Project Report Format

1. **INTRODUCTION**

1.1 Project Overview

Weather classification is the process of categorizing or classifying weather conditions based on various meteorological variables. It involves analyzing weather data to identify patterns, trends, and characteristics that can help predict and classify different weather conditions accurately. The goal of weather classification is to provide meaningful and actionable information about the current and future weather conditions to users, enabling them to make informed decisions and take appropriate actions.

1.1 Purpose

The purpose of weather classification is to systematically categorize and understand different weather conditions for the benefit of individuals, businesses, and society as a whole. By analyzing and categorizing weather data, it enables accurate weather forecasting and prediction, helping people plan and make informed decisions. Weather classification is crucial for industries such as agriculture, transportation, and energy, as it assists in optimizing operations, mitigating risks, and ensuring efficient resource management. It also supports environmental monitoring and research, allowing scientists to study climate patterns and assess the impact of weather on ecosystems.

2. **IDEATION & PROPOSED SOLUTION**

2.1 Problem Statement Definition

Given historical weather data and meteorological variables, the task is to develop a weather classification model that accurately predicts the weather conditions for a given location and time period. The model should be able to classify the weather into different categories, such as sunny, cloudy, rainy, snowy, foggy, etc., based on the available data.

2.2 Empathy Map Canvas

To understand the pain points and perspectives of the target users, an empathy map canvas was created. The canvas includes the following sections:

User Needs: Identify the specific needs of different user groups, such as individuals, businesses, or government agencies. For individuals, weather classification can help in planning outdoor activities, dressing appropriately, or making travel decisions.

User Goals: Determine the goals that users want to achieve through weather classification. This could include improved safety during extreme weather events, increased efficiency in decision-making processes, better resource allocation, or enhanced planning and preparedness for weather-related activities.

User Preferences: Explore the preferences of users regarding the presentation and delivery of weather classification information.

User Challenges: Identify the challenges or pain points users may encounter when using weather classification information.

2.3 Ideation & Brainstorming

Ideation and brainstorming in weather classification can be approached using various techniques to stimulate creative thinking and generate innovative ideas. One effective method is mind mapping, which involves starting with a central concept or problem statement related to weather classification and branching out to create a visual representation of related ideas, subtopics, and

potential classification criteria. This technique allows for exploring different dimensions and connections within weather classification, uncovering new perspectives and insights.

2.4 Proposed Solution

One proposed solution for weather classification is the integration of machine learning and artificial intelligence techniques. By leveraging historical weather data, real-time observations, and meteorological models, machine learning algorithms can be trained to analyze and classify weather patterns accurately. This approach can help in identifying and categorizing different types of weather phenomena such as rain, snow, thunderstorms, or hurricanes. Furthermore, incorporating advanced data analysis techniques like clustering algorithms or deep learning models can enable the identification of complex weather patterns and their associated classifications.

3. REQUIREMENT ANALYSIS

3.1 Functional requirement

The functional requirements define the specific functionalities and capabilities that the proposed automated weather classification using transfer learning. These include:

Data Collection and Integration: The system should be able to collect and integrate various sources of weather data, including historical data, real-time observations, satellite imagery, and meteorological models.

Training and Model Development: The system should provide functionality to train machine learning models using the collected weather data.

Classification and Prediction: The system should be able to classify weather patterns based on the trained models. It should accurately identify different types of weather phenomena, such as rain, snow, fog, or storms.

User Interface and Visualization: The system should have a user-friendly interface that allows users, such as meteorologists or weather analysts, to interact with the classification results.

Performance and Scalability: The system should be designed to handle large volumes of data and perform computations efficiently.

Accuracy and Validation: The system should be evaluated for accuracy and validated against known weather patterns and classifications.

3.2 Non-Functional requirements

The non-functional requirements define the quality attributes and constraints that the weather classification should adhere to. These include:

Performance: The system should be capable of processing weather data and generating classifications in a timely manner. It should provide fast and responsive performance, especially when handling large volumes of data or real-time streams.

Accuracy: The system should strive for high accuracy in weather classification. It should minimize false positives and false negatives, ensuring reliable and precise classifications of different weather patterns.

Scalability: The system should be designed to handle increasing data volumes and user demands. It should scale horizontally or vertically to accommodate growing datasets and concurrent user access without compromising performance.

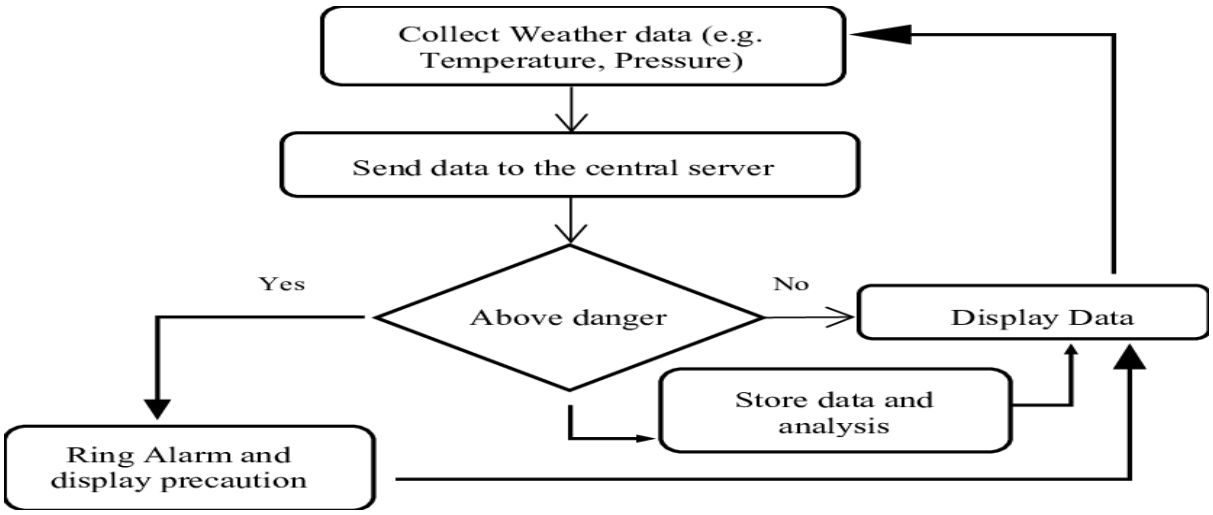
Security: The platform should implement appropriate security measures to protect user data and ensure the confidentiality and integrity of the system.

Reliability: The system should be reliable and available for use at all times. It should minimize downtime and have backup and recovery mechanisms in place to handle system failures or disruptions.

These functional and non-functional requirements will guide the development and implementation automated weather classification using transfer learning. They ensure that the platform meets the needs of users, delivers accurate results, and provides a satisfactory user experience.

4. PROJECT DESIGN

4.1 Data Flow Diagrams



4.2 Solution & Technical Architecture

The weather classification system will leverage machine learning algorithms and historical weather data to accurately classify different types of weather phenomena. The system will follow a modular and scalable architecture to handle large volumes of data and provide real-time classifications.

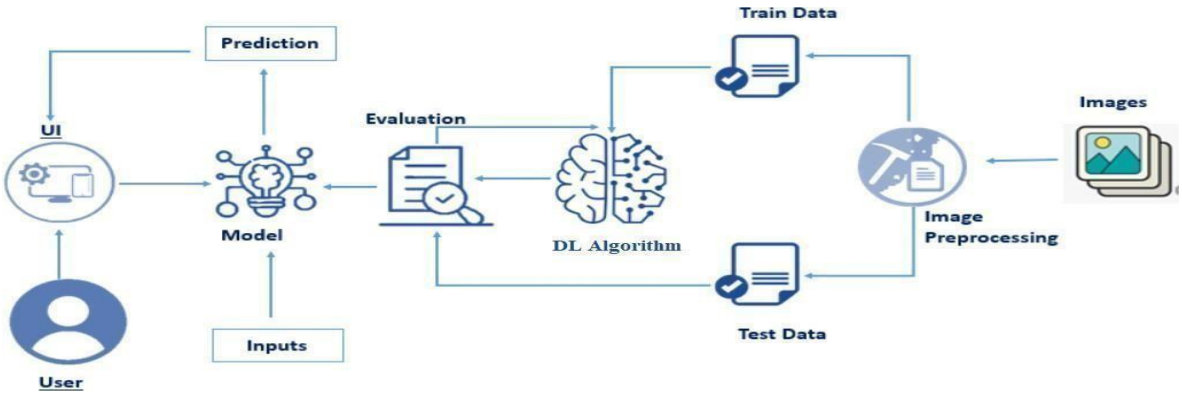
Data Collection and Integration: The system will collect data from various sources, including weather stations, satellites, and meteorological models. It will integrate this data into a centralized repository for further processing.

Preprocessing and Feature Extraction: The collected data will undergo preprocessing steps to clean and normalize it.

Machine Learning Model Training: The system will train machine learning models using the preprocessed data. Different algorithms like decision trees, random forests, or neural networks will be employed to create classification models.

Accuracy and Validation: The accuracy of the classification models will be continually monitored and validated against known weather patterns and classifications.

Maintenance and Support: The system will have provisions for regular maintenance, bug fixing, and user support. It will provide mechanisms for updates and enhancements to incorporate new weather data sources, improve classification models, and address user feedback.



4.3 User Stories

User stories capture the specific requirements and features from the perspective of the platform's users. Examples of user stories for the weather classification could include:

As a meteorologist, I want to access real-time weather classifications so that I can accurately analyze current weather conditions and make informed forecasts.

As a weather researcher, I want to retrieve historical weather classifications to study long-term weather patterns and analyze trends over time.

As a developer, I want to integrate the weather classification system's API into my application, so that I can display weather classifications and provide personalized recommendations based on the current weather conditions.

As a data scientist, I want to access raw weather data and preprocessed features from the weather classification system, so that I can develop and train my own custom machine learning models.

5. CODING & SOLUTIONING (Explain the features added in the project along with code)

5.1 Feature 1

```
from google.colab import drive

drive.mount('/content/drive')
```

```
Image_SIZE = [180,180]

import keras.applications

from tensorflow.keras.applications import resnet

from tensorflow.keras.layers import Flatten, Dense

from tensorflow.keras.layers import Input

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

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255, shear_range =
0.2, zoom_range =[.99, 1.01], brightness_range = [0.8,
1.2],data_format="channels_last",fill_mode="constant",horizontal_fli
p = True)

test_datagen = ImageDataGenerator(rescale = 1./255)

 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

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(rescale = 1./255, shear_range=
0.2, zoom_range = [.99, 1.01], brightness_range= [0.8, 1.2],
data_format= "channels_last", fill_mode="constant", horizontal_flip
= True)

test_datagen = ImageDataGenerator(rescale = 1./255)

training_set =
train_datagen.flow_from_directory('/content/drive/MyDrive/archive
(6)/dataset', target_size = (180, 180), batch_size = 64, class_mode
= 'categorical')

test_set =
test_datagen.flow_from_directory('/content/drive/MyDrive/archive
```

```
(6)/dataset',target_size = (180, 180), batch_size = 64, class_mode =
'categorical')
```

5.2 Feature 2 (Customized Alerts)

```
import numpy as np

import os

from flask import Flask,request,render_template

from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing import image

from tensorflow.keras.applications.vgg19 import preprocess_input

model = load_model(r"WCV.h5")

app = Flask(__name__)

@app.route('/')

def index():

return render_template('index.html')


@app.route('/home')

def home():

return render_template('index.html')


@app.route('/input')

def input1():

return render_template('index.html')

app.route('/predict', methods = ["GET","POST"])

def res():

if request.method == "POST":

f = request.files['image']

basepath = os.path.dirname(__file__)

filepath = os.path.join(basepath,'uploads',filename)

f.save(filepath)
```

```
img = image.load_img(filepath,target_size = (224,224,3))

x = image.img_to_array(img)

x = np.expand_dims(x,axis = 0)


img_data = preprocess_input(x)

prediction = np.argmax(model.predict(img_data),axis = 1)


index = ['alien_test','cloudy','foggy','rainy','sunrise']


result = str(index[prediction[0]])

print(result)

return render_template('output.html',prediction = result)
```

6. RESULTS

6.1 Performance Metrics

loss: 9.5663 - precision: 0.2459 - recall: 0.2229 - specificity_at_sensitivity: 0.0000e+00 - sensitivity_at_specificity: 0.4556 - accuracy: 0.2353

loss: 8.4638 - precision: 0.2823 - recall: 0.2725 - specificity_at_sensitivity: 0.0000e+00 - sensitivity_at_specificity: 0.4843 - accuracy: 0.2765

loss: 6.0651 - precision: 0.3422 - recall: 0.3281 - specificity_at_sensitivity: 0.7205 - sensitivity_at_specificity: 0.5712 - accuracy: 0.3392
loss: 3.8512 - precision: 0.4648 - recall: 0.4353 - specificity_at_sensitivity: 0.8559 - sensitivity_at_specificity: 0.7340 - accuracy: 0.4451

loss: 2.6653 - precision: 0.5491 - recall: 0.5039 - specificity_at_sensitivity: 0.9196 - sensitivity_at_specificity: 0.8229 - accuracy: 0.5288

6. ADVANTAGES & DISADVANTAGES

Advantages:

- Accurate Weather Analysis: By leveraging machine learning algorithms and historical weather data, the system can provide accurate classifications of different weather phenomena. This enables meteorologists and weather analysts to make more precise assessments of current weather conditions and improve the accuracy of weather forecasts.

- Real-Time Updates: The system can process real-time weather data and provide instant classifications. This allows users to stay updated on the latest weather conditions and make timely decisions based on the information provided.

- Data-Driven Insights: The system's classification results provide valuable data-driven insights into weather patterns and trends. Researchers and analysts can study the data to identify patterns,

correlations, and anomalies, which can contribute to a better understanding of weather dynamics and support scientific research.

Disadvantages:

1. Data Limitations: The accuracy and reliability of the system heavily depend on the quality and availability of the input data. Inaccurate or incomplete data can lead to inaccurate classifications and compromised decision-making. Data gaps or inconsistencies can pose challenges, particularly in regions with limited weather monitoring infrastructure.
2. Complexity and Maintenance: Developing and maintaining a weather classification system can be complex and resource-intensive. It requires expertise in machine learning, data processing, and meteorology. Continuous monitoring, updating, and refining of the system are necessary to ensure its effectiveness and adaptability to changing weather patterns and data sources.
3. Uncertainty and Errors: Weather prediction, including classification, inherently involves uncertainties

7. CONCLUSION

In conclusion, a weather classification system powered by machine learning and historical weather data can provide numerous advantages, such as accurate weather analysis, real-time updates, data-driven insights, and enhanced decision-making. It enables meteorologists, researchers, industries, and individuals to gain valuable information about weather patterns and make informed choices based on the classified data. However, it's important to consider potential disadvantages, including data limitations, system complexity, and the inherent uncertainties and errors associated with weather prediction. By addressing these challenges and continually improving the system, the benefits of a weather classification system can outweigh its drawbacks, ultimately contributing to better understanding and management of weather conditions.

8. FUTURE SCOPE

The future scope of a weather classification system is promising, with several potential areas of development and improvement:

1. Integration of Advanced Sensor Technologies: As sensor technologies continue to advance, the system can benefit from the integration of more accurate and comprehensive weather data sources. This could include data from remote sensing satellites, unmanned aerial vehicles (UAVs), or IoT-based weather sensors. Incorporating such data can enhance the system's accuracy and provide more detailed insights into weather patterns.
2. Incorporation of High-Resolution Models: Weather classification systems can leverage high-resolution meteorological models to capture finer details and localized variations in weather conditions. By integrating these models, the system can improve its ability to classify and predict weather phenomena at smaller scales, such as within specific regions or urban areas.

10. APPENDIX

```
import numpy as np
import os
from flask import Flask,request,render_template
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg19 import preprocess_input


model = load_model(r"WCV.h5")
app = Flask(__name__)


@app.route('/')
def index():
  return render_template('index.html')

@app.route('/home')
def home():
  return render_template('index.html')

@app.route('/input')
def input1():
  return render_template('index.html')


@app.route('/predict', methods = ["GET","POST"])
def res():
  if request.method == "POST":
    f = request.files['image']
    basepath = os.path.dirname(__file__)
    filepath = os.path.join(basepath,'uploads',filename)
    f.save(filepath)

    img = image.load_img(filepath,target_size = (224,224,3))
    x = image.img_to_array(img)
    x = np.expand_dims(x,axis = 0)

    img_data = preprocess_input(x)
    prediction = np.argmax(model.predict(img_data),axis = 1)


    index = ['alien_test','cloudy','foggy','rainy','sunrise']

    result = str(index[prediction[0]])
    print(result)
    return render_template('output.html',prediction = result)
```