

# ***AUTOMATED WEATHER CLASSIFICATION USING TRANSFER LEARNING***

***TEAM MEMBERS:***

***BALA MURUGAN S (TEAM LEADER)***

***KARTHI S (TEAM MEMBER 1)***

***GOWTHAMAN A M (TEAM MEMBER 2)***

***BALAJI S (TEAM MEMBER 3)***

***TEAM ID: NM2023TMID00061***

# INDEX

## 1. INTRODUCTION

1.1 Project Overview

1.2 Purpose

## 2. IDEATION & PROPOSED SOLUTION

2.1 Problem Statement

2.2 Empathy Map Canvas

2.3 Brainstorming & Idea Prioritization

2.4 Proposed Solution

## 3. REQUIREMENT ANALYSIS

3.1 Functional requirement

3.2 Non-Functional requirements

## 4. PROJECT DESIGN

4.1 Data Flow Diagrams

4.2 Solution & Technical Architecture

4.3 User Stories

## 5. IBM CLOUD MANAGE

5.1 Registration

5.2 Watson Studio

5.3 Watson Machine Learning

5.4 Object Storage

5.5 Model Training and Deployment

## 6. CODING & SOLUTIONING

6.1 Loading the images

6.2 Image Preprocessing

6.3 Building the model

6.4 Training the model

6.5 Testing the model

6.6 Web Application Development

## 7. RESULTS

7.1 Performance Metrics

## 8. CONCLUSION

## 9. APPENDIX

# 1. INTRODUCTION

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems. AI requires a foundation of specialized hardware and software for writing and training machine learning algorithms. In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. Our project aims to develop an AI Model by building neural networks to classify the weather condition based on the images.

## 1.1 PROJECT OVERVIEW

This project aims to develop a weather classification system using deep learning techniques, specifically leveraging the capabilities of the IBM Cloud platform. The system will utilize deep neural networks to automatically classify weather conditions based on images.

By utilizing the IBM Cloud infrastructure, the project will benefit from its scalable computing power, extensive AI toolkits, and efficient data management capabilities. The project will encompass the development of deep learning models, training and evaluation processes, as well as the deployment and integration of the crime classification system on the IBM Cloud platform.

This report provides an overview of the project objectives, methodology, and the utilization of IBM Cloud resources for the successful implementation of the weather classification system.

## 1.2 PURPOSE

The purpose of this project is to develop a weather classification system using deep learning techniques. This project will be immensely valuable to the weather forecasters, meteorologists to know about the weather conditions in different places at the same time. The ultimate goal of this project is to assist the meteorologists in their field of work by updating them the weather conditions of different places time to time.

### **Specific Goals :**

- Developing a robust and accurate weather classification tasks
- Using Transfer Learning Techniques to achieve a better accuracy.
- Evaluating and optimizing the model
- Create a web Application that enables the user to upload the images and get the results.

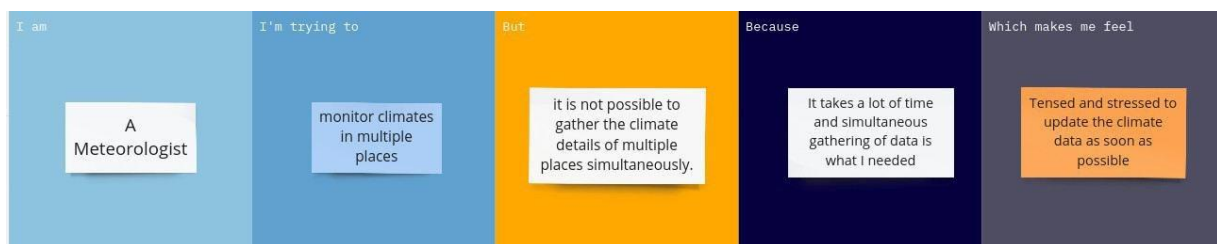
# 2. IDEATION & PROPOSED SOLUTION

## 2.1 Problem Statement

My Customer is a meteorologist/a scientist who monitors weather in multiple places. But it is not possible to gather data simultaneously without human assistance and satellite assistance. So he needs a solution to get the climate data up to date and in an accurate manner. So we built a solution to detect images using Transfer Learning which solves our customer's problem better.

<b>Problem Statement (PS)</b>	<b>I am (Customer)</b>	<b>I'm trying to</b>	<b>But</b>	<b>Because</b>	<b>Which makes me feel</b>
I need a tool to detect weather on multiple places at the same time without any human or satellite assistance.	A Meteorologist	Monitor climates in multiple places.	It is not possible to gather the climate details of multiple places simultaneously.	It takes a lot of time and simultaneous gathering of data is what I needed	Tensed and stressed to update the climate data as soon as possible.
I need a tool to know the weather on a place when I am visiting it.	A Traveller	Know the climate in a particular place.	It is not possible to get an up to date information.	The traditional system has much delays.	Worried about the unknown weather in a place and it spoils my travel plan.
I am in need of a tool to automatically adjust indoor temperature, lighting, or other environmental factors based on the weather conditions.	A Smart home Company	Know the climate of a particular region	It is not accurate and updated.	The traditional system is not adaptive as fast as the climate changes.	Hard in adjusting the indoor temperature, lighting, etc.

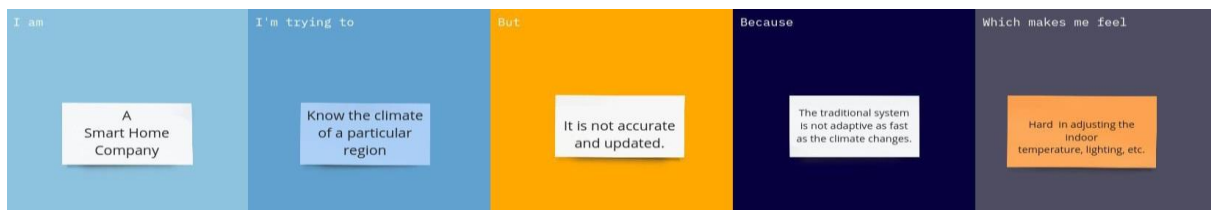
### *Problem Statement of Meteorologist*



### *Problem Statement of A Traveller*



### *Problem Statement of a Smart Home Company*



## 2.2 EMPTAHY MAP



## 2.3 BRAINSTORMING AND IDEA PRIORITIZATION



## Brainstorm & idea prioritization

Use this template in your own brainstorming sessions so your team can unleash their imagination and start shaping concepts even if you're not sitting in the same room.

🕒 10 minutes to prepare

🕒 1 hour to collaborate

👤 2-8 people recommended

💬 Share template feedback



### Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

🕒 10 minutes



#### A Team gathering

Define who should participate in the session and send an invite. Share relevant information or pre-work ahead.



#### B Set the goal

Think about the problem you'll be focusing on solving in the brainstorming session.



#### C Learn how to use the facilitation tools

Use the Facilitation Superpowers to run a happy and productive session.

[Open article](#) →



### 1 Define your problem statement

What problem are you trying to solve? Frame your problem as a How Might We statement. This will be the focus of your brainstorm.

🕒 5 minutes

#### PROBLEM

**Need to monitor weather in multiple places. But it is not possible to gather data simultaneously without human assistance and satellite assistance.**



#### Key rules of brainstorming

To run a smooth and productive session



Stay in topic.



Encourage wild ideas.



Defer judgment.



Listen to others.



Go for volume.



If possible, be visual.

2

## Brainstorm

Write down any ideas that come to mind that address your problem statement.

🕒 10 minutes

### TIP

You can select a sticky note and hit the pencil [switch to sketch] icon to start drawing!

Person 1



Determine the classes such as sunny, cloudy, rainy, foggy, etc.

Use publicly available datasets or collection of own images using weather cameras, satellite images, or webcams.

Collecting high-quality images of weather conditions

Person 2



Appropriate deep learning algorithm, should be chosen such as convolutional neural network (CNN).

Preprocess the data to ensure that the images are in the same format and resolution

The images should be cropped, resized or normalized.

Person 3



Use color filters to enhance certain features of the images.

Data cleaning should be performed to remove duplicate or irrelevant images that do not belong to any of the weather classes.

Quality of the data significantly impacts the accuracy of the model.

Person 4



Model should be fine-tuned by adjusting hyper-parameters, such as the learning rate, batch size, or number of epochs.

ROC curve should be used for graphical representation of the performance of the model at different classification thresholds.

Precision and recall are two metrics that are used to measure the accuracy of the model for each class.

3

## Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

🕒 20 minutes

### TIP

Add customizable tags to sticky notes to make it easier to find, browse, organize, and categorize important ideas as themes within your mural.

User interface provides an intuitive and user-friendly way for users to interact with the model and get real-time weather predictions. We can use user interface to create a web-based or mobile application that allows users to upload images or enter weather conditions, and get instant feedback on the weather predictions

Transfer learning is a technique that leverages pre-trained models to speed up the training process and improve the accuracy of the model. We can use transfer learning to adapt a pre-trained model, such as VGG, ResNet, or Inception, to our Weather classification task.

Image augmentation is a technique that generates new images by applying random transformations to the original images. We can use image augmentation to increase the size of our dataset and improve the robustness of your model. As a team, we can experiment with different types of image augmentations, such as rotation, scaling, cropping, and flipping, and compare their effects on the model's performance.



4

## Prioritize

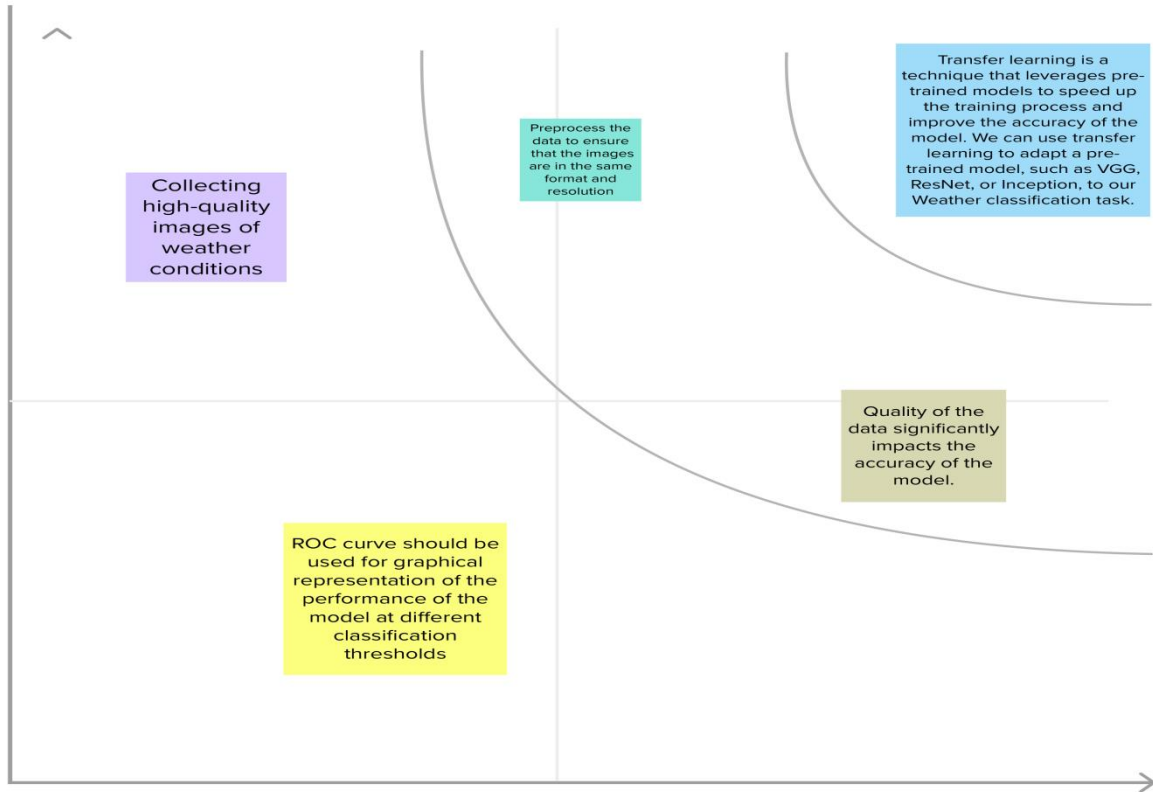
Your team should all be on the same page about what's important moving forward. Place your ideas on this grid to determine which ideas are important and which are feasible.

⌚ 20 minutes

### TIP

Participants can use their cursors to point at where sticky notes should go on the grid. The facilitator can confirm the spot by using the laser pointer holding the **H** key on the keyboard.

**Importance**  
If each of these tasks could get done without any difficulty or cost, which would have the most positive impact?



**Feasibility**  
Regardless of their importance, which tasks are more feasible than others? (Cost, time, effort, complexity, etc.)

## 2.4 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	Often times, Meteorologists and weather forecasters need to monitor weather in multiple places simultaneously without human and satellite assistance.
2.	Idea / Solution description	We can use transfer learning to adapt a pre-trained model, such as VGG, ResNet, or Inception, to our Weather classification task. Transfer Learning is a technique that leverages pre-trained models to speed up the training process

		and improve the accuracy of the model.
3.	Novelty / Uniqueness	We have collected a novel dataset for weather classification from images. This dataset could help to improve the accuracy of weather classification models. We believe that our model would achieve a state-of-the-art performance on a benchmark dataset for weather classification.
4.	Social Impact / Customer Satisfaction	Our customer needs to know the weather condition of multiple places accurately and simultaneously. Our model enables the customer to update the weather from the images of multiple places.
5.	Business Model (Revenue Model)	Our customers include businesses in the agriculture, energy industries, as well as government agencies involved in disaster response. We can make our model accessible to customers by deploying it using the cloud based APIs and the revenue could be earned by employing charge for the number of API calls and creating subscription models based on that. Fine tuning the Neural network can also improve the accuracy of the model which helps to earn better. Our customers include businesses in the agriculture, energy industries, as well as government agencies involved in disaster response. We can make our model accessible to customers by deploying it using the cloud based APIs and the revenue could be earned by employing charge for the number of API calls and creating subscription models based on that. Fine tuning the Neural network can also improve the accuracy of the model which helps to earn better.
6.	Scalability of the Solution	A large dataset could improve the accuracy of the model thereby improving the scalability, Also the need for scalability of our model depends on the number of customers, if there are more number of customers then the number of API calls to be handled is also so high. When we deploy our model on Amazon Web Services(AWS) or Google Cloud they use automatic scalability and rapid Elasticity. Thus scalability of the proposed solution is possible.

### 3. REQUIREMENT ANALYSIS

#### Functional Requirements

Following are the functional requirements of the proposed solution.

FR NO	Functional Requirement (Epic)	Sub Requirement (Story / sub-Task)
FR-1	Data Collection	Implement a data collection module that retrieves weather data from different sources and stores it in a structured format for further processing.
FR-2	Preprocessing and Data Cleaning	Develop preprocessing algorithms and techniques to handle missing data, remove outliers, and normalize the data for training.
FR-3	Transfer Learning Model	Utilize pre-trained models, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs). Fine-tune the pre-trained model on the weather dataset using transfer learning approaches.
FR-4	Feature Extraction	Design feature extraction methods to determine weather conditions based on the images
FR-5	Model Training and Evaluation	Train the transfer learning model on the preprocessed data using appropriate classification algorithms . Evaluate the model's performance using metrics like accuracy, precision etc....
FR-6	Weather Classification	Deploy the trained model to classify weather conditions into categories such as sunny, cloudy, rainy, foggy, or stormy
FR-7	Real-time Weather Classification	Implement a real-time processing module that continuously collects and processes the latest weather data.

FR-8	Maintenance and Updates	Design the solution with modular and well-documented code to facilitate easy maintenance and updates. Implement version control and monitoring mechanisms to track changes and ensure system stability.
FR-9	API Integration	Develop a well-documented and secure API that enables integration with other applications or services, allowing them to retrieve weather classification results programmatically.

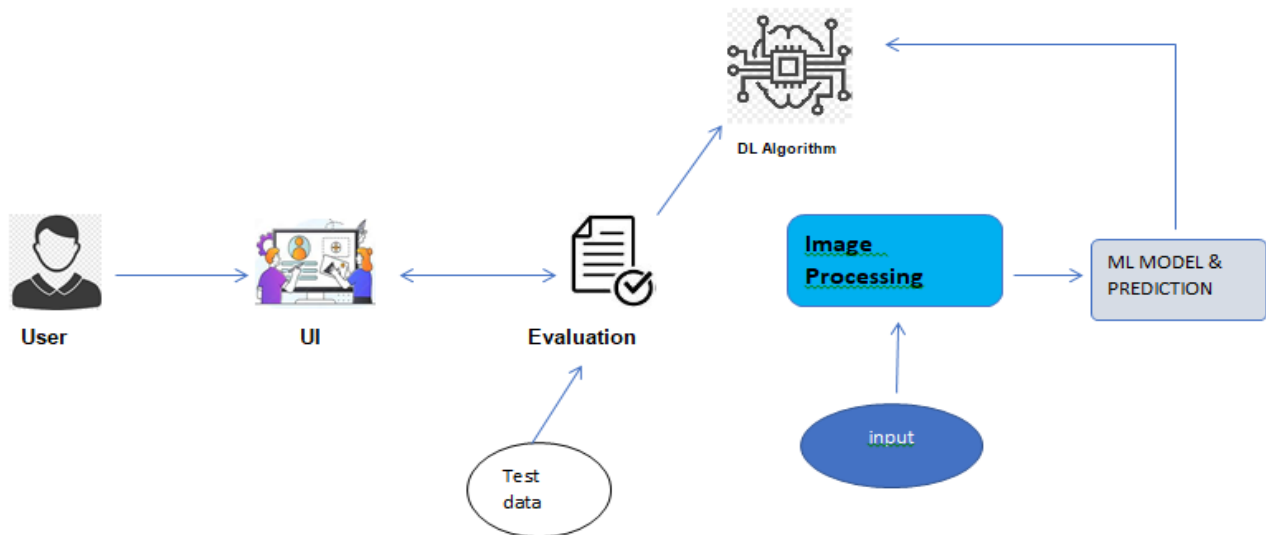
## Non-Functional Requirements:

Following are the non-functional Requirements of the proposed solution

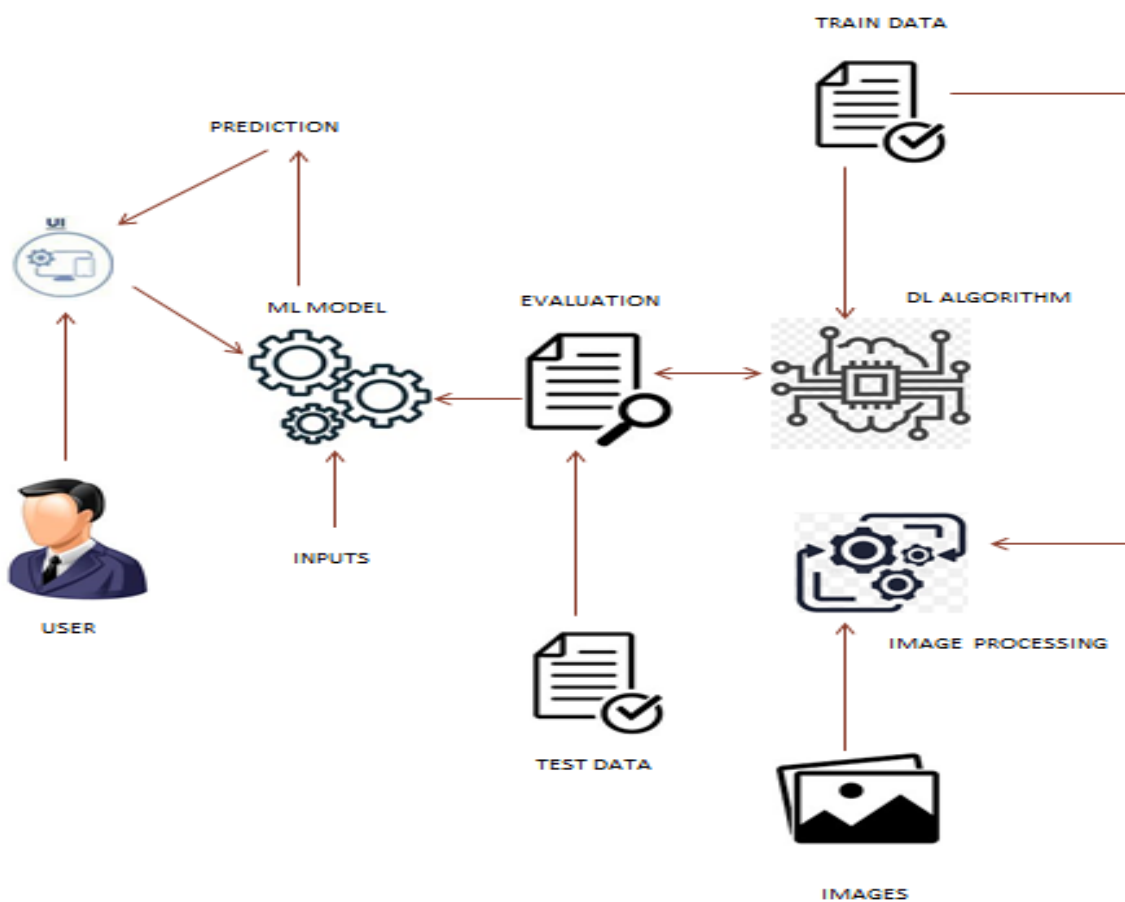
FR NO	Non-Functional Requirements	Description
FR-1	Usability	Conduct user-centered design and usability testing to ensure a well-designed and intuitive user interface. Provide clear instructions, tooltips, and error messages to guide users in interacting with the system effectively
FR-2	Security	Implement robust security measures, such as encryption, access controls, and user authentication, to protect user data. Follow security best practices and comply with relevant data protection regulations, such as GDPR or CCPA.
FR-3	Reliability	Implement fault-tolerant mechanisms, such as redundant servers or failover systems, to ensure high availability. Conduct regular system monitoring and implement automated error detection and recovery procedures.
FR-4	Performance	Optimize the code and algorithms to minimize computational overhead and improve processing speed. Utilize parallel processing or distributed computing techniques to enhance performance, especially when dealing with large-scale datasets.
FR-5	Availability	The system should be always available and accessible to user, minimizing downtime and disruptions. It should have robust infrastructure, fault- tolerant design, and appropriate backup mechanisms to ensure high availability
FR-6	Scalability	Employ scalable infrastructure, such as cloud-based services or distributed computing, to handle increased data loads. Implement efficient data storage and retrieval mechanisms to accommodate growing datasets.

## 4. PROJECT DESIGN:

### 4.1 DATA FLOW DIAGRAMS



## 4.2 SOLUTION AND TECHNICAL ARCHITECTURE



### IMAGE PROCESSING:

Images are loaded from the training dataset and they are resized according to the model's parameters. This process is called image processing. These images are split into two parts namely training set and validation set.

### **DL ALGORITHM :**

The deep learning algorithm used here is an artificial neural network called VGG19 belonging to the Tensorflow package. This algorithm takes in the learning rate, activation function and number of epochs which effectively influences the accuracy of the model. The model is trained using the images in the training dataset and validated at the end of each epoch using the validation dataset.

### **EVALUATION PHASE :**

During the evaluation phase, the test dataset which is foreign from the training set is given as an input to the model, to test its accuracy. This phase determines our model's capacity to detect weather conditions and this helps in fine-tuning the hyperparameters of the neural network if needed.

### **ML MODEL :**

The model is loaded and it is now ready to take images from the user. It takes in an image input and that is processed as discussed above. Then the model gives an output. This is called Prediction. The Predicted output is given to the Web Application using API calls.

### **USER INTERFACE :**

A user interface (UI) for a web application is the visual and interactive design that allows users to interact with the application through a web browser. The User interface enables the user to upload the images , then it feeds the images to the model and gives the predicted weather as the output. The User interface needs to be simple enough and here it is built using flask (Python). Thus a user (Meteorologist/weather Forecaster) can use this model to effectively detect the weather conditions in multiple places simultaneously without any human or satellite assistance thereby achieving the goal of this project.

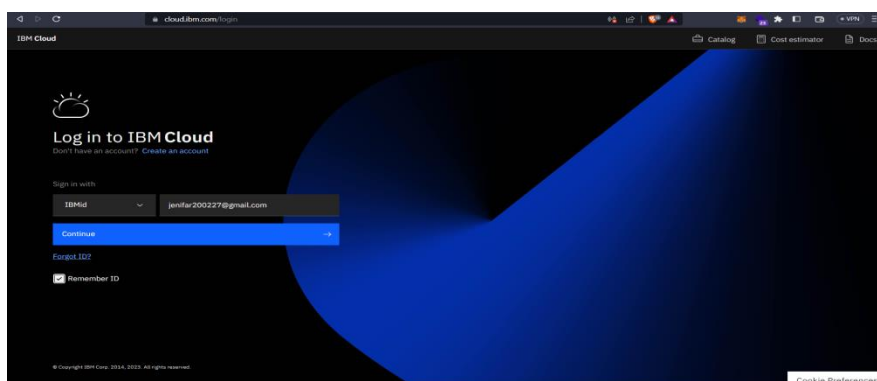
## **4.3 USER STORIES**

User Type	Functional Requirement (Epic)	User Story Number	User Story/ Task	Acceptance criteria	Priority	Team Member
Data Scientist/Researcher	Decide weather	USN-1	As a researcher, I want to decide the weather based on the categories of images	The model should provide the conditions of detection and classification of weather such as sunny, rainy, cloudy, or foggy	medium	GOWTHAMAN A M
	Improve the accuracy	USN-2	As a researcher, I want to improve the accuracy upto valid level.	The model should achieve an accuracy of at least 85% on a labeled weather dataset during training and validation.	high	BALA MURUGAN S
	Train the weather detection system	USN-3	As a data scientist, I want to train a weather detection model using transfer learning	The system should be based on a pre-trained deep learning architecture suitable for image classification.	low	KARTHI S
Developer	Get different image formats	USN-4	As a developer, I want to handle different image formats	The model should handle different image formats commonly used in weather data, such as JPEG, PNG, or TIFF.	low	BALAJI S
	Integrate the model into automated system	USN-5	As a developer, I want to integrate the weather detection model into an automated system.	The model should provide a user-friendly API or SDK for easy integration with the existing system.	high	GOWTHAMAN A M
End User/Operator	Generate weather information	USN-6	As an end user, I want to use the automated	The system should provide a user interface that allows me to upload an image or capture an image using a camera for weather detection.	high	KARTHI S

## 5. IBM CLOUD MANAGE

### 5.1 REGISTRATION

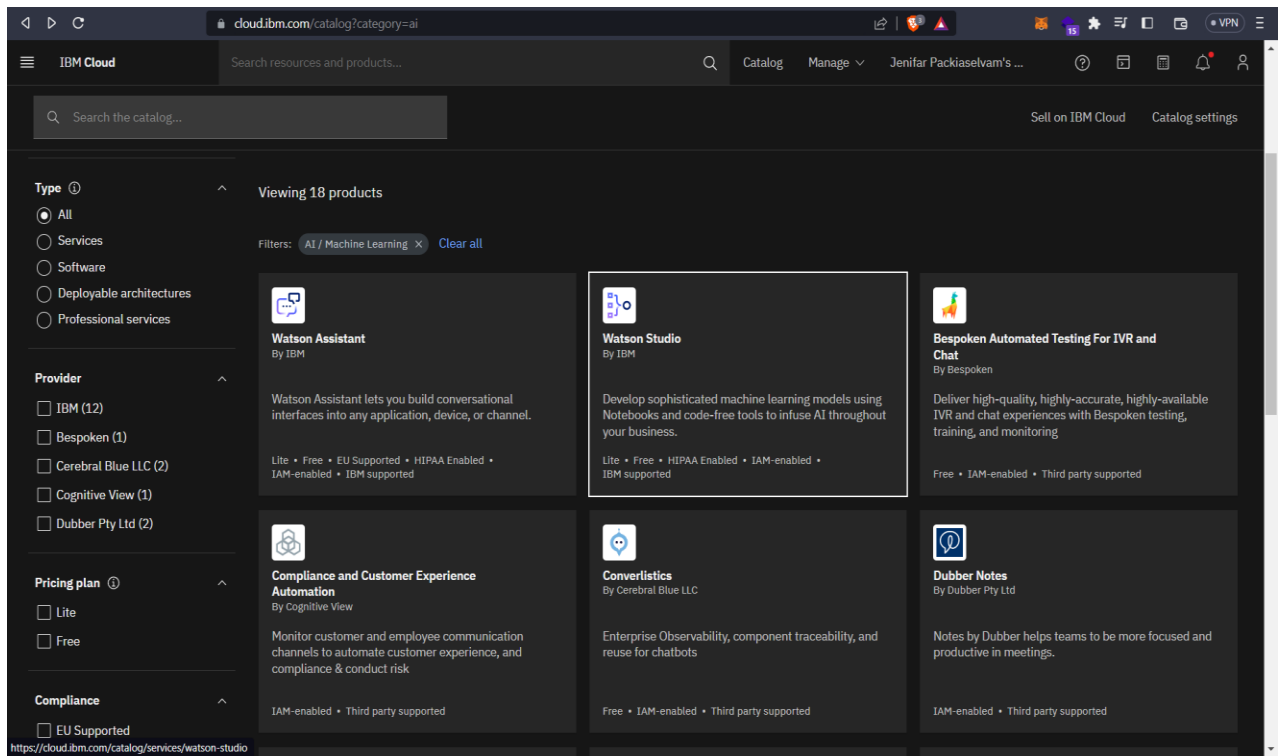
In IBM Cloud, a register is a process of creating an account to access the services and resources provided by the IBM platform. Go to IBM Cloud website (<https://cloud.ibm.com>). Click on the "Create an IBM Cloud account" or "Sign up" button. It's directed to the registration page. Fill in the required information to create IBM cloud account.



### 5.2 WATSON STUDIO

In this project, we leverage the capabilities of IBM Watson, a comprehensive suite of AI-powered tools and services, to enhance the development and implementation of our advanced crime classification system.

Specifically, we utilize the IBM Watson platform to access and utilize its deep learning frameworks, such as IBM Watson Studio and IBM Watson Machine Learning. These tools enable us to develop sophisticated machine learning models using Notebooks and code-free tools to infuse AI throughout your business.

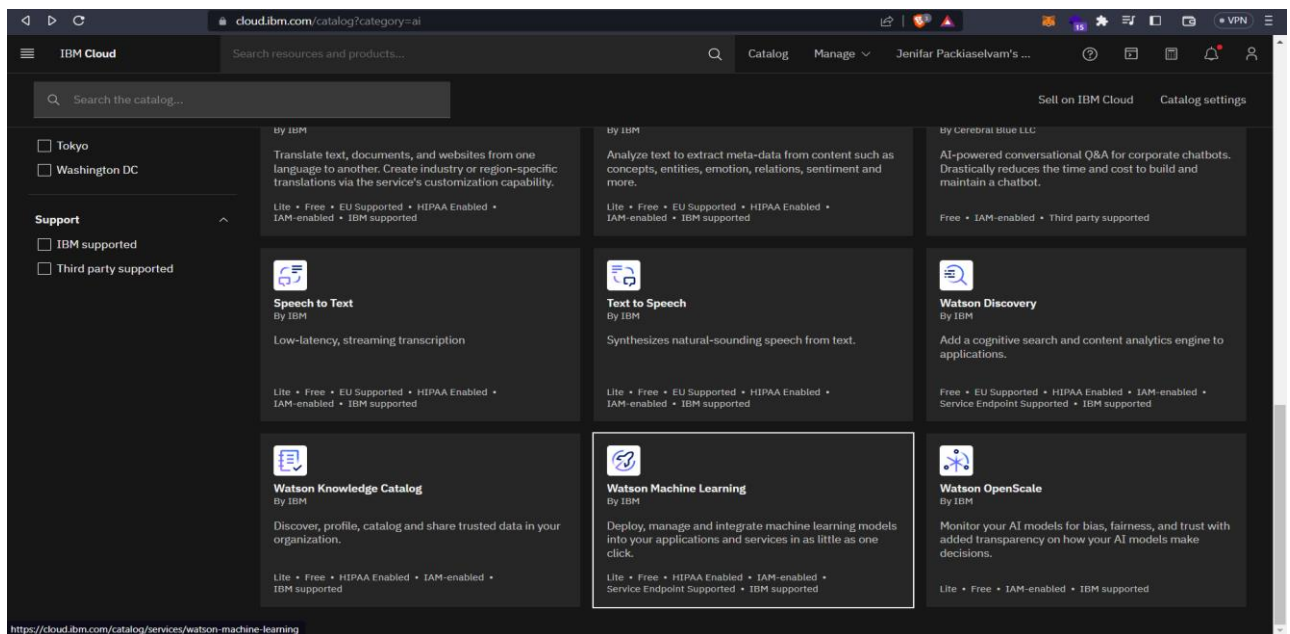


Additionally, the scalability and reliability of the IBM Cloud infrastructure provided by IBM Watson facilitate efficient data management, model deployment, and integration into our crime classification system. By utilizing IBM Watson, we ensure access to cutting-edge AI technologies and resources, empowering us to achieve accurate and efficient crime classification results.

### 5.3 WATSON MACHINE LEARNING

This project leverages IBM Watson Machine Learning, a vital component of the IBM Watson platform, to optimize the training, deployment, and seamless integration of our deep learning models for crime classification.

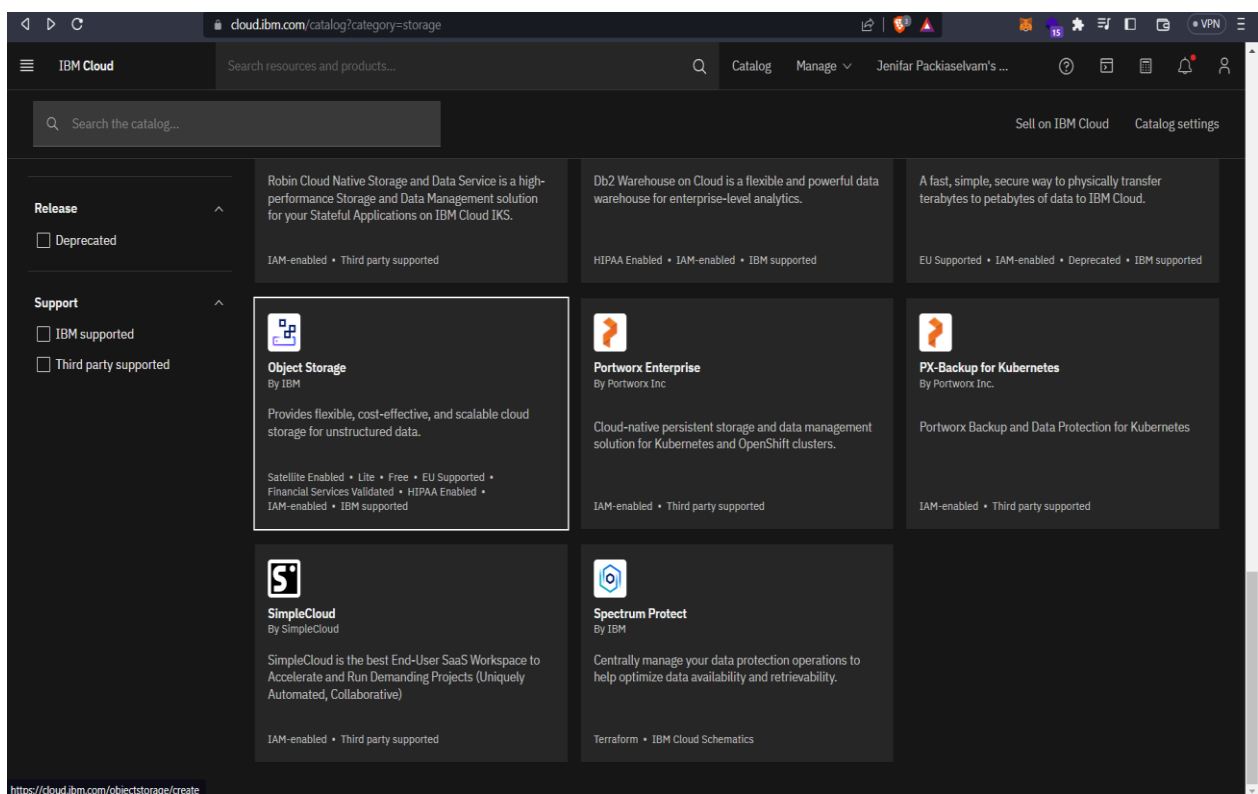
By harnessing the capabilities of IBM Watson Machine Learning, we enhance the performance of our models, fine-tune hyperparameters, and streamline the training process. Furthermore, we benefit from the efficient deployment mechanisms offered by IBM Watson Machine Learning, enabling smooth integration of our models into the broader crime classification system.



## 5.4 CLOUD OBJECT STORAGE

Our project relies on the robust capabilities of IBM Object Storage for the efficient management and storage of extensive crime data utilized in our deep learning models. With IBM Object Storage, we benefit from a cost-effective and highly durable cloud storage infrastructure, ensuring the secure retention and retrieval of crime scene images, textual descriptions, and other relevant data.

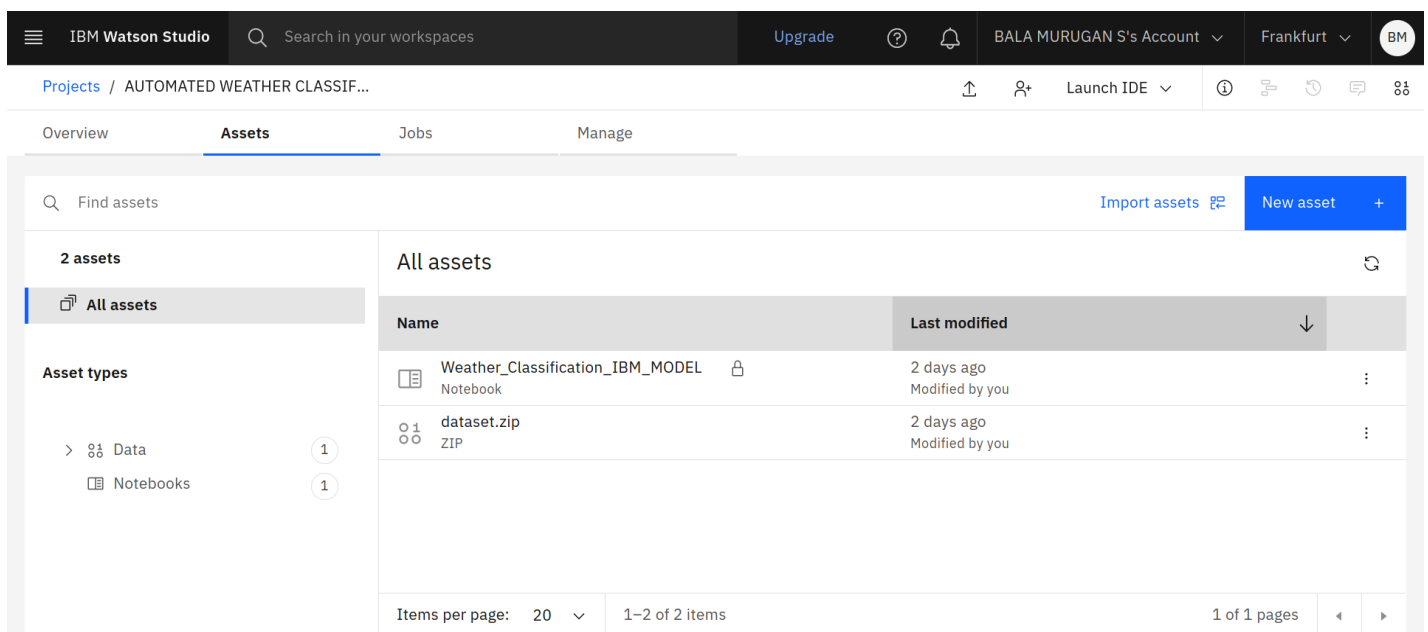
This solution enables seamless availability and accessibility of the data throughout the training and evaluation phases of our weather classification system.





## 5.5 MODEL TRAINING AND DEPLOYMENT

Cloud's Model Training and Deployment services optimize deep learning model training and deployment for crime classification, enhancing efficiency. The parallelization of training tasks offered by IBM Cloud Model Training significantly reduces training time. We achieve faster training times for complex models on large datasets. Following figures show how we deployed on IBM cloud.



The screenshot displays the IBM Watson Studio interface. At the top, there's a navigation bar with 'IBM Watson Studio', a search bar, and user account information. Below this, a breadcrumb trail shows 'Projects / AUTOMATED WEATHER CLASSIF...'. The main content area is divided into tabs: 'Overview', 'Assets' (selected), 'Jobs', and 'Manage'. On the left, a sidebar shows '2 assets' and 'Asset types' with 'Data' and 'Notebooks' categories. The main panel, titled 'All assets', contains a table with two items:

Name	Last modified
Weather_Classification_IBM_MODEL Notebook	2 days ago Modified by you
dataset.zip ZIP	2 days ago Modified by you

At the bottom of the table, it indicates 'Items per page: 20' and '1-2 of 2 items'.

## 6. CODING & SOLUTIONING

### 6.1 LOADING THE IMAGES

The dataset is downloaded from Kaggle and the zip file is unzipped using a small snippet of code. The necessary libraries are imported such as tensorflow, numpy, etc. The images are loaded using the flow from directory().

#### UNZIPPING THE LOADED DATASET

```
In [47]: from io import BytesIO
import zipfile

zip_ref = zipfile.ZipFile(BytesIO(streaming_body.read()), 'r')
zip_ref.extractall('/home/wsuser/work/')
zip_ref.close()
```

#### IMAGE DATA GENERATION AND PREPROCESSING

```
In [63]: from tensorflow.keras.applications.vgg19 import VGG19, preprocess_input
from tensorflow.keras.layers import Flatten, Dense
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from PIL import ImageFile
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
```

```
train_datagen = ImageDataGenerator(rescale = 1./255, shear_range = 0.2, zoom_range = [.99, 1.01], brightness_range = [0.8
training_set = train_datagen.flow_from_directory('/home/wsuser/work/home/wsuser/work/dataset/train', target_size = (18
test_set = train_datagen.flow_from_directory('/home/wsuser/work/home/wsuser/work/dataset/train', target_size = (180, 18
```

```
Found 1225 images belonging to 5 classes.
Found 304 images belonging to 5 classes.
```

## 6.2 IMAGE PREPROCESSING

Image preprocessing is done using the Image Data Generator(). It is used for the generation of the batches containing the data of tensor images and is used in the domain of real-time data augmentation. We can loop over the data in batches when we make use of the images. The images need to be resized and the zoom range and shear range are passed as parameters.

## 6.3 BUILDING THE MODEL

Here we make use of a pretrained model VGG19 as a feature extractor. Pre-trained models, especially those trained on large and diverse datasets (e.g., ImageNet), have already learned meaningful and generalizable features from the data. These models have gone through extensive training to extract relevant patterns and features that can be useful for a wide range of tasks, including weather classification. By utilizing a pre-trained model as a feature extractor, you can leverage these learned features instead of starting from scratch, which can significantly speed up the training process and improve the performance of our model. Added Dense layers and the activation function “Softmax” is used for the Multiclass Classification.

#### ADDING A PRETRAINED MODEL VGG19 AS A FEATURE EXTRACTOR

```
In [65]: IMAGE_SIZE = [180,180]
VGG19 = VGG19(input_shape= IMAGE_SIZE+[3],weights = 'imagenet',include_top = False)
for layer in VGG19.layers:
    layer.trainable = False
x = Flatten()(VGG19.output)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg19/vgg19_weights_tf_dim_orderi
ng_tf_kernels_notop.h5
80134624/80134624 [=====] - 1s 0us/step
```

```
In [66]: prediction = Dense(5,activation = 'softmax')(x)
model = Model(inputs = VGG19.input,outputs = prediction)
model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[None, 180, 180, 3]	0
block1_conv1 (Conv2D)	(None, 180, 180, 64)	1792
block1_conv2 (Conv2D)	(None, 180, 180, 64)	36928
block1_pool (MaxPooling2D)	(None, 90, 90, 64)	0
block2_conv1 (Conv2D)	(None, 90, 90, 128)	73856
block2_conv2 (Conv2D)	(None, 90, 90, 128)	147584
block2_pool (MaxPooling2D)	(None, 45, 45, 128)	0
block3_conv1 (Conv2D)	(None, 45, 45, 256)	295168
block3_conv2 (Conv2D)	(None, 45, 45, 256)	590080
block3_conv3 (Conv2D)	(None, 45, 45, 256)	590080

## 6.4 TRAINING THE MODEL

The optimizer used by our model is “Adam”, it is the most common optimizer used in image classification. The Loss function used here is “Categorical Crossentropy” which is usually used in the multiclass classification problems where the target variable is labelled not numbered. After training the model we achieved an Accuracy of 85.71% . Then the model file with h5 extension is saved and it is downloaded as “wcv.h5”

## TRAINING THE MODEL

```
In [67]: model.compile(loss = 'categorical_crossentropy',optimizer = 'adam',metrics = ['accuracy'])
r = model.fit(training_set,validation_data = test_set,epochs = 50,steps_per_epoch = len(training_set),validation_step
loss,accuracy = model.evaluate(test_set,steps = 11,verbose = 2,use_multiprocessing=True,workers=2)
print(f'model performance on test images:\n Accuracy = {accuracy}\n Loss = {loss}')
|

Epoch 46/50
20/20 [=====] - 314s 16s/step - loss: 0.0481 - accuracy: 0.9918 - val_loss: 0.4488 - val_ac
curacy: 0.8553
Epoch 47/50
20/20 [=====] - 310s 16s/step - loss: 0.0507 - accuracy: 0.9894 - val_loss: 0.4384 - val_ac
curacy: 0.8586
Epoch 48/50
20/20 [=====] - 312s 16s/step - loss: 0.0434 - accuracy: 0.9943 - val_loss: 0.4091 - val_ac
curacy: 0.8783
Epoch 49/50
20/20 [=====] - 309s 16s/step - loss: 0.0457 - accuracy: 0.9927 - val_loss: 0.4445 - val_ac
curacy: 0.8684
Epoch 50/50
20/20 [=====] - 312s 16s/step - loss: 0.0455 - accuracy: 0.9943 - val_loss: 0.4496 - val_ac
curacy: 0.8322
11/11 - 140s - loss: 0.4196 - accuracy: 0.8571 - 140s/epoch - 13s/step
model performance on test images:
Accuracy = 0.8571428656578064
Loss = 0.4195686876773834

In [68]: model.save('wcv.h5')
```

## 6.5 TESTING THE MODEL

After saving the model we started testing the model with random images from the test dataset which our model never seen before I.e., the images in the test set and training set are different. We tested our model with two random images both are predicted correctly.

```
In [77]: from tensorflow.keras.preprocessing import image
from PIL import Image
im = Image.open(r"/home/wsuser/work/home/wsuser/work/dataset/TEST/rain_2.png")
im = im.resize((300,180))
im.show()
model = load_model("/home/wsuser/wcv.h5")
img = image.load_img("/home/wsuser/work/home/wsuser/work/dataset/TEST/rain_2.png",target_size = (180,180))

x = image.img_to_array(img)
x = np.expand_dims(x,axis = 0)
preds = model.predict(x)
pred = np.argmax(preds,axis = 1)
index = ['cloudy','foggy','rainy','sunshine','sunrise']
result = str(index[pred[0]])
print(result)
```



WARNING:tensorflow:5 out of the last 5 calls to <function Model.make\_predict\_function.<locals>.predict function at 0x7fe7bad2f910> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce\_retracing=True option that can avoid unnecessary retracing. For (3), please refer to [https://www.tensorflow.org/guide/function#controlling\\_retracing](https://www.tensorflow.org/guide/function#controlling_retracing) and [https://www.tensorflow.org/api\\_docs/python/tf/function](https://www.tensorflow.org/api_docs/python/tf/function) for more details.

```
1/1 [=====] - 0s 324ms/step
rainy
```

```
In [80]: im = Image.open(r"/home/wsuser/work/home/wsuser/work/dataset/TEST/sunrise_1.jpg")
im = im.resize((300,180))
im.show()
model = load_model("/home/wsuser/wcv.h5")
img = image.load_img("/home/wsuser/work/home/wsuser/work/dataset/TEST/sunrise_1.jpg",target_size = (180,180))

x = image.img_to_array(img)
x = np.expand_dims(x,axis = 0)
preds = model.predict(x)
pred = np.argmax(preds,axis = 1)
index = ['cloudy','foggy','rainy','sunshine','sunrise']
result = str(index[pred[0]])
print(result)
```

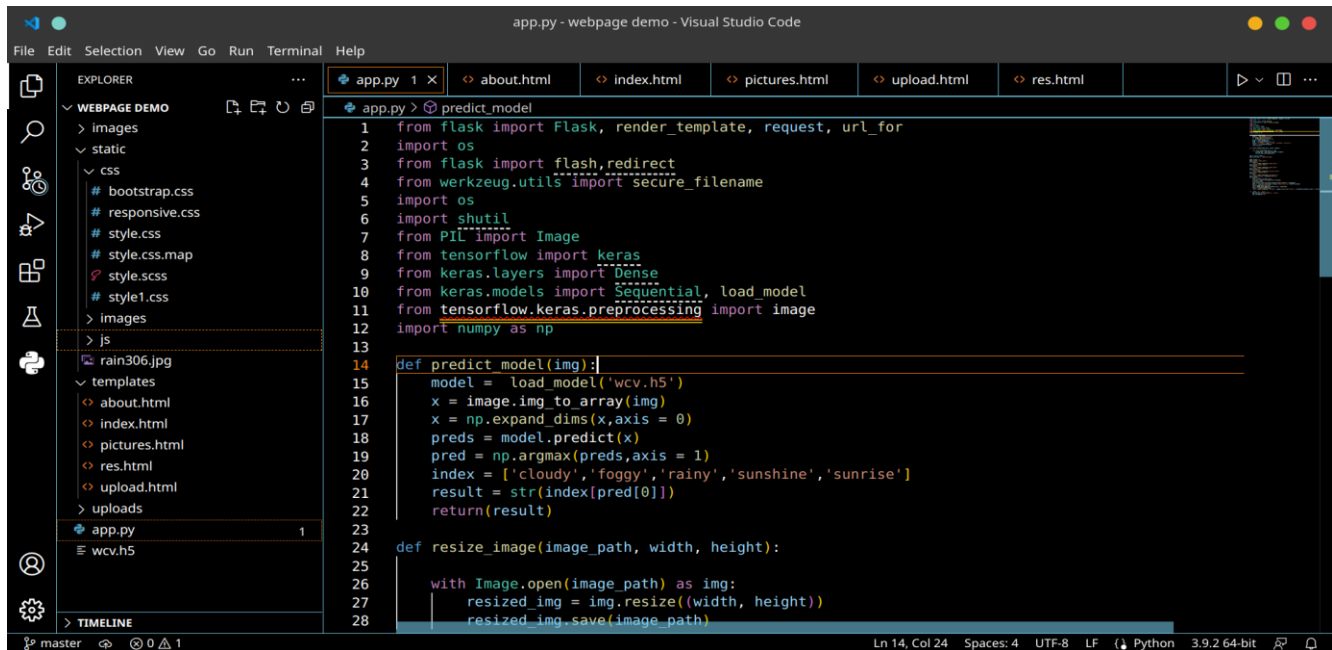


1/1 [=====] - 0s 316ms/step  
sunrise

## 6.5 BUILDING A WEB APPLICATION

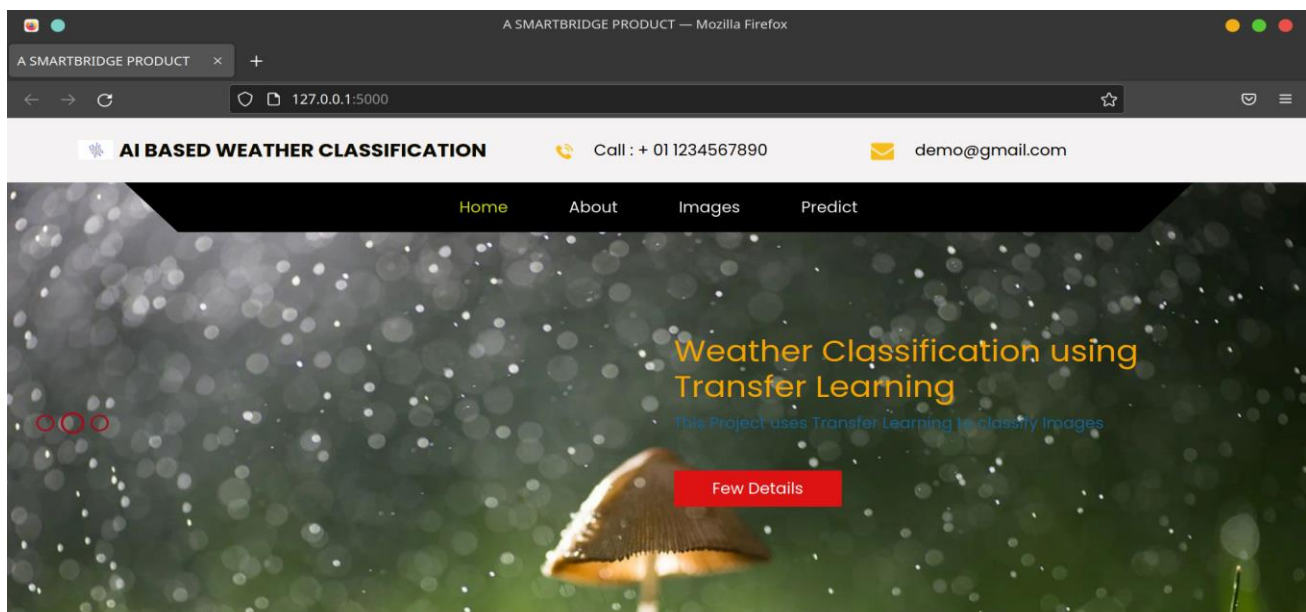
After saving the model file we started working on the front end and backend web development. Flask framework is used to create the backend web integration. HTML and CSS are used to develop the front-end design. Then the flask file “app.py” was run and the webpage is hosted locally. There are basically four tabs in the web page. They are “HOME”, “ABOUT”, “IMAGES”, “PREDICT”. The Home page contains a slider and gives the basic information of the web page. The about tab gives the user additional details and redirects the user to the images tab where the user can see the images that our model has predicted.

The following figure shows the app.py file,

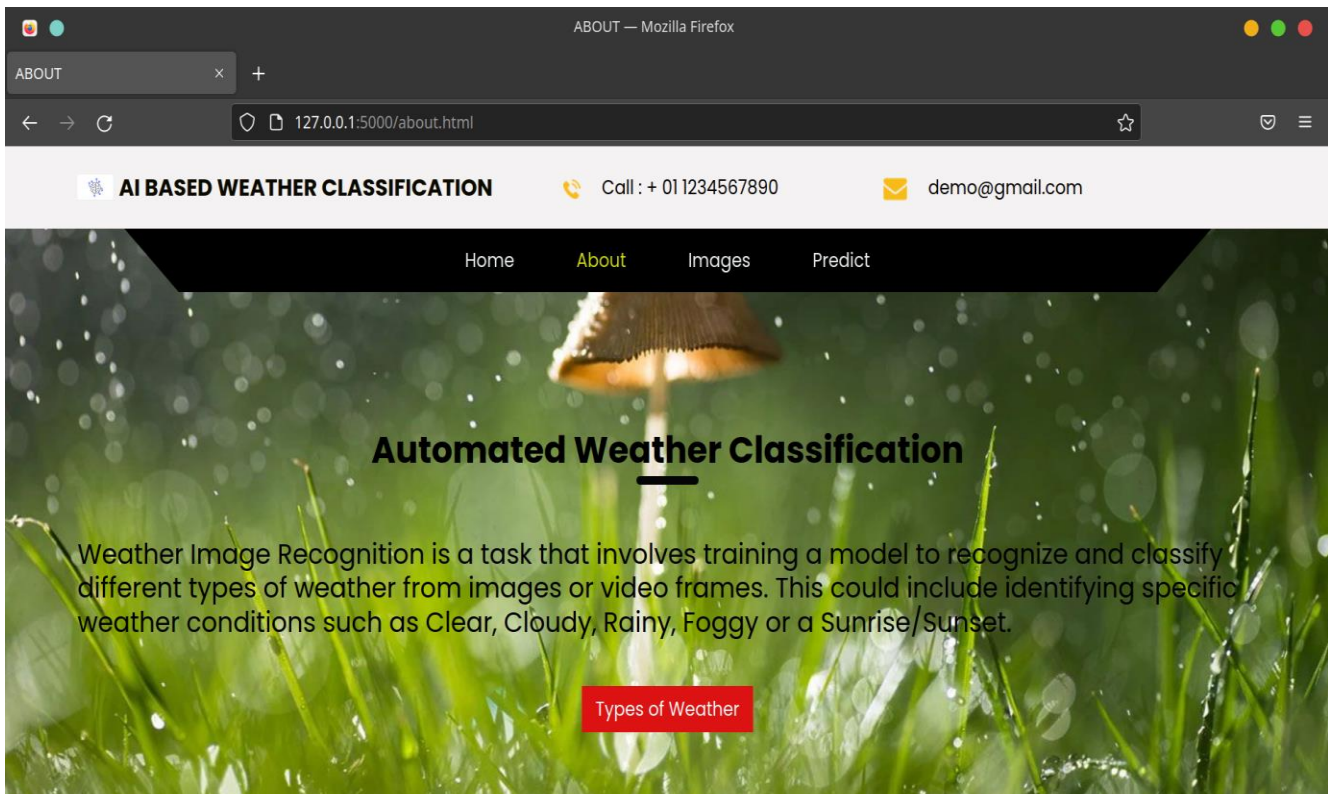


```
1 from flask import Flask, render_template, request, url_for
2 import os
3 from flask import flash, redirect
4 from werkzeug.utils import secure_filename
5 import os
6 import shutil
7 from PIL import Image
8 from tensorflow import keras
9 from keras.layers import Dense
10 from keras.models import Sequential, load_model
11 from tensorflow.keras.preprocessing import image
12 import numpy as np
13
14 def predict_model(img):
15     model = load_model('wcv.h5')
16     x = image.img_to_array(img)
17     x = np.expand_dims(x, axis = 0)
18     preds = model.predict(x)
19     pred = np.argmax(preds, axis = 1)
20     index = ['cloudy', 'foggy', 'rainy', 'sunshine', 'sunrise']
21     result = str(index[pred[0]])
22     return(result)
23
24 def resize_image(image_path, width, height):
25
26     with Image.open(image_path) as img:
27         resized_img = img.resize((width, height))
28         resized_img.save(image_path)
```

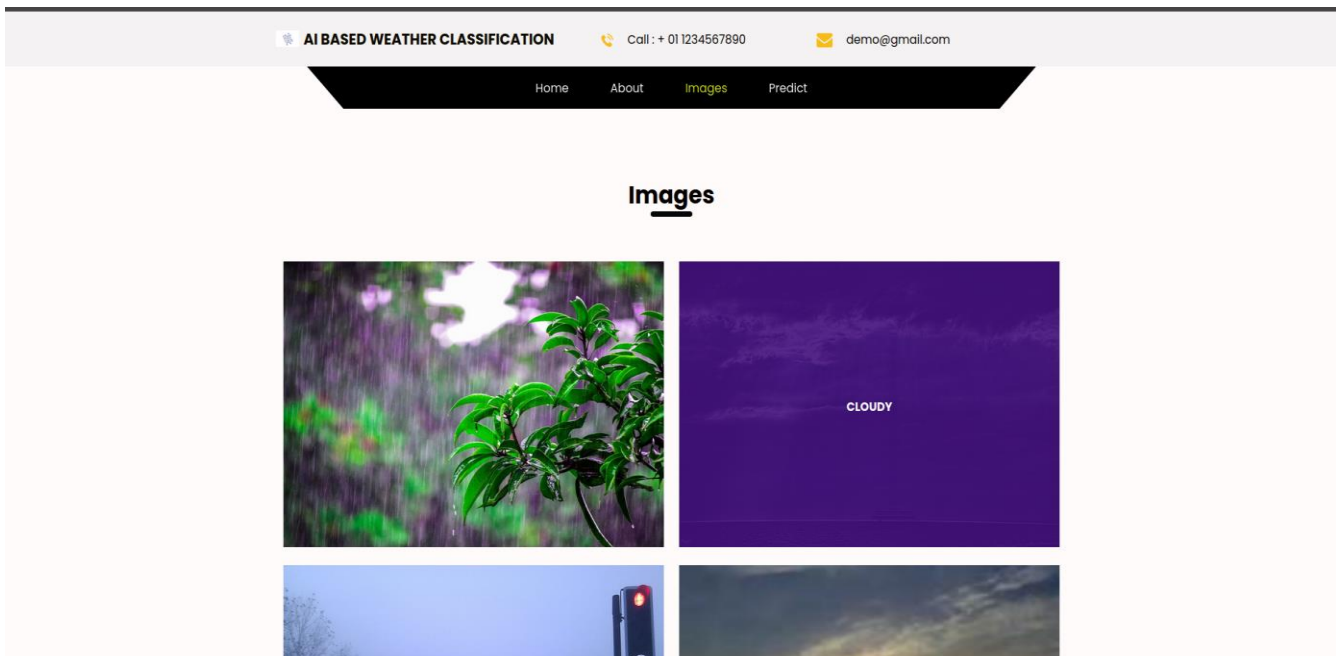
The “Home” Tab and “about” Tab is shown in the figure



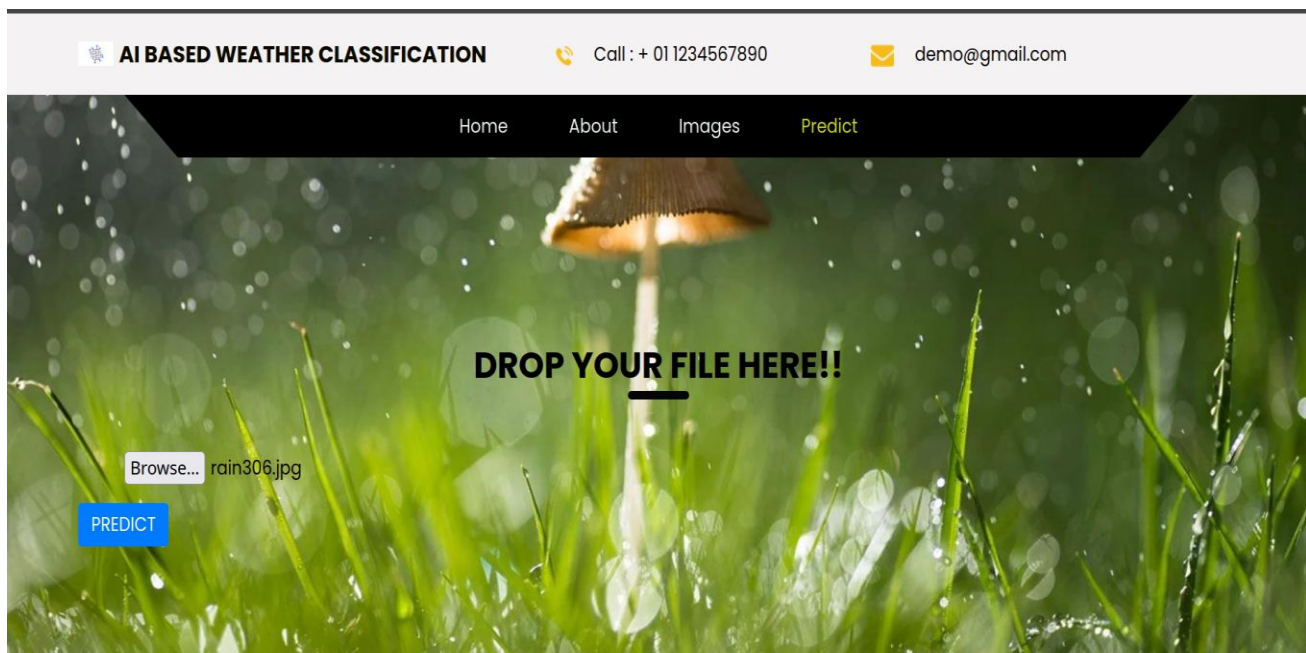




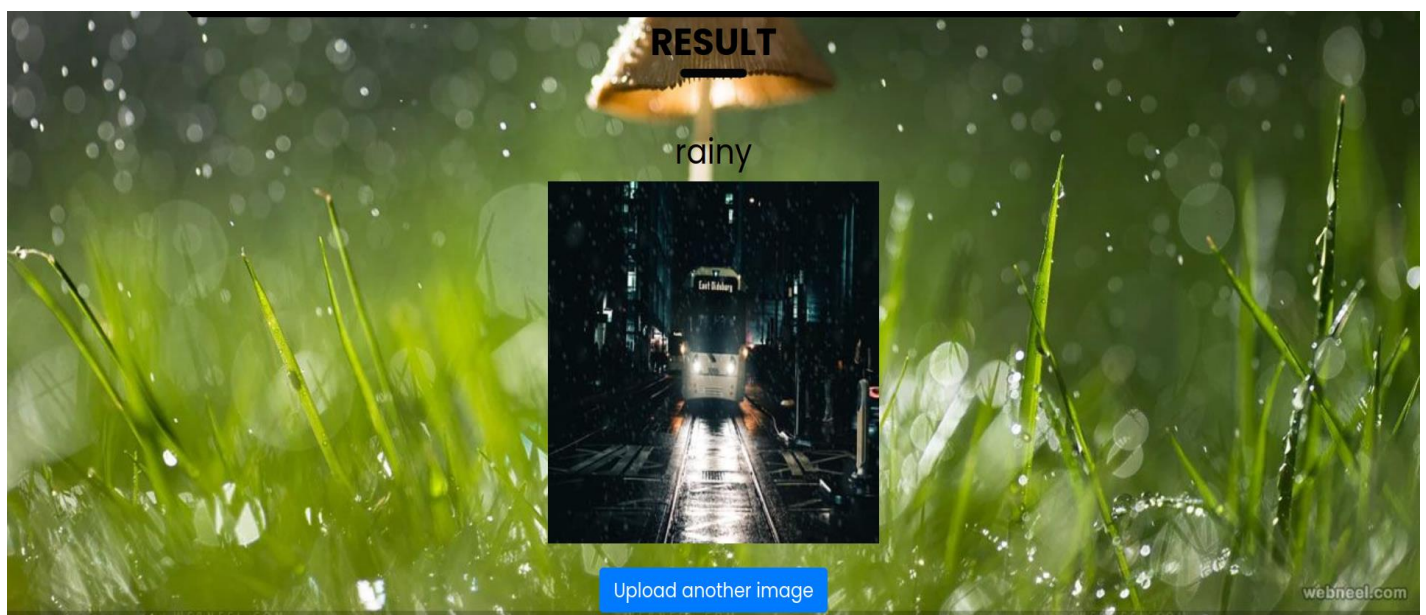
The types of Weather button in “About” Tab will redirect you to the “Images” tab. The “Images” tab is shown in the following figure.



Then the “Predict” tab allows the users to upload the images from their local device and once the users click on the predict button, they can see the result page.



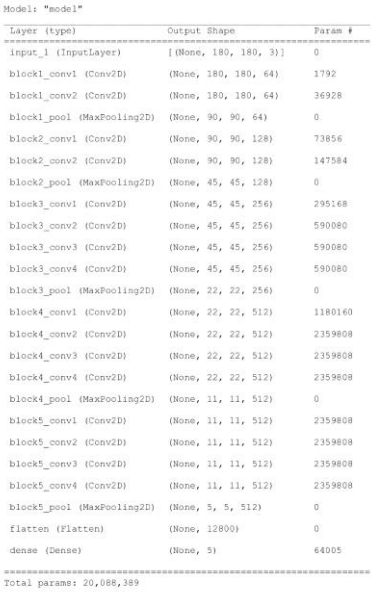
Then the result tab appears as follows,

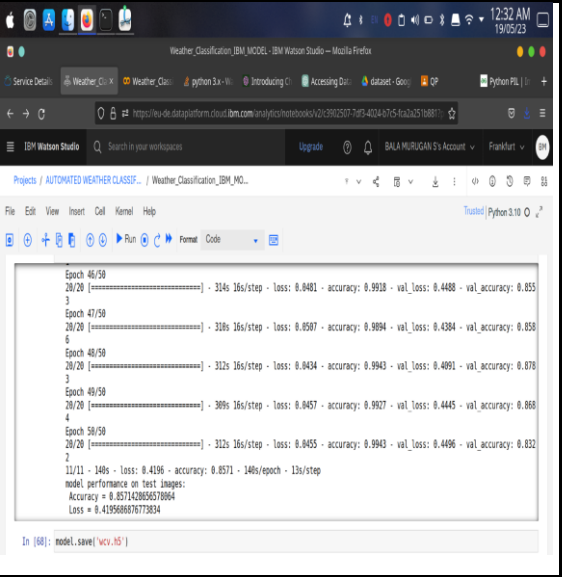




## 7. RESULTS

### 7.1 PERFORMANCE TESTING

S.No.	Parameter	Values	Screenshot
1.	Model Summary	<p>Total params: 20,088,389  Trainable params: 64,005  Non-trainable params: 20,024,384</p> <p>The model has:</p> <p>16 Convolutional layers, 5 Pooling layers and 1 Dense layer with one Input layer</p>	 <pre> Model: "model" Layer (type)                 Output Shape              Param # ----- input_1 (InputLayer)         [None, 180, 180, 3]      0 block1_conv1 (Conv2D)        (None, 180, 180, 64)     1792 block1_conv2 (Conv2D)        (None, 180, 180, 64)     36928 block1_pool (MaxPooling2D)   (None, 90, 90, 64)       0 block2_conv1 (Conv2D)        (None, 90, 90, 128)      73856 block2_conv2 (Conv2D)        (None, 90, 90, 128)      147584 block2_pool (MaxPooling2D)   (None, 45, 45, 128)      0 block3_conv1 (Conv2D)        (None, 45, 45, 256)      295168 block3_conv2 (Conv2D)        (None, 45, 45, 256)      590080 block3_conv3 (Conv2D)        (None, 45, 45, 256)      590080 block3_conv4 (Conv2D)        (None, 45, 45, 256)      590080 block3_pool (MaxPooling2D)   (None, 22, 22, 256)      0 block4_conv1 (Conv2D)        (None, 22, 22, 512)      1180160 block4_conv2 (Conv2D)        (None, 22, 22, 512)      2359808 block4_conv3 (Conv2D)        (None, 22, 22, 512)      2359808 block4_conv4 (Conv2D)        (None, 22, 22, 512)      2359808 block4_pool (MaxPooling2D)   (None, 11, 11, 512)      0 block5_conv1 (Conv2D)        (None, 11, 11, 512)      2359808 block5_conv2 (Conv2D)        (None, 11, 11, 512)      2359808 block5_conv3 (Conv2D)        (None, 11, 11, 512)      2359808 block5_conv4 (Conv2D)        (None, 11, 11, 512)      2359808 block5_pool (MaxPooling2D)   (None, 5, 5, 512)        0 flatten (Flatten)            (None, 12800)             0 dense (Dense)                 (None, 5)                 64005 ----- Total params: 20,088,389 </pre>

2.	Accuracy	Training Accuracy – 99.43%  Validation Accuracy –85.71%	
----	----------	---	--

8. CONCLUSION

In conclusion, this project successfully developed a weather classification system using deep learning techniques. By leveraging deep neural networks and utilizing IBM Cloud resources, we were able to enhance the accuracy and efficiency of weather categorization. Through extensive training and deployment, our models achieved an impressive accuracy of 85.71%. The integration of IBM Watson Machine Learning and IBM Object Storage facilitated seamless model training, deployment, and data management. The results obtained from this project highlight the potential impact of deep learning in weather classification, Thus this project will be a useful tool for meteorologists and weather forecasters to cater their needs of knowing about the weather from time to time.

9. APPENDIX

9.1 GITHUB AND PROJECT DEMO VIDEO LINKS:

Github link: [Click here](#)  
Project Video Demo Link: [Click Here](#)