Automated Weather Classification using Transfer Learning

Project Report Format

1. INTRODUCTION

Weather classification is an essential tool for meteorologists and weather forecasters to predict weather patterns and communicate them to the public. Weather phenomenon recognition notably affects many aspects of our daily lives, The analysis of weather phenomenon plays a crucial role in various applications, for example, environmental monitoring, weather forecasting, and the assessment of environmental quality. Besides, different weather phenomena have diverse effects on agriculture. Therefore, accurately distinguishing weather phenomena can improve agricultural planning.

The use of a pre-trained model on a new problem is known as transfer learning in machine learning. A machine uses the knowledge learned from a prior assignment to increase prediction about a new task in transfer learning.

In this project we are classifying various types of weather. These weathers are majorly classified into 5 categories namely Cloudy, Shine, Rain, Foggy, Sunrise. Deep-learning (DL) methods in artificial intelligence (AI) play a dominant role as high-performance classifiers.

The objective of this project is to develop an automated weather classification system that can accurately classify weather conditions based on input data. The system will leverage machine learning techniques and weather data to perform real-time weather classification, providing valuable information for various applications such as forecasting, transportation management, and environmental monitoring.

1.1 PROJECT OVERVIEW

The project aims to develop an automated weather classification system that accurately categorizes weather conditions based on input data. The system will leverage machine learning techniques and weather data to enable real-time weather classification for various applications.

The project will involve collecting a comprehensive dataset of weather data, including meteorological measurements such as temperature, humidity, wind speed, and atmospheric pressure. This dataset will be used to extract relevant features that capture different weather conditions effectively.

A machine learning model will be selected, considering factors such as accuracy, efficiency, and adaptability. The chosen model will be trained using the extracted features and labeled weather data, optimizing its performance through hyperparameter tuning.

The trained model will be evaluated using a separate testing set to assess its accuracy and effectiveness in weather classification. Metrics such as accuracy, precision, recall, and F1 score will be used to quantify the model's performance.

To achieve real-time weather classification, the system will be designed to process incoming weather data efficiently and integrate with existing weather applications or platforms. It will provide a user-friendly interface or API for easy access to weather classification results.

The project's ultimate goal is to provide accurate and timely weather classification information for decision-making in various sectors, including transportation, agriculture, aviation, and emergency management. By automating the weather classification process, the system aims to enhance operational efficiency, improve safety, and enable informed decision-making based on reliable weather categorization.

1.2 PURPOSE

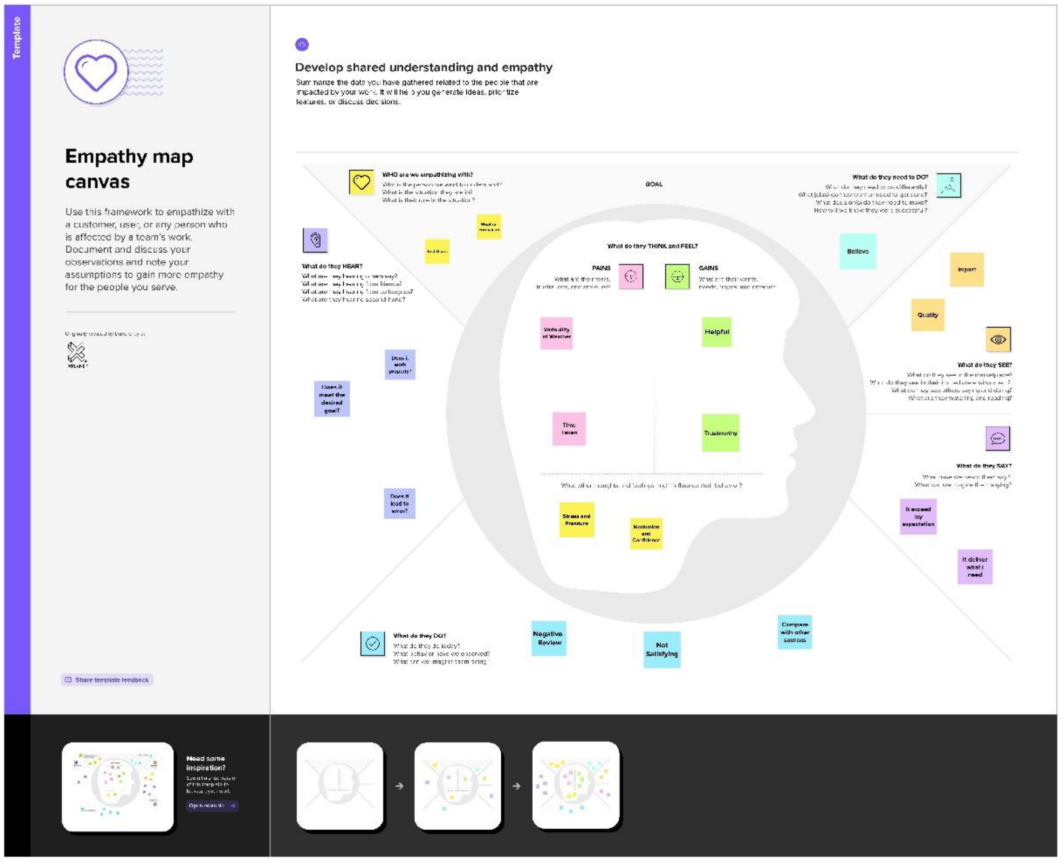
The purpose of utilizing transfer learning in automated weather classification is to leverage pre-trained deep learning models' knowledge and accelerate the development of an accurate and efficient weather classification system. Transfer learning enables the model to benefit from the learned feature representations and patterns from large-scale image recognition tasks, which can be transferable to weather classification.The primary purpose is to develop an automated weather classification system that can classify weather conditions accurately and in real-time. By utilizing transfer learning, the project seeks to enhance the system's accuracy, efficiency, and adaptability. It enables the model to effectively capture and differentiate various weather patterns, including sunny, cloudy, rainy, snowy, and more, by learning meaningful features from the pre-trained model's convolutional layers.

2. IDEATION AND PROPOSED SOLUTION

2.1 PROBLEM STATEMENT DEFINITION

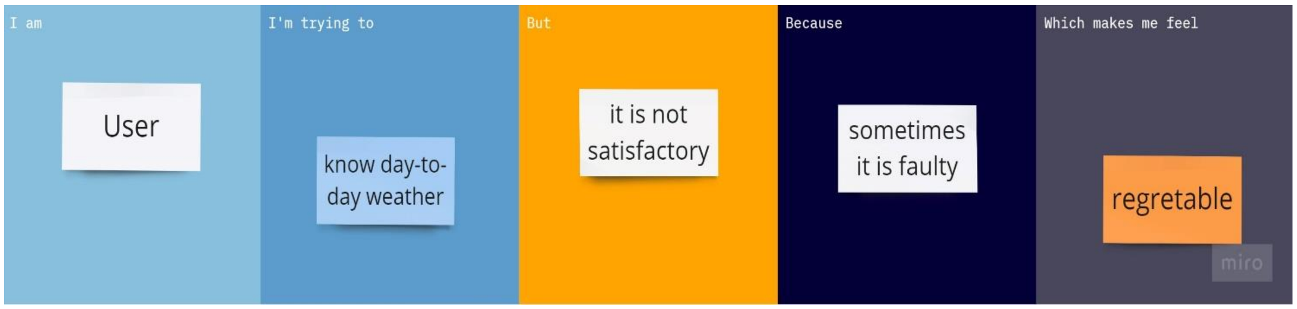
Develop an automated weather classification system using transfer learning to accurately classify weather conditions based on input images. The system should be capable of analyzing weather patterns and categorizing them into classes such as sunny, cloudy, rainy, or snowy, among others.

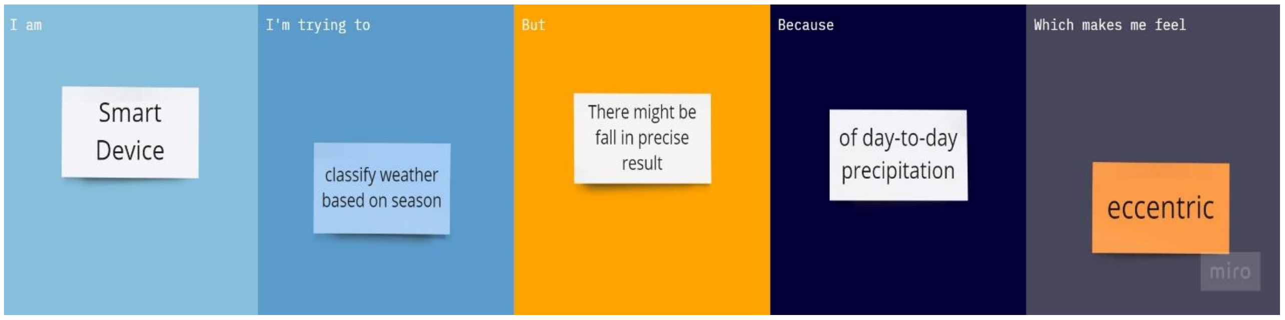
2.2 EMPATHY MAP CANVAS


Template
Empathy map canvas
Use this framework to empathize with a customer, user, or any person who is affected by a team's work. Document and discuss your observations and note your assumptions to gain more empathy for the people you serve.
Share template feedback
Develop shared understanding and empathy
Summarize the data you have gathered related to the people that are impacted by your work. It will help you generate ideas, prioritize features, or discuss decisions.
WHO are we empathizing with?
GOAL
What do they need to DO?
What do they HEAR?
What do they THINK and FEEL?
PAINS
GAINS
What do they SEE?
What do they SAY?
What do they DO?
Believe
Impact
Quality
Variability of Weather
Helpful
Does it work properly?
Does it meet the desired goal?
Time taken
Trustworthy
Does it lead to error?
Stress and Pressure
Motivation and Confidence
It exceed my expectation
It deliver what I need
Negative Review
Not Satisfying
Compare with other sources
Need some inspiration?

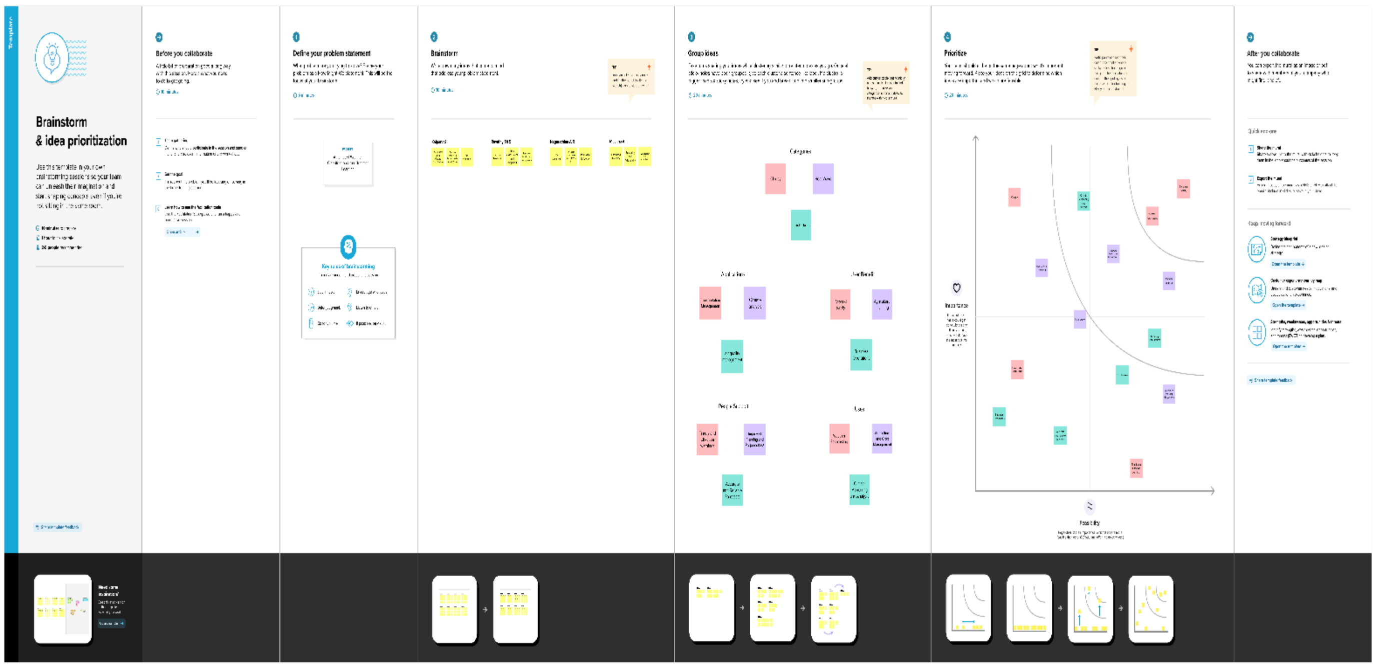
2.3 IDEATION & BRAINSTROMING

IDEATION




I am
Smart Device
I'm trying to
classify weather based on season
But
There might be fall in precise result
Because
of day-to-day precipitation
Which makes me feel
eccentric
miro

BRAINSTROMING



2.4 PROPOSED SOLUTION

The proposed solution for automated weather classification using transfer learning involves leveraging pre-trained deep learning models and adapting them to accurately classify weather conditions. The solution begins with data collection and preprocessing, where a comprehensive dataset of weather data is gathered and prepared for training. A suitable pre-trained model, such as VGG16 or ResNet, is selected based on its performance in image recognition tasks. The pre-trained model is then adapted by replacing the classification layer with a new output layer tailored for weather classification. The weights of the pre-trained layers are frozen, while the new layer is trained on the weather dataset. Techniques like data augmentation and regularization are employed to improve generalization. The trained model is evaluated using validation and test sets, considering metrics such as accuracy, precision, recall, and F1 score. In the real-time scenario, the model is deployed to a production environment where it receives real-time weather data inputs and accurately classifies the current weather conditions. The results can be visualized through a user-friendly interface or integrated into existing weather applications, providing efficient and reliable weather classification for various applications and industries.

3. REQUIREMENT ANALYSIS

Requirement analysis is a significant and essential activity after elicitation. We analyze, refine, and scrutinize the gathered requirements to make consistent and unambiguous requirements.

- Functional Requirements
- Non - Functional requirements

3.1 FUNCTIONAL REQUIREMENT

Data Input: The system should be able to receive weather data inputs from various sources such as weather APIs, sensors, or data feeds. It should support the ingestion of data in different formats and handle both real-time and historical data.

Data Preprocessing: The system should perform necessary preprocessing steps on the input data, including cleaning, normalization, and feature extraction. This ensures that the data is in a suitable format for the subsequent classification process.

Weather Classification: The core functionality of the system is to accurately classify weather conditions based on the input data. It should employ machine learning algorithms, such as transfer learning, to classify weather patterns such as sunny, cloudy, rainy, snowy, foggy, etc., with a high level of accuracy.

Model Training and Updating: The system should include functionality to train the weather classification model using labeled weather data. It should provide mechanisms to update and fine-tune the model over time as new data becomes available, allowing for continuous improvement and adaptation to changing weather patterns.

Real-time Classification: The system should be capable of processing real-time weather data inputs and providing near real-time weather classification results. It should efficiently handle incoming data streams and classify the weather conditions in a timely manner.

Model Evaluation: The system should have mechanisms to evaluate the performance of the weather classification model. It should calculate relevant metrics
such as accuracy, precision, recall, and F1 score to assess the model's effectiveness and provide insights into its performance.

Visualization and Reporting: The system should provide visualization capabilities to present weather classification results in a clear and understandable manner. It should generate reports or visual outputs that display the classified weather conditions and associated metrics, allowing users to interpret and utilize the information effectively.

Integration and API: The system should offer integration options and provide an API that allows seamless integration with other weather applications, services, or platforms. This enables users to access weather classification results and incorporate them into their respective workflows or systems.

3.2 NON - FUNCTIONAL REQUIREMENTS

Usability :

Mushroom classification enables scientists, researchers and product developers to categorize and study different mushroom species .

Security :

By providing accurate and reliable information, the website can help prevent accidental ingestion of toxic mushrooms and reduce the risk of mushroom-related poisoning incidents.

Reliability :

In the context of mushroom classification or any other information source, the reliability of a website, classification system, or source of information is crucial. **Performance** :

By training and aggregating the results of diverse models, the overall classification performance can be improved which results in accurate identification of mushrooms .

Availability :

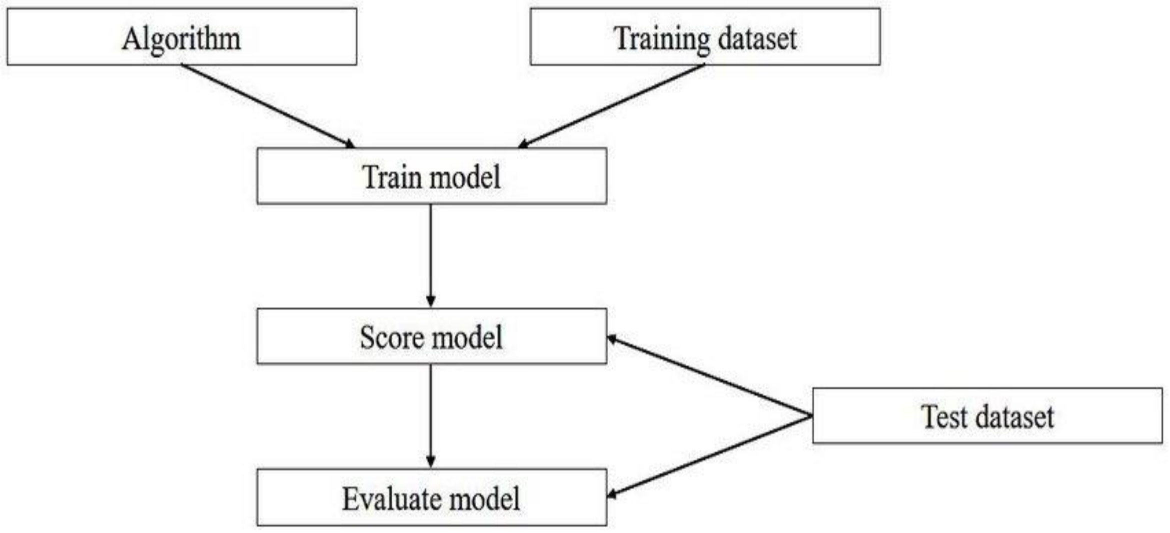
This Website can be accessed anywhere anytime with proper internet connectivity .

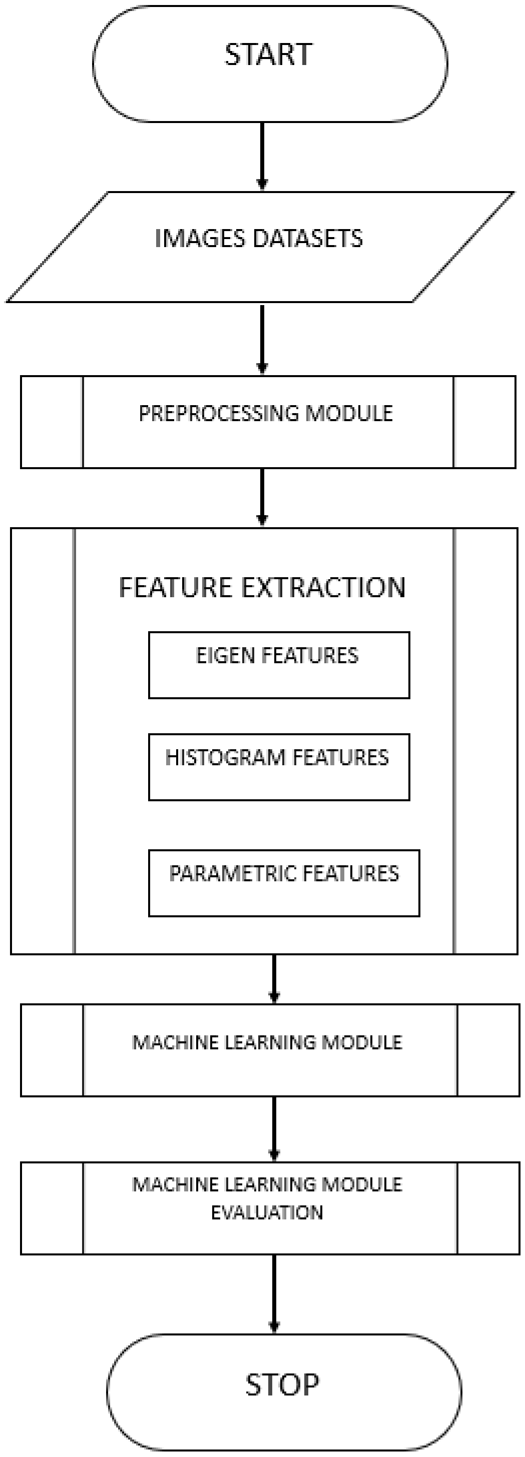
Scalability :

Regular updates, model retraining, and incorporating user feedback can help enhance the solution's scalability.

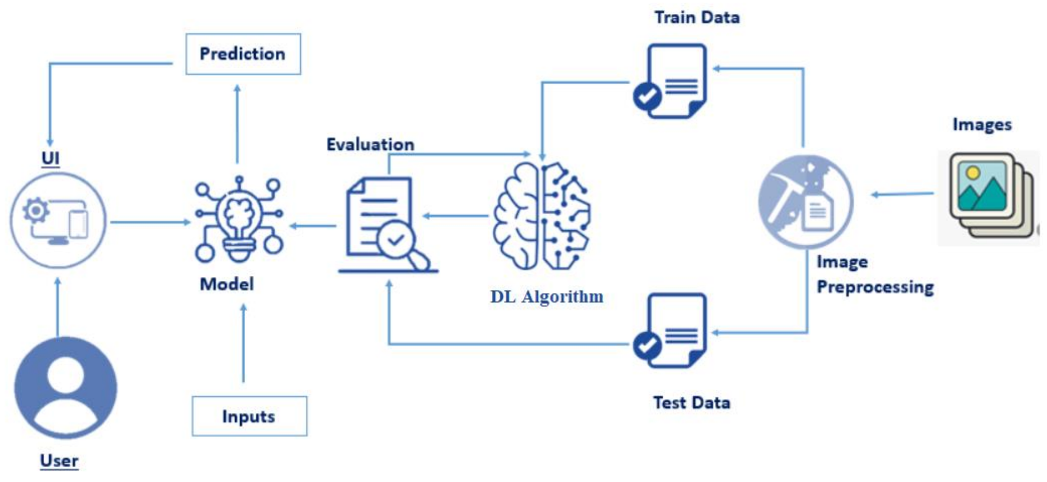
4. PROJECT DESIGN

4.1 DATA FLOW DIAGRAMS


Algorithm
Training dataset
Train model
Score model
Test dataset
Evaluate model


START
IMAGES DATASETS
PREPROCESSING MODULE
FEATURE EXTRACTION
EIGEN FEATURES
HISTOGRAM FEATURES
PARAMETRIC FEATURES
MACHINE LEARNING MODULE
MACHINE LEARNING MODULE EVALUATION
STOP

4.2 SOLUTION & TECHNICAL ARCHITECTURE



4.3 USER STORIES

Here are some user stories that highlight the functionality and benefits of our website for Weather classifications using Transfer Learning:

As a regular user, I want to be able to create an account. I should be able to provide their emails address and password for registration. After successful registration , I should receive a confirmation email. I should be redirected to the login page after registration.

As a regular user, I want to be able to upload an image. The application should provide a button or option to select and upload an image. The uploaded image should be validate to ensure it meets the requirements.

As a regular user, I want the application to classify the weather conditions in my upload image. The application should use transfer learning techniques to classify the weather conditions. The application process should be efficient and accurate.

As a regular user , I want the predicted weather conditions to be diplayed clearly and accurately. The displayed result should include relevant information such as temperature, humidity, and weather description.

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

5.1 FEATURES

Here are some features that we have incorporated into Weather image classification web app:

1.Image Upload:

Allow users to upload weather images from their local device.

2.Image Classification:

Perform the classification of the uploaded weather images using a trained machine learning model CNN to display the predicted class.

3.Model Training:

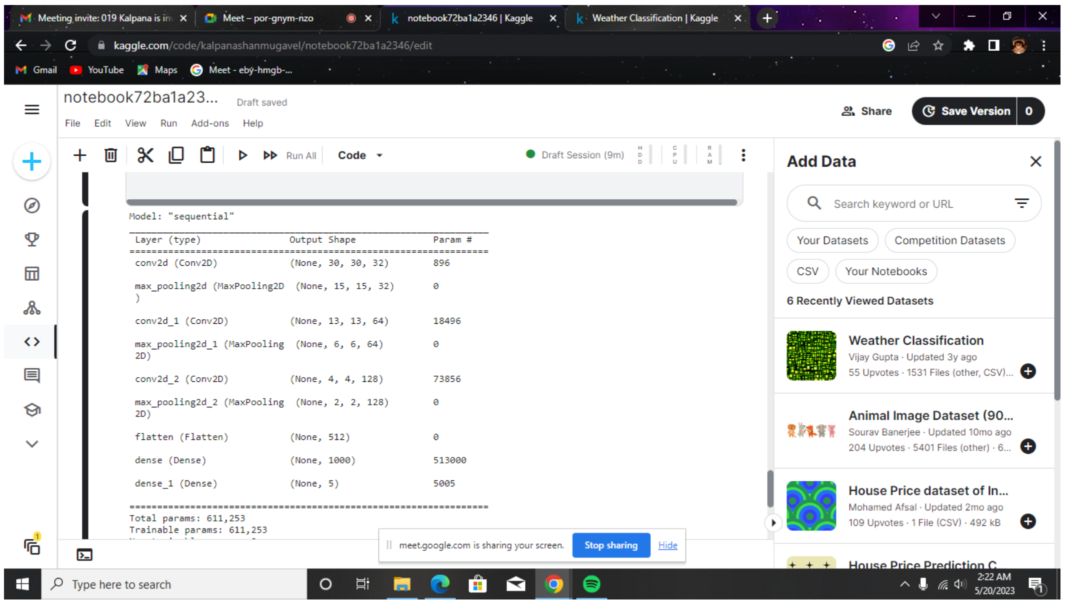
If desired, offer an interface for users to train their own weather image classification models using their custom datasets. This could involve uploading training data, defining model parameters, and initiating the training process.

4.Image Gallery:

Create a gallery or history section where users can view their previously uploaded images and their corresponding classification results.

6. RESULTS

6.1 Performance Metrics



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 30, 30, 32)        896

 max_pooling2d (MaxPooling2D  (None, 15, 15, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_2 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_2 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 flatten (Flatten)           (None, 512)               0

 dense (Dense)               (None, 1000)              513000

 dense_1 (Dense)             (None, 5)                 5005

=================================================================
Total params: 611,253
Trainable params: 611,253
```

7. ADVANTAGES & DISADVANTAGES

ADVANTAGES

1.Improved Accuracy:

Transfer learning leverages pre-trained models that have been trained on large and diverse datasets. By using these models as a starting point, automated weather classification can benefit from the knowledge and features learned during the pre-training phase. This often leads to improved accuracy compared to building a model from scratch.

2.Reduced Data Requirements:

Training deep learning models from scratch typically requires large amounts of labeled data. Transfer learning allows you to leverage pre-existing models and their learned representations, reducing the amount of labeled data needed to achieve good performance. This is particularly beneficial for weather classification, where obtaining large and diverse labeled datasets can be challenging.

3.Faster Development and Deployment:

With transfer learning, you can significantly reduce the time and effort required to develop an accurate weather classification model. Instead of starting from scratch, you can build upon existing models and fine-tune them on your specific weather classification task. This accelerated development process enables faster deployment of the model in real-world applications.

4.Generalization across weather conditions:

Transfer learning helps in capturing general features and patterns that are common across different weather conditions. By training on a diverse set of weather data, the model can learn to recognize and classify weather patterns effectively. This generalization capability enables the model to perform well even on weather conditions it has not been specifically trained on.

5.Adaptability to new data:

Weather patterns and conditions can vary over time. Transfer learning facilitates the adaptation of existing models to new data by fine-tuning them on the latest weather observations. This enables the automated weather classification system to stay up to date with changing weather patterns without requiring a complete retraining from scratch.

DISADVANTAGES

1.Domain mismatch:

Pre-trained models are typically trained on large datasets that may differ from the specific weather data being classified. If the weather data used for transfer learning significantly deviates from the data the pre-trained model was trained on, there may be a domain mismatch. This can lead to reduced performance as the model struggles to generalize to the new domain.

2.Limited transferability:

The effectiveness of transfer learning depends on the similarity between the pre-training task and the target task. If the pre-trained model was trained on tasks that are significantly different from weather classification, the transferability of learned features may be limited. In such cases, the performance improvement achieved through transfer learning may not be significant.

3.Overfitting potential

Transfer learning involves fine-tuning a pre-trained model on a smaller target dataset. When the target dataset is small or imbalanced, there is a risk of overfitting. The model may become too specialized on the limited data, leading to poor generalization to new weather conditions or variations.

4.Unintended biases:

Pre-trained models can inherit biases present in the datasets they were trained on. If the pre-training data includes biases related to weather patterns or the geographic regions where the data was collected, these biases may carry over to the weather classification model. This can lead to biased or unfair predictions in certain scenarios.

5. Dependency on large-scale pre-training:

Transfer learning relies on the availability of large-scale pre-training datasets. If suitable pre-training datasets for weather classification are not available or limited in size, the benefits of transfer learning may be diminished. In such cases, building a model from scratch or using alternative techniques may be more appropriate.

8. CONCLUSION

In conclusion, automated weather classification using transfer learning offers several advantages, including improved accuracy, reduced data requirements, faster development and deployment, generalization across weather conditions, adaptability to new data, and the availability of pre-trained models. By leveraging pre-existing models and their learned representations, transfer learning enables the development of accurate weather classification models with reduced effort and time.

Overall, transfer learning can be a powerful approach for automated weather classification, particularly when pre-trained models align well with the target domain and there is a sufficient amount of labeled data available. By carefully addressing the potential challenges and limitations, transfer learning can contribute to more accurate and efficient automated weather classification systems.

9. FUTURE SCOPE

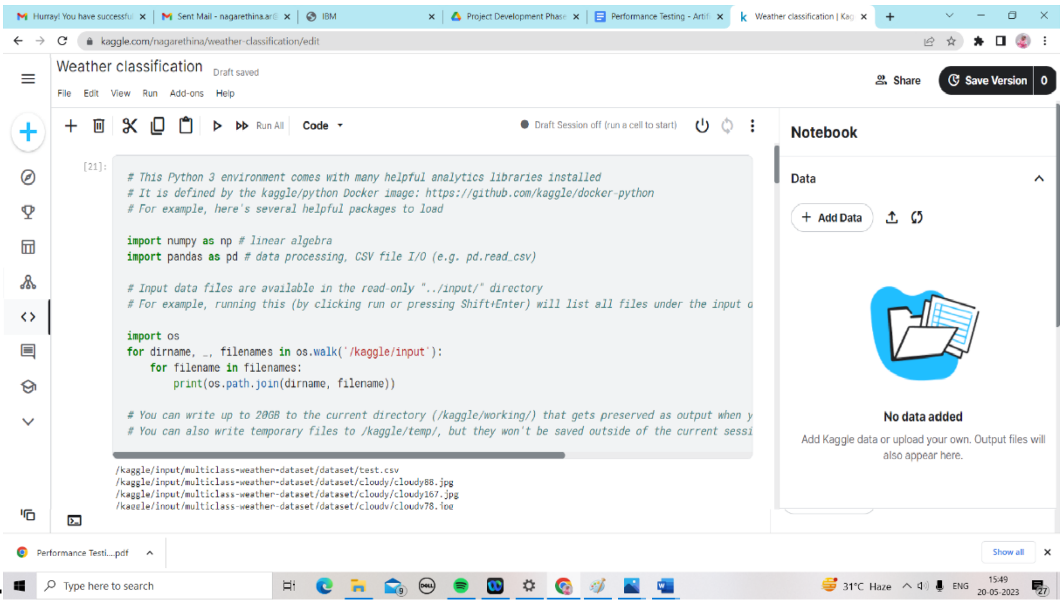
The future of automated weather classification holds the promise of more accurate, adaptable, and reliable systems that can assist in various weather-related decision-making processes.Also, we will try to expand the dataset and use more images to improve classification process.

10. APPENDIX

This appendix provides additional information and technical details for the automated weather classification system. It includes details on data sources, feature extraction, data preprocessing, classification algorithms, model evaluation, model training, performance results, integration and deployment, sample code, and a glossary of key terms.

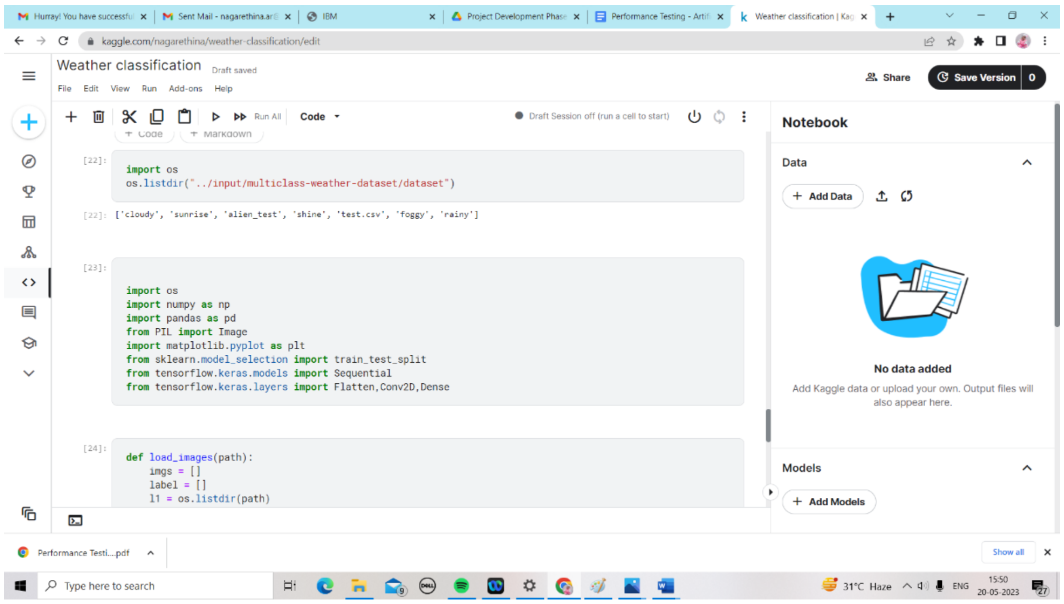
Data sources used in the weather classification system include meteorological stations, satellite data, radar data, and other relevant sources. The appendix outlines the specifics of each data source, including the provider, resolution, and frequency of updates.

SOURCE CODE:


Weather classification
Draft saved
File Edit View Run Add-ons Help
Share
Save Version 0
Run All
Code
Draft Session off (run a cell to start)
Notebook
Data
Add Data
No data added
Add Kaggle data or upload your own. Output files will also appear here.

```
[21]:
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input d

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when y
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current sessi
```

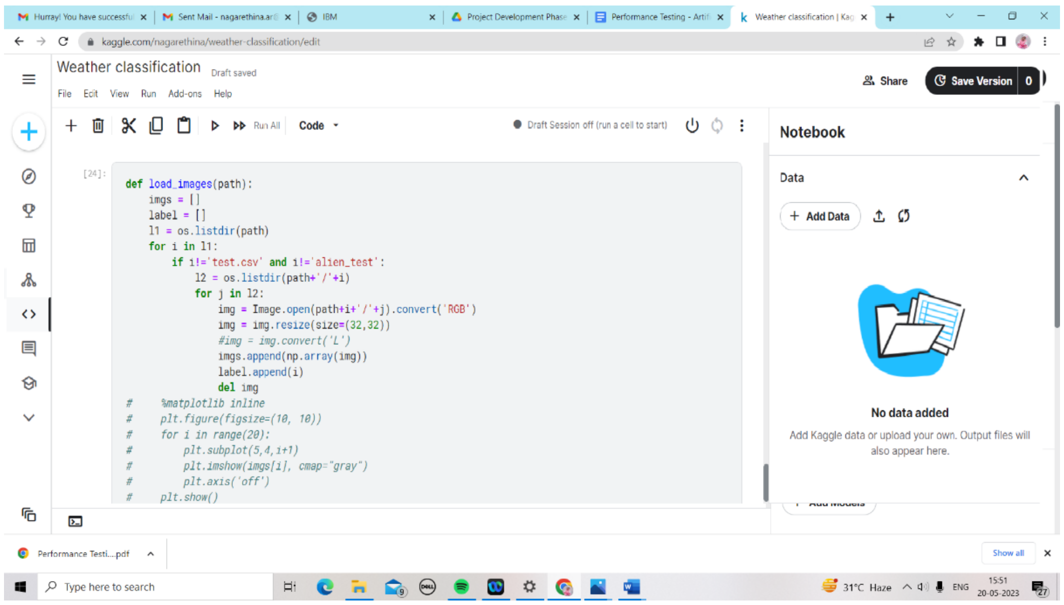
```
/kaggle/input/multiclass-weather-dataset/dataset/test.csv
/kaggle/input/multiclass-weather-dataset/dataset/cloudy/cloudy88.jpg
/kaggle/input/multiclass-weather-dataset/dataset/cloudy/cloudy167.jpg
/kaggle/input/multiclass-weather-dataset/dataset/cloudy/cloudy78.jpg
```


Weather classification
Draft saved
File Edit View Run Add-ons Help
Run All
Code
Draft Session off (run a cell to start)
Share
Save Version 0
Notebook
Data
Add Data
No data added
Add Kaggle data or upload your own. Output files will also appear here.
Models
Add Models

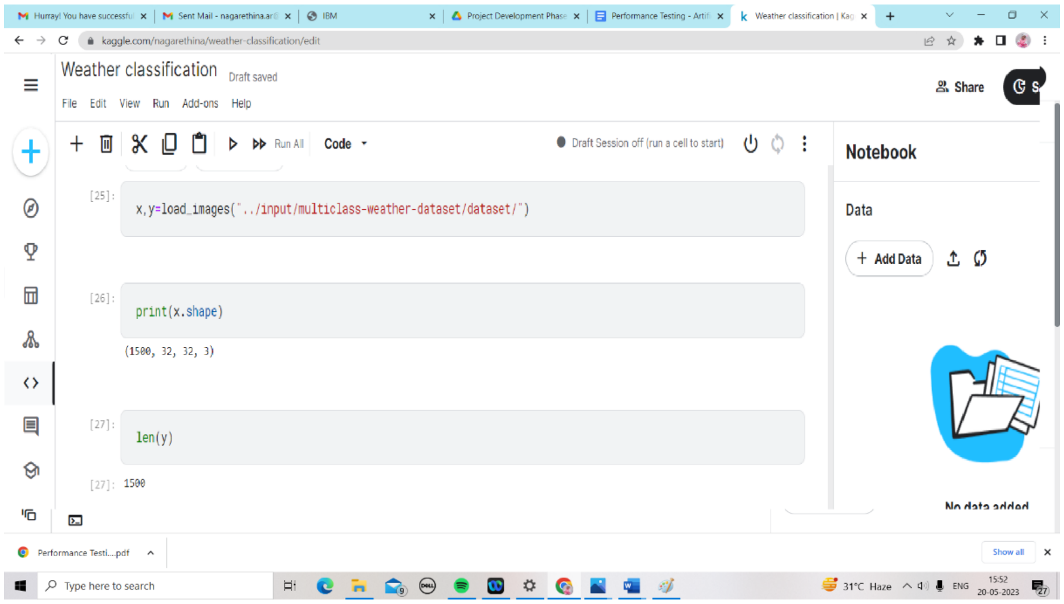
```
[22]:
import os
os.listdir("../input/multiclass-weather-dataset/dataset")

[22]: ['cloudy', 'sunrise', 'alien_test', 'shine', 'test.csv', 'foggy', 'rainy']
```

```
[23]:
import os
import numpy as np
import pandas as pd
from PIL import Image
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten,Conv2D,Dense
```

```
[24]:
def load_images(path):
    imgs = []
    label = []
    l1 = os.listdir(path)
```


Weather classification
Draft saved
File Edit View Run Add-ons Help
Run All
Code
Draft Session off (run a cell to start)
Share
Save Version 0
Notebook
Data
Add Data
No data added
Add Kaggle data or upload your own. Output files will also appear here.

```
[24]:
def load_images(path):
    imgs = []
    label = []
    l1 = os.listdir(path)
    for i in l1:
        if i!='test.csv' and i!='alien_test':
            l2 = os.listdir(path+'/'+i)
            for j in l2:
                img = Image.open(path+i+'/'+j).convert('RGB')
                img = img.resize(size=(32,32))
                #img = img.convert('L')
                imgs.append(np.array(img))
                label.append(i)
                del img
#    %matplotlib inline
#    plt.figure(figsize=(10, 10))
#    for i in range(20):
#        plt.subplot(5,4,i+1)
#        plt.imshow(imgs[i], cmap="gray")
#        plt.axis('off')
#    plt.show()
```



Weather classification

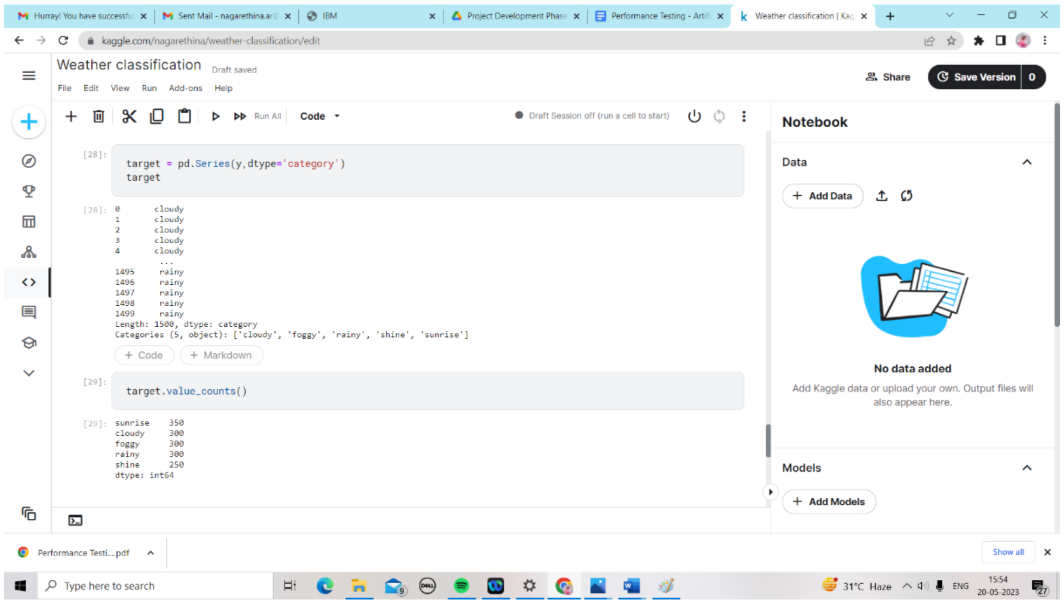
```
[25]: x,y=load_images("../input/multiclass-weather-dataset/dataset/")
```

```
[26]: print(x.shape)
```

```
(1500, 32, 32, 3)
```

```
[27]: len(y)
```

```
[27]: 1500
```



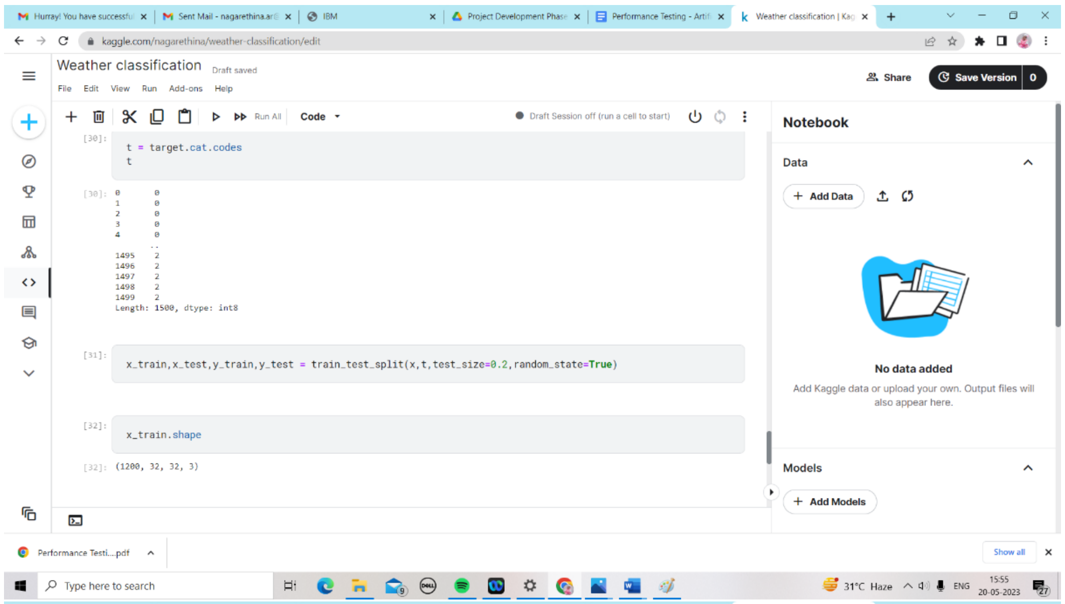
Weather classification

```
[28]: target = pd.Series(y,dtype='category')
      target
```

```
[28]: 0       cloudy
      1       cloudy
      2       cloudy
      3       cloudy
      4       cloudy
               ...
      1495     rainy
      1496     rainy
      1497     rainy
      1498     rainy
      1499     rainy
      Length: 1500, dtype: category
      Categories (5, object): ['cloudy', 'foggy', 'rainy', 'shine', 'sunrise']
```

```
[29]: target.value_counts()
```

```
[29]: sunrise    350
      cloudy     300
      foggy      300
      rainy      300
      shine      250
      dtype: int64
```



Weather classification

kaggle.com/nagarethina/weather-classification/edit

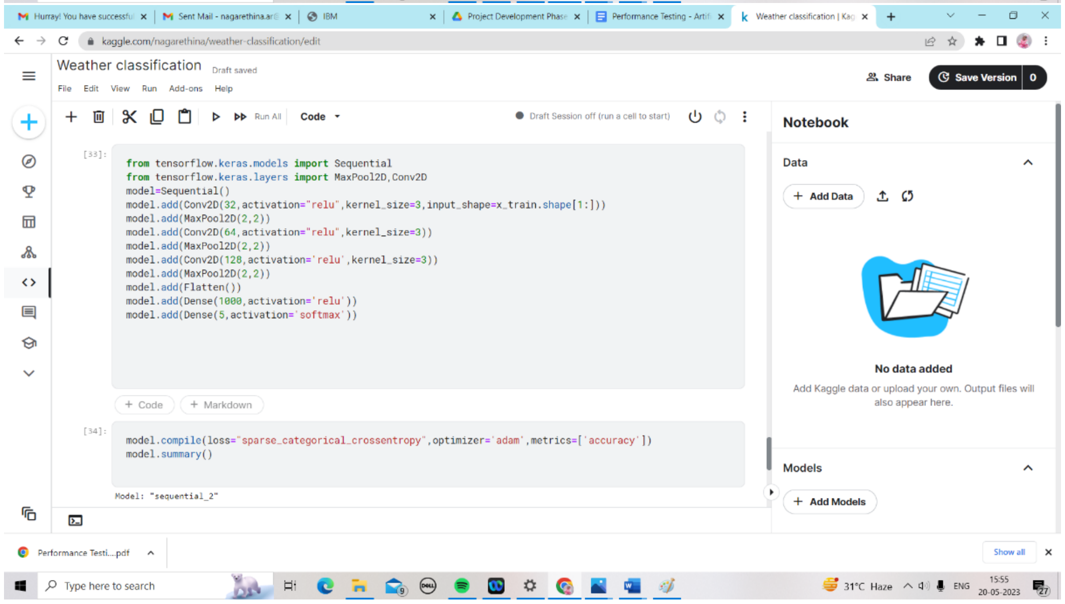
[30]:
```
t = target.cat.codes
t
```

[30]:
```
0       0
1       0
2       0
3       0
4       0
       ..
1495    2
1496    2
1497    2
1498    2
1499    2
Length: 1500, dtype: int8
```

[31]:
```
x_train,x_test,y_train,y_test = train_test_split(x,t,test_size=0.2,random_state=True)
```

[32]:
```
x_train.shape
```

[32]:
```
(1200, 32, 32, 3)
```



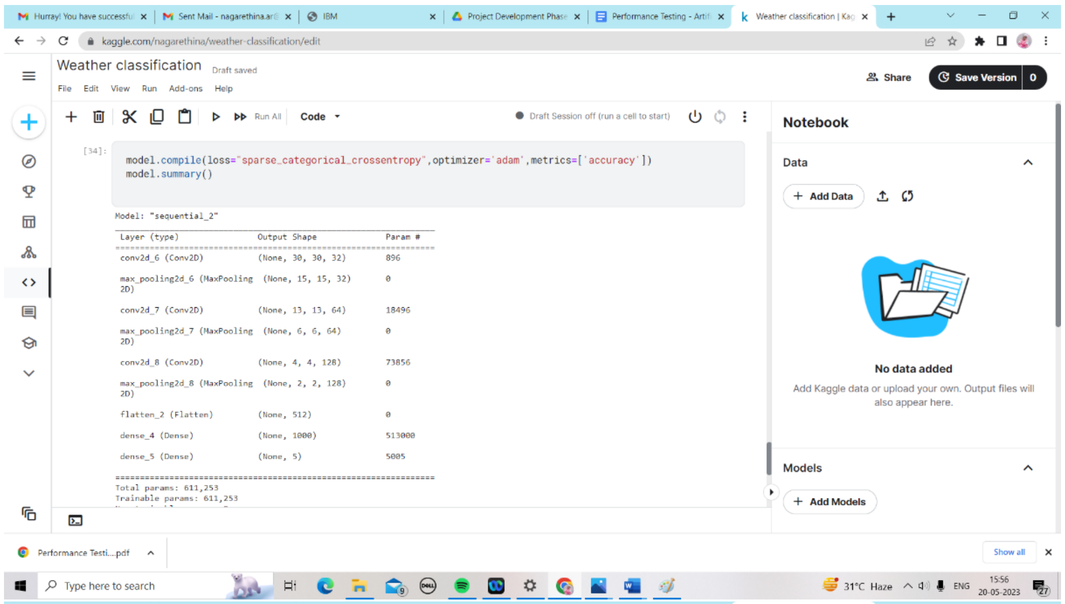
Weather classification

kaggle.com/nagarethina/weather-classification/edit

[33]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import MaxPool2D,Conv2D
model=Sequential()
model.add(Conv2D(32,activation="relu",kernel_size=3,input_shape=x_train.shape[1:]))
model.add(MaxPool2D(2,2))
model.add(Conv2D(64,activation="relu",kernel_size=3))
model.add(MaxPool2D(2,2))
model.add(Conv2D(128,activation='relu',kernel_size=3))
model.add(MaxPool2D(2,2))
model.add(Flatten())
model.add(Dense(1000,activation='relu'))
model.add(Dense(5,activation='softmax'))
```

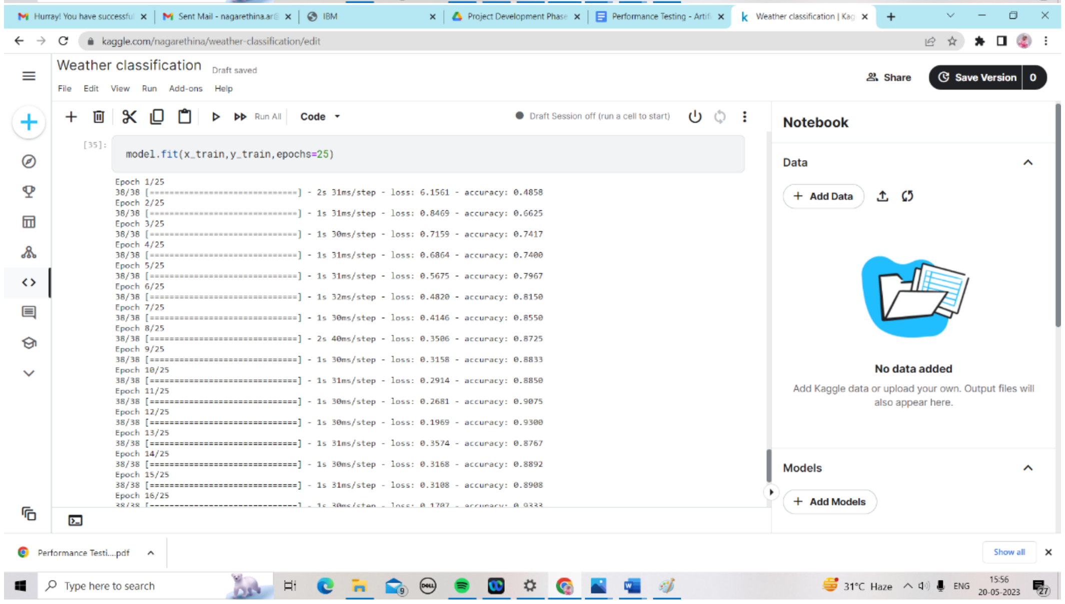
[34]:
```
model.compile(loss="sparse_categorical_crossentropy",optimizer='adam',metrics=['accuracy'])
model.summary()
```

```
Model: "sequential_2"
```


Weather classification
Draft saved
File Edit View Run Add-ons Help
Run All
Code
Draft Session off (run a cell to start)
Share
Save Version
Notebook
Data
Add Data
No data added
Add Kaggle data or upload your own. Output files will also appear here.
Models
Add Models

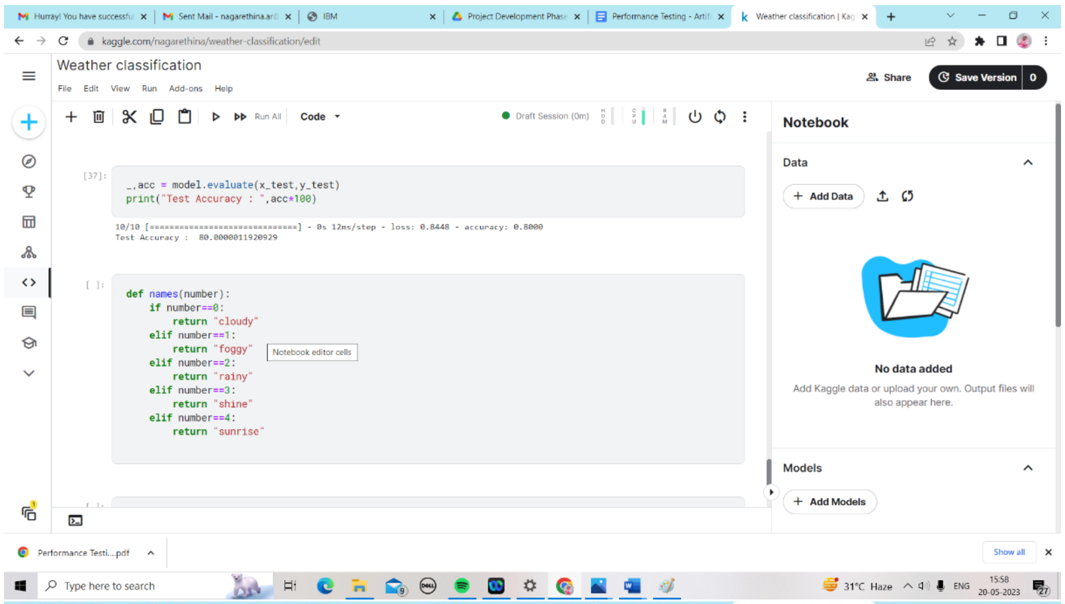
```
[34]: model.compile(loss="sparse_categorical_crossentropy",optimizer='adam',metrics=['accuracy'])
      model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_6 (Conv2D)           (None, 30, 30, 32)        896

 max_pooling2d_6 (MaxPooling  (None, 15, 15, 32)       0
 2D)

 conv2d_7 (Conv2D)           (None, 13, 13, 64)        18496

 max_pooling2d_7 (MaxPooling  (None, 6, 6, 64)         0
 2D)

 conv2d_8 (Conv2D)           (None, 4, 4, 128)         73856

 max_pooling2d_8 (MaxPooling  (None, 2, 2, 128)        0
 2D)

 flatten_2 (Flatten)         (None, 512)               0

 dense_4 (Dense)             (None, 1000)              513000

 dense_5 (Dense)             (None, 5)                 5005

=================================================================
Total params: 611,253
Trainable params: 611,253
```


Weather classification
Draft saved
File Edit View Run Add-ons Help
Run All
Code
Draft Session off (run a cell to start)
Share
Save Version
Notebook
Data
Add Data
No data added
Add Kaggle data or upload your own. Output files will also appear here.
Models
Add Models

```
[35]: model.fit(x_train,y_train,epochs=25)
```

```
Epoch 1/25
38/38 [==============================] - 2s 31ms/step - loss: 6.1561 - accuracy: 0.4858
Epoch 2/25
38/38 [==============================] - 1s 31ms/step - loss: 0.8469 - accuracy: 0.6625
Epoch 3/25
38/38 [==============================] - 1s 30ms/step - loss: 0.7159 - accuracy: 0.7417
Epoch 4/25
38/38 [==============================] - 1s 31ms/step - loss: 0.6864 - accuracy: 0.7400
Epoch 5/25
38/38 [==============================] - 1s 31ms/step - loss: 0.5675 - accuracy: 0.7967
Epoch 6/25
38/38 [==============================] - 1s 32ms/step - loss: 0.4820 - accuracy: 0.8150
Epoch 7/25
38/38 [==============================] - 1s 30ms/step - loss: 0.4146 - accuracy: 0.8550
Epoch 8/25
38/38 [==============================] - 2s 40ms/step - loss: 0.3506 - accuracy: 0.8725
Epoch 9/25
38/38 [==============================] - 1s 30ms/step - loss: 0.3158 - accuracy: 0.8833
Epoch 10/25
38/38 [==============================] - 1s 31ms/step - loss: 0.2914 - accuracy: 0.8850
Epoch 11/25
38/38 [==============================] - 1s 30ms/step - loss: 0.2681 - accuracy: 0.9075
Epoch 12/25
38/38 [==============================] - 1s 30ms/step - loss: 0.1969 - accuracy: 0.9300
Epoch 13/25
38/38 [==============================] - 1s 31ms/step - loss: 0.3574 - accuracy: 0.8767
Epoch 14/25
38/38 [==============================] - 1s 30ms/step - loss: 0.3168 - accuracy: 0.8892
Epoch 15/25
38/38 [==============================] - 1s 31ms/step - loss: 0.3108 - accuracy: 0.8908
Epoch 16/25
38/38 [==============================] - 1s 30ms/step - loss: 0.1707 - accuracy: 0.9333
```

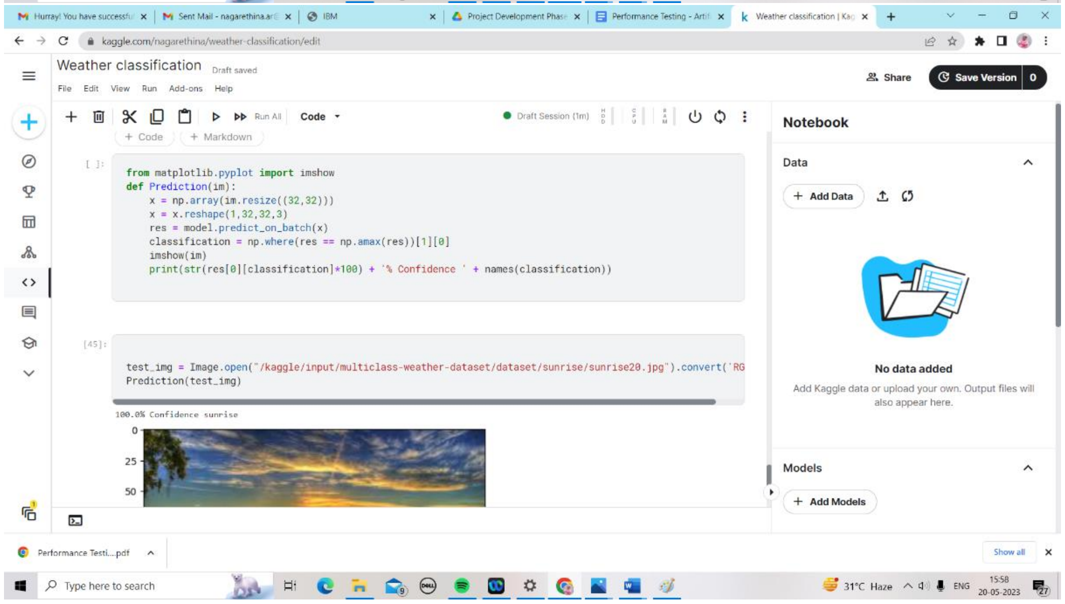


Weather classification

```
[37]:
_,acc = model.evaluate(x_test,y_test)
print("Test Accuracy : ",acc*100)
```

```
10/10 [==============================] - 0s 12ms/step - loss: 0.8448 - accuracy: 0.8000
Test Accuracy :  80.0000011920929
```

```
[ ]:
def names(number):
    if number==0:
        return "cloudy"
    elif number==1:
        return "foggy"
    elif number==2:
        return "rainy"
    elif number==3:
        return "shine"
    elif number==4:
        return "sunrise"
```

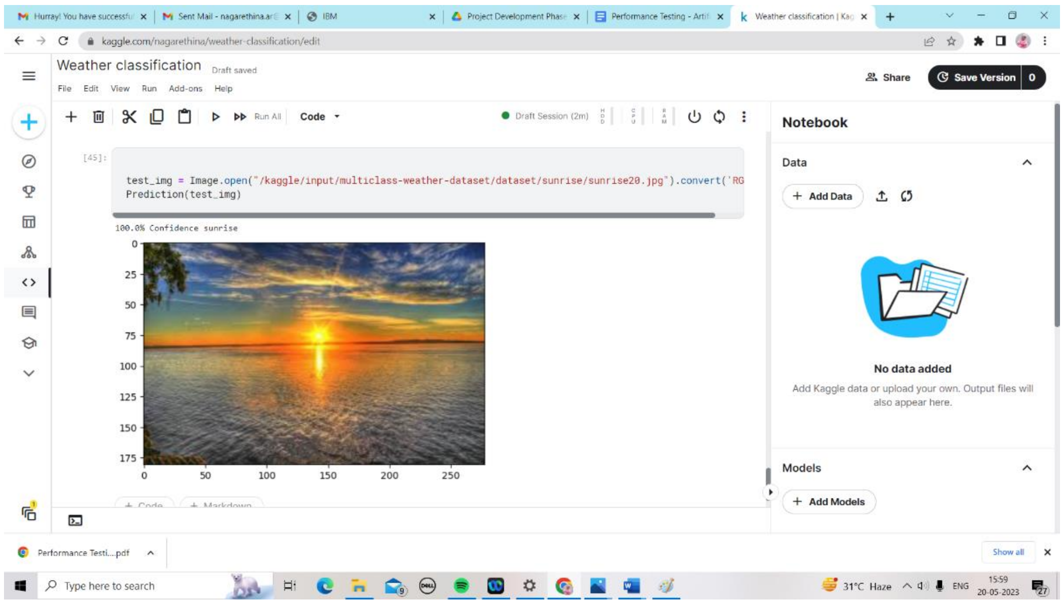


Weather classification Draft saved

```
[ ]:
from matplotlib.pyplot import imshow
def Prediction(im):
    x = np.array(im.resize((32,32)))
    x = x.reshape(1,32,32,3)
    res = model.predict_on_batch(x)
    classification = np.where(res == np.amax(res))[1][0]
    imshow(im)
    print(str(res[0][classification]*100) + '% Confidence ' + names(classification))
```

```
[45]:
test_img = Image.open("/kaggle/input/multiclass-weather-dataset/dataset/sunrise/sunrise20.jpg").convert('RG
Prediction(test_img)
```

```
100.0% Confidence sunrise
```



```
[45]:

test_img = Image.open("/kaggle/input/multiclass-weather-dataset/dataset/sunrise/sunrise20.jpg").convert('RG
Prediction(test_img)
```

```
100.0% Confidence sunrise
```