

**PREDICTIVE ANALYTICS PROJECT**

**REPORT on**

Flower Prediction Model

**Submitted By:**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Name** | **Course** | **Sap ID** |
| 1 | Lavanya Arora | B.Tech-CSE-AIML-hons | 500101910 |
| 3 | Varun Gupta | B.Tech-CSE-AIML-hons | 500101681 |

**Under the guidance of**

Prof. Achala Shakya

# 

**School of Computer Science**

University of Petroleum & Energy Studies, Dehradun

**Index**

| **Sr. No.** | **Contents** |
| --- | --- |
| 1 | Project Title |
| 2 | Abstract |
| 3 | Introduction |
| 5 | Problem Statement |
| 6 | Objectives |
| 7 | Methodology |
| 8 | Implementation |

# 

# 1.Project Title

Flower Prediction Machine Learning Model

# 2. Abstract

The **Flower Prediction Model** is a machine learning project designed to automate the classification of flower species based on measurable physical features such as petal length, petal width, sepal length, and sepal width. Traditional methods of species identification rely heavily on manual examination, which is time-consuming, prone to error, and dependent on expert knowledge.

This project utilizes structured datasets like the Iris dataset and applies machine learning techniques such as Decision Trees, Support Vector Machines (SVM), and Neural Networks to build a scalable, accurate, and efficient classification model. The system is designed to handle both small and large datasets while maintaining reliability.

The project is of significant value in fields like botany, environmental science, gardening, and agriculture, where quick and precise flower identification is crucial. Furthermore, this work lays the groundwork for future advancements, including the integration of image-based recognition for real-time applications. By harnessing the potential of artificial intelligence, the project bridges the gap between technology and environmental sciences, offering an innovative solution to an age-old problem.

# 3. Introduction

Identifying flower species has traditionally been a manual process requiring significant expertise and effort. While effective in controlled environments, manual identification methods are impractical for large-scale applications or non-expert users. This challenge is particularly acute in fields like botany, agriculture, and environmental science, where accurate classification is essential for research, conservation, and productivity.

This project leverages machine learning to create an automated system capable of classifying flower species using measurable physical features. The model analyses patterns. in datasets to make reliable predictions, significantly reducing the time and effort required. for identification.

The key motivation behind this project is to demonstrate how machine learning can transform traditional practices by providing a scalable, efficient, and user-friendly solution. The broader goal is to integrate this model into real-world applications, making flower species identification accessible to researchers, farmers, gardeners, and educators alike.

# 4. Problem statement

The manual identification of flower species is highly dependent on expert knowledge, making it inaccessible to non-specialists. This process is time-consuming, particularly when dealing with large datasets or field surveys. Errors and inconsistencies in manual identification can lead to flawed research findings and inefficiencies in practical applications. There is a critical need for an automated system that can classify flowers with high accuracy, efficiency, and scalability.

# 5. Objectives

* **Build a Machine Learning Model:** Develop a system that uses measurable features (e.g., petal and sepal dimensions) to classify flower species.
* **Optimize Accuracy and Efficiency:** Employ advanced preprocessing techniques, feature selection, and model optimization to achieve high predictive performance.
* **Scalability:** Ensure that the model can handle datasets of varying sizes and adapt to classify a wide range of flower species.
* **User Accessibility:** Design an interactive, user-friendly web interface for practical applications in research, farming, and gardening.
* **Future Developments:** Lay the groundwork for enhancements, such as integrating image-based recognition for real-time prediction and expanding the dataset to include more species.

# 6. Methodology

**1. Data Collection:**

* Utilize the Iris dataset, a well-known structured dataset, as a baseline.
* If required, incorporate additional datasets with labeled flower species and their respective features.
* Features considered include petal length, petal width, sepal length, and sepal width.

**2.** **Data Preprocessing:**

* Clean the data to remove missing values, outliers, or inconsistencies.
* Normalize numerical values to ensure uniformity in scale across features.
* Conduct exploratory data analysis (EDA) to identify patterns, correlations, and feature importance.

**3.** **Model Selection and Development:**

* Experiment with multiple algorithms, such as Logistic Regression, Decision Trees, K-Nearest Neighbors (KNN), Support Vector Machines (SVM), and Neural Networks.
* Evaluate each algorithm’s performance to identify the most suitable model for the classification task.

**4.** **Training and Testing:**

* Split the dataset into training (80%) and testing (20%) subsets.
* Train the model using the training dataset and fine-tune hyperparameters for optimal performance.
* Validate the model on the testing dataset to ensure generalizability.

**5.** **Evaluation Metrics:**

* Use performance metrics such as accuracy, precision, recall, F1 score, and confusion matrix to assess the model’s effectiveness.
* Compare the results with baseline models to validate improvements.

**6.** **Deployment:**

* Package the final model into a web-based application or user-friendly interface for real-world usage.
* Ensure the interface is accessible to both technical and non-technical users.

**7. Implementation**

The flower prediction model is built using the **Iris dataset**, which contains four features (sepal length, sepal width, petal length, petal width) to predict the flower species (Iris-setosa, Iris-versicolor, Iris-virginica).

##### **1. Image Preprocessing**

The function load\_and\_prep(filepath, normalize=False) is responsible for preparing images before feeding them into a model:

* **Input**: The file path of an image.
* **Steps**:
  1. **Read File**: Uses tf.io.read\_file to load the image file as raw data.
  2. **Decode Image**: Converts raw data into a tensor using tf.io.decode\_image.
  3. **Resize Image**: Ensures uniform size (224x224) across all images using tf.image.resize.
  4. **Normalization**: Optionally normalizes pixel values to the range [0, 1] if normalize=True.
* **Output**: A preprocessed image tensor ready for model input.

##### **2. Random Image Plotting**

* Demonstrates how images are loaded and labeled correctly.
* Randomly selects 9 images, preprocesses them using load\_and\_prep, and displays them with their labels.
* Ensures diversity in data visualization by choosing random images.

##### **3. Data Augmentation**

* A tf.keras.Sequential model named data\_augmentation is created to apply random transformations to images for diversity and robustness.
* **Augmentation Techniques**:
  1. **Random Flip**: Horizontally flips the image.
  2. **Random Rotation**: Rotates the image by up to 20%.
  3. **Random Zoom**: Zooms in or out by 20%.
  4. **Random Height and Width**: Adjusts height and width dimensions by 20%.
* **Benefits**:
  1. Improves model generalization by exposing it to variations.
  2. Reduces the risk of overfitting to the training dataset.

##### **4. Augmentation Visualization**

* Two visualization experiments are conducted:
  1. **Single Image Augmentation**:
     + Original and augmented versions of a specific image (daisy) are displayed side-by-side.
     + Highlights how augmentation transforms the image.
  2. **Multiple Image Augmentation**:
     + Original and augmented versions of multiple random images are displayed.
     + Each original image is paired with its augmented counterpart, labeled appropriately.
     + Enhances understanding of the augmentation effects on various samples.

#### **5. Model Training**

* A **K-Nearest Neighbors (KNN)** classifier is used due to its simplicity and effectiveness.
* The optimal number of neighbors (K) is determined using cross-validation.
  + 1. **Model-1**
* The base model used is **EfficientNetB0**, a pre-trained convolutional neural network known for its efficiency and accuracy.
* The top layers of the base model were excluded (include\_top=False), and it was set to non-trainable (base\_model.trainable = False) to retain pre-trained feature extraction capabilities while avoiding overfitting.
* The custom architecture includes:
  + An **input layer** with an input shape of (224, 224, 3) to accommodate RGB images.
  + **Data augmentation** applied to enhance generalization.
  + A **GlobalAveragePooling2D layer**, reducing the spatial dimensions into a feature vector.
  + A **Dense output layer** with 5 units (indicating 5 classes) and a softmax activation function.

#### **Compilation and Optimization**

* **Loss Function**: Categorical Crossentropy, suitable for multi-class classification problems.
* **Optimizer**: Adam, chosen for its adaptive learning rate and computational efficiency.
* **Evaluation Metric**: Accuracy, used to monitor performance during training and validation.

#### **Training Performance**

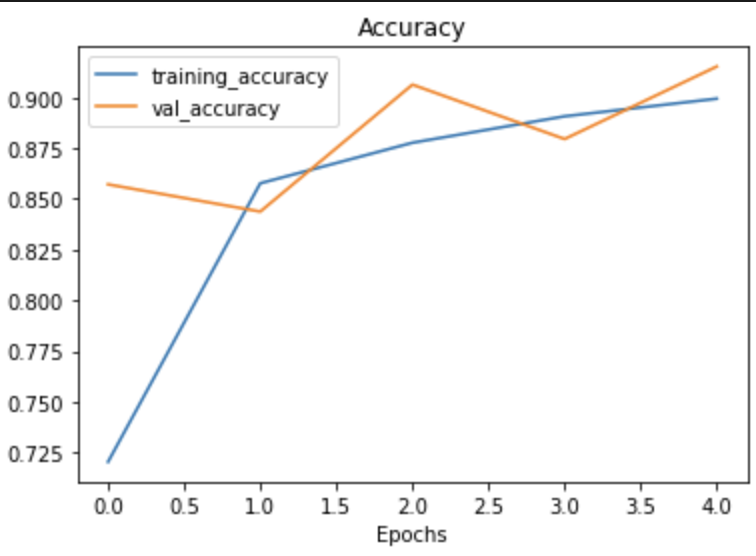
The model was trained for **5 epochs**, with the following key observations:

* **Training Loss and Accuracy**:
  + Starting loss: 0.8168, with an accuracy of 72.06%.
  + Final loss: 0.3036, with an accuracy of 89.93%.
  + The consistent reduction in loss and improvement in accuracy over epochs indicates effective learning.
* **Validation Loss and Accuracy**:
  + Starting loss: 0.4429, with an accuracy of 85.71%.
  + Final loss: 0.2366, with an accuracy of 91.52%.
  + The validation metrics show a stable improvement, suggesting good generalization.

#### **Epoch-wise Highlights**

1. **Epoch 1**: Significant performance improvement, with validation accuracy starting at 85.71%.
2. **Epoch 3**: Validation accuracy peaked at 90.62%, indicating strong generalization.
3. **Epoch 5**: Validation accuracy reached 91.52%, with a significant drop in validation loss to 0.2366.

A graph with blue and orange lines

Description automatically generated

#### **Insights**

* The model demonstrated consistent improvement in both training and validation metrics, with minimal signs of overfitting.
* Validation accuracy consistently exceeded training accuracy, suggesting that the base model's pre-trained features were leveraged effectively.
  + 1. **Model-2**

1. **Base Model**:
   * **Pre-trained Model**: ResNet50V2, pre-trained on ImageNet, used for feature extraction.
   * **Freezing Layers**: All layers of the base model were frozen (trainable=False) to preserve pre-trained weights and prevent overfitting.
   * **No Top Layers**: The top layers were excluded (include\_top=False), allowing custom layers to adapt the model to the dataset.
2. **Input Layer**:
   * **Input Shape**: Images resized to (224, 224, 3) to match ResNet50V2's input requirements.
   * **Rescaling**: Pixel values normalized to the [0, 1] range via Rescaling(1./255) for faster convergence.
3. **Custom Layers**:
   * **Data Augmentation**: Random augmentations (flipping, rotation, zoom, height/width changes) to enhance generalization.
   * **Global Average Pooling**: Reduces feature maps into a single vector for classification.
   * **Output Layer**: A Dense layer with 5 units (for 5 classes) and softmax activation for multi-class classification.
4. **Compilation**:
   * **Loss Function**: Categorical Crossentropy, suitable for multi-class classification.
   * **Optimizer**: Adam, selected for its adaptability and efficiency.
   * **Evaluation Metric**: Accuracy, used for monitoring performance.

### **Training Performance**

The model was trained for **5 epochs** on the given dataset. Below is an epoch-wise performance summary:

#### **Training Loss and Accuracy**:

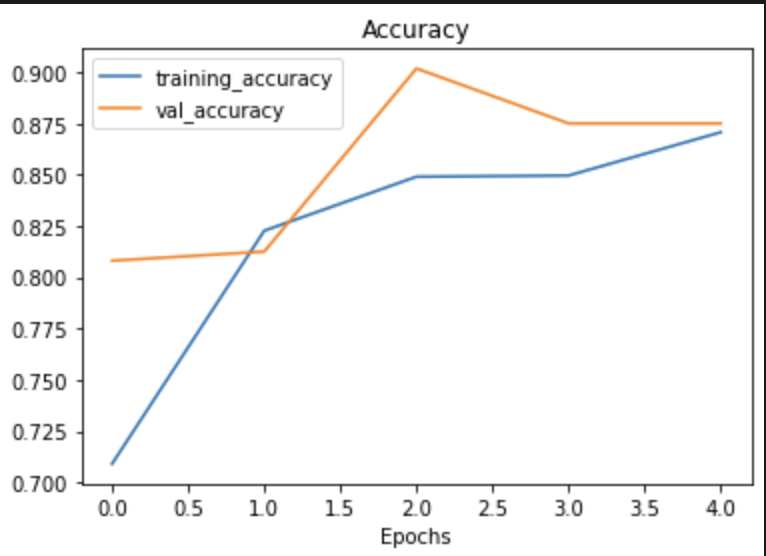
* **Epoch 1**:
  + Loss: 0.8014
  + Accuracy: 70.90%
* **Epoch 5**:
  + Loss: 0.3755
  + Accuracy: 87.07%
* **Observation**: Progressive improvement in training metrics indicates effective learning of the model.

A graph with blue and orange lines

Description automatically generated

#### **Validation Loss and Accuracy**:

* **Epoch 1**:
  + Loss: 0.4933
  + Accuracy: 80.80%
* **Epoch 5**:
  + Loss: 0.3707
  + Accuracy: 87.50%
* **Observation**: The validation accuracy closely tracks training accuracy, suggesting minimal overfitting.



### **Performance Insights**

1. **Strengths**:
   * **Consistent Accuracy**: Both training and validation accuracies improved consistently, reaching approximately 87% at the final epoch.
   * **Generalization**: The gap between training and validation accuracy remained small, indicating robust generalization.
   * **Pre-trained Features**: ResNet50V2's pre-trained weights significantly boosted performance without requiring additional fine-tuning.
2. **Challenges**:
   * **Epoch 2 Plateau**: Validation accuracy stagnated temporarily, possibly due to the optimizer requiring more time to adjust weights.
   * **Slight Overfitting Trend**: A small increase in validation loss during later epochs (0.3896 at epoch 4) might suggest early signs of overfitting.