Configuration Manual

MSc Research Project
MSc Data Analytics

Shraddha Eknath Gargote
Student ID: X23106263

School of Computing
National College of Ireland

1 Supervisor: Dr Catherine Mulwa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Shraddha Eknath Gargote

Student ID: X23106263

Programme: MSc Data Analytics **Year:** 2023-2024

Module: MSc Research Project

Supervisor: Dr Catherine Mulwa

Submission Due Date: 2nd September 2024

Project Title: Real-Time Wildfire Progression Analysis and Prediction using Hybrid Model

Word Count: **919** **Page Count: 31**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Shraddha Eknath Gargote

Date: 31/08/2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	YES
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	YES
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	YES

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Shraddha Ekath Gargote

X23106263

1. System Configuration

The project is done on 64-Bits windows 11operating system with 8 GB ram with intel(R) core(TM) i5-8300H CPU @ 2.30 GHz

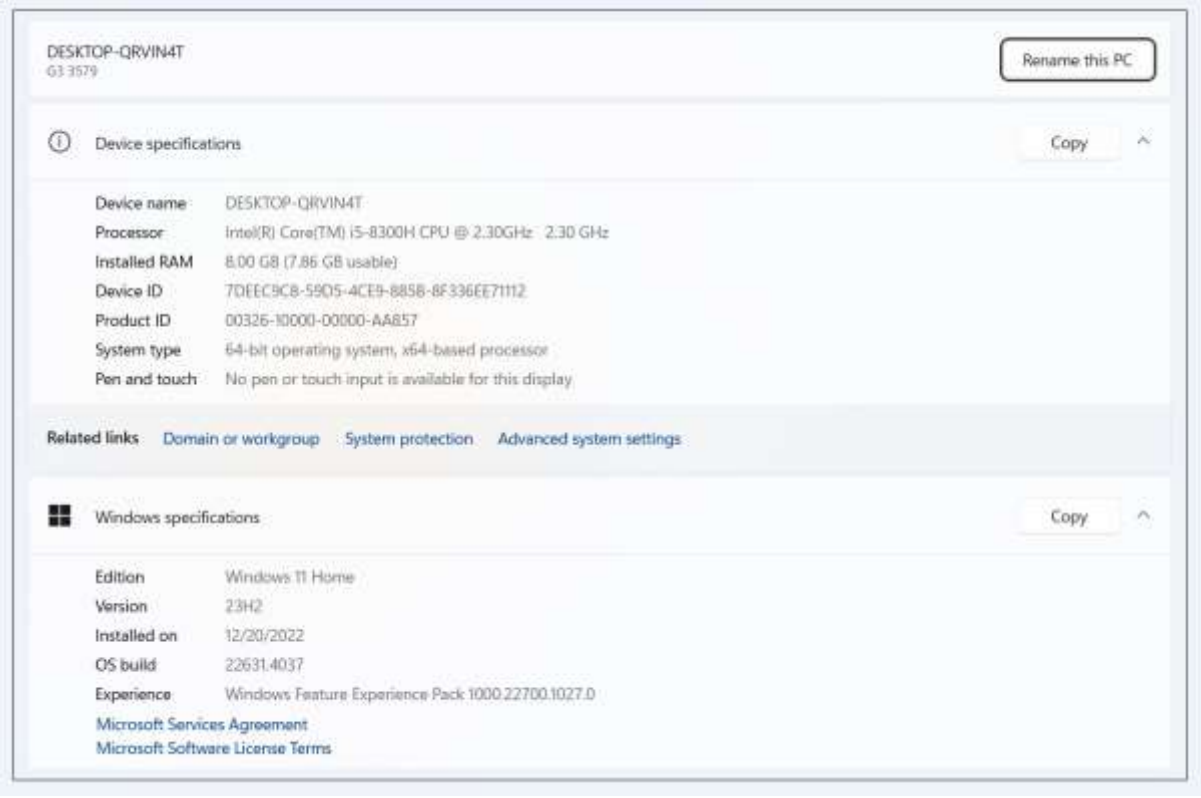


Figure 1. System Configuration

2. Software Requirements

For the project we have used following software

1. Anaconda 2.3.3
2. Python 3.9.19
3. Jupyter Notebook
4. Google Colab
5. Spyder 5.5.1

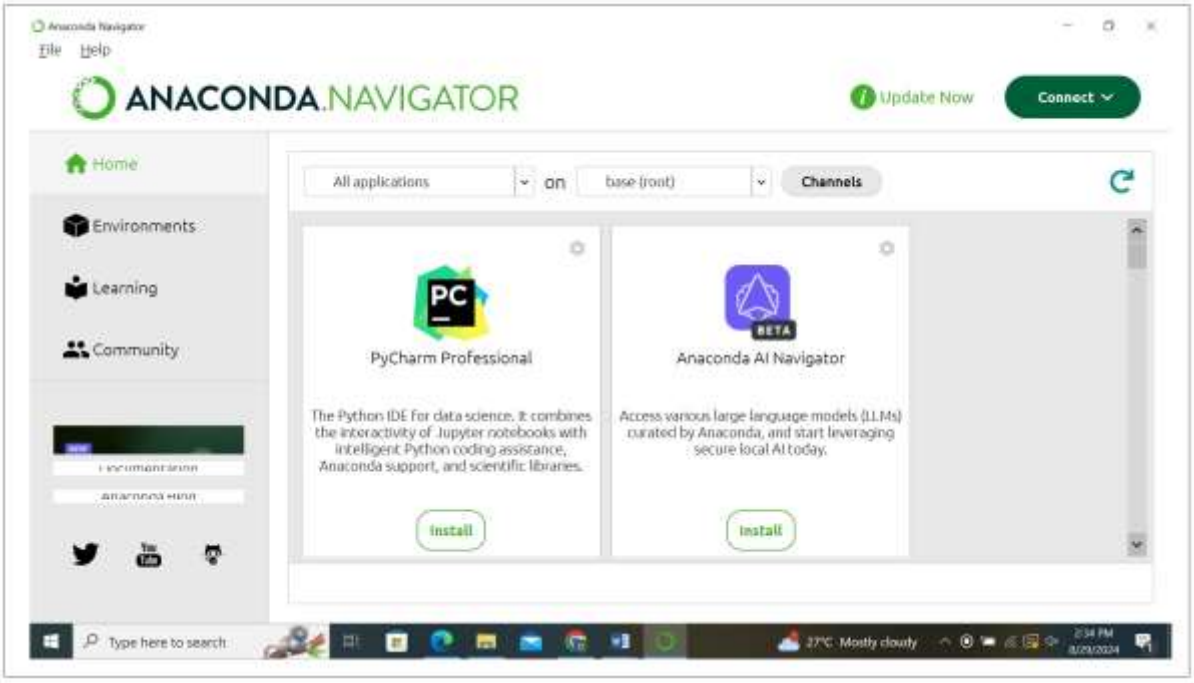


Figure 2. Anaconda Navigator

3. Creating Environment in anaconda navigator

Figure 3. Creating Environment

4. Anaconda Navigator CMD to install libraries

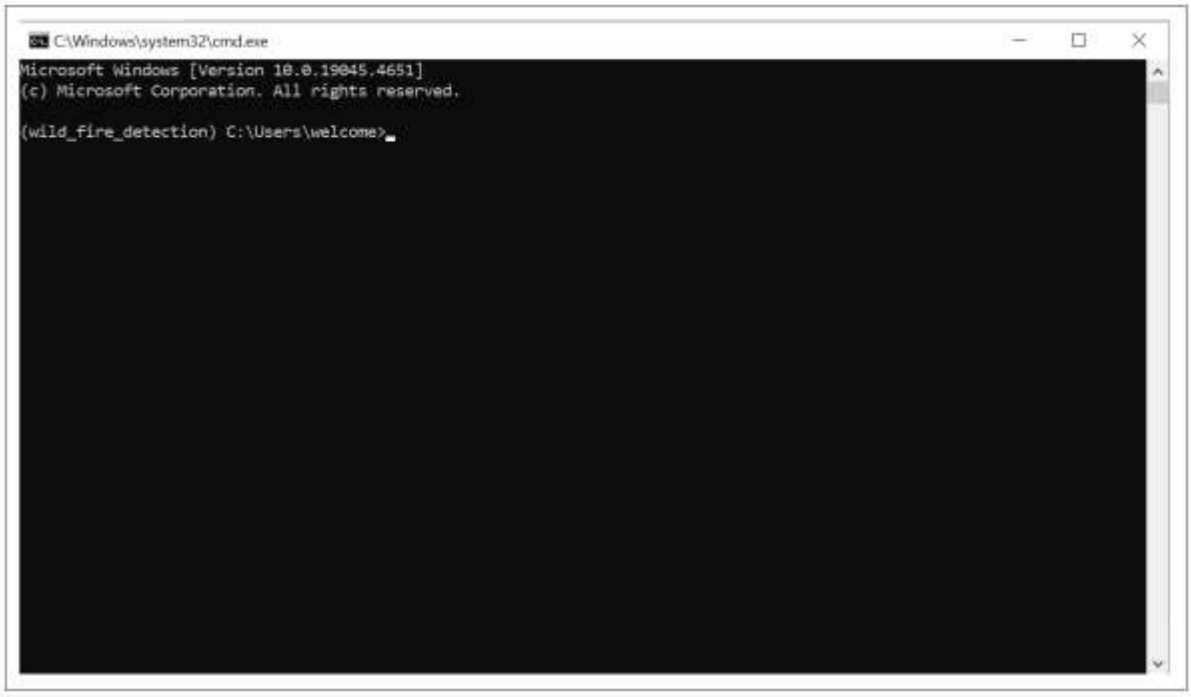


Figure 4. CMD.exe console to install the libraries

5. Anaconda Navigator Spyder IDE

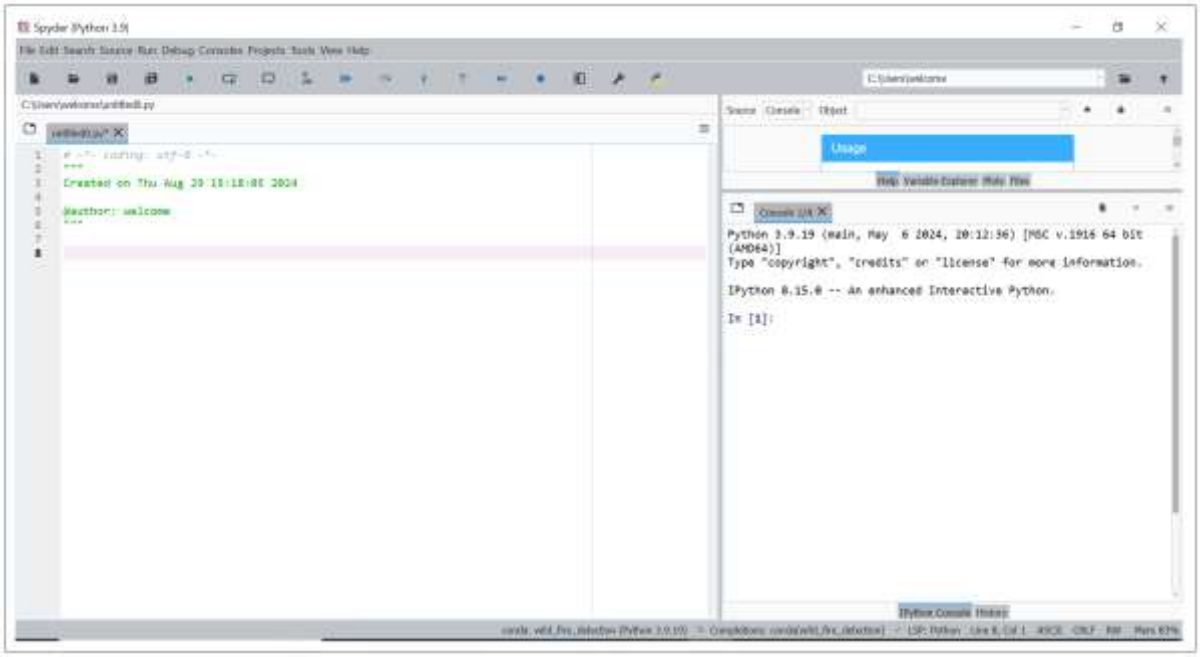


Figure 5. Spyder IDE 5.5.1

6. Creation of google colab notebook

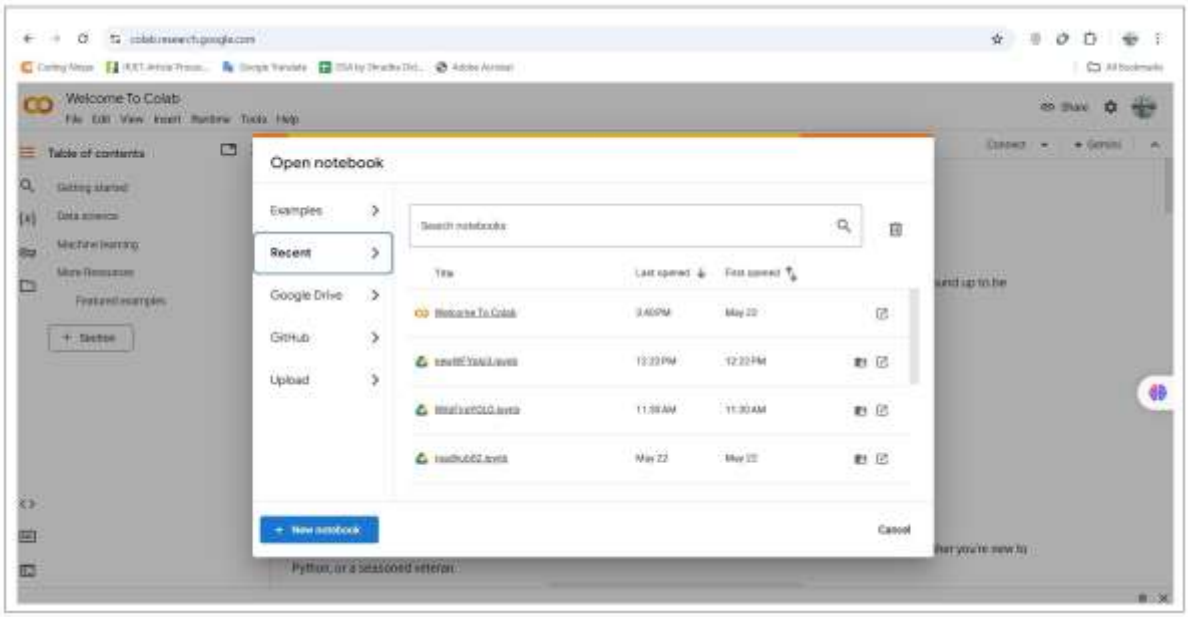


Figure 6. Creating google colab new notebook

7. Google colab new notebook view

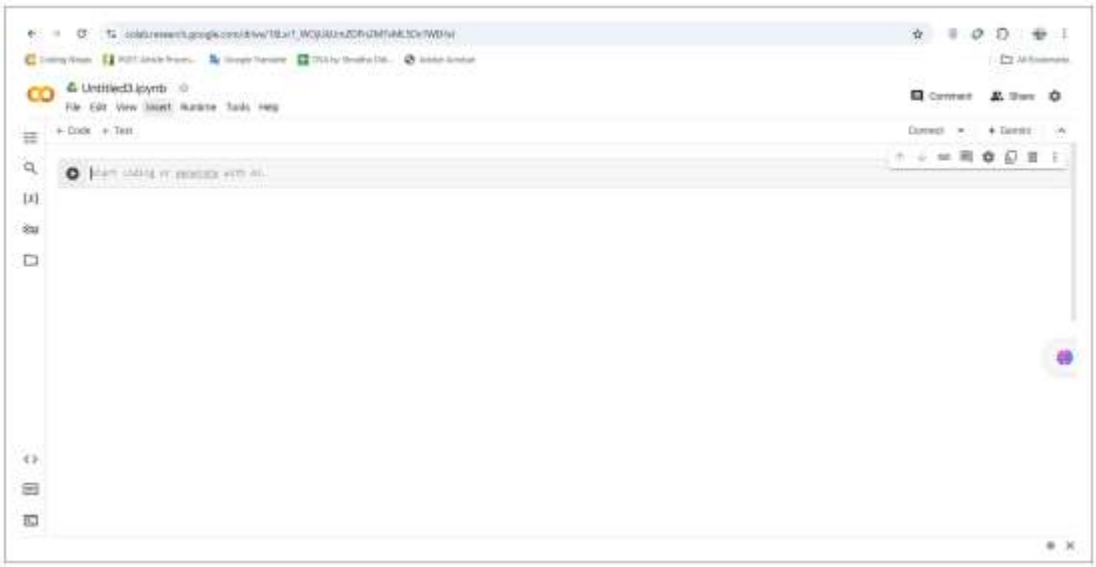


Figure 7. Google colab new notebook view

8. Selection of runtime type

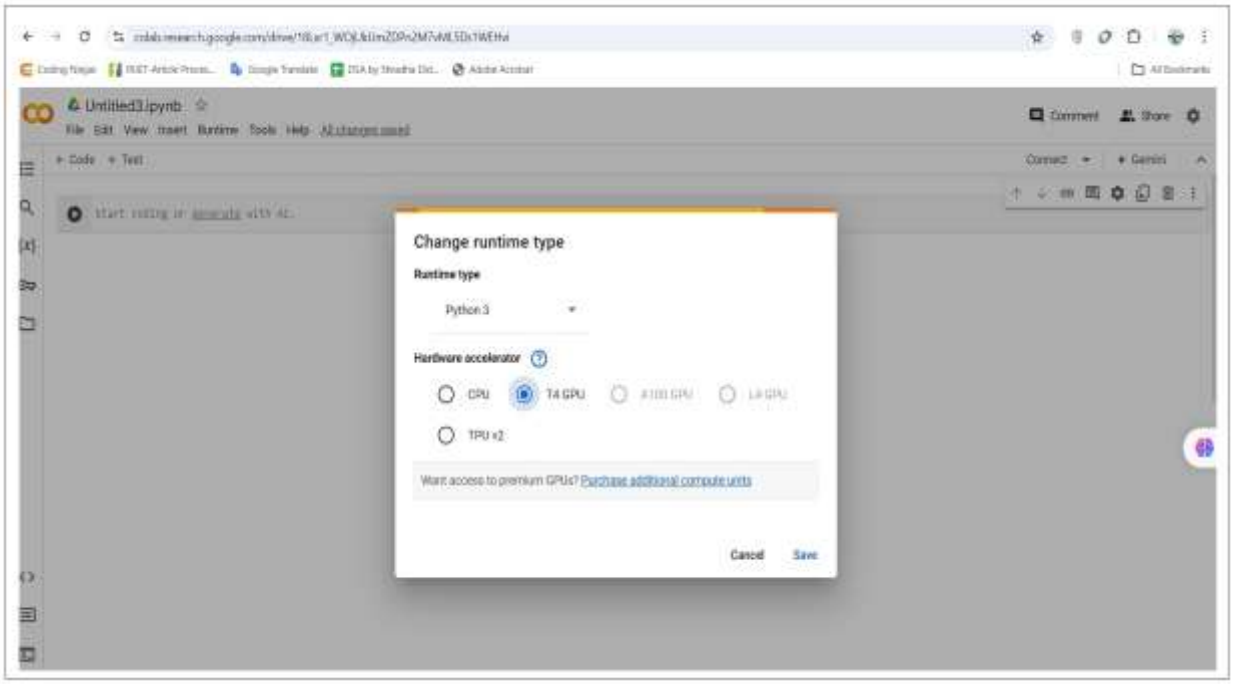


Figure 8. Selection of runtime type

9. Selecting home directory and installing ultralytics in Google colab environment

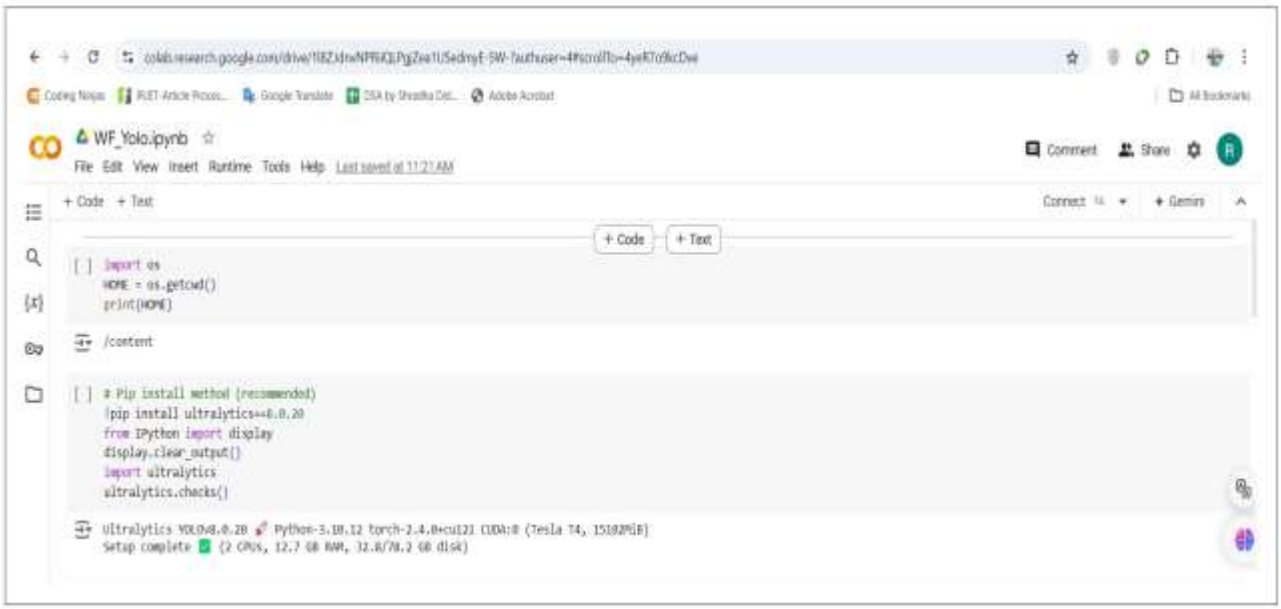
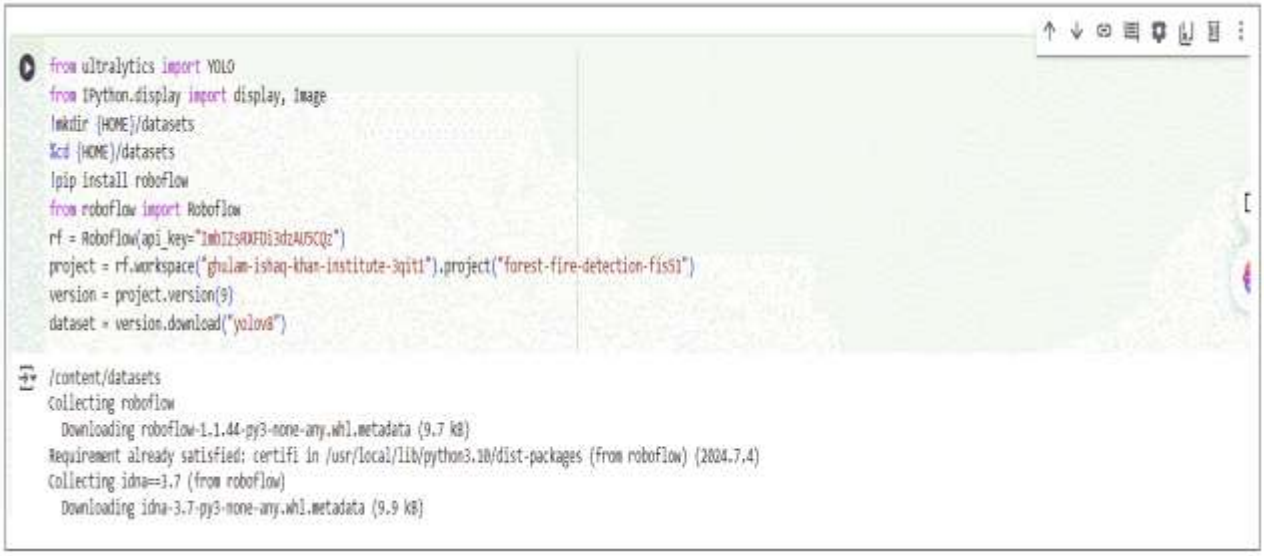


Figure 9. Selecting home directory and installing ultralytics in google colab environment

10. Downloading roboflow dataset



```
from ultralytics import YOLO
from IPython.display import display, Image
!mkdir {HOME}/datasets
%cd {HOME}/datasets
!pip install roboflow
from roboflow import Roboflow
rf = Roboflow(api_key="ImbIIsRXFDi3dzAU5CQz")
project = rf.workspace("ghulam-ishaq-khan-institute-3qit1").project("forest-fire-detection-fiss1")
version = project.version(9)
dataset = version.download("yolov8")
```

```
/content/datasets
Collecting roboflow
  Downloading roboflow-1.1.44-py3-none-any.whl.metadata (9.7 kB)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from roboflow) (2024.7.4)
Collecting idna==3.7 (from roboflow)
  Downloading idna-3.7-py3-none-any.whl.metadata (9.9 kB)
```

Figure 10. Downloading roboflow dataset

11.Downloaded dataset for Wildfire

Figure 11. Downloaded Dataset for wildfire

12. Training epochs of YOLO V8

```
[ ] %cd {HOME}
!yolo task=detect mode=train model=yolov8n.pt data={dataset.location}/data.yaml epochs=200 imgsz=640 plots=True cache=True batch=32

    75/200     12.2G      1.694       1.64      1.541        392        640:  65% 33/51 [00:24<00:13,  1.29it/s]/usr/local/lib/python3.10/dist-packages/ultralytics/yolo/engine/tr
  with torch.cuda.amp.autocast(self.amp):
    75/200     12.2G       1.69      1.639      1.539        280        640:  67% 34/51 [00:25<00:10,  1.55it/s]/usr/local/lib/python3.10/dist-packages/ultralytics/yolo/engine/tr
  with torch.cuda.amp.autocast(self.amp):
    75/200     12.2G       1.69      1.637      1.538        351        640:  69% 35/51 [00:25<00:10,  1.48it/s]/usr/local/lib/python3.10/dist-packages/ultralytics/yolo/engine/tr
  with torch.cuda.amp.autocast(self.amp):
    75/200     12.2G      1.689      1.638      1.539        318        640:  71% 36/51 [00:26<00:09,  1.56it/s]/usr/local/lib/python3.10/dist-packages/ultralytics/yolo/engine/t
  with torch.cuda.amp.autocast(self.amp):
    75/200     12.2G      1.688      1.636      1.538        358        640:  73% 37/51 [00:27<00:08,  1.63it/s]/usr/local/lib/python3.10/dist-packages/ultralytics/yolo/engine/tr
  with torch.cuda.amp.autocast(self.amp):
    75/200     12.2G      1.688      1.636      1.536        290        640:  75% 38/51 [00:27<00:08,  1.58it/s]/usr/local/lib/python3.10/dist-packages/ultralytics/yolo/engine/tr
  with torch.cuda.amp.autocast(self.amp):
    75/200     12.2G      1.688      1.637      1.538        304        640:  76% 39/51 [00:28<00:06,  1.78it/s]/usr/local/lib/python3.10/dist-packages/ultralytics/yolo/engine/tr
  with torch.cuda.amp.autocast(self.amp):
    75/200     12.2G       1.69      1.638       1.54        288        640:  78% 40/51 [00:28<00:06,  1.73it/s]/usr/local/lib/python3.10/dist-packages/ultralytics/yolo/engine/tr
  with torch.cuda.amp.autocast(self.amp):
    75/200     12.2G      1.692      1.639       1.54        294        640:  80% 41/51 [00:29<00:05,  1.92it/s]/usr/local/lib/python3.10/dist-packages/ultralytics/yolo/engine/tr
```

Figure 12. Training epochs of YOLO V8

13. Downloading trained data of YOLO V8

```
import zipfile
import os
 # Set the directory path for the folder you want to download
dir_path = "/content/runs"
 # Set the path for the zip file you want to create
zip_path = "/content/FIRE_RUNS.zip"
 # Create a zipfile object
zipf = zipfile.ZipFile(zip_path, 'w', zipfile.ZIP_DEFLATED)
 # Close the zipfile object

 # Walk through all the directories and subdirectories in the given path
for root, dirs, files in os.walk(dir_path):
  for file in files:
    # Get the absolute path of the file
    file_path = os.path.join(root, file)
 # Add the file to the zipfile object
    zipf.write(file_path)
 # Download the zipfile
zipf.close()
from google.colab import files
files.download(zip_path)
```

Figure 13. Downloading trained data of YOLO V8

14.Confusion Matrix of YOLO V8

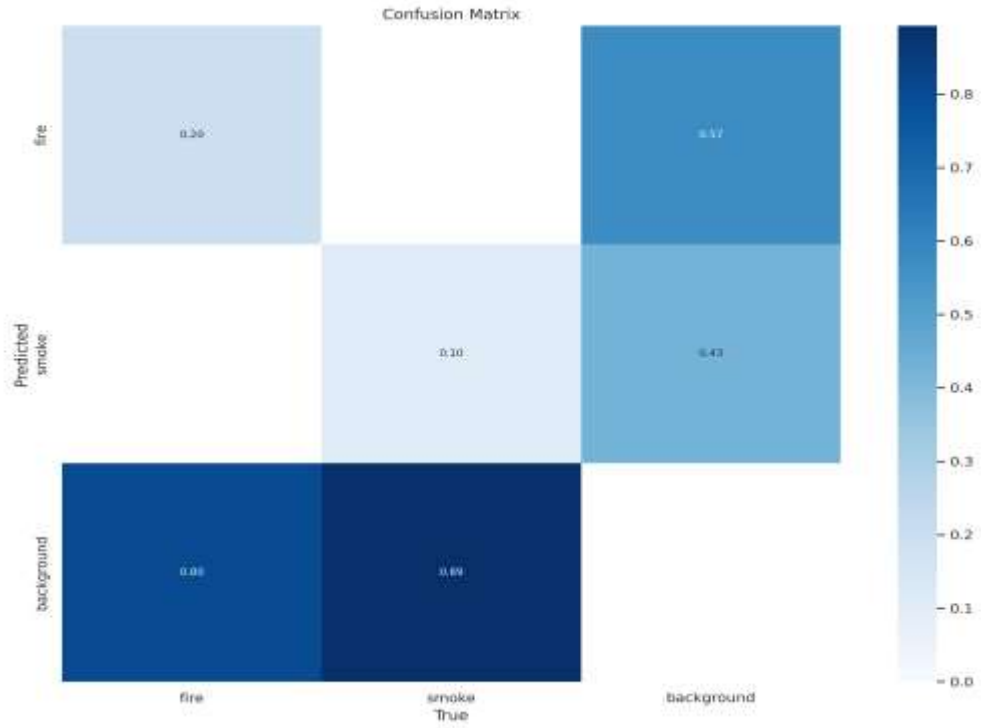


Figure 14. Confusion Matrix of YOLO V8

15. Confidence Curve Of YOLO V8

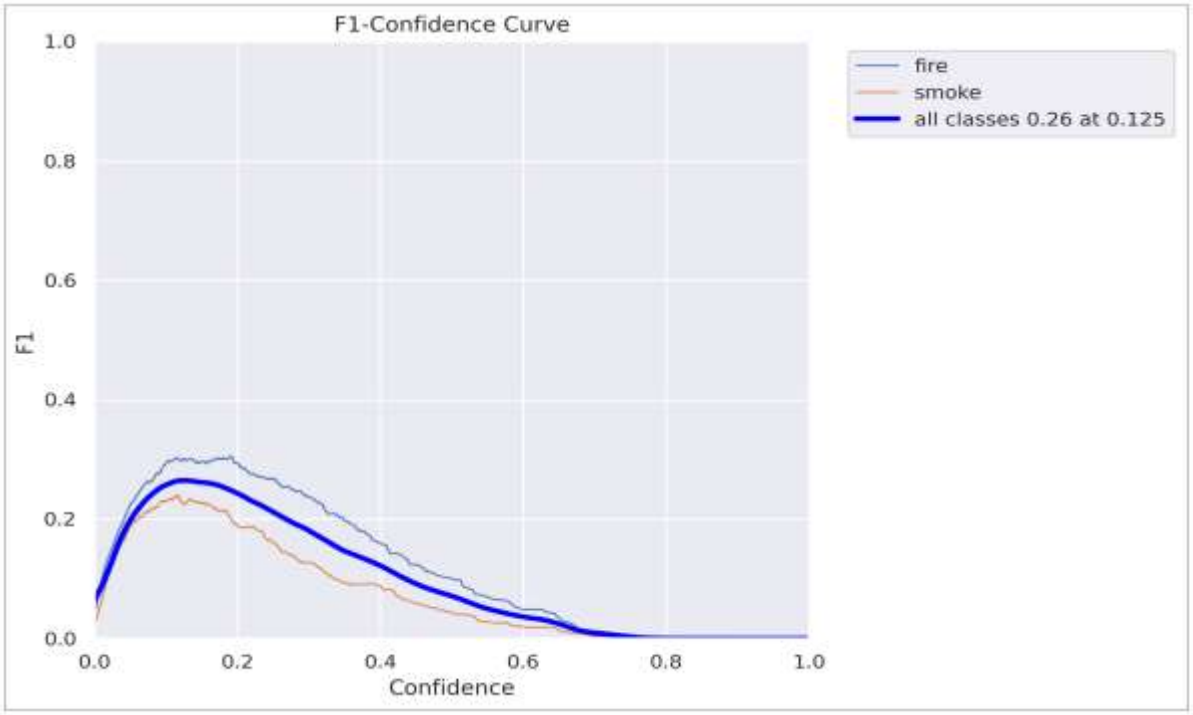


Figure 15. Confidence Curve of YOLO V8

16. Precision Curve of YOLO V8

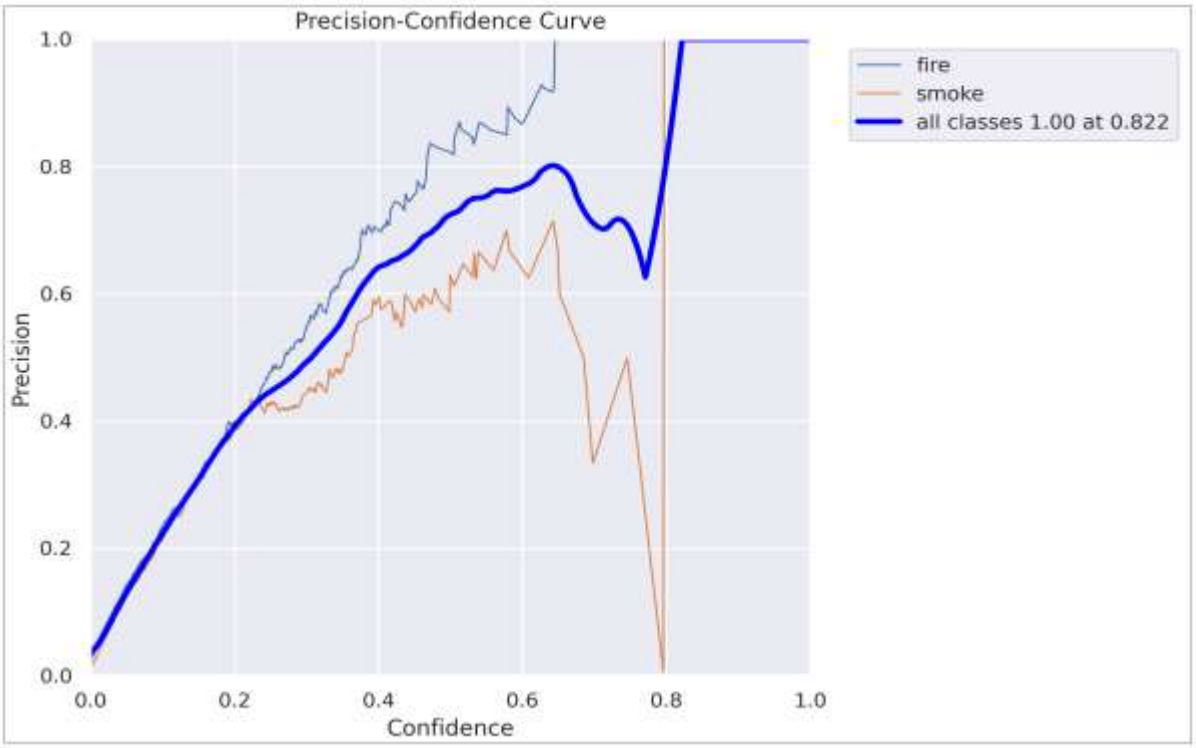


Figure 16. Precision Curve of YOLO V8

17. Precision and recall curve of YOLO V8

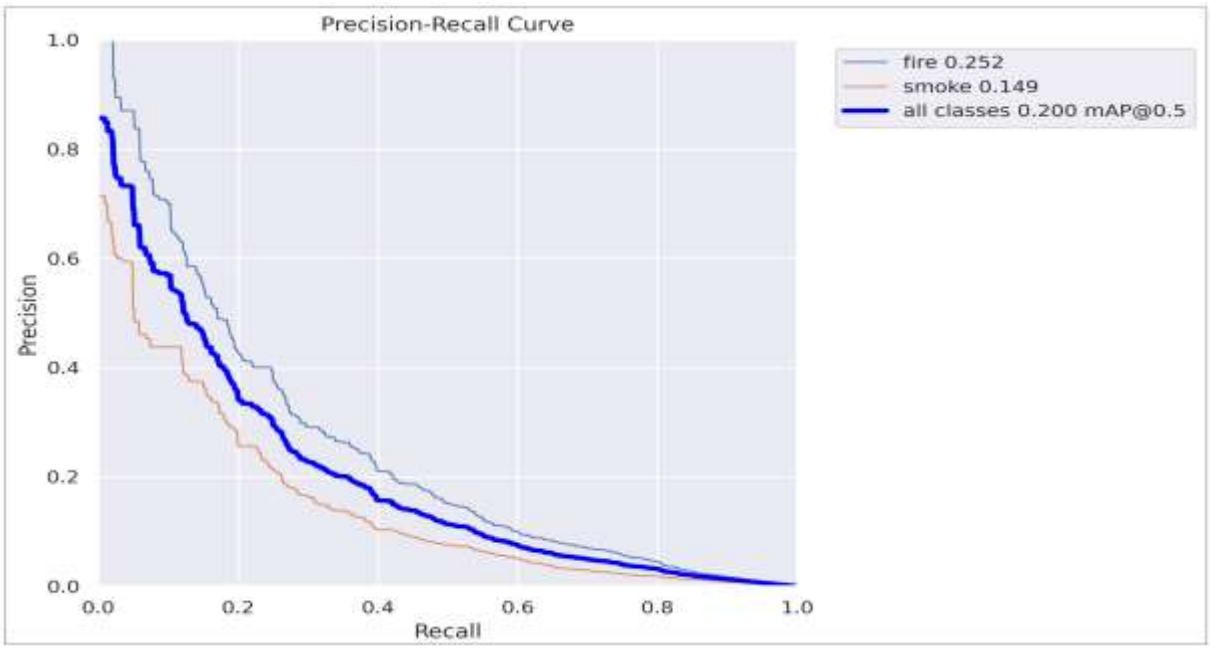


Figure 17. Precision and recall curve of YOLO V8

18. Recall curve of YOLO V8

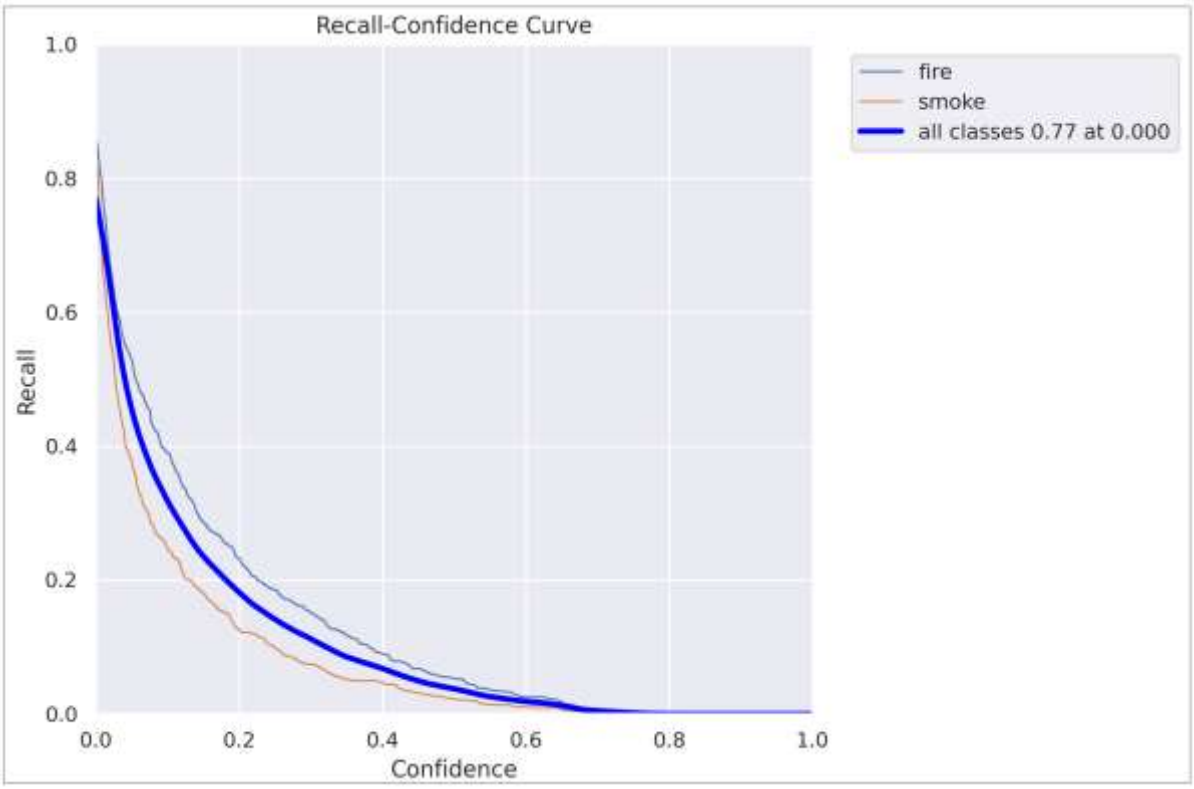


Figure 18. Recall curve of YOLO V8

19. All YOLO V8 Result

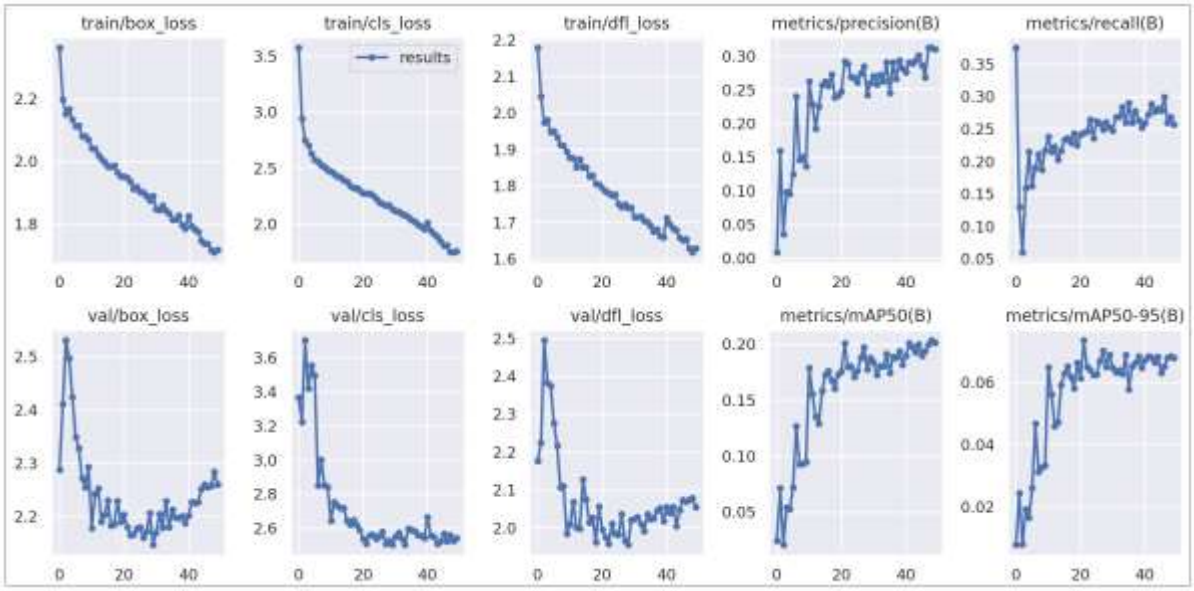
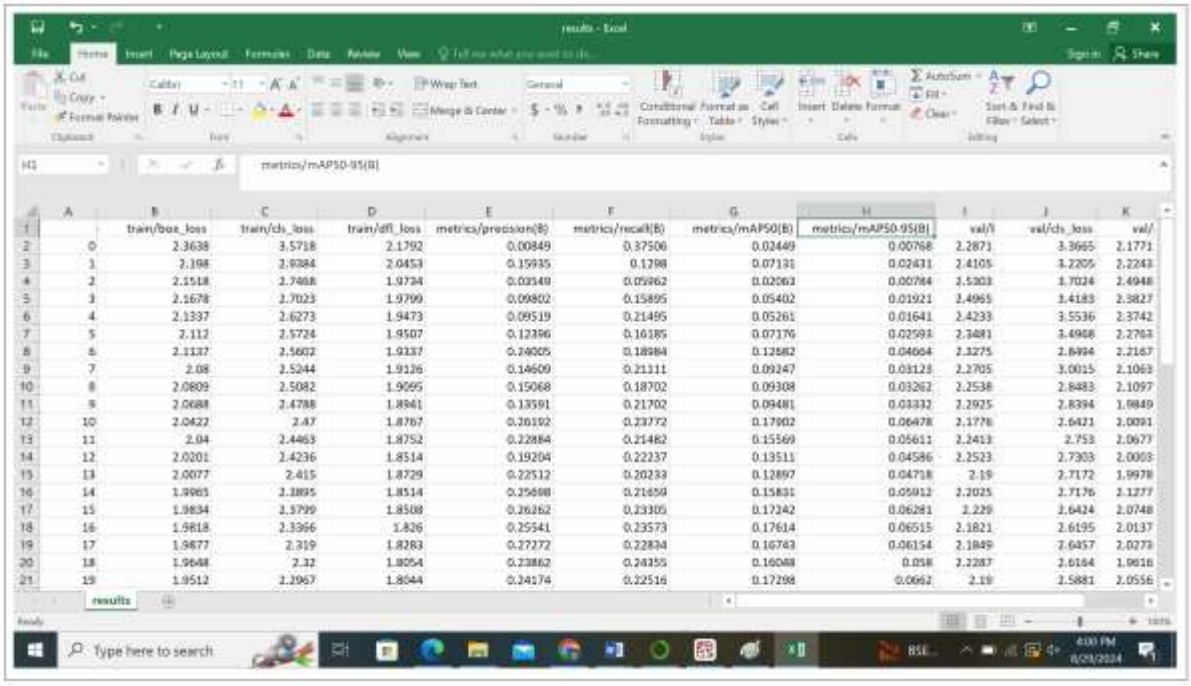


Figure 19. All YOLO V8 Result

20. YOLO V8 Wildfire Detection Results

Figure 20. YOLO V8 Wildfire Detection Results

21. Obtained Results Data for training of YOLO V8



	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision(B)	metrics/recall(B)	metrics/mAP50(B)	metrics/mAP50-95(B)	val/l	val/cls_loss	val/
0	2.3638	3.5718	2.1792	0.00849	0.37506	0.02449	0.00768	2.2871	3.3665	2.1771
1	2.198	2.9384	2.0453	0.15935	0.1298	0.07131	0.02431	2.4105	3.2205	2.2243
2	2.1518	2.7468	1.9734	0.03549	0.05962	0.02063	0.00784	2.5303	3.7024	2.4948
3	2.1678	2.7023	1.9799	0.09802	0.15895	0.05402	0.01921	2.4965	3.4183	2.3827
4	2.1337	2.6273	1.9473	0.09519	0.21495	0.05261	0.01641	2.4233	3.5536	2.3742
5	2.112	2.5724	1.9507	0.12396	0.16185	0.07176	0.02593	2.3481	3.4968	2.2763
6	2.1137	2.5602	1.9337	0.24005	0.18984	0.12682	0.04664	2.3275	2.8494	2.2167
7	2.08	2.5244	1.9126	0.14609	0.21111	0.09247	0.03123	2.2705	3.0015	2.1063
8	2.0809	2.5082	1.9095	0.15068	0.18702	0.09308	0.03262	2.2538	2.8483	2.1097
9	2.0688	2.4788	1.8941	0.13591	0.21702	0.09481	0.03332	2.2925	2.8394	1.9849
10	2.0422	2.47	1.8767	0.26192	0.23772	0.17902	0.06478	2.1778	2.6421	2.0091
11	2.04	2.4463	1.8752	0.22884	0.21482	0.15569	0.05611	2.2413	2.753	2.0677
12	2.0201	2.4236	1.8514	0.19204	0.22237	0.13511	0.04586	2.2523	2.7303	2.0003
13	2.0077	2.415	1.8729	0.22512	0.20233	0.12897	0.04718	2.19	2.7172	1.9978
14	1.9965	2.3895	1.8514	0.25698	0.21659	0.15831	0.05912	2.2025	2.7176	1.9777
15	1.9834	2.3799	1.8508	0.26262	0.23305	0.17242	0.06281	2.229	2.6424	2.0748
16	1.9818	2.3366	1.826	0.25541	0.23573	0.17614	0.06515	2.1821	2.6195	2.0137
17	1.9677	2.319	1.8283	0.27272	0.22834	0.16743	0.06154	2.1849	2.6457	2.0273
18	1.9648	2.32	1.8054	0.23862	0.24355	0.16048	0.058	2.2287	2.6164	1.9816
19	1.9512	2.2967	1.8044	0.24174	0.22516	0.17298	0.0662	2.19	2.5881	2.0556

22. Python Libraries

The project uses following python libraries:

a)

```
# Ultralytics requirements
# Usage: pip install -r requirements.txt

# Base ----------------------------------------
matplotlib>=3.2.2
numpy>=1.18.5
opencv-python>=4.6.0
Pillow>=7.1.2
PyYAML>=5.3.1
requests>=2.23.0
scipy>=1.4.1
torch>=1.7.0
torchvision>=0.8.1
tqdm>=4.64.0

# Logging -------------------------------------
tensorboard>=2.4.1
# clearml
# comet

# Plotting ------------------------------------
pandas>=1.1.4
seaborn>=0.11.0

# Export --------------------------------------
# coremltools>=6.0  # CoreML export
onnx>=1.12.0  # ONNX export
onnxsim>=0.4.1  # ONNX simplifier
# nvidia-pyindex  # TensorRT export
# nvidia-tensorrt  # TensorRT export
# scikit-learn==0.19.2  # CoreML quantization
# tensorflow>=2.4.1  # TF exports (-cpu, -aarch64, -macos)
# tflite-support
# tensorflowjs>=3.9.0  # TF.js export
# openvino-dev>=2022.3  # OpenVINO export

# Extras --------------------------------------
psutil  # system utilization
thop>=0.1.1  # FLOPs computation
wheel>=0.38.0  # Snyk vulnerability fix
# ipython  # interactive notebook
# albumentations>=1.0.3
# pycocotools>=2.0.6  # COCO mAP
roboflow
```

Figure 22. Libraries in requirements

b) Pillow
c) SVM
d) Tensorflow
e) Keras
f) Matplotlib
g) Opencv-python
h) Ultralytics
i) Scikit-learn
j) Plot_keras_history
k) Openpyxl

23. Collection of weather dataset

Field Name	Description	Units	Data Type
DailyAverageDewPointTemperature	Average daily dew point temperature	Fahrenheit (° F)	Int
DailyAverageDryBulbTemperature	The average daily dry bulb temperature recorded	Fahrenheit (° F)	Int
DailyAverageRelativeHumidity	Average daily Humidity	Percentage(%)	Int
DailyAverageSeaLevelPressure	Average daily sea level pressure	Inch of mercury (Hg)	Int
DailyAverageStationPressure	Daily average station pressure	Inch of mercury (Hg)	Int
DailyAverageWetBulbTemperature	Average daily wet bulb temperature	Fahrenheit (° F)	Int
DailyAverageWindSpeed	Average wind speed in mph	Miles Per Hour (mph)	Int
DailyAverageDewPointTemperature	Average daily dew point temperature	Fahrenheit (° F)	Int
DailyAverageDryBulbTemperature	The average daily dry bulb temperature recorded	Fahrenheit (° F)	Int
DailyPeakWindDirection	Daily peak wind direction	Compass Degrees	Int
DailyDryBulbTemperature	Daily dry bulb temperature	Fahrenheit (° F)	Int
DailyPrecipitation	Daily precipitation	Inch(in)	Int
PeakWindDirection	Maximum wind direction	Compass Degrees	Int
DryBulbTemperature	Dry bulb temperature	Fahrenheit (° F)	Int
Date	recorded	MM DD YYYY	Date

Figure 23. Collection of weather dataset

24. Historical wildfire Dataset

Field Name	Description	Data Type
X	Latitude	Decimal
Y	Longitude	Decimal
Discover_Year	Year of the fire	Number
Fire_Number	Alternative ID for each fire that occurred	Number
Total_acres_burnt	Number of acres burnt by fire	Decimal
County	Region of the fire incident	Text
Slope	A Integer value indicating slope of the region	Integer
Elevation	A Integer value indicating elevation of the region	Integer
Fire_Name	Name of the fire	Text
Ignition	Start date of the fire	DateTime
Fire_Out	End date of the fire	DateTime
Location	Area of the fire in the county	Text
Fuel_model	Presence of fuel levels in the area	Text
Cover_Class	Type of land cover in the area	Text
Fire_Intensity	Length of the flame of a fire	Text

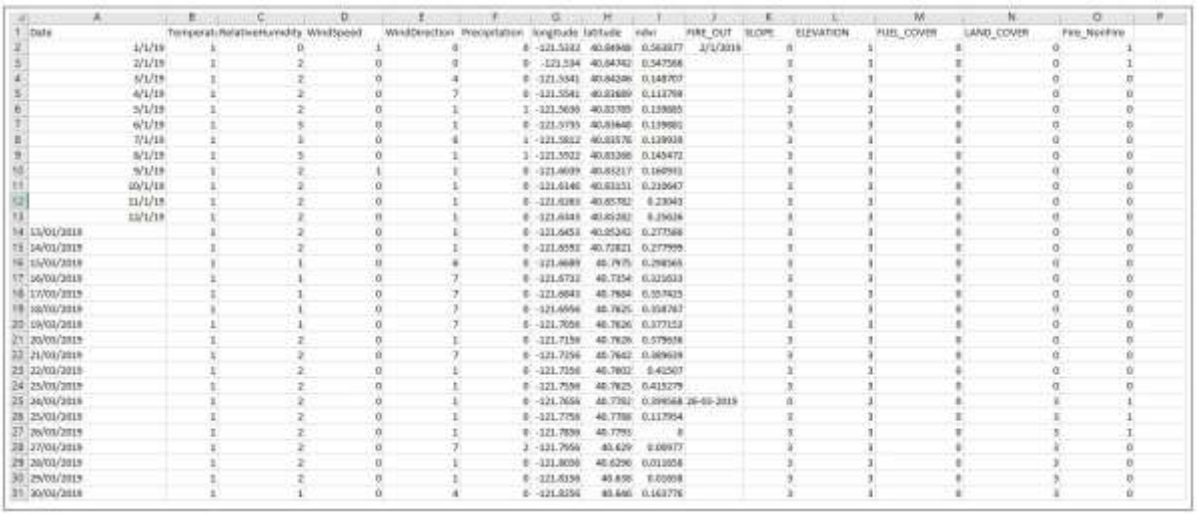
Figure 24. Historical Wildfire Dataset

25. Vegetation Dataset

Index	Description	Formula
NDVI	Quantifies the greenness and is useful to understand density and health of the plants	(NIR - R) / (NIR + R)
NDMI	Used to detect the water content in the vegetation	(NIR - SWIR) / (NIR + SWIR)
EVI	Quantifies the greenness and corrects the atmospheric conditions	G * ((NIR - R) / (NIR + C1 * R – C2 * B + L))
NBR	Used to identify the area burnt after the fire	(NIR - SWIR) / (NIR + SWIR)
NBR 2	It modifies the NBR to highlight the water content in the vegetation after the fire	(SWIR1 – SWIR2) / (SWIR1 + SWIR2)

Figure 25. Vegetation Dataset

26. Input dataset for LSTM



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	Date	Temperature	RelativeHumidity	WindSpeed	WindDirection	Precipitation	longitude	latitude	ndvi	FIRE_OUT	SLOPE	ELEVATION	FUEL_COVER	LAND_COVER	Fire_NonFire	
2	1/1/19	1	0	1	0	0	-121.5102	40.84948	0.568877	2/1/2019	0	1	0	0	1	
3	2/1/19	1	2	0	0	0	-121.534	40.84742	0.547568		1	1	0	0	1	
4	3/1/19	1	2	0	4	0	-121.5541	40.84586	0.148707		1	1	0	0	0	
5	4/1/19	1	2	0	7	0	-121.5541	40.83689	0.111759		1	1	0	0	0	
6	5/1/19	1	2	0	1	1	-121.5639	40.83705	0.139885		1	1	0	0	0	
7	6/1/19	1	5	0	1	0	-121.5735	40.83648	0.139891		1	1	0	0	0	
8	7/1/19	1	1	0	4	1	-121.5812	40.83576	0.139918		1	1	0	0	0	
9	8/1/19	1	5	0	1	1	-121.5922	40.83268	0.145472		1	1	0	0	0	
10	9/1/19	1	2	1	1	0	-121.6039	40.83217	0.166943		1	1	0	0	0	
11	10/1/19	1	2	0	1	0	-121.6140	40.83151	0.210647		1	1	0	0	0	
12	11/1/19	1	2	0	1	0	-121.6283	40.83782	0.23043		1	1	0	0	0	
13	12/1/19	1	2	0	1	0	-121.6343	40.83282	0.19634		1	1	0	0	0	
14	13/01/2019	1	2	0	1	0	-121.6453	40.85242	0.277568		1	1	0	0	0	
15	14/01/2019	1	2	0	1	0	-121.6553	40.72821	0.277999		1	1	0	0	0	
16	15/01/2019	1	1	0	4	0	-121.6689	40.7975	0.298365		1	1	0	0	0	
17	16/01/2019	1	1	0	7	0	-121.6732	40.7254	0.321633		1	1	0	0	0	
18	17/01/2019	1	1	0	7	0	-121.6843	40.7984	0.357425		1	1	0	0	0	
19	18/01/2019	1	1	0	7	0	-121.6956	40.7625	0.358767		1	1	0	0	0	
20	19/01/2019	1	1	0	7	0	-121.7056	40.7626	0.377153		1	1	0	0	0	
21	20/01/2019	1	2	0	1	0	-121.7159	40.7626	0.379038		1	1	0	0	0	
22	21/01/2019	1	2	0	7	0	-121.7256	40.7642	0.389619		1	1	0	0	0	
23	22/01/2019	1	2	0	1	0	-121.7356	40.7802	0.42507		1	1	0	0	0	
24	23/01/2019	1	2	0	1	0	-121.7539	40.7825	0.415279		1	1	0	0	0	
25	24/01/2019	1	2	0	1	0	-121.7656	40.7781	0.399568	26-01-2019	0	1	0	1	1	
26	25/01/2019	1	2	0	1	0	-121.7756	40.7788	0.117954		1	1	0	1	1	
27	26/01/2019	1	2	0	1	0	-121.7839	40.7795	0		1	1	0	1	1	
28	27/01/2019	1	2	0	7	2	-121.7956	40.629	0.08977		1	1	0	1	0	
29	28/01/2019	1	2	0	1	0	-121.8036	40.6296	0.011858		1	1	0	1	0	
30	29/01/2019	1	2	0	1	0	-121.8159	40.638	0.03008		1	1	0	1	0	
31	30/01/2019	1	1	0	4	0	-121.8256	40.646	0.163776		1	1	0	1	0	

Figure 26. Input Dataset for LSTM

27. Pre-processing Of LSTM

```
class Preprocessing:
    def __init__(self,dataframe):
        self.df=dataframe

    def dataPreprocessing(self):
        print("\n")
        print('Shape of the file')#Number of Rows and Columns
        print('*************************************************************************')
        print(self.df.shape)
        print("\n")
        print('Number of Missing Values per Column')
        print('*************************************************************************')
        print("\n")
        print(self.df.isnull().sum())
        print("\n")
        print('Variable types')
        print('*************************************************************************')
        print(self.df.dtypes)
        limitPer = len(self.df) * .94
        self.df = self.df.dropna(thresh=limitPer, axis=1)
        print("\n")
        print('Number of Missing Values per Column After Data Cleaning')
        print('*************************************************************************')
        print("\n")
        print(self.df.isnull().sum())

        # Converting date format
        self.df['Date'] = pd.to_datetime(self.df["Date"]).dt.strftime('%d%m%Y').astype(int)

        print("\n")
        print('Find Correlation Between Different Attributes')
        print('*************************************************************************')
        print("\n")

        corrmatrix=self.df.corr(method="pearson")
        print("The correlation matrix: ")
        print("\n")
        print(corrmatrix)
        plt.subplots()
        sns.set(rc={'figure.figsize':(15,11)})
        sns.heatmap(corrmatrix, vmax=0.9, square=True)
        plt.show()
```

Figure 27. Pre-processing of LSTM

28. Data Segmentation of LSTM

```
    def initLSTM(self):

        # Code to split dataset into testing and training
        X_train, X_test, Y_train, Y_test= train_test_split(self.X, self.Y, test_size=0.25,random_state=52,shuffle=True)

        print(X_train.shape)
        print(Y_train.shape)
        print(X_test.shape)
        print(Y_test.shape)



        # Adefine min max scaler and transfor data
        scaler = MinMaxScaler()
        train_X=scaler.fit_transform(X_train)
        test_X=scaler.transform(X_test)
        train_Y=scaler.fit_transform(Y_train)
        test_Y=scaler.transform(Y_test)


        # reshape input to be 3D for LSTM [samples, timesteps, features]
        train_X1 = train_X.reshape((train_X.shape[0],1,train_X.shape[1]))
        train_Y1 = train_Y.reshape((train_Y.shape[0],1,train_Y.shape[1]))
        test_X1= test_X.reshape((test_X.shape[0],1,test_X.shape[1]))
        test_Y1= test_Y.reshape((test_Y.shape[0],1,test_Y.shape[1]))



        print(train_X1.shape, train_Y1.shape, test_X1.shape)
```

Figure 28. Data Segmentation Of LSTM

29. LSTM Model

```
56        # Create LSTM network
57        model = Sequential()
58        model.add(LSTM(units=20,return_sequences=True,input_shape=(train_X1.shape[1], train_X1.shape[2])))
59        model.add(Dropout(0.2))
60        model.add(LSTM(units=40,return_sequences=True))
61        model.add(Dropout(0.2))
62        model.add(LSTM(units=80,return_sequences=True))
63        model.add(Dropout(0.2))
64        model.add(LSTM(units=80))
65        model.add(Dropout(0.2))
66        model.add(Dense(units=40,activation='tanh'))
67        model.add(Dense(units=1, activation='tanh'))
68        model.compile( optimizer='adam',loss='mean_squared_error', metrics=['accuracy'])
69        history=model.fit(train_X1, train_Y1, epochs=self.number_of_interations, batch_size=10,validation_data=(test_X1, test_Y1), shuffle=False)
70        model.save("LSTMModelData.h5")
71        show_history(history)
72        plot_history(history, path="Loss_Accuracy.png")
73        plt.close()
```

Figure 29. LSTM Model

30. LSTM Accuracy Graph

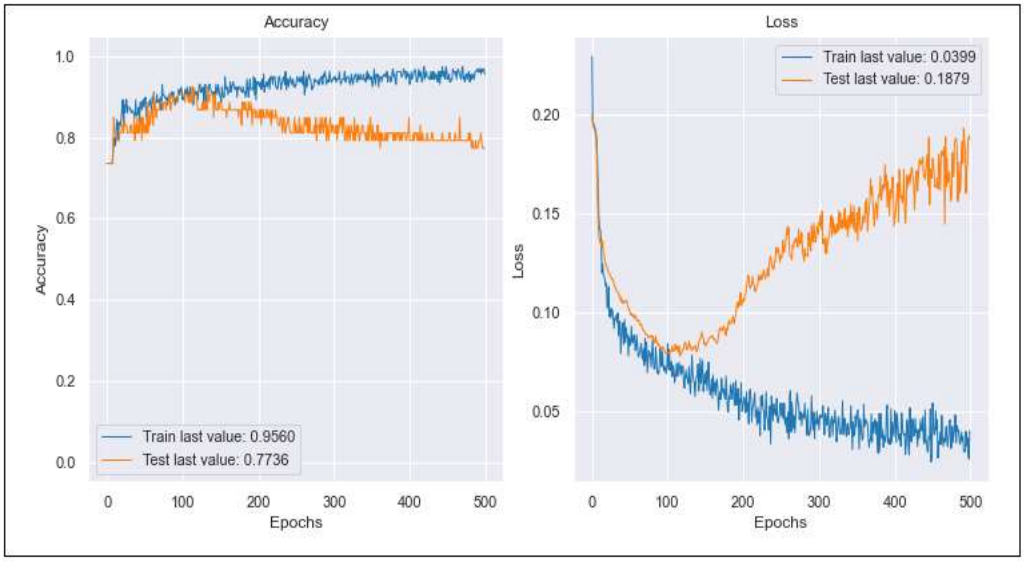


Figure 30. LSTM Accuracy Graph

31. Main GUI

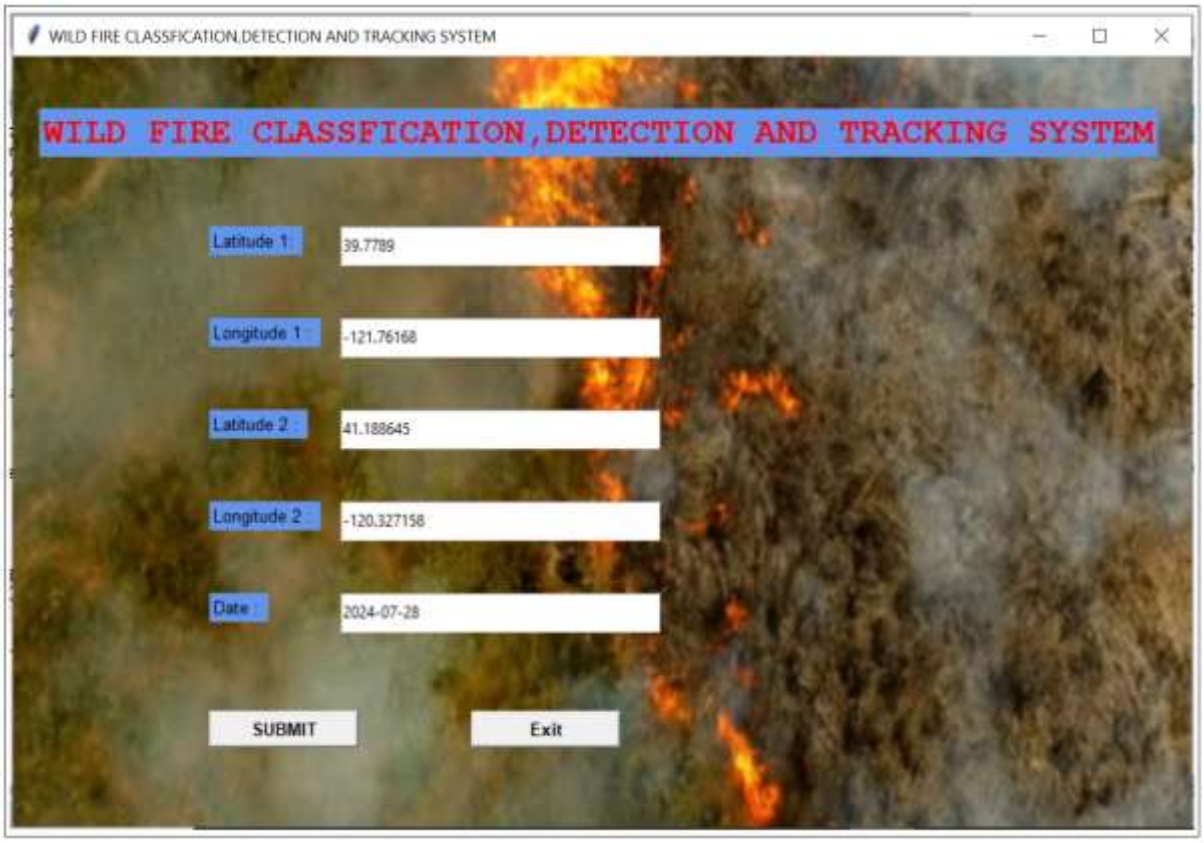


Figure 31. Main GUI

32. Image Pulling

```
import os
import requests

def getImage(date, cord):
    url = 'https://wvs.earthdata.nasa.gov/api/v1/snapshot?REQUEST=GetSnapshot&CRS=EPSG:4326&WRAP=DAY&LAYERS=MODIS_Terra_CorrectedReflectanc
    page = requests.get(url)
    f_ext = 'Input_map.jpg'
    with open(f_ext, 'wb') as f:
        f.write(page.content)

    return 1
```

Figure 32. Image Pulling

33. Pulled Image for given co-ordinates

Figure 33. Pulled image for given co-ordinates

34. SVM Test

```
10  def isFireDetected(img):
11    print("inside SVM")
12    retsultstr="Fire"
13    test_image = load_img(img, target_size=(224,224))
14   # plt.imshow(test_image)
15    test_image = img_to_array(test_image)
16    test_image=test_image/255
17    test_image = np.expand_dims(test_image, axis = 0)
18    result = model.predict(test_image)
19    #print(result)
20    if result[0]>0.3 and result[0]<1:
21        retsultstr="No Fire Detected"
22    else:
23        retsultstr="Fire Detected"
24
25    return retsultstr
```

Figure 34. SVM Test

35. Fire Detection

Figure 35. Fire Detection

36. YOLO V8 Detection

```
35  def startDetection(input_imagepath):
36      model = YOLO("model/best.pt","v8")
37      frame=cv2.imread(input_imagepath)
38      detect_params=model.predict(frame, conf=0.4, save=False)
39      print(detect_params)
40      for box in detect_params[0].boxes:
41          clsID = box.cls.numpy()[0]
42          conf = box.conf.numpy()[0]
43          bb = box.xyxy.numpy()[0]
44          x1 = int(bb[0])
45          x2 = int(bb[2])
46          y1 = int(bb[1])
47          y2 = int(bb[3])
48          value=round(conf, 3)
49        #  text=class_list[int(clsID)]
50          print("Reached 1")
51
52          text = " Pothole" + str(round(conf, 3)) + "%"
53          cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 255), 3)
54          font = cv2.FONT_HERSHEY_COMPLEX
55          cv2.putText(frame, text, (x1, y1), font, 1, (0, 0, 255), 2)
```

Figure 36. YOLO V8 Detection

37. YOLO V8 result display

```
53                  from PIL import Image
54                  img = Image.open('Yolo_Detected.jpg')
55                  img.show()
```

Figure 37. YOLO V8 result display Code

38. YOLO Detected Image for Wildfire

Figure 38. YOLO Detected Image for Wildfire

39. LSTM Testing Pre-processing

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns



class Preprocessing:
    def __init__(self,dataframe):
        self.df=dataframe

    def dataPreprocessing(self):

        limitPer = len(self.df) * .94

        self.df = self.df.dropna(thresh=limitPer, axis=1)


        self.df['Date'] = pd.to_datetime(self.df["Date"]).dt.strftime('%d%m%Y').astype(int)


        corrmatrix=self.df.corr(method='pearson')


        return self.df

```

Figure 39. LSTM Testing Pre-processing

40. LSTM Parameterized Constructor

```
class LSTMNeuralNetwork:

    def __init__(self, X,Y,name,no_of_days,date,co_ordinates,dataframe,pulled_image_path):
        self.X=X
        self.Y=Y
        self.name=name

        self.no_of_days=no_of_days
        self.date=date
        self.co_ordinates=co_ordinates
        self.df=dataframe
        self.pulled_image_path=pulled_image_path

```

Figure 40. LSTM Parameterized Constructor

41. LSTM Test Model

```
    def initLSTM(self):
        # Code to split dataset into testing and training
        X_train, X_test, Y_train, Y_test= train_test_split(self.X, self.Y, test_size=0.25,random_state=52,shuffle=Tru
        print(X_train.shape)
        # #define min max scaler and transfor data
        scaler = MinMaxScaler()
        train_X=scaler.fit_transform(X_train)
        test_X=scaler.transform(X_test)
        train_Y=scaler.fit_transform(Y_train)
        test_Y=scaler.transform(Y_test)
        # reshape input to be 3D for LSTM [samples, timesteps, features]
        train_X1 = train_X.reshape((train_X.shape[0],1,train_X.shape[1]))
        train_Y1 = train_Y.reshape((train_Y.shape[0],1,train_Y.shape[1]))
        test_X1= test_X.reshape((test_X.shape[0],1,test_X.shape[1]))
        test_Y1= test_Y.reshape((test_Y.shape[0],1,test_Y.shape[1]))
        print(train_X1.shape, train_Y1.shape, test_X1.shape)
        model = Sequential()
        model.add(LSTM(units=20,return_sequences=True,input_shape=(train_X1.shape[1], train_X1.shape[2])))
        model.add(Dropout(0.2))
        model.add(LSTM(units=40,return_sequences=True))
        model.add(Dropout(0.2))
        model.add(LSTM(units=80,return_sequences=True))
        model.add(Dropout(0.2))
        model.add(LSTM(units=80))
        model.add(Dropout(0.2))
        model.add(Dense(units=40,activation='tanh'))
        model.add(Dense(units=1, activation='tanh'))
        model.load_weights('model/LSTMModelData.h5')
        predicted_fire_status = model.predict(test_X1)
        df= pd.DataFrame(X_test['Date'])
        df= pd.to_datetime(df['Date'], format='%d%m%Y').dt.date
        #print(df.head())
        inputdate=pd.Series(df.unique()).tolist()
```

Figure 41. LSTM Test Model

42. LSTM Test Root Mean Square Error

```
        testresult=[]
        datalist=self.df.values.tolist()
        cordinates=[]
        for val in datalist:
            drow=[]
            drow.append(val[7])
            drow.append(val[6])
            cordinates.append(drow)
        with open('predicted_Cord.csv', 'w') as f:
            write = csv.writer(f)
            write.writerows(cordinates)
        i=0
        for row in inputdate:
            value=predicted_fire_status[i]
            value1=str(row)
            #print(value1)

            if value<=0.04:
                value=0
            else:
                value=1
            testresult.append(value)
            i+=1
        #calculate root mean squared error
        MapInit.drawFirePattern(self.no_of_days,self.date,self.co_ordinates,self.pulled_image_path,testresult)
        print("\n")
        print("*****************Root Mean Square Error Using LSTM*********************")
        print("\n")
        lstmtestScore = math.sqrt(mean_squared_error(test_Y, predicted_fire_status))
        print('Root Mean Square Error for LSTM', lstmtestScore)
        frame_folder="Results"
        frames = [Image.open(image) for image in glob.glob(f"{frame_folder}/*.JPG")]
        frame_one = frames[0]
        frame_one.save('LSTM_output.gif', format="GIF", append_images=frames,save_all=True, duration=300, loop=0)
```

Figure 42. LSTM Test Root Mean Square Error

43. LSTM Flow

```
import pandas as pd
import preprocessing
import Test_Processing



def startLSTM(co_ordinates,date,pulled_image_path):

    datasetpath='LSTM Train/Final_combined_datasets1.xlsx'
    no_of_days=5

    #*******************Code to Preprocess Dataset**************************************
    df=pd.read_excel(datasetpath)
    print(df.head())
    pre=preprocessing.Preprocessing(df)
    df=pre.dataPreprocessing()

    #*******************Code for LSTM**************************************

    X=df[['Date','Temperature' ,'RelativeHumidity', 'WindSpeed','WindDirection','Precipitation',
          'longitude','latitude','ndvi' ,'SLOPE', 'ELEVATION','FUEL_COVER','LAND_COVER']]


    Y=df[['Fire_NonFire']]

    l=Test_Processing.LSTMNeuralNetwork(X, Y,"LSTM",no_of_days,date,co_ordinates,df,pulled_image_path)
    l.initLSTM()
```

Figure 43. LSTM Flow

44.GPS Mapper

```
    def plot_map(self, output='save', save_as='Fire_Pattern_Predicted.png'):
        self.get_ticks()
        fig, axis1 = plt.subplots(figsize=(10, 10))
        axis1.imshow(self.result_image)
        axis1.set_xlabel('Longitude')
        axis1.set_ylabel('Latitude')
        axis1.set_xticklabels(self.x_ticks)
        axis1.set_yticklabels(self.y_ticks)
        axis1.grid()
        if output == 'save':
            plt.savefig(save_as)
        img_points = []
        if os.path.isfile('Fire_Pattern_Predicted.png'):
            imageob = Image.open('Fire_Pattern_Predicted.png').convert('RGB')
            width, height = imageob.size
            pix=imageob.load()
            for i in range(width):
                for j in range(height):
                    col=pix[i,j]
                    R=col[0]
                    G=col[1]
                    B=col[2]
                    if(R>=200 and R<=240):
                        if(G>=140 and G<=165):
                            if(B>=120 and B<=165):
                                img_points.append((i, j))
        return   img_points
```

Figure 44. GPS Mapper

45. Grid Marker

```python
import matplotlib.pyplot as plt
import matplotlib.ticker as plticker

def getGriDImage():

    try:
        from PIL import Image
    except ImportError:
        import Image

    image = Image.open('Fire_Pattern_Predicted.png')
    gridLineWidth=5
    fig=plt.figure(figsize=(float(image.size[0])/gridLineWidth, float(image.size[1])/gridLineWidth), dpi=gridLineWidth)
    axes=fig.add_subplot(111)
    gridInterval=127.
    location = plticker.MultipleLocator(base=gridInterval)
    axes.xaxis.set_major_locator(location)
    axes.yaxis.set_major_locator(location)
    axes.grid(which='major', axis='both', linestyle='-', linewidth=10,color='#000000')
    nx=abs(int(float(axes.get_xlim()[1]-axes.get_xlim()[0])/float(gridInterval)))
    ny=abs(int(float(axes.get_ylim()[1]-axes.get_ylim()[0])/float(gridInterval)))
    for j in range(ny):
        y=gridInterval/2+j*gridInterval
        for i in range(nx):
            x=gridInterval/2.+float(i)*gridInterval
            axes.text(x,y,'{:d}'.format(i+j*nx), color='#000000', fontsize='25',  ha='center', va='center')
```

Figure Grid Marker

46.Grid Marked Image

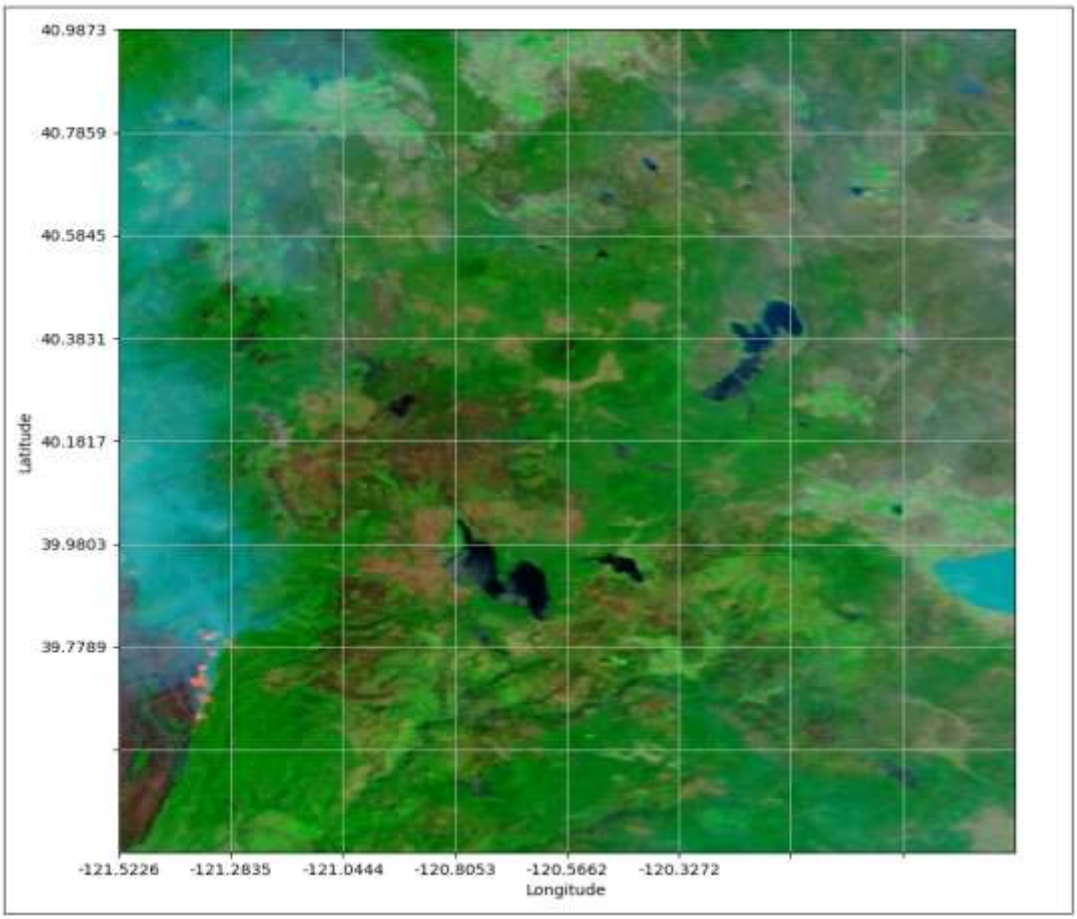


Figure 46. Grid Marked Image

47. Unique Co-ordinate Estimation

```
52          for points in img_points:
53              x=points[0]
54              y=points[1]
55              if((x>p1 and x<p3) and (y>p2 and y<p4)):
56                  newrow=[]
57                  newrow.append(x)
58                  newrow.append(y)
59                  newpoints.append(newrow)
60              else:
61                  if(x<p1):
62                      x=p1+abs(p1-x)
63                  if(y<p2):
64                      y=p2+abs(p2-y)
65                  if(x>p3):
66                      x=x-abs(p3-x)
67                  if(y>p4):
68                      y=y-abs(p4-y)
69
70                  newrow=[]
71                  newrow.append(x)
72                  newrow.append(y)
73                  newpoints.append(newrow)
74          uniquepoints=[]
75          uniquepoints=remove(newpoints)
```

Figure 47. Unique Co-ordinates Estimation

48. Nearest Co-ordinate Estimator

```
106         for row in sumlist:
107             if(k<no_of_days):
108                 testimgob = Image.open(finpath).convert('RGB')
109                 x1=row[0]
110                 y1=row[1]
111                 k=k+1
112                 if(k==1):
113                     mx=x1+pval
114                     my=y1+pval
115                     point1 = np.array((mx, my))
116                     point2 = np.array((x1, y1))
117                     dist = np.linalg.norm(point1 - point2)
118                     temprow=[]
119                     temprow.append(x1)
120                     temprow.append(y1)
121                     temprow.append(dist)
122                     finlist.append(temprow)
123                 else:
124                     val1=random.randint(0,100)
125                     val2=random.randint(0,100)
126                     x1=x1+val1
127                     y1=y1+val2
128                     point1 = np.array((mx, my))
129                     point2 = np.array((x1, y1))
130                     dist = np.linalg.norm(point1 - point2)
131                     temprow=[]
132                     temprow.append(x1)
133                     temprow.append(y1)
134                     temprow.append(dist)
135                     finlist.append(temprow)
```

Figure 48. Nearest Co-ordinate Estimator

49. GIF File Opener

```
class ImageLabel(tk.Label):
    """a label that displays images, and plays them if they are gifs"""
    def load(self, im):
        if isinstance(im, str):
            im = Image.open(im)
        self.loc = 0
        self.frames = []
        try:
            for i in count(1):
                self.frames.append(ImageTk.PhotoImage(im.copy()))
                im.seek(i)
        except EOFError:
            pass

        try:
            self.delay = im.info['duration']
        except:
            self.delay = 100

        if len(self.frames) == 1:
            self.config(image=self.frames[0])
        else:
            self.next_frame()

    def unload(self):
        self.config(image="")
        self.frames = None

    def next_frame(self):
        self.update()
        if self.frames:
            self.loc += 1
            self.loc %= len(self.frames)
            self.config(image=self.frames[self.loc])
            self.after(self.delay, self.next_frame)
```

Figure 49. GIF File Opener

50. LSTM Root Call Flow

```
from PIL import Image
img = Image.open('Yolo_Detected.jpg')
img.show()

# cv2.imshow('Fire Detected Result', image)
# cv2.waitKey(0)
# cv2.destroyAllWindows()
co_ordinates=lat1+","+long1+","+lat2+","+long2
print("calling LSTM test")
import LSTMTEST_Init
LSTMTEST_Init.startLSTM(co_ordinates, date, image_path)
import GIFFileOpener
from threading import Thread
guithread1 = Thread(target = GIFFileOpener.OpenGIF, args = ())

guithread1.start()
```

Figure 50. LSTM Root Call Flow

51. Tracking path for 5 days

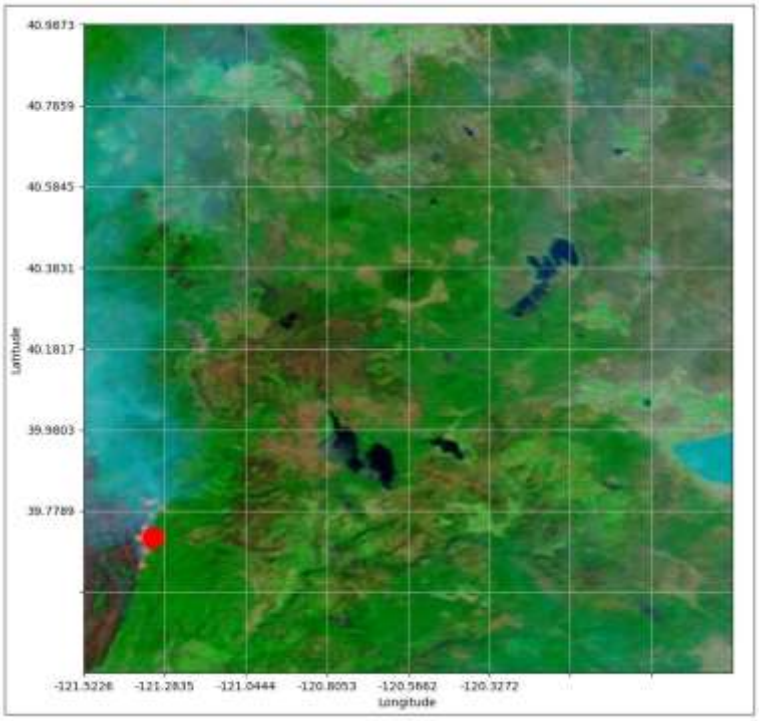


Figure 51. Tracking path 1

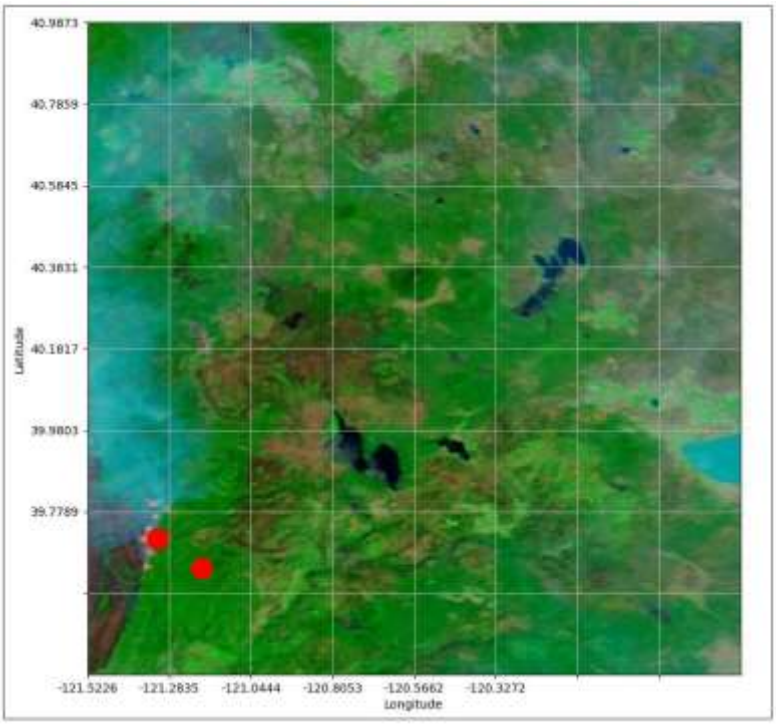


Figure 52. Tracking path 2

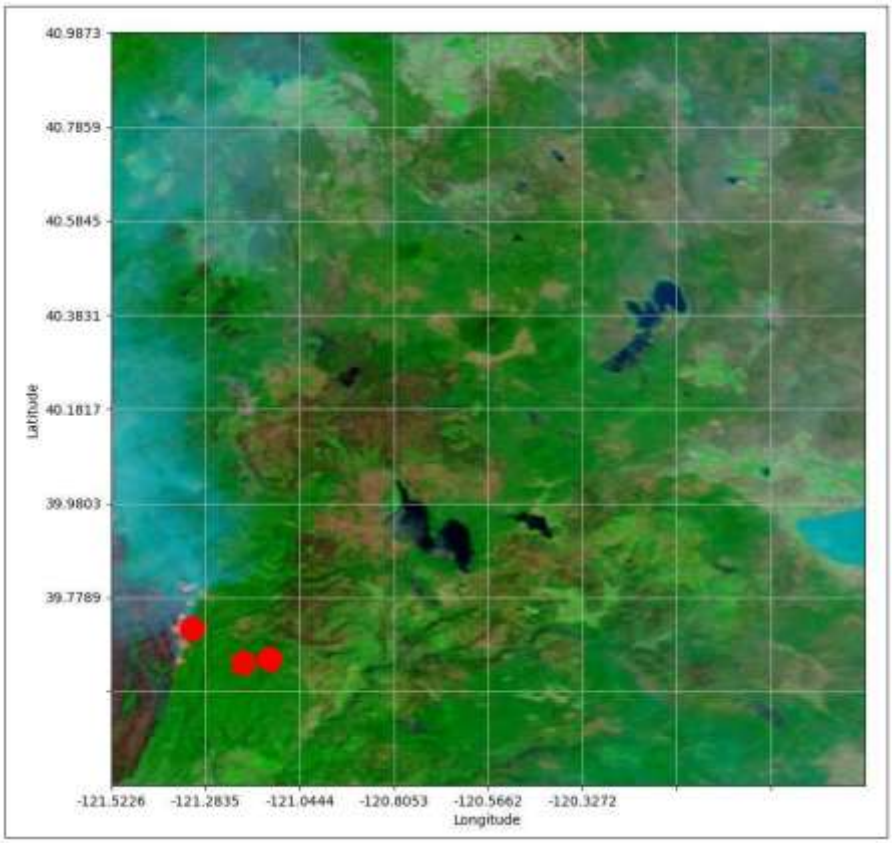


Figure 53. Tracking path 3

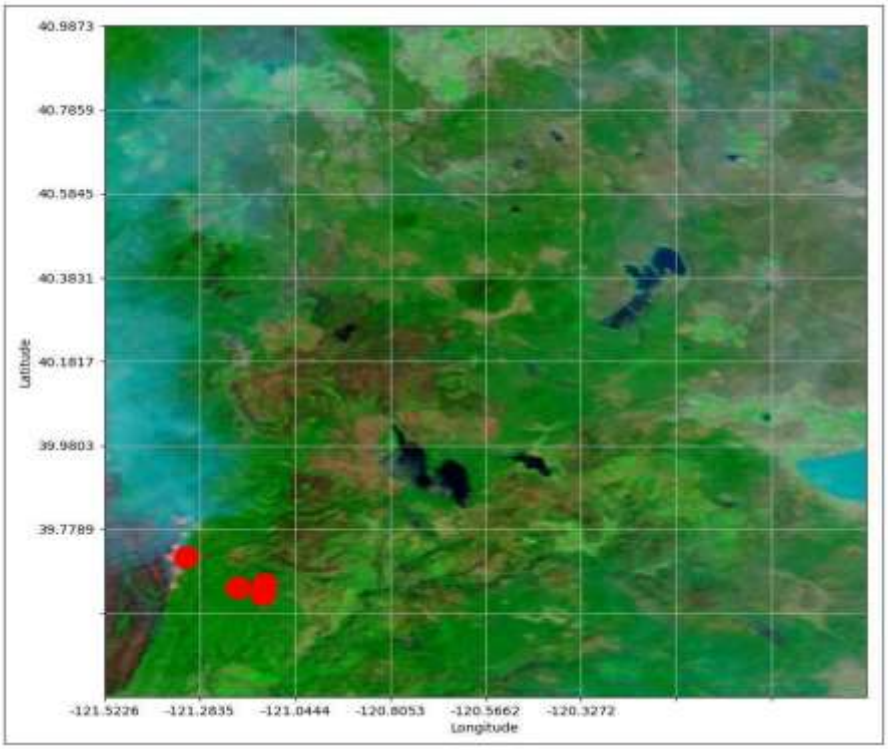


Figure 54. Tracking path 4

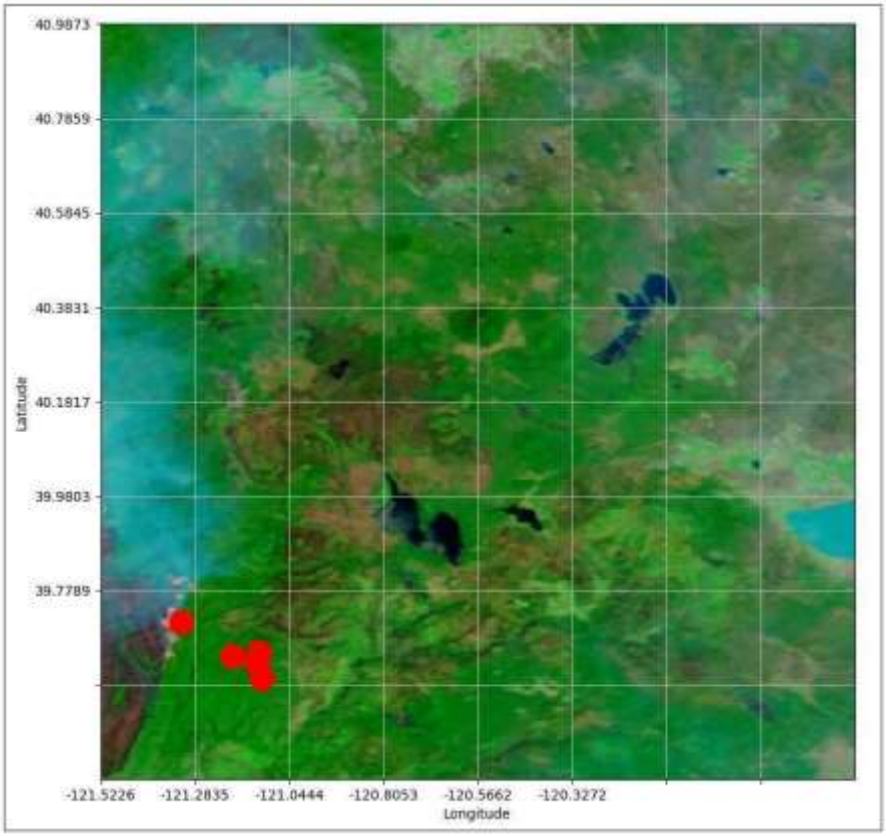


Figure 55. Tracking path 5