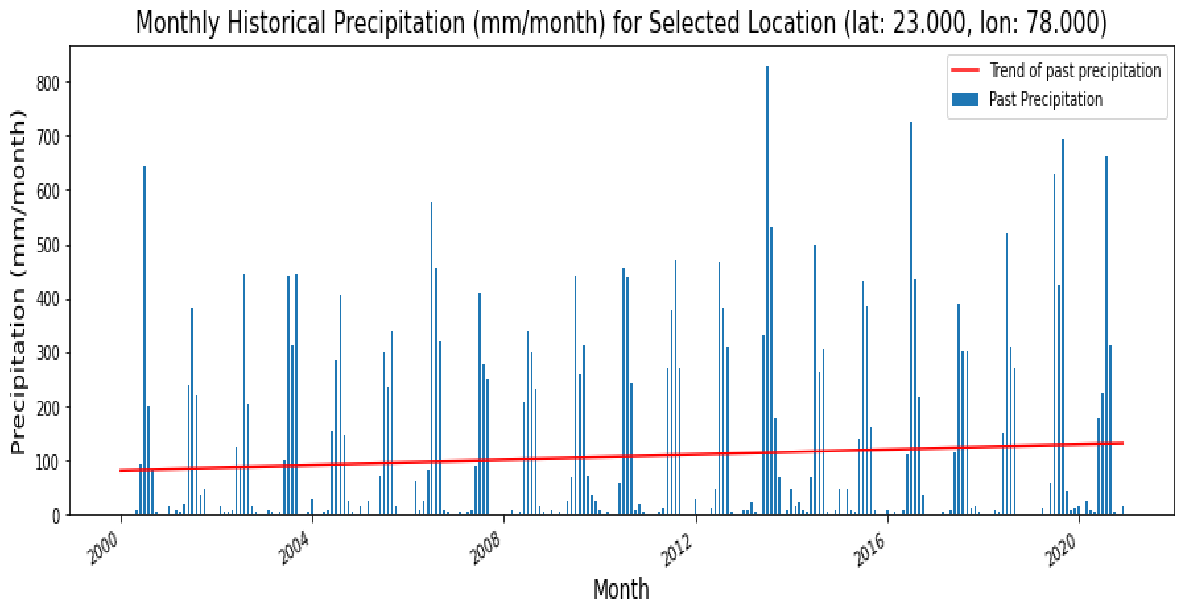


Fig: 12 Difference in Rainfall Radiance Map

This Radiance Map helps us understand the difference in rainfall pattern over the past decade here we can see there is a large difference in pattern near the equatorial region along with the regions with extreme cold climate.

Our model can spot these differences and learn from them to produce accurate predictions wherever necessary.



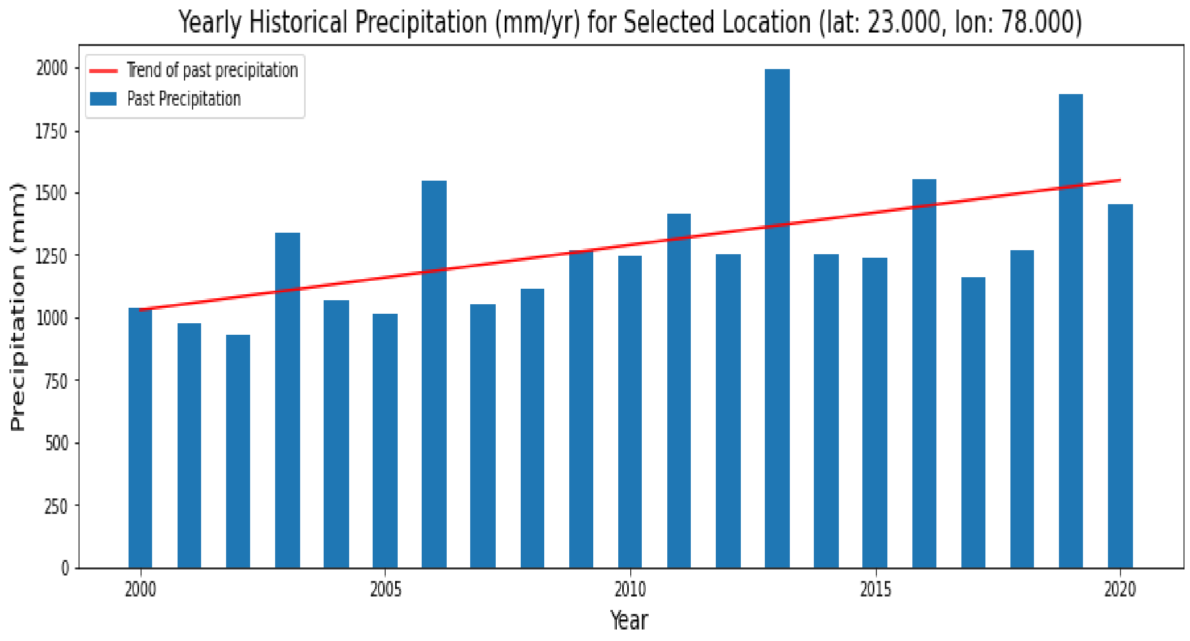


Fig: 13 Change in Rainfall Pattern over the Decade

We also came to the conclusion that some locations have seen a sharp rise in rainfall, which renders the earlier forecast models invalid because they are based on outdated information that is incongruent with the increase in rainfall for these areas. Our model aids in the analysis of these trends, and if there has been a sharp increase over the previous ten years, it uses that information in its prediction to improve accuracy and prevent overfitting. If the trends are consistent, we are able to use the old data as per usual for our prediction.

There are several key advantages to using assimilation techniques in weather prediction systems:

1. **Improved accuracy:** Assimilation techniques allow meteorologists to incorporate observations of the weather into the computer model in a more systematic and accurate way, which can help to reduce the error in the model's forecast and improve the accuracy of the forecast.
2. **Continuous updating:** Assimilation techniques allow the model to be continuously updated and improved as new observations become available, which can help to reduce the error in the model's forecast and improve the accuracy of the forecast.
3. **Increased realism:** By incorporating observations of the weather into the model, assimilation techniques can help to improve the realism of the model's forecast, particularly in regions where there are fewer observations available.
4. **Increased computational efficiency:** Assimilation techniques can be more computationally efficient than other methods of incorporating observations into the model, as they allow the model to be updated without the need to re-run the entire simulation.

Despite these advantages, there are also limitations to using assimilation techniques in weather prediction systems. For example, the accuracy of the forecast may be limited by the quality and quantity of the observations used, as well as the complexity and realism of the computer model. In addition, assimilation techniques may be less effective for forecasting weather events that are poorly represented by the computer model.

The accuracy of a weather prediction system using assimilation techniques depends on a variety of factors, including the quality and quantity of the observations used, the complexity and realism of the computer model, and the effectiveness of the assimilation technique being used. In general, weather prediction systems using assimilation techniques can produce highly accurate forecasts, particularly for weather events that are well-represented by the computer model.

One of the key benefits of using assimilation techniques in weather prediction systems is that they allow the model to be continuously updated and improved as new observations become available. This can help to reduce the error in the model's forecast and improve the accuracy of the forecast.

However, it is important to note that weather prediction is a complex and challenging task, and there are always limits to the accuracy of weather forecasts. Factors such as the inherent variability of the weather and the complexity of the physical processes that govern the weather can make it difficult to produce highly accurate forecasts. As a result, weather prediction systems using assimilation techniques are generally best at forecasting weather events on a relatively short time scale (e.g., a few days) and may be less accurate for longer-term forecasts.

CHAPTER 4

CONCLUSION AND FUTURE WORK

In conclusion, weather prediction systems using assimilation techniques are an effective tool for forecasting the weather. By combining observations of the weather with computer models, these systems can produce accurate forecasts of the weather, particularly for weather events that are well-represented by the computer model.

Assimilation techniques play a crucial role in these systems by allowing the model to be continuously updated and improved as new observations become available, which can help to reduce the error in the model's forecast and improve the accuracy of the forecast. Overall, weather prediction systems using assimilation techniques are a valuable resource for a wide range of organizations, including government agencies, aviation companies, and energy utilities, as they help these organizations to make informed decisions and plan for potential weather events. Despite their limitations, these systems are an important tool for improving our understanding of the weather and helping us to prepare for and respond to potential weather events.

The prediction skill of numerical weather and climate models is mainly dependent on accurate model’s initial conditions (ICs). The generation of model ICs is tightly reliant on weather observations from Automated Weather Stations (AWS), ground weather radar network, and weather satellites. From our research we observed that there has been a drastic change in rainfall patterns over the previous decade due to multiple factors such as rise in temperature, variation in pressure and annual rainfall.

Lot can be done in this area. There is a large scope which could be ventured, and new designs or systems could be made to improve the conditions and efficiency of the vehicles and by using AI we can figure out if in near future any of the components might need attention.

CHAPTER 5

APPENDIX

1. 3DVar - Three dimensional variational
2. 4DVar - Four dimensional variational
3. AWS - Automated Weather Stations
4. CSV - Comma-separated values
5. EnKF - Ensemble Kalvin Filter
6. Folium - Python Library to create several types of maps
7. GPM - Global Precipitation Measurement
8. IC - Initial Conditions
9. INSAT - Indian National Satellite System
10. INSAT 3DR - Indian National Satellite 3D repeat.
11. netCDF4 - Network Common Data Form (Disk Format)
12. Neural Prophet - Python Library for autoregression and lagged covariance
13. NWP - Numerical Weather Prediction
14. WRF - Weather Research and Forecasting
15. XArray - Python package for dealing with netCDF4 and numpy

CHAPTER 6

REFERENCES

[1] Stensrud, D. J., Xue, M., Wicker, L. J., Kelleher, K. E., Foster, M. P., Schaefer, J. T., ... & Tuell, J. P. (2009). Convective-scale warn-on-forecast system: A vision for 2020. *Bulletin of the American Meteorological Society*, *90*(10), 1487-1500.

[2] Hartman, C.M., Chen, X. and Chan, M.Y., 2022. Improving Tropical Cyclogenesis Forecasts of Hurricane Irma (2017) through the Assimilation of All-Sky Infrared Brightness Temperatures. *Monthly Weather Review*.

[3] Chawang, N., & Kutty, G. (2022). Ensemble-based forecast sensitivity approach to estimate the impact of satellite-derived atmospheric motion vectors in a limited area model. *Journal of Earth System Science*, *131*(4), 1-11.

[4] Bilgiç, E., Tuygun, G. T., & Gündüz, O. (2022). Determination of Air Pollution from Wildfires with Satellite Observations. *Ahmet ÖZTOPAL Melek AKIN Abdurrahman DURMAZ*, 133.

[5] Honda, T., Amemiya, A., & Miyoshi, T. (2022). *Implications of a 30-second Update Cycle for a Convective-Scale Ensemble Radar Data Assimilation System* (No. EGU22-9244). Copernicus Meetings.

[6] Yeh, H. L., Yang, S. C., Terasaki, K., Miyoshi, T., & Liou, Y. C. (2022). Including observation error correlation for ensemble radar radial wind assimilation and its impact on heavy rainfall prediction. *Quarterly Journal of the Royal Meteorological Society*.

[7] Thodsan, T., Wu, F., Torsri, K., Cuestas, E. M. A., & Yang, G. (2022). Satellite Radiance Data Assimilation Using the WRF-3DVAR System for Tropical Storm Dianmu (2021) Forecasts. *Atmosphere*, *13*(6), 956.

[8] Reimann, L., Simmer, C., Potthast, R., & Trömel, S. (2022). *On the assimilation of dual-polarization radar observations via estimators of hydrometeor mixing ratios in Germany* (No. EMS2022-197). Copernicus Meetings.

[9] Tan, P. H., Soong, W. K., Tsao, S. J., Chen, W. J., & Chen, I. H. (2022). Impact of Lidar Data Assimilation on Simulating Afternoon Thunderstorms near Pingtung Airport, Taiwan: A Case Study. *Atmosphere*, *13*(9), 1341.

[10] Lei, L., Ge, Y., Tan, Z. M., Zhang, Y., Chu, K., Qiu, X., & Qian, Q. (2022). Evaluation of a Regional Ensemble Data Assimilation System for Typhoon Prediction. *Advances in Atmospheric Sciences*, *39*(11), 1816-1832.

[11] Eyre, J. R., English, S. J., & Forsythe, M. (2020). Assimilation of satellite data in numerical weather prediction. Part I: The early years. *Quarterly Journal of the Royal Meteorological Society*, *146*(726), 49-68.

[12] Wei, S. W., Lu, C. H., Liu, Q., Collard, A., Zhu, T., Grogan, D., ... & Bhattacharjee, P. S. (2021). The impact of aerosols on satellite radiance data assimilation using NCEP global data assimilation system. *Atmosphere*, *12*(4), 432.

[13] Schultz, M. G., Betancourt, C., Gong, B., Kleinert, F., Langguth, M., Leufen, L. H., ... & Stadtler, S. (2021). Can deep learning beat numerical weather prediction? *Philosophical Transactions of the Royal Society A*, *379*(2194), 20200097.

[14] Ren, X., Li, X., Ren, K., Song, J., Xu, Z., Deng, K., & Wang, X. (2021). Deep learning-based weather prediction: a survey. *Big Data Research*, *23*, 100178.

[15] Liu, C., Zhang, X., Mei, S., Zhen, Z., Jia, M., Li, Z., & Tang, H. (2022). Numerical weather prediction enhanced wind power forecasting: Rank ensemble and probabilistic fluctuation awareness. *Applied Energy*, *313*, 118769.

[16] Zhao, P., Wang, Q. J., Wu, W., & Yang, Q. (2022). Extending a joint probability modelling approach for post-processing ensemble precipitation forecasts from numerical weather prediction models. *Journal of Hydrology*, *605*, 127285.

[17] Gad, I., & Hosahalli, D. (2022). A comparative study of prediction and classification models on NCDC weather data. *International Journal of Computers and Applications*, *44*(5), 414-425.

[18] Pope, J. O., Brown, K., Fung, F., Hanlon, H. M., Neal, R., Palin, E. J., & Reid, A. (2022). Investigation of future climate change over the British Isles using weather patterns. *Climate Dynamics*, *58*(9), 2405-2419.

[19] Zhang, K., & Zou, G. (2022, August). Photovoltaic Output Prediction Method Based on Weather Forecast and Machine Learning. In *Journal of Physics: Conference Series* (Vol. 2320, No. 1, p. 012032). IOP Publishing.

[20] Moosavi, A., Rao, V., & Sandu, A. (2021). Machine learning based algorithms for uncertainty quantification in numerical weather prediction models. *Journal of Computational Science*, *50*, 101295.

[21] Ren, X., Li, X., Ren, K., Song, J., Xu, Z., Deng, K., & Wang, X. (2021). Deep learning-based weather prediction: a survey. *Big Data Research*, *23*, 100178.

[22] Sønderby, C. K., Espeholt, L., Heek, J., Dehghani, M., Oliver, A., Salimans, T., ... & Kalchbrenner, N. (2020). Metnet: A neural weather model for precipitation forecasting. *arXiv preprint arXiv:2003.12140*.

[23] Chattopadhyay, A., Nabizadeh, E., & Hassanzadeh, P. (2020). Analog forecasting of extreme-causing weather patterns using deep learning. *Journal of Advances in Modeling Earth Systems*, *12*(2), e2019MS001958.

[24] Bannister, R. N., Chipilski, H. G., & Martinez-Alvarado, O. (2020). Techniques and challenges in the assimilation of atmospheric water observations for numerical weather prediction towards convective scales. *Quarterly Journal of the Royal Meteorological Society*, *146*(726), 1-48.

[25] Abatzoglou, J. T., Dobrowski, S. Z., Parks, S. A., & Hegewisch, K. C. (2018). TerraClimate, a high-resolution global dataset of monthly climate and climatic water balance from 1958–2015. *Scientific data*, *5*(1), 1-12.

[26] Qin, Y., Abatzoglou, J. T., Siebert, S., Huning, L. S., AghaKouchak, A., Mankin, J. S., ... & Mueller, N. D. (2020). Agricultural risks from changing snowmelt. *Nature Climate Change*, *10*(5), 459-465.

BIODATA OF ALL THE TEAM MEMBERS:

NAME: ABHAY CHAUDHARY
Mobile no. +91 94144 21646
Email : abhay.19bce7290@vitap.ac.in
Permanent Address: Jaipur, Rajasthan, India

NAME: DHRITI RANJAN
Mobile no. +91 8790261135
Email : ranjan.19bce7249@vitap.ac.in
Permanent Address: Bangalore, Karnataka, India

NAME: HRIDESH SINGH
Mobile no. +91 90016 85921
Email : hridesh.19bce7102@vitap.ac.in
Permanent Address: Delhi, India

Google drive link for supplementary material:

https://drive.google.com/drive/folders/1WH-3eqN7y5P_p1pWb8sP7dM_GGcF8JmX?usp=share_link